LLC: A LINEAR LOW-RANK CODEC FOR JOINT KEY-VALUE COMPRESSION BEFORE TOKEN EVICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Token eviction controls KV cache growth but permanently discards states that may become relevant later. Retaining these states in compressed form requires a compact representation that remains useful to attention. We introduce LLC, a closed-form, training-free codec that jointly encodes post-RoPE keys and values across all heads into a shared low-rank code. Fixed linear maps enable direct attention over the codes without reconstructing full-width K/V states. We deploy the codec in a two-tier cache: a base eviction policy manages exact residence, while a separately budgeted compressed tier retains demoted tokens with the lowest reconstruction errors. The joint representation improves reconstruction fidelity and yields stronger accuracy–memory trade-offs across models and eviction policies. Across RULER at 8k, 16k, and 32k, LLC approaches dense accuracy with less KV cache memory; at 32k, it stays within 1.5 points of dense across four models using 31–38% of the dense cache. On LongBench, LLC outperforms a prior codec in all eight matched-memory comparisons. Under CUDA Graph replay, an A100 implementation achieves 1.31–$4.10\times$ speedups over matched dense SDPA in single-token, per-layer attention microbenchmarks from 8k to 64k. INT8 quantization of the stored codes further halves their storage with negligible accuracy loss. Together, LLC mitigates irreversible eviction without sacrificing memory efficiency.

1 INTRODUCTION

Autoregressive LLMs cache K/V states for every past token, causing KV memory to grow linearly with context length. Token eviction controls this cost with a fixed number of KV entries, estimating importance and evicting the least important. Despite this efficiency, eviction is irreversible: retained tokens remain available, while evicted tokens are permanently lost. Because future relevance is unknown at eviction time, even strong policies may remove information needed later. Can we mitigate this loss without giving up the active-token budget that makes eviction attractive?

Retaining compressed states for tokens selected for eviction offers a middle ground between exact retention and removal. The stored representation determines how many demoted tokens fit within a memory budget, how faithfully their states can be recovered, and how attention can use them. This raises a representation question: how can demoted K/V states be stored compactly while remaining directly usable by attention?

LLC addresses this question with a shared low-rank code for all-head post-RoPE keys and values. Their common source in a token's hidden state motivates fitting a joint subspace, which allocates its rank budget across both K/V and heads. For each layer, LLC fits a closed-form linear codec by eigendecomposing the second-moment matrix of cached K/V states. Because the maps operate after RoPE, decoding can be absorbed into attention: queries are projected into code space, and value decoding follows code accumulation. This avoids position-dependent reconstruction and materialized full-width K/V states.

Figure 1 shows how we use this codec for eviction recovery. A base policy manages a fixed exact tier, and demoted tokens enter a separately budgeted compressed tier. The same basis provides a normalized reconstruction residual, which we use to retain the lowest-error codes when that tier is full. This two-stage design keeps both capacities bounded and leaves the base policy's exact-selection rule unchanged. We evaluate the admission rule separately from the codec's representation and execution

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.

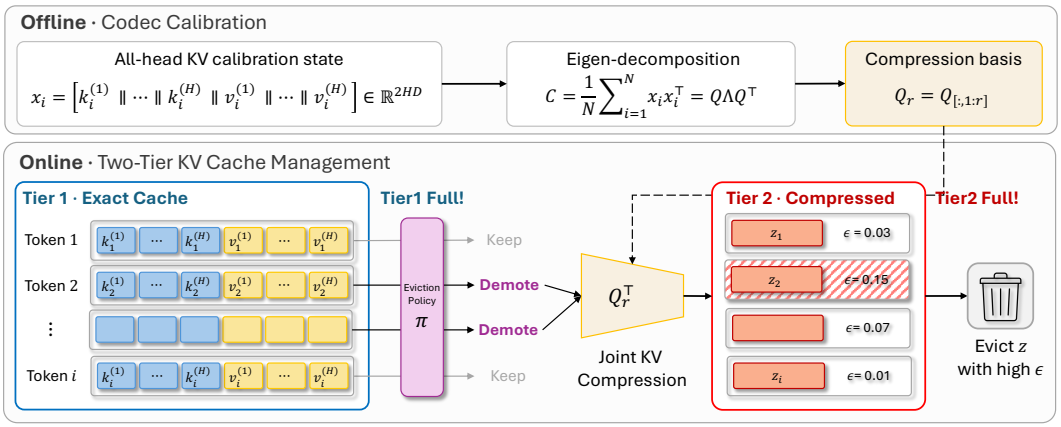

Figure 1: Overview of LLC. Offline, a shared low-rank basis is fitted to all-head post-RoPE K/V states. Online, a base eviction policy manages the fixed exact tier; demoted tokens are jointly encoded into a fixed compressed tier, which retains the codes with the lowest normalized reconstruction errors.

properties. RULER and LongBench evaluations show favorable accuracy–memory trade-offs across models and eviction policies. Fixed maps also remain effective beyond the 8k calibration length.

Our contributions can be summarized as follows:

- **One code for all K/V.** We introduce a closed-form post-RoPE codec that stores all KV heads in one shared rank-r code, allocating its dimensions jointly across keys, values, and heads without model retraining. A bounded two-tier cache retains these codes beyond the exact cache.
- **Near-dense accuracy at 31% of the KV payload.** At 32k, LLC stays within 0.6 RULER points of dense across four models using 40 rather than 128 MiB of KV payload per full-attention layer (31.25%, excluding fixed codec maps). This common setting uses $r = 512$, 8k exact and 8k compressed tokens under ValueDiff, with maps calibrated at up to 8k and reused without refitting (Table 3).
- **Attention directly over codes.** We move decoding outside the token scan without materializing reconstructed K/V states. An A100 implementation achieves 1.31–4.10× speedups over matched dense SDPA in single-token, per-layer attention microbenchmarks under CUDA Graph replay from 8k to 64k (Section 5.3). INT8 codes further halve compressed-state storage with negligible accuracy loss.

2 Related Work

Token eviction. Token eviction bounds the number of cached positions by retaining tokens selected through attention statistics, key geometry, or other importance signals. Representative methods include H_2O (Zhang et al., 2023), SnapKV (Li et al., 2024), StreamingLLM (Xiao et al., 2024), KeyDiff (Park et al., 2025), and ValueDiff (Park et al., 2026). ValueDiff, in particular, ranks tokens by their value-vector distance from the cache mean. These methods primarily differ in how they determine which tokens remain exact. LLC is complementary to this line of work: it uses an existing eviction policy as the base policy and operates only on tokens that the policy selects for removal.

Low-precision and low-dimensional KV compression. Approximate K/V states can reduce memory along two complementary axes: the number of bits per stored element and the number of elements stored per token. Quantization targets the first. Representative methods such as KIVI (Liu et al., 2024), KVQuant (Hooper et al., 2024), and SKVQ (Duanmu et al., 2024) improve low-bit fidelity through asymmetric schemes, explicit outlier handling, and window-aware quantization. MiKV (Yang et al., 2024) further couples precision with token importance, retaining important states at high precision and states selected for eviction at lower precision. This reduces bits per retained position, whereas LLC fixes the capacities of both cache tiers and reduces the number of elements

stored for each compressed token. The two choices are complementary: LLC codes can themselves be quantized, as demonstrated in Section 5.4.

Low-rank methods target the second axis by projecting keys and values into smaller feature spaces. Representative methods include EigenAttention (Saxena et al., 2024), Palu (Chang et al., 2024), and LoRC (Zhang et al., 2024). These approaches include both architectural modifications trained with the model and post-hoc codecs. OjaKV (Zhu et al., 2026) uses online-adapted, separate K/V bases, representing keys before RoPE and retaining high-error states at full rank. Its reconstruction error helps determine exact residence; LLC applies that signal to tokens demoted by the base policy at each update, using a fixed joint all-head post-RoPE subspace. We therefore use factorized codecs as the direct representation comparison.

Reconstructability-aware cache management. VECTOR (Lin et al., 2026) uses reconstruction error within a joint three-way allocation over an expanded importance-selected pool. It retains each selected token's full-width key and predicts only its value from the corresponding pre-RoPE key, obtained through inverse RoPE. LLC instead uses the base policy's exact-selection rule, then uses reconstruction error to admit demoted tokens to a separate compressed tier. It jointly encodes and reconstructs post-RoPE K/V rather than retaining a full-width K. OjaKV's key codec and VECTOR's value predictor therefore depend on pre-RoPE keys, whereas LLC enables the token-independent decoder absorption in Section 3.2. We evaluate VECTOR as the most relevant recovery baseline.

3 LLC

LLC represents each demoted token's all-head post-RoPE K/V states with a shared low-rank code. We first define the codec and its reconstruction residual, then describe its use in a bounded two-tier cache and derive attention directly over the codes.

3.1 Joint Low-Rank Codec

At a layer and token, keys and values across heads derive from the same hidden state and may therefore be correlated. LLC exploits this shared source by fitting one low-dimensional subspace over their joint representation.

Formally, fix one layer with H KV heads of width D. For each token, let $k, v \in \mathbb{R}^{HD}$ denote its cached post-RoPE key and value, concatenated across heads, and form

$$x := [k \| v] \in \mathbb{R}^{2HD}. \tag{1}$$

The shared basis allocates a fixed code budget jointly across keys, values, and heads. We fit one codec per layer, offline, from N calibration vectors. Stack their keys and values row-wise as $K, V \in \mathbb{R}^{N \times HD}$. LLC accumulates the uncentered second-moment matrix

$$C = \frac{1}{N} \sum_{i=1}^{N} x_i x_i^\top = \frac{1}{N} \begin{bmatrix} K^\top K & K^\top V \\ V^\top K & V^\top V \end{bmatrix}, \qquad C = Q \Lambda Q^\top, \tag{2}$$

where the eigenvalues are sorted in descending order. Let $Q_r = Q_{:,1:r} \in \mathbb{R}^{2HD \times r}$ contain the leading r eigenvectors. They give the optimal rank-r orthogonal projection for the empirical squared reconstruction objective and define both maps of the codec:

$$z = Q_r^\top x, \qquad \widehat{x} = Q_r z = [\widehat{k} \| \widehat{v}]. \tag{3}$$

Equivalently, $W_{\text{enc}} = Q_r^\top$ and $W_{\text{dec}} = Q_r$. Calibration requires neither model retraining nor gradient descent, and the fixed maps are reused for every token. The compressed KV cache stores only the r-dimensional code z for each token; neither x nor $\widehat{x}$ is retained.

For a token i, LLC ranks codes for admission by their normalized reconstruction residual. The orthonormality of Q_r lets this score be evaluated without materializing $\widehat{x}_i$:

$$\epsilon_i = \|x_i - Q_r Q_r^\top x_i\|_2^2 / \|x_i\|_2^2 = 1 - \|z_i\|_2^2 / \|x_i\|_2^2. \tag{4}$$

Table 1: Post-RoPE K/V reconstruction R^2 (six-model mean). All variants are PCA codecs with the same per-token code size. Check marks indicate which inputs are concatenated before projection. LLC achieves the highest mean in every column.

	Concatenated		$r = 256$		$r = 512$	
Codec	K/V	Heads	RULER-8K	C4	RULER-8K	C4
Sep	–	–	.618	.641	.742	.749
KV-Joint	✓	–	.706	.717	.816	.819
Head-Joint	–	✓	.802	.807	.881	.884
LLC	✓	✓	**.849**	**.853**	**.914**	**.917**

Algorithm 1 Two-tier cache management with LLC

Require: incoming K/V states $\mathcal{I} = \{x_i = [k_i \| v_i]\}$; exact cache $\mathcal{E}$ (capacity B_e) under policy π; compressed cache $\mathcal{C}$ (capacity B_c); basis Q_r
Ensure: updated $\mathcal{E}$ and $\mathcal{C}$

$\mathcal{E} \leftarrow \mathcal{E} \cup \mathcal{I}$ ▷ admit incoming states
if $|\mathcal{E}| > B_\mathrm{e}$ **then**
  $\mathcal{D} \leftarrow \pi(\mathcal{E}, B_\mathrm{e})$; $\mathcal{E} \leftarrow \mathcal{E} \setminus \mathcal{D}$ ▷ demote excess states
  for each $x_i = [k_i \| v_i]$ in $\mathcal{D}$ **do**
    $z_i \leftarrow Q_r^\top x_i$; $\epsilon_i \leftarrow 1 - \|z_i\|_2^2 / \|x_i\|_2^2$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{(z_i, \epsilon_i)\}$
  end for
end if
$\mathcal{C} \leftarrow$ the $\min(B_\mathrm{c}, |\mathcal{C}|)$ entries with smallest ϵ_i
return $\mathcal{E}, \mathcal{C}$

Why joint K/V across heads? The shared-source argument motivates sharing a basis across two axes: K/V and heads. We isolate these choices with four PCA codecs at fixed total code dimension r. Decomposed variants divide r uniformly across blocks; Sep, for example, assigns $r/(2H)$ to each KV head. Table 1 reports held-out post-RoPE R^2 on RULER-8K and C4.

Table 1 reveals a clear ordering: reconstruction improves each time the code can share capacity more broadly. Joining K and V within each head improves over Sep, sharing capacity across heads produces a larger gain, and LLC combines both choices to perform best at both code dimensions and on both distributions. This progression supports the shared-source intuition: a fixed rank budget is better allocated where the joint K/V representation concentrates its energy than preassigned to separate signals or heads. Controlled pre-/post-RoPE results are reported in Section C.2.

3.2 Decoupled Two-Stage Cache Management

LLC gives each token a one-way lifecycle, summarized in Algorithm 1. New tokens enter the exact cache $\mathcal{E}$, which holds at most B_e tokens; a base eviction policy π determines which tokens remain exact. Rather than immediately discarding a token selected by π, LLC encodes its K/V state and moves the resulting code to the compressed cache $\mathcal{C}$, which holds at most B_c codes. When a new code would exceed this budget, LLC retains the codes with the smallest reconstruction residuals from Equation (4) and evicts the rest. Tokens therefore move only from exact to compressed to evicted, and the number of active cache entries never exceeds $B_\mathrm{e} + B_\mathrm{c}$. The residual measures reconstruction fidelity, while future token relevance remains unknown. We evaluate this admission heuristic in Section 5.2.

Code-space attention. Writing $Q_r = [A_K; A_V]$, post-RoPE codes give $q^\top \widehat{k}_t = (A_K^\top q)^\top z_t$ and $\sum_t \alpha_t \widehat{v}_t = A_V(\sum_t \alpha_t z_t)$. Thus the transformed query is reused across tokens and value decoding follows code accumulation. A pre-RoPE code instead gives $q^\top R_t A_K z_t = (A_K^\top R_t^\top q)^\top z_t$; because R_t varies by position, this transform remains inside the token scan. LLC therefore avoids materialized K/V reconstruction and position-dependent decoding. The complete derivation and cost analysis appear in Section B.

Table 2: Stored state and reconstruction for one token assigned to an approximate or compressed cache state. For Sep, $z_h^K = (Q_h^K)^\top k_h$ and $z_h^V = (Q_h^V)^\top v_h$ each have dimension $r/(2H)$, giving r stored elements in total. Fixed per-layer codec parameters are excluded.

Method	Stored state	Reconstruction	Elements/token
Sep	$\{z_h^K, z_h^V\}_{h=1}^H$	$\widehat{k}_h = Q_h^K z_h^K,\ \widehat{v}_h = Q_h^V z_h^V$	r
VECTOR	k	$\widehat{k} = k,\ \widehat{v} = W_{\mathrm{OLS}} k$	HD
LLC	$z = Q_r^\top [k\|v]$	$[\widehat{k}\|\widehat{v}] = Q_r z$	r

4 EXPERIMENTS

We test whether LLC recovers evicted information across policies and context lengths while improving end-task quality–memory tradeoffs over alternative codecs.

Eviction-policy coverage. To test whether LLC's benefit depends on a particular base eviction policy, we evaluate four distinct selection criteria: attention-sink retention (StreamingLLM; Xiao et al., 2024), attention scores (SnapKV; Li et al., 2024), key diversity (KeyDiff; Park et al., 2025), and value-space diversity (ValueDiff; Park et al., 2026).

Codec baselines. To isolate the effect of LLC's joint codec design within the proposed two-tier cache, we compare against two reconstruction baselines. (1) Sep uses independent low-rank bases for each key and value head under the same total rank budget and cache management as LLC (Section 3.1). Sep is motivated by the per-head projections in OjaKV (Zhu et al., 2026). (2) VECTOR retains a full-width key and reconstructs only its value using a per-layer linear map calibrated by ordinary least squares (OLS), providing the closest alternative recovery design (Lin et al., 2026). Table 2 summarizes the state stored by each codec and its reconstruction. All three codecs are calibrated offline once per model using post-RoPE states from the same token-balanced, five-domain corpus, with each sample capped at 8,192 tokens. The fitting split contains approximately 41M token positions; Appendix A details the sources, sampling, and holdout protocol.

Memory accounting. Reported KV cache memory includes persistent exact KV and compressed token states but excludes fixed codec maps and temporary kernel workspace, which are quantified in Section B. Caches use BF16 unless otherwise noted.

4.1 RULER RESULTS

Protocol. We evaluate the 13-task mean on RULER-8K (Hsieh et al., 2024) using 50 samples per task. The five exact/compressed cache splits are $2\text{k} + \{2\text{k}, 4\text{k}, 6\text{k}\}$ and $4\text{k} + \{2\text{k}, 4\text{k}\}$. The corresponding $B_\text{c} = 0$ configurations provide exact-only baselines, and an 8k exact cache provides the dense reference. All codecs operate on post-RoPE states; LLC and Sep use $r \in \{256, 512\}$.

Coverage. Figure 2 compares ValueDiff, KeyDiff, StreamingLLM, and SnapKV on three 4B-class models chosen to span distinct attention architectures: Qwen3-4B (full attention), Gemma-3-4B (sliding-window/full-attention hybrid), and Qwen3.5-4B (GatedDeltaNet/full-attention hybrid). For the hybrid models, LLC is applied only to the full-context attention layers. The appendix extends each architecture class with an additional model and reports the complete budget grid (Section D.1).

Under ValueDiff and KeyDiff, the exact-only caches already retain much of the dense accuracy. LLC generally narrows the remaining gap at both exact-cache capacities, with non-monotonic cases as B_c grows. The two code dimensions expose a consistent tradeoff: $r = 256$ provides lower-memory operating points, whereas $r = 512$ generally achieves higher accuracy.

StreamingLLM provides a stringent recovery setting, as its exact-only cache suffers substantial accuracy loss. At larger compressed capacities, LLC recovers much of this gap. SnapKV exhibits an even cleaner pattern: LLC approaches dense accuracy across all three models while using less than half of the dense KV cache memory. Together, these results show that compressed recovery remains effective across eviction policies with substantially different selection criteria.

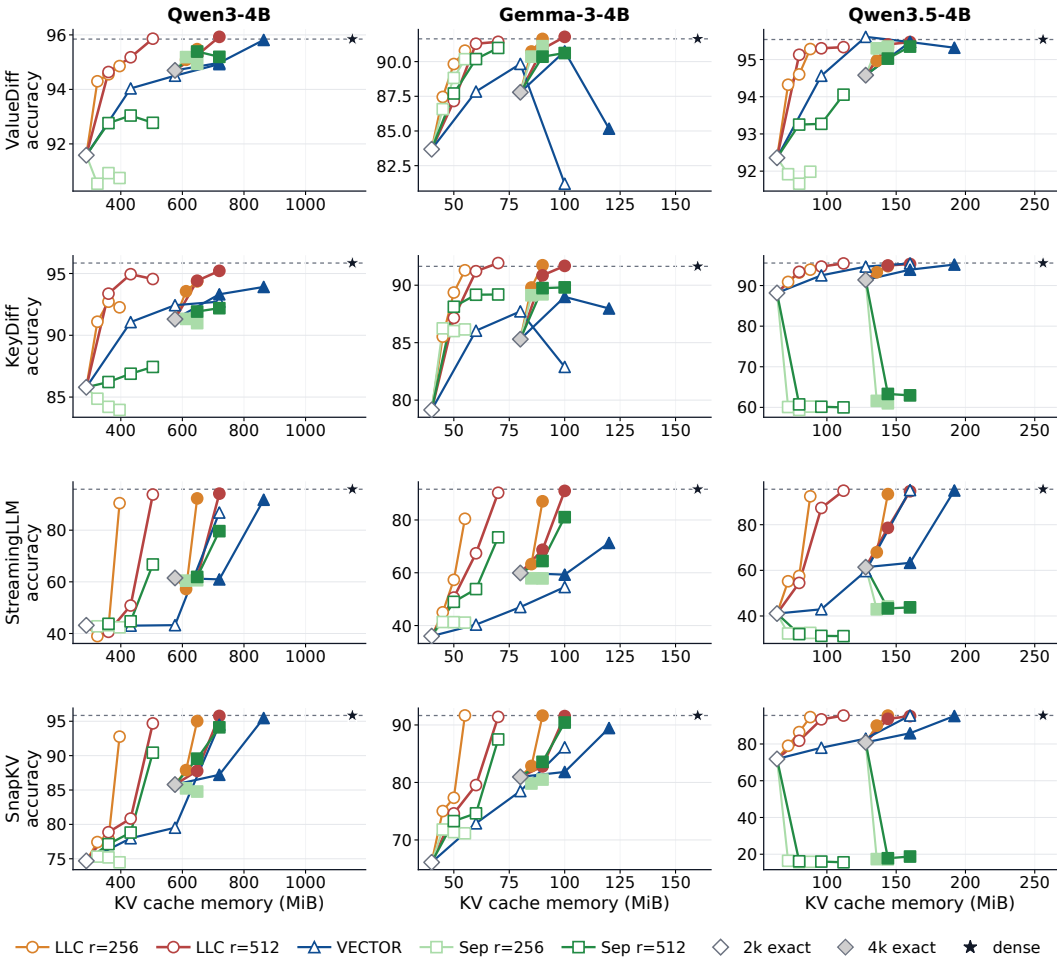

Figure 2: RULER-8K accuracy versus total KV cache memory under ValueDiff (top), KeyDiff, StreamingLLM, and SnapKV (bottom), post-RoPE. Columns show Qwen3-4B, Gemma-3-4B, and Qwen3.5-4B. LLC uses $r = 256$ (orange) or 512 (crimson), VECTOR is blue, and Sep uses $r = 256$ (light green) or 512 (dark green). Hollow markers use $B_e = 2048$ and filled markers use $B_e = 4096$; diamonds mark exact-only baselines. The dotted line and star show the dense reference. Full results for LLC and VECTOR are in Section D.1.

VECTOR also recovers discarded information, confirming the general benefit of a second tier. However, it retains a full-width key for each compressed token and therefore requires more memory. Sep and VECTOR can become non-monotonic as B_c grows or fall below exact-only accuracy, and LLC also has such exceptions. On Qwen3-4B at $r = 512$ and $B_e = 2$k, KeyDiff accuracy falls from 94.94 to 94.55 as B_c grows from 4k to 6k; under StreamingLLM, adding a 2k compressed tier lowers 43.17 to 40.65 (Section D.1). Thus, retaining more compressed states does not guarantee better accuracy, despite the overall recovery gains.

4.2 Extrapolating Beyond the Calibration Length

An offline codec is most useful when it can be reused beyond the context lengths seen during calibration. We therefore freeze the per-layer maps fitted on sequences of at most 8k tokens and evaluate them on RULER at 16k and 32k, without collecting additional statistics or refitting. We use the ValueDiff eviction policy and preserve two proportional cache allocations across lengths: the exact and compressed capacities together retain 50% or 75% of the context positions, as shown in Table 3.

Table 3: RULER accuracy and KV cache memory beyond the 8k calibration length. All codecs reuse fixed post-RoPE maps under the ValueDiff eviction policy. Each cell reports accuracy followed by total exact+compressed KV cache memory per layer in MiB. Columns show exact + compressed token capacities. Bold marks the best compressed method for each model and allocation.

		16k context		32k context	
Model	Method	4k + 4k	4k + 8k	8k + 8k	8k + 16k
Qwen3-4B	Dense	95.47 (64)		95.05 (128)	
	LLC (r=256)	93.68 (18)	93.86 (20)	93.37 (36)	93.72 (40)
	LLC (r=512)	**93.96** (20)	**94.44** (24)	**95.17** (40)	**94.83** (48)
	VECTOR	93.41 (24)	93.73 (32)	93.09 (48)	78.23 (64)
Gemma-3-4B	Dense	89.61 (64)		86.17 (128)	
	LLC (r=256)	84.89 (18)	**87.78** (20)	85.88 (36)	**86.22** (40)
	LLC (r=512)	85.01 (20)	87.76 (24)	**86.01** (40)	85.86 (48)
	VECTOR	**85.06** (24)	82.04 (32)	83.31 (48)	70.78 (64)
Qwen3.5-4B	Dense	95.88 (64)		95.72 (128)	
	LLC (r=256)	95.44 (18)	95.30 (20)	95.48 (36)	95.36 (40)
	LLC (r=512)	95.21 (20)	**95.67** (24)	95.84 (40)	95.67 (48)
	VECTOR	**95.66** (24)	95.52 (32)	**95.89** (48)	**96.09** (64)
Llama-3.1-8B	Dense	94.39 (64)		89.35 (128)	
	LLC (r=256)	90.78 (18)	89.67 (20)	76.25 (36)	78.98 (40)
	LLC (r=512)	**93.23** (20)	**91.67** (24)	**89.92** (40)	**87.94** (48)
	VECTOR	92.31 (24)	89.07 (32)	45.87 (48)	0.00 (64)

At 32k, LLC with r=512 remains within 1.5 accuracy points of dense across all four models and both cache allocations, while using only 40–48 MiB per layer instead of 128 MiB. VECTOR exhibits a markedly model-dependent transfer pattern: it remains near dense on Qwen3.5, but degrades sharply on Qwen3, Gemma-3, and Llama as the compressed history grows. This contrast shows that retaining more approximate states is not inherently beneficial; their utility depends on whether the codec remains reliable beyond its calibration length. LLC provides the more consistent trade-off, although the Llama results indicate that sufficient joint rank is still necessary. Broader 16k grids and additional 32k RoPE and budget ablations appear in Sections D.2 and D.3.

4.3 LONGBENCH RESULTS

On LongBench (Bai et al., 2024), we compare post-RoPE LLC (r=512) with VECTOR under ValueDiff across Qwen3-4B, Gemma-3-4B, Qwen3.5-4B, and Llama-3.1-8B. We report mean scores under both matched token capacities and matched KV cache memory.

At matched token capacities, LLC uses less memory and loses only once. At each exact budget, LLC with 4k compressed tokens matches the memory of VECTOR with 2k and wins all eight. Qwen3.5 is particularly informative: although VECTOR transfers strongly in Table 3, LLC wins all four LongBench comparisons and reaches 99.8% of dense, remaining advantageous even when VECTOR is competitive. LLC is also non-monotonic here: on Llama with $B_e = 8k$, increasing B_c from 2k to 4k lowers accuracy from 48.51 to 47.93.

5 ANALYSIS AND PRACTICAL CONSIDERATIONS

5.1 PRE- VS. POST-ROPE COMPRESSION

Prior work finds that pre-RoPE keys have lower effective rank because position-dependent rotations spread their variance across more directions (Sun et al., 2025; Mu et al., 2025), motivating pre-RoPE compression in Palu and OjaKV; VECTOR similarly applies inverse RoPE before reconstruction (Chang et al., 2024; Zhu et al., 2026; Lin et al., 2026). This representation may be easier to reconstruct but leaves position-dependent rotations in the approximation path. LLC compresses cached post-RoPE keys, enabling the token-independent absorption in Section 3.2 and avoiding inverse and forward RoPE.

Table 4: LongBench under post-RoPE ValueDiff. Cells report the 21-task mean (% dense) and per-layer BF16 KV cache memory; bold marks the matched-token winner. LLC uses r=512.

		VECTOR		LLC (r=512)	
Model	B_e+B_c	Mem. (MiB)	Acc. (% dense)	Mem. (MiB)	Acc. (% dense)
Qwen3-4B	4k + 2k	20	45.85 (95.9%)	18	**46.53 (97.3%)**
	4k + 4k	24	46.08 (96.4%)	20	**46.53 (97.3%)**
	8k + 2k	36	47.13 (98.6%)	34	**47.35 (99.0%)**
	8k + 4k	40	**47.53 (99.4%)**	36	47.32 (99.0%)
Gemma-3-4B	4k + 2k	20	37.26 (90.9%)	18	**40.17 (98.0%)**
	4k + 4k	24	37.89 (92.5%)	20	**40.10 (97.9%)**
	8k + 2k	36	40.22 (98.2%)	34	**42.20 (103.0%)**
	8k + 4k	40	39.38 (96.1%)	36	**42.00 (102.5%)**
Qwen3.5-4B	4k + 2k	20	46.35 (97.3%)	18	**46.67 (97.9%)**
	4k + 4k	24	46.57 (97.7%)	20	**47.10 (98.8%)**
	8k + 2k	36	47.31 (99.1%)	34	**47.53 (99.5%)**
	8k + 4k	40	47.35 (99.1%)	36	**47.67 (99.8%)**
Llama-3.1-8B	4k + 2k	20	35.78 (72.3%)	18	**43.00 (86.9%)**
	4k + 4k	24	41.40 (83.7%)	20	**43.80 (88.5%)**
	8k + 2k	36	43.24 (87.4%)	34	**48.51 (98.0%)**
	8k + 4k	40	45.38 (91.7%)	36	**47.93 (96.8%)**

Table 5: Pre- versus post-RoPE compression: (a) layer-averaged reconstruction R^2 (LLC $r = 256$); (b) representative ValueDiff accuracy at 2k + 2k on RULER-8K (LLC $r = 256$) and 2k + 6k on LongBench (LLC $r = 512$). Broader RULER results are in Section C.2.

(a) Representation fidelity (R^2)

		LLC		VECTOR	
Data	Model	Pre	Post	Pre	Post
RULER-8K	Qwen3	.94	.89	.76	.56
	Llama-3.1	.98	.93	−1.67	.33
C4	Qwen3	.93	.89	.76	.57
	Llama-3.1	.96	.91	.62	−1.08

(b) End-task accuracy

		LLC		VECTOR	
Benchmark	Model	Pre	Post	Pre	Post
RULER-8K	Qwen3	93.99	94.30	94.35	94.04
	Llama-3.1	82.15	83.88	87.73	85.73
LongBench	Qwen3	45.45	44.52	44.84	43.61
	Llama-3.1	37.28	37.19	38.13	33.56
	Gemma-3	39.21	39.33	36.48	33.65

Table 5 confirms the trade-off: post-RoPE lowers reconstruction R^2 by 0.04–0.05, but end-task accuracy changes modestly and without a consistent direction; five of six models in the broader RULER sweep remain within one point of their pre-RoPE result (Table 16). Post-RoPE compression thus preserves comparable accuracy while enabling direct code-space attention.

5.2 SECOND-TIER ADMISSION

Once the compressed cache is full, LLC needs an admission rule. We test whether favoring states the codec can reconstruct faithfully improves end-task accuracy on Qwen3-4B. Holding the codec, base policy, and cache capacities fixed within each setting, we compare low-error admission with FIFO, which evicts the oldest code when the second tier is full, and high-error admission, which reverses LLC's criterion.

Low-error admission beats high-error admission in every setting and FIFO in three of four (Table 6). Relative to FIFO, its differences are +0.89 and −0.14 points under ValueDiff, and +3.00 and +1.26 under KeyDiff. These results support low-error admission on this model, with larger observed gains under KeyDiff. The comparison holds the two-stage cache structure fixed and evaluates the admission rule within it; the magnitude of the benefit depends on the base policy and budget.

Table 6: Second-tier admission accuracy on Qwen3-4B (RULER-8K, r=512); bold marks the best rule per setting.

	ValueDiff		KeyDiff	
Admission	2k + 2k	2k + 4k	2k + 2k	2k + 4k
Low error (ours)	**94.64**	95.18	**93.37**	**94.94**
FIFO	93.75	**95.32**	90.37	93.68
High error	93.02	94.38	88.39	92.49

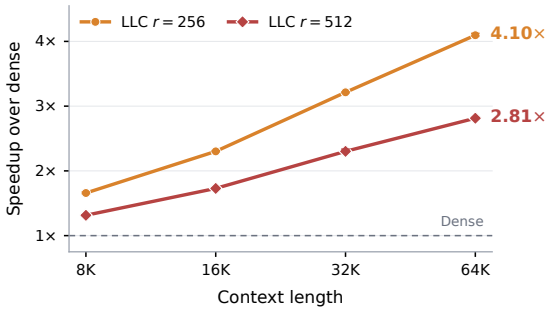

Figure 3: Per-layer attention speedup over dense BF16 SDPA on A100 (batch 1; CUDA Graph replay; hardware-cache reuse perturbed). Error bars show 95% confidence intervals across five independent runs; their full widths are at most 1.7% of the point estimates and are barely visible at this scale.

Method	$B_e + B_c$	r	Code	KV MiB	Accuracy
Dense	8k + 0k	–	–	32.00	95.85
ValueDiff	2k + 0k	–	–	8.00	91.58
ValueDiff	4k + 0k	–	–	16.00	94.70
LLC	2k + 6k	512	BF16	14.00	95.86
			INT8	11.01	**95.62**
			INT4	9.51	93.94
		256	BF16	11.00	94.86
			INT8	9.51	**94.99**
			INT4	8.76	93.39

Table 7: LLC code quantization on Qwen3-4B, RULER-8K, ValueDiff. KV memory: MiB/layer; bold: iso-memory pairs.

5.3 Efficient Attention over Compressed Codes

We benchmark one-token, per-layer attention on an A100 at batch size one, with 32 query heads, 8 KV heads, head width 256, and $B_e = 2k$. Under CUDA Graph replay, LLC is faster than matched dense BF16 SDPA from 8k to 64k, reaching $4.10\times$ at $r = 256$ and $2.81\times$ at $r = 512$ (Figure 3). The benchmark's KV width $HD = 2048$ differs from $HD = 1024$ in the four main accuracy models, giving different compression ratios at the same rank. Prepared-cache timings exclude encoding, admission, cache updates, and other model computation (Section B).

5.4 Compatibility with Quantized Codes

Low rank and quantization reduce the number of elements per token and bits per element, respectively. We apply signed symmetric per-tensor quantization to each code while keeping the exact tier in BF16. Assuming packed codes and one BF16 scale per token, INT8 nearly preserves accuracy while halving code storage (Table 7). At similar memory, $r = 512$ INT8 outperforms $r = 256$ BF16 by 0.76 points, showing that lower precision can support a larger code. Relative to INT8, INT4 saves memory but loses 1.6–1.7 points; at 9.51 MiB/layer, $r = 256$ INT8 outperforms $r = 512$ INT4 by 1.05 points.

6 Conclusion

Token eviction controls KV cache growth but permanently removes states that may become useful later. LLC retains demoted states with a closed-form, training-free codec that jointly encodes all-head post-RoPE K/V. The shared representation supports direct code-space attention and INT8 code quantization. Deployed in a bounded two-tier cache, LLC recovers accuracy across models, policies, budgets, and context lengths while improving accuracy–memory trade-offs over the evaluated alternative codecs. A focused Qwen3-4B ablation supports low-error admission within the compressed tier, with gains that depend on the base policy and budget. LLC thus combines compact retention beyond the exact cache with an efficient attention path.

AI USE STATEMENT

In this work, we used generative AI tools to refine the conceptual framing, provide feedback on methodology and experimental design, assist with code implementation and debugging, and support the interpretation of experimental results. We also used these tools to organize and edit the manuscript.

REPRODUCIBILITY STATEMENT

Section 3 specifies the closed-form codec construction, two-tier cache lifecycle, admission criterion, and attention computation. Section 4 describes the calibration data, evaluated models, eviction policies, cache budgets, benchmark protocols, and memory accounting. The appendices provide full RULER grids, calibration and holdout details, and additional implementation and inference details.

REFERENCES

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, 2024. doi: 10.18653/v1/2024.acl-long.172.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S. Abdelfattah, and Kai-Chiang Wu. Palu: Compressing KV-cache with low-rank projection. *arXiv preprint arXiv:2407.21118*, 2024.

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

Haojie Duanmu, Zhihang Yuan, Xiuhong Li, Jiangfei Duan, Xingcheng Zhang, and Dahua Lin. SKVQ: Sliding-window key and value cache quantization for large language models. *arXiv preprint arXiv:2405.06219*, 2024.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024. doi: 10.52202/079017-0040.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024. doi: 10.52202/079017-0722.

Yuping Lin, Jiayuan Ding, Yue Xing, Pengfei He, Jiliang Tang, and Subhabrata Mukherjee. A simple plug-in for improving eviction-based KV cache compression. *arXiv preprint arXiv:2605.23258*, 2026.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. *arXiv preprint arXiv:2402.02750*, 2024.

Junlin Mu, Hantao Huang, Jihang Zhang, Minghui Yu, Tao Wang, and Yidong Li. SALS: Sparse attention in latent space for KV cache compression. In *Advances in Neural Information Processing Systems*, volume 38, pp. 428–447, 2025. doi: 10.52202/085713-0013.

Junyoung Park, Dalton Jones, Matthew Morse, Raghavv Goel, Mingu Lee, and Christopher Lott. KeyDiff: Key similarity-based KV cache eviction for long-context LLM inference in resource-constrained environments. *Advances in Neural Information Processing Systems*, 38:6771–6807, 2025. doi: 10.52202/085713-0210.

Junyoung Park, Jungwook Choi, and Mingu Lee. ValueDiff: Value-geometric KV cache eviction for sink-suppressed LLMs. *arXiv preprint arXiv:2609.23314*, 2026.

Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. Eigen attention: Attention in low-rank space for KV cache compression. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 15332–15344, 2024. doi: 10.18653/v1/2024.findings-emnlp.899.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. ShadowKV: KV cache in shadows for high-throughput long-context LLM inference. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57355–57373, 2025.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. No token left behind: Reliable KV cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*, 2024.

Rongzhi Zhang, Kuang Wang, Liyuan Liu, Shuohang Wang, Hao Cheng, Chao Zhang, and Yelong Shen. LoRC: Low-rank compression for LLMs KV cache with a progressive compression strategy. *arXiv preprint arXiv:2410.03111*, 2024.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023. doi: 10.52202/075280-1506.

Yuxuan Zhu, David H. Yang, Mohammad Mohammadi Amiri, Keerthiram Murugesan, Tejaswini Pedapati, and Pin-Yu Chen. OjaKV: Context-aware online low-rank KV cache compression. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 10161–10178, 2026. doi: 10.18653/v1/2026.findings-acl.494.

LLC
Supplementary Material

A Calibration and Fitting Details

All LLC, Sep, and VECTOR results use the same model-specific offline calibration split. We construct this calibration mixture from five domains: web prose, instruction-following code, mathematical text, Chinese-language text, and multi-turn dialogue. The web slice draws from the C4, Common-Crawl, and Wikipedia subsets of SlimPajama; the code slice uses Magicoder evol-instruct examples; the math slice combines NuminaMath with SlimPajama's ArXiv subset; the Chinese slice uses FineWeb-Edu-Chinese; and the dialogue slice uses UltraChat. We first collect an equal approximate token budget from each domain (9.7M tokens per slice). A sample's contribution to this collection budget is capped at the approximate length of an 8,192-token sequence, preventing a few long samples from dominating a slice.

After a deterministic shuffle, we retain at most 12,000 samples from each domain, yielding a 60,000-sample pool. We then form a stratified 15% holdout within each domain and fit on the remaining samples. Because samples have variable token lengths, the resulting fitting set contains approximately 41M token positions rather than $51{,}000 \times 8{,}192$ positions.

Each sample is one calibration example. We apply the checkpoint's chat template to dialogue samples and tokenize every example independently with a maximum length of 8,192 tokens; samples shorter than 32 tokens are excluded. We do not concatenate samples to fill a context window, and we do not split a long sample into multiple examples: its suffix beyond the model-specific 8,192-token limit is discarded. We collect post-RoPE K/V states from the fitting split and accumulate the layerwise Gram statistics once per checkpoint. LLC and Sep obtain their PCA bases from the joint statistics, while VECTOR obtains its K-to-V OLS map from the same calibration pass. The resulting maps are fixed for evaluation and reused at longer context lengths without additional statistics collection or refitting.

B Efficient Code-Space Attention

This appendix derives the code-space computation used by the implementation evaluated in Section 5.3. The derivation is algebraically equivalent to attention over reconstructed K/V states but avoids materializing those states in memory. We then describe the prepared Triton execution path, its cost, and the complete latency measurements underlying Figure 3.

B.1 The joint two-tier softmax

At decode time, each query head attends over both tiers *jointly*: the n_{e} exact tokens of $\mathcal{E}$ and the n_{c} codes of $\mathcal{C}$, under a single softmax. Let H_{q} be the number of query heads and let $g(h) \in \{1, \dots, H\}$ map query head h to its KV head (the identity map for multi-head attention and the usual group map for GQA). Writing $W_{\mathrm{dec}} = \begin{bmatrix} A_K \\ A_V \end{bmatrix}$ with $A_K, A_V \in \mathbb{R}^{HD \times r}$, denote the $D \times r$ row blocks for KV head g by $A_K^{(g)}$ and $A_V^{(g)}$. A code z_i reconstructs $\hat{k}_{g,i} = A_K^{(g)} z_i$ and $\hat{v}_{g,i} = A_V^{(g)} z_i$. Absorbing the usual $1/\sqrt{D}$ scale into q_h, the output for query head h is

$$o_h \;=\; \frac{\sum_{j=1}^{n_{\mathrm{e}}} e^{s_{h,j}}\, v_{g(h),j} \;+\; \sum_{i=1}^{n_{\mathrm{c}}} e^{s'_{h,i}}\, A_V^{(g(h))} z_i}{\sum_{j=1}^{n_{\mathrm{e}}} e^{s_{h,j}} \;+\; \sum_{i=1}^{n_{\mathrm{c}}} e^{s'_{h,i}}}, \qquad s_{h,j} = q_h^\top k_{g(h),j}, \quad s'_{h,i} = q_h^\top A_K^{(g(h))} z_i. \tag{B.1}$$

The codec's linearity lets us evaluate Equation (B.1) without ever materializing $\widehat{K}$ or $\widehat{V}$: decompression is *absorbed* into the attention computation, on both sides of the softmax.

Absorbing A_K into the query. The compressed score is linear in the relevant decoder block, so

$$s'_{h,i} \;=\; \tilde{q}_h^\top z_i, \qquad \tilde{q}_h := A_K^{(g(h))\top} q_h, \tag{B.2}$$

which forms one r-dimensional projected query per query head at $O(H_{\mathrm{q}} r D)$ cost per decode step, independent of both cache lengths, and reuses it for every code.

Absorbing A_V into the output. Each tier is scanned once, online-softmax style, carrying its running maximum, its normalizer mass, and an *unnormalized* output accumulator:

$$\begin{aligned} m_{e,h} &= \max_j s_{h,j}, \quad \ell_{e,h} = \textstyle\sum_j e^{s_{h,j}-m_{e,h}}, \quad \tilde{o}_{e,h} = \textstyle\sum_j e^{s_{h,j}-m_{e,h}}\, v_{g(h),j} \in \mathbb{R}^D, \\ m_{c,h} &= \max_i s'_{h,i}, \quad \ell_{c,h} = \textstyle\sum_i e^{s'_{h,i}-m_{c,h}}, \qquad u_h = \textstyle\sum_i e^{s'_{h,i}-m_{c,h}}\, z_i \in \mathbb{R}^r. \end{aligned} \tag{B.3}$$

The compressed tier's accumulator u_h stays in *code space*: A_V does not depend on i, so it factors out of the weighted sum and is deferred until after the scan. Consequently, r floats are carried instead of D, and n_{c} never meets A_V. The two scans then merge *exactly*: with $m_h^\star = \max(m_{e,h}, m_{c,h})$, $\alpha_h = e^{m_{e,h}-m_h^\star}$, and $\beta_h = e^{m_{c,h}-m_h^\star}$,

$$o_h \;=\; \frac{\alpha_h \tilde{o}_{e,h} + \beta_h A_V^{(g(h))} u_h}{\alpha_h \ell_{e,h} + \beta_h \ell_{c,h}}, \tag{B.4}$$

which recovers Equation (B.1) identically (multiply numerator and denominator by $e^{m_h^\star}$). This is the same merge rule that flash-decoding applies across sequence splits: the compressed tier is one more partial scan, heterogeneous in representation but not in bookkeeping, and the relevant block of A_V multiplies one r-vector per query head, *after* the merge. The exactness of Equation (B.4) is what lets Section 4 report attention over both tiers as a single softmax with no tiering approximation.

Per-head structure. Stacking all projected query heads gives $\widetilde{Q} \in \mathbb{R}^{H_{\mathrm{q}} \times r}$, and the only contractions that touch n_{c} are the two GEMMs

$$S' \;=\; \widetilde{Q}\, Z^\top \in \mathbb{R}^{H_{\mathrm{q}} \times n_{\mathrm{c}}}, \qquad U \;=\; \widetilde{P}\, Z \in \mathbb{R}^{H_{\mathrm{q}} \times r}, \tag{B.5}$$

where $\widetilde{P}$ holds the unnormalized weights $e^{s'_{h,i}-m_{c,h}}$. The compressed tier thus behaves as *multi-query attention with head width* r: all H_{q} query heads share the codes Z as both keys and values, stored once per layer. Under grouped-query attention, each query head reads the A_K, A_V block of the KV head assigned by g.

B.2 Cost: FLOPs and memory traffic

Persistent-state accounting. Main-text memory values count token-dependent cache state rather than fixed codec parameters. LLC stores one basis of $2HDr$ elements per layer; the encoder is its transpose and need not be stored separately. VECTOR stores an $(HD)^2$-element OLS map, while Sep's independent bases contain Dr elements in total. At the main models' HD=1024 in BF16, LLC's basis occupies 1 MiB for r=256 and 2 MiB for r=512, and VECTOR's map occupies 2 MiB. The prepared CUDA Graph reuses 1.28 MiB and 2.16 MiB of temporary workspace at ranks 256 and 512, respectively; these buffers do not grow with context length.

The exact tier's scan is ordinary decode attention. For the compressed tier the two accountings diverge, and we report both. *FLOPs*: the cache-length-dependent contractions of Equation (B.5) cost $4H_{\mathrm{q}} r n_{\mathrm{c}}$, plus $4H_{\mathrm{q}} Dr$ for forming the projected queries and applying the value-decoder blocks. Dense attention over the same n_{c} tokens costs $4H_{\mathrm{q}} D n_{\mathrm{c}}$. The leading scan ratio is therefore r/D, independent of either head count: $2\times$ *more* scan compute at r=256, D=128, neutral at D=256, and up to $4\times$ at r=512. *Bytes*: in an ideal single-read accounting, dense attention accesses $2n_{\mathrm{c}}HD$ K/V elements per decode step, versus $n_{\mathrm{c}} r$ code elements shared across heads (Equation (B.5)). At BF16 these are $4n_{\mathrm{c}}HD$ and $2n_{\mathrm{c}}r$ bytes, respectively. The ratio is $r/(2HD)$: $1/8$ at $r = 256$ and $1/4$ at $r = 512$ for $HD = 1024$. This exchange of arithmetic for a smaller cache payload can benefit

bandwidth-bound decoding, but measured traffic and latency also depend on reuse, intermediate buffers, and kernel scheduling. A related trade is deployed at scale by multi-head latent attention (DeepSeek-AI, 2024), whose absorbed form likewise contracts each head's scores against a shared latent wider than the head dimension (r_{MLA}=512 vs. D=128), accepting the identical r/D FLOPs factor for the bandwidth win. The joint decode-step total is

$$\underbrace{4H_{\mathrm{q}}n_{\mathrm{e}}D}_{\text{exact tier}} + \underbrace{4H_{\mathrm{q}}r(n_{\mathrm{c}}+D)}_{\text{compressed tier}} + O(H_{\mathrm{q}}D) \tag{B.6}$$

FLOPs over $2n_{\mathrm{e}}HD + n_{\mathrm{c}}r$ elements read, the $O(H_{\mathrm{q}}D)$ merge of Equation (B.4) being negligible. The same representation reduces persistent cache capacity and ideal per-step cache-read traffic by $r/(2HD)$, with actual execution costs assessed by the measurements below.

Table 8: Attending over the compressed cache with absorption (Equations (B.2) and (B.4)) vs. dense attention over an equal number of full-precision tokens, per layer per decode step, at this paper's aggressive compression point (n_{c}=6144, r=256; bf16; reads count K/V vs. codes only). HD is the KV cache width (HD) of each model. The FLOP column reports the leading scan ratio r/D and omits the cache-length-independent query/output transforms; the read ratio is $r/(2HD)$. Both are independent of query head count, so models sharing (D, HD) share a row regardless of size. Absorption trades a modest FLOPs increase for a large traffic reduction; decode attention is bandwidth-bound, so the ratios are FLOPs/bytes accounting only and imply no wall-clock claim. Measured latencies are in Table 9. †Hybrid architecture with full attention every fourth layer (8 of 32); values are for those layers, the only ones holding a KV cache and hence the only ones LLC applies to (calibration cost shrinks accordingly). Its partial RoPE (rotary on 25% of head dimensions) also limits how much position can contaminate the post-RoPE eigenbasis.

			FLOPs	Reads (MB)		
Model	D	HD	ratio	Absorbed	Dense	ratio
Qwen3-4B / Llama-3.1-8B	128	1024	2.0×	3.1	25.2	1/8
Gemma-3-4B / Qwen3.5-4B†	256	1024	1.0×	3.1	25.2	1/8

B.3 Prepared Triton Implementation

Algorithm 2 Code-space two-tier decode attention (one layer, one query; per head)

Require: query head q_h (post-RoPE); assigned KV head $g(h)$; exact cache $\mathcal{E} = (K, V)$, n_{e} tokens; codes $Z \in \mathbb{R}^{n_{\mathrm{c}} \times r}$ from $\mathcal{C}$; decoder blocks $A_K^{(g(h))}, A_V^{(g(h))}$

$\tilde{q}_h \leftarrow A_K^{(g(h))\top} q_h$ ▷ once per head and step; $O(rD)$

$(m_{e,h}, \ell_{e,h}, \tilde{o}_{e,h}) \leftarrow$ online-softmax scan of exact tiles ▷ Equation (B.3)

$(m_{c,h}, \ell_{c,h}, u_h) \leftarrow$ online-softmax scan of code tiles, scores $s'_{h,i} = \tilde{q}_h^\top z_i$ ▷ $u_h \in \mathbb{R}^r$

$m_h^\star \leftarrow \max(m_{e,h}, m_{c,h})$; $\alpha_h \leftarrow e^{m_{e,h} - m_h^\star}$; $\beta_h \leftarrow e^{m_{c,h} - m_h^\star}$

return $o_h = \left(\alpha_h \tilde{o}_{e,h} + \beta_h A_V^{(g(h))} u_h\right) / \left(\alpha_h \ell_{e,h} + \beta_h \ell_{c,h}\right)$ ▷ exact joint softmax

The implementation realizes Algorithm 2 as a prepared four-stage CUDA Graph rather than a monolithic kernel. It first projects the query into code space. The exact-cache scan then runs on one stream while the projected query feeds the compressed-cache scan on another. A joint online-softmax reduction waits for both branches, merges their stable partial statistics, and applies A_V once after the weighted-code reduction. Long caches are split across programs flash-decoding-style; the same log-sum-exp rule handles both split and tier boundaries. The selected graph schedule uses 12 exact splits and 54 compressed splits at both evaluated ranks.

This organization preserves the important dataflow properties of the derivation: reconstructed K/V states are never written to HBM, each compressed scan operates directly on Z, and value decoding is deferred until after the compressed splits have been reduced. Because the post-RoPE design places RoPE before A_K, the query-to-code projection is evaluated explicitly at each decoding step rather than folded into the model's query projection.

B.4 Benchmark Protocol and Additional Results

We benchmark one-token decoding on an NVIDIA A100-SXM4-80GB with batch size one, 32 query heads, 8 KV heads, and head width D=256. The exact cache holds 2,048 tokens and the compressed cache holds the remaining positions at each total length. Inputs, exact K/V, codes, and decoder matrices use BF16; normalized compressed partials use FP16, while projection and softmax statistics use FP32. The dense comparator is PyTorch scaled-dot-product attention with grouped-query attention enabled over full-length BF16 K/V.

This geometry has $HD = 2048$, compared with $HD = 1024$ for the four main accuracy models. At $r = 512$, codes occupy $1/8$ of full K/V payload per compressed token in the benchmark, versus $1/4$ in those models; at $r = 256$, the ratios are $1/16$ and $1/8$, respectively. The timing includes query projection, both attention scans, and output decoding and reduction over prepared cache states. Encoding demoted tokens, admission, cache updates, and the rest of the model are outside the timed path.

Both methods use preallocated buffers. We report ordinary eager execution and the prepared CUDA Graph path, under both hardware-cache reuse and a condition that reads a separate 96-MiB buffer before each timed invocation. Each rank is confirmed in five independent processes with 50 warmups and 500 paired repetitions per condition. Every run is checked against absorbed FP32 and staged Triton references with relative ℓ_2 error at most 2×10^{-3}.

Table 9: Per-layer one-token decode latency in μs. Each entry reports LLC/dense median latency from the independent confirmation runs. Graph uses the prepared parallel schedule; eager uses the selected serialized schedule. "Perturbed" reads a separate 96-MiB buffer before timing to reduce hardware-cache reuse.

		CUDA Graph		Eager	
r	Length	Reused	Perturbed	Reused	Perturbed
256	8k	44.0/72.8	41.5/69.1	141.9/119.8	119.8/70.8
	16k	44.6/106.5	48.7/112.2	144.1/147.4	118.1/110.4
	32k	57.4/177.0	60.1/193.1	138.1/190.3	110.8/195.3
	64k	83.4/352.1	85.8/351.4	146.9/360.3	146.9/350.4
512	8k	51.3/69.0	53.7/70.6	138.0/110.7	112.3/68.9
	16k	62.5/104.9	64.5/111.4	140.8/141.6	114.3/109.1
	32k	81.9/177.5	83.3/191.8	139.3/181.0	136.9/193.1
	64k	120.9/334.9	122.8/346.1	222.7/354.9	223.3/349.8

The prepared graph path is faster than dense at every measured length for both ranks, and the gap grows with context length. Eager LLC is slower at 8k, mixed at 16k, and faster from 32k onward. The reused and perturbed conditions show the same qualitative trend. The main figure reports the perturbed graph condition; Table 9 exposes the corresponding absolute latencies and the other measured execution modes.

C Additional Codec Analyses

C.1 Joint K/V and All-Head Design

Section 3.1 compares the four combinations of sharing a basis across K/V and across heads. This appendix gives their block construction, per-model reconstruction results, decode cost, and a representative end-task comparison. All variants use the same calibration statistics and store the same total number of code elements per token. Sep is the fully factorized, prior-style reference motivated by per-head projection methods such as EigenAttention and OjaKV (Saxena et al., 2024; Zhu et al., 2026); it is not a reproduction of either full method.

The four variants. Each variant fits its basis by eigendecomposition of the appropriate sub-block(s) of the same joint $2HD \times 2HD$ second-moment matrix C (Equation (2)), accumulated in a single

calibration pass; they differ only in *which* covariance structure the projection is permitted to see (Table 10). Because every block uses orthonormal eigenvectors, all four remain globally orthonormal and reuse LLC's exact two-tier machinery and residual shortcut (Section 3.2) without modification.

Table 10: The 2×2 concatenation factorial. All variants are PCA codecs (top eigenvectors of a covariance block) sharing one total code dimension r; they differ only in whether keys/values and heads are pooled before projection. "Blocks" counts the independent eigendecompositions and each block's width. The last row is LLC; the first row is the prior-style separate per-head reference.

Variant	KV	Heads	Basis blocks (per layer)
Sep	separate	separate	$2H$ blocks, $D \times D$
KV-Joint	joint	separate	H blocks, $2D \times 2D$
Head-Joint	separate	concat	2 blocks, $HD \times HD$
LLC	joint	concat	1 block, $2HD \times 2HD$

Rank accounting: iso-memory is iso-r. Per-token cache cost is the number of stored code floats; the decoder matrices are a one-time fixed cost (like model weights), excluded from per-token memory, following the same convention used for VECTOR vs LLC in the main text. Fixing the *total* code dimension r therefore makes all four variants iso-memory. The decomposed variants must *partition* that budget a priori: each block is truncated to an equal share ($r/2$ per signal for separate-KV, r/H per head, and $r/(2H)$ per (head, signal) for the fully separate corner), which requires only r divisible by $2H$. LLC places no such partition on r: its single eigendecomposition spends the whole budget on the top-r directions of the joint covariance, wherever the shared-source structure concentrates energy. Fixing r gives the same per-token code payload across models, but the fraction of full K/V storage depends on $2HD$. This width is 2048 for five models and 4096 for Gemma2-9B, whose compression factor is therefore twice as large at the same rank. We use the default even split ($r_K = r_V$, uniform per head) throughout.

C.1.1 RECONSTRUCTION FIDELITY

The ablation's claim is about *codec quality*, so the primary evaluation is reconstruction R^2 of the true K/V pair at rank r, the direct, policy-free measure (cf. Table 14). This isolates the codec from the two-tier structure entirely: it is what each projection can recover before any token is routed or evicted, and thus reads off the main effect of each concatenation axis and their interaction with no eviction confound. Table 11 reports this on RULER-8K and C4, post-RoPE.

Table 11: **Primary.** Reconstruction R^2 of the K/V pair (layer-averaged, higher is better, $\leq$1.0) for the four concatenation corners at r=256 and r=512, post-RoPE, on RULER-8K and C4, across six models spanning both head counts ($H \in \{4, 8\}$) and head widths ($D \in \{256, 128\}$). Columns are ordered by increasing concatenation: **Sep** uses separate K/V bases per head, **KV-Joint** joins K/V within each head, **Head-Joint** joins heads separately for K and V, and LLC joins both axes. LLC is highest in every column, and the order Sep < KV-Joint < Head-Joint < LLC holds for all six models on both distributions and at both ranks.

		RULER-8K				C4			
Model	(H, D)	Sep	KV-Joint	Head-Joint	LLC	Sep	KV-Joint	Head-Joint	LLC
Code dimension $r = 256$									
Qwen3-4B-Instruct	$(8, 128)$	0.667	0.747	0.856	**0.893**	0.686	0.757	0.860	**0.898**
Llama-3.1-8B	$(8, 128)$	0.667	0.768	0.893	**0.928**	0.655	0.759	0.878	**0.916**
Gemma-3-4B-it	$(4, 256)$	0.754	0.871	0.882	**0.926**	0.817	0.892	0.902	**0.936**
Qwen3.5-4B†	$(4, 256)$	0.582	0.679	0.789	**0.861**	0.622	0.701	0.798	**0.865**
Qwen3.5-9B†	$(4, 256)$	0.580	0.669	0.777	**0.846**	0.615	0.689	0.783	**0.846**
Gemma2-9B-it‡	$(8, 256)$	0.455	0.502	0.612	**0.639**	0.451	0.503	0.623	**0.656**
Code dimension $r = 512$									
Qwen3-4B-Instruct	$(8, 128)$	0.780	0.850	0.914	**0.943**	0.788	0.856	0.918	**0.948**
Llama-3.1-8B	$(8, 128)$	0.774	0.889	0.934	**0.966**	0.766	0.882	0.924	**0.960**
Gemma-3-4B-it	$(4, 256)$	0.889	0.935	0.944	**0.967**	0.909	0.943	0.955	**0.972**
Qwen3.5-4B†	$(4, 256)$	0.729	0.814	0.896	**0.942**	0.744	0.821	0.900	**0.944**
Qwen3.5-9B†	$(4, 256)$	0.723	0.803	0.883	**0.932**	0.734	0.809	0.886	**0.932**
Gemma2-9B-it‡	$(8, 256)$	0.559	0.604	0.714	**0.731**	0.552	0.602	0.723	**0.743**

Columns are the four factorial corners in increasing concatenation: Sep (separate K/V per head), KV-Joint, Head-Joint, and LLC. LLC is highest in every column at both ranks, and the monotone order Sep < KV-Joint < Head-Joint < LLC holds without exception across all six models, both distributions, and both code dimensions. At r=512, every entry rises and the spread compresses, as expected when the budget is less scarce. Each concatenation axis increases R^2 on its own, and both together (LLC) is best. This central design claim is now robust across two head-count regimes ($H \in \{4, 8\}$), two head widths ($D \in \{128, 256\}$), and two model families' sizes. †Qwen3.5 are hybrid (GatedDeltaNet + full attention) with partial rotary; LLC applies to their full-attention layers, and the same ordering holds there. ‡Gemma2-9B reconstructs lower in *absolute* terms for two compounding, codec-independent reasons: its concatenated K/V representation is twice as wide ($2HD$=4096), so a fixed r=256 is a 16× compression versus 8× for the $2HD$=2048 models; and its K/V distribution is intrinsically higher-rank (needing 35% of its dimensions for 90% variance, versus 7–12% for the others). At iso-compression (r=512, i.e. 8×) it recovers to 0.731, and the factorial ordering is unchanged throughout; the absolute level is a property of the model's K/V distribution, not of any codec.

Reconstruction–compute trade-off. Head-Joint retains concatenation across heads while separating K and V. It comes within 0.02–0.07 R^2 of LLC at half its leading compressed-scan FLOPs (Section C.1.2). At $r = 256$ and $D = 128$, its leading scan FLOPs equal those of dense attention. The measured reconstruction gap varies across models; head count is not varied independently of model architecture in this comparison. LLC achieves the highest reconstruction R^2, while Head-Joint offers a lower arithmetic cost. Their latency trade-off requires matched kernel measurements.

C.1.2 Decode FLOPs

The four variants are iso-*memory* but not iso-*FLOP*, and the two concatenation choices that buy LLC's reconstruction quality are also the choices that raise its compressed-tier scan cost. In an ideal single-read accounting, all four variants access $n_c r$ code elements, or $2n_c r$ bytes in BF16, per step. Their code-payload ratio to full K/V is $r/(2HD)$ (Table 8); actual traffic also depends on the kernel's reuse and intermediate buffers. Their leading scan arithmetic differs: the compressed-tier scan (Equation (B.5)) costs $4H_q r n_c$ FLOPs for LLC, where H_q is the query-head count, and each

concatenation axis moves this independently. Joining keys with values makes the *full* width-r code participate in both the score and the output contraction; splitting them lets z_K serve only the score and z_V only the output, halving the work. Concatenating heads makes every one of the H heads contract against the shared width-r code; a per-head code lets head h touch only its own r/H slice. Higher reconstruction fidelity thus accompanies higher leading scan FLOPs (Table 12): Sep has the lowest scan cost and reconstruction R^2, and LLC has the highest of both.

Table 12: Compressed-tier decode FLOPs per variant, relative to LLC and to dense attention over the same n_c tokens, at r=256. All four store the same code payload. Join-KV contributes a factor 2 and head-concat a factor H; the vs-dense factor is LLC's r/D (Section B.2) scaled down accordingly. Columns use $(H, D) = (8, 128)$ for Qwen3/Llama and $(4, 256)$ for Gemma-3. These leading scan counts exclude query/output transforms and do not establish latency equivalence.

	FLOP factor		vs. dense (FLOPs)	
Codec variant	vs. LLC	structural	Qwen3/Llama	Gemma-3
Sep	$1/(2H)$	$\text{join}^{-1}\text{head}^{-1}$	0.125×	0.125×
KV-Joint	$1/H$	head^{-1}	0.25×	0.25×
Head-Joint	1/2	join^{-1}	1.0×	0.5×
LLC	1	n/a	2.0×	1.0×

The ablation exposes a reconstruction–compute trade-off at fixed code payload. If execution is bandwidth-bound and code reuse is effective, additional arithmetic may have a limited latency cost. In a compute-bound regime, the lower scan FLOPs of factorized codecs may matter more. Equal payload alone does not establish equal measured traffic or latency across the variants. The measurements in Section B.4 compare LLC with dense attention, rather than benchmarking all four factorial codecs.

C.1.3 End-Task Accuracy

To test how reconstruction fidelity relates to downstream accuracy, we evaluate all four variants on RULER-8K under a single eviction policy (ValueDiff, the policy common to every model) at the highest codec-engagement operating point (B_e=2048, B_c=6144, filling the 8192-token context). The combined capacity can cover all prompt positions at prefill, limiting hard eviction at that stage. During generation, new tokens can exceed this capacity, and codec-dependent hidden states can change subsequent exact and compressed selections. All variants use the same budgets, $r = 256$, BF16, and calibration corpus; the experiment measures the end-task effect of changing codec structure within this lifecycle.

Table 13: **Secondary.** RULER-8K mean accuracy for the four concatenation variants, post-RoPE, under the ValueDiff eviction policy, at B_e=2048, B_c=6144, $r = 256$, context 8192, 50 samples/task. The last row is LLC.

Codec variant	Qwen3-4B	Llama-3.1-8B	Gemma-3-4B
Sep	90.75	89.76	90.17
KV-Joint	92.25	89.81	90.19
Head-Joint	94.52	89.86	90.74
LLC	**94.86**	**90.93**	**90.79**

A caveat from the broader grid. The single operating point above is representative, not universal. Sweeping the full B_e/B_c budget grid at both ranks across all six models of Table 11 surfaces two exceptions to LLC's lead, both at small margins. On Llama-3.1-8B at r=512, Head-Joint wins outright in 7/9 budget pairs (e.g. 93.74 vs. LLC's 93.63 at B_e=2048, B_c=6144); there, the cheaper join-KV concatenation costs LLC almost nothing to drop. On Gemma2-9B-it, Sep is nominally best in roughly half the grid at both ranks (e.g. 86.64 vs. LLC's 85.47 at B_e=2048, B_c=2048, r=512), which sits in tension with Table 11's clear R^2 ordering for that model, since fidelity and end-task accuracy do not always agree at this granularity. We flag rather than resolve this: end-task accuracy is a noisier signal than reconstruction R^2, and margins throughout are small (single-digit points),

so neither exception overturns the paper's central claim, but LLC is not a strict per-cell Pareto win everywhere in the grid.

Calibration cost. All four variants of a model are derived from a *single* calibration pass: one forward pass accumulates the joint Gram C, and each variant is obtained by re-slicing C into the relevant sub-blocks and running the corresponding eigendecomposition. No additional forward passes or calibration data are required beyond LLC's own.

C.2 Pre- vs. Post-RoPE Compression

C.2.1 Reconstruction Fidelity

Table 14 compares reconstruction in the two RoPE spaces. This is a diagnostic ablation: all main-text codec evaluations use the cached post-RoPE representation. LLC remains stable across datasets and RoPE spaces, whereas VECTOR is substantially more sensitive to this choice.

Table 14: Reconstruction R^2 of the K/V pair (layer-averaged) for VECTOR and LLC ($r = 256$; higher is better, ≤ 1.0), each calibrated in pre- and post-RoPE space on a single calibration datamix and evaluated on two datasets.

	RULER-8K				C4			
	VECTOR		LLC		VECTOR		LLC	
Model	pre	post	pre	post	pre	post	pre	post
Qwen3-4B-Instruct	0.76	0.56	**0.94**	**0.89**	0.76	0.57	**0.93**	**0.89**
Llama-3.1-8B	−1.67	0.33	**0.98**	**0.93**	0.62	−1.08	**0.96**	**0.91**

C.2.2 End-Task Accuracy on RULER-8K

Table 15 isolates the effect of the RoPE space a codec is calibrated in, holding everything else fixed. Each codec is fit twice from a single forward pass: once using K/V states before applying RoPE and once using K/V states after applying RoPE. Both are evaluated on RULER-8K under both the ValueDiff and KeyDiff eviction policies. Every cell shares the same budget (B_e=2048, B_c=2048, r=256 for LLC, bf16, context 8192), so Δ = post − pre is a clean A/B on RoPE space alone; memory differs between codecs (LLC stores a rank-256 code, VECTOR stores the full-width post-RoPE key) but is identical within each pre/post pair.

At this budget, VECTOR's task-accuracy changes lie in $\Delta \in [-2.00, +1.64]$, improving under KeyDiff and decreasing under ValueDiff on both models. LLC's changes lie in $\Delta \in [-0.99, +1.73]$, including a decrease for Llama under KeyDiff. These task-level changes need not follow the reconstruction R^2 changes in Table 14. This comparison measures the combined effect within the two-tier cache; it does not isolate a causal explanation for the gap between reconstruction fidelity and task accuracy.

Table 15: RULER-8K mean accuracy for each codec calibrated in pre- vs post-RoPE space, both eviction policies. Δ = post − pre. Shared budget B_e=B_c=2048, r=256, context 8192, 50 samples/task. VECTOR stores no low-rank latent (full-width post-RoPE key), so its rank is n/a. Reference floors (no codec): dense 95.85/95.94; ValueDiff eviction at B_e=2048 91.58/82.18; KeyDiff eviction at B_e=2048 85.79/85.86 (Qwen3-4B/Llama-3.1-8B).

Eviction policy π	Model	Codec	Pre-RoPE	Post-RoPE	Δ
ValueDiff	Qwen3-4B-Instruct	LLC	93.99	94.30	+0.31
		VECTOR	94.35	94.04	−0.31
	Llama-3.1-8B	LLC	82.15	83.88	+1.73
		VECTOR	87.73	85.73	−2.00
KeyDiff	Qwen3-4B-Instruct	LLC	90.89	91.10	+0.21
		VECTOR	89.43	91.07	+1.64
	Llama-3.1-8B	LLC	89.06	88.07	−0.99
		VECTOR	89.35	89.59	+0.24

C.2.3 Generality Across Model Families

Table 15 isolates RoPE space on two models at one budget. Table 16 widens the check to LLC's post-RoPE penalty specifically, across six model instances spanning three architecture families: standard full rotary (Qwen3, Llama), full rotary with sliding-window attention (Gemma2, Gemma3), and partial rotary (25%) combined with GatedDeltaNet linear attention (Qwen3.5). Δ = post − pre at the two *engaged* budgets of the main RULER sweep (2048+6144 and 4096+4096), under the ValueDiff eviction policy. These larger retained-token budgets complement the 2048+2048 comparison above.

Table 16: LLC post-RoPE minus pre-RoPE RULER-8K accuracy under the ValueDiff eviction policy, at the two engaged budgets, across six model instances. †Qwen3.5-9B reports both code dimensions (r=256 / r=512); all other rows are r=256.

Model	Δ @ 2048+6144	Δ @ 4096+4096
Qwen3-4B-Instruct	−0.12	−0.18
Gemma2-9B-it	+0.83	+0.95
Gemma-3-4B-it	+0.10	+0.01
Qwen3.5-4B	−0.05	+0.01
Qwen3.5-9B†	−0.05 / +0.03	−0.36 / −0.31
Llama-3.1-8B	**−3.23**	**−1.76**

Five of the six model instances remain within one point of their pre-RoPE accuracy at both engaged budgets ($\Delta \in [-0.36, +0.95]$). Llama-3.1-8B is the exception, with decreases of 3.23 and 1.76 points. The post-RoPE effect is therefore model- and budget-dependent in this sweep. These RULER-8K differences are smaller than the large long-context degradations reported for the post-RoPE VECTOR codec in Sections D.2 and D.3.

D Complete RULER Results

D.1 RULER-8K: Full Eviction-Policy Grid

Tables 17 to 22 report the complete available RULER-8K grid across all four eviction policies (KeyDiff, ValueDiff, StreamingLLM, SnapKV) and both codecs (LLC at $r \in \{256, 512\}$, VECTOR), for all six evaluated models. The main figure shows three representative models; these tables provide the LLC and VECTOR cells behind that figure and the policy comparison in Section D.4.

All six models include the full KeyDiff/ValueDiff and StreamingLLM/SnapKV sweeps. Where a separate no-eviction run is unavailable, the full-context B_e=8192, B_c=0 result is labeled as a dense-equivalent reference rather than a true dense measurement.

Table 17: Qwen3-4B-Instruct, RULER-8K, post-RoPE. Dense score: 95.85.

Eviction policy π	Codec	B_e	B_c	r	RULER-8K
Dense					95.85
KD	–	1024	0	–	82.91
KD	LLC	1024	2048	256	88.10
KD	LLC	1024	2048	512	91.11
KD	VEC	1024	2048	–	87.87
KD	LLC	1024	4096	256	90.42
KD	LLC	1024	4096	512	94.06
KD	VEC	1024	4096	–	90.98
KD	LLC	1024	6144	256	91.23
KD	LLC	1024	6144	512	94.11
KD	VEC	1024	6144	–	91.23
KD	LLC	2048	2048	256	91.10
KD	LLC	2048	2048	512	93.37
KD	VEC	2048	2048	–	91.07
KD	LLC	2048	4096	256	92.72
KD	LLC	2048	4096	512	94.94
KD	VEC	2048	4096	–	92.44
KD	LLC	2048	6144	256	92.26
KD	LLC	2048	6144	512	94.55
KD	VEC	2048	6144	–	92.72
KD	LLC	4096	2048	256	93.56
KD	LLC	4096	2048	512	94.42
KD	VEC	4096	2048	–	93.32
KD	LLC	4096	4096	256	94.36
KD	LLC	4096	4096	512	95.22
KD	VEC	4096	4096	–	93.91
KD	LLC	6144	2048	256	95.52
KD	LLC	6144	2048	512	95.63
KD	VEC	6144	2048	–	95.46
KD	–	8192	0	–	95.68
VD	–	1024	0	–	88.00
VD	LLC	1024	2048	256	91.55
VD	LLC	1024	2048	512	93.17
VD	VEC	1024	2048	–	92.43
VD	LLC	1024	4096	256	93.27
VD	LLC	1024	4096	512	94.73
VD	VEC	1024	4096	–	93.05
VD	LLC	1024	6144	256	93.42
VD	LLC	1024	6144	512	95.35
VD	VEC	1024	6144	–	93.29
VD	–	2048	0	–	91.58
VD	LLC	2048	2048	256	94.30
VD	LLC	2048	2048	512	94.64
VD	VEC	2048	2048	–	94.04
VD	LLC	2048	4096	256	94.55
VD	LLC	2048	4096	512	95.18
VD	VEC	2048	4096	–	94.50
VD	LLC	2048	6144	256	94.86
VD	LLC	2048	6144	512	95.86
VD	VEC	2048	6144	–	94.93
VD	–	4096	0	–	94.70
VD	LLC	4096	2048	256	95.09
VD	LLC	4096	2048	512	95.26
VD	VEC	4096	2048	–	94.97
VD	LLC	4096	4096	256	95.48
VD	LLC	4096	4096	512	95.93
VD	VEC	4096	4096	–	95.82
VD	LLC	6144	2048	256	95.71

continued on next page

Eviction policy π	Codec	B_e	B_c	r	RULER-8K
VD	LLC	6144	2048	512	95.72
VD	VEC	6144	2048	–	95.74
VD	–	8192	0	–	95.68
StreamingLLM	–	2048	0	–	43.17
StreamingLLM	LLC	2048	2048	256	38.98
StreamingLLM	LLC	2048	2048	512	40.65
StreamingLLM	VEC	2048	2048	–	43.03
StreamingLLM	LLC	2048	4096	256	42.44
StreamingLLM	LLC	2048	4096	512	50.83
StreamingLLM	VEC	2048	4096	–	43.23
StreamingLLM	LLC	2048	6144	256	90.46
StreamingLLM	LLC	2048	6144	512	93.79
StreamingLLM	VEC	2048	6144	–	86.83
StreamingLLM	–	4096	0	–	61.51
StreamingLLM	LLC	4096	2048	256	57.22
StreamingLLM	LLC	4096	2048	512	61.63
StreamingLLM	VEC	4096	2048	–	60.95
StreamingLLM	LLC	4096	4096	256	92.30
StreamingLLM	LLC	4096	4096	512	94.14
StreamingLLM	VEC	4096	4096	–	91.79
StreamingLLM	LLC	6144	2048	256	94.42
StreamingLLM	LLC	6144	2048	512	95.26
StreamingLLM	VEC	6144	2048	–	95.41
StreamingLLM	–	8192	0	–	95.68
SnapKV	–	2048	0	–	74.70
SnapKV	LLC	2048	2048	256	77.43
SnapKV	LLC	2048	2048	512	78.86
SnapKV	VEC	2048	2048	–	77.99
SnapKV	LLC	2048	4096	256	77.89
SnapKV	LLC	2048	4096	512	80.85
SnapKV	VEC	2048	4096	–	79.52
SnapKV	LLC	2048	6144	256	92.76
SnapKV	LLC	2048	6144	512	94.69
SnapKV	VEC	2048	6144	–	94.50
SnapKV	–	4096	0	–	85.79
SnapKV	LLC	4096	2048	256	87.89
SnapKV	LLC	4096	2048	512	87.77
SnapKV	VEC	4096	2048	–	87.22
SnapKV	LLC	4096	4096	256	95.02
SnapKV	LLC	4096	4096	512	95.77
SnapKV	VEC	4096	4096	–	95.47
SnapKV	LLC	6144	2048	256	95.65
SnapKV	LLC	6144	2048	512	95.92
SnapKV	VEC	6144	2048	–	95.61
SnapKV	–	8192	0	–	95.68

Table 18: Llama-3.1-8B, RULER-8K, post-RoPE. Dense score: 95.94; the separately measured KeyDiff full-context exact result is 96.01.

Eviction policy π	Codec	B_e	B_c	r	RULER-8K
Dense					95.94
KD	–	1024	0	–	81.95
KD	LLC	1024	2048	256	84.51
KD	LLC	1024	2048	512	84.20
KD	VEC	1024	2048	–	84.79
KD	LLC	1024	4096	256	86.39
KD	LLC	1024	4096	512	91.44

continued on next page

Eviction policy π	Codec	B_{e}	B_{c}	r	RULER-8K
KD	VEC	1024	4096	–	87.37
KD	LLC	1024	6144	256	85.50
KD	LLC	1024	6144	512	92.41
KD	VEC	1024	6144	–	88.95
KD	LLC	2048	2048	256	88.07
KD	LLC	2048	2048	512	88.68
KD	VEC	2048	2048	–	89.59
KD	LLC	2048	4096	256	89.49
KD	LLC	2048	4096	512	94.06
KD	VEC	2048	4096	–	90.50
KD	LLC	2048	6144	256	92.84
KD	LLC	2048	6144	512	94.57
KD	VEC	2048	6144	–	95.15
KD	LLC	4096	2048	256	91.89
KD	LLC	4096	2048	512	91.86
KD	VEC	4096	2048	–	91.09
KD	LLC	4096	4096	256	95.27
KD	LLC	4096	4096	512	95.34
KD	VEC	4096	4096	–	95.15
KD	LLC	6144	2048	256	95.27
KD	LLC	6144	2048	512	95.73
KD	VEC	6144	2048	–	95.78
KD	–	8192	0	–	96.01
VD	–	1024	0	–	77.42
VD	LLC	1024	2048	256	80.81
VD	LLC	1024	2048	512	81.58
VD	VEC	1024	2048	–	85.80
VD	LLC	1024	4096	256	82.16
VD	LLC	1024	4096	512	91.17
VD	VEC	1024	4096	–	84.53
VD	LLC	1024	6144	256	80.61
VD	LLC	1024	6144	512	92.04
VD	VEC	1024	6144	–	83.63
VD	–	2048	0	–	82.18
VD	LLC	2048	2048	256	83.88
VD	LLC	2048	2048	512	87.02
VD	VEC	2048	2048	–	85.73
VD	LLC	2048	4096	256	84.81
VD	LLC	2048	4096	512	94.21
VD	VEC	2048	4096	–	84.71
VD	LLC	2048	6144	256	90.93
VD	LLC	2048	6144	512	93.63
VD	VEC	2048	6144	–	85.32
VD	–	4096	0	–	84.91
VD	LLC	4096	2048	256	87.88
VD	LLC	4096	2048	512	88.03
VD	VEC	4096	2048	–	87.57
VD	LLC	4096	4096	256	93.36
VD	LLC	4096	4096	512	94.69
VD	VEC	4096	4096	–	90.79
VD	LLC	6144	2048	256	95.22
VD	LLC	6144	2048	512	95.27
VD	VEC	6144	2048	–	93.48
VD	–	8192	0	–	95.94
StreamingLLM	–	2048	0	–	34.13
StreamingLLM	LLC	2048	2048	256	34.04
StreamingLLM	LLC	2048	2048	512	35.12
StreamingLLM	VEC	2048	2048	–	41.46
StreamingLLM	LLC	2048	4096	256	39.85
StreamingLLM	LLC	2048	4096	512	42.96
StreamingLLM	VEC	2048	4096	–	43.04

continued on next page

Eviction policy π	Codec	B_e	B_c	r	RULER-8K
StreamingLLM	LLC	2048	6144	256	41.39
StreamingLLM	LLC	2048	6144	512	56.72
StreamingLLM	VEC	2048	6144	–	50.51
StreamingLLM	LLC	4096	2048	256	59.90
StreamingLLM	LLC	4096	2048	512	61.42
StreamingLLM	VEC	4096	2048	–	62.40
StreamingLLM	LLC	4096	4096	256	63.43
StreamingLLM	LLC	4096	4096	512	76.90
StreamingLLM	VEC	4096	4096	–	70.04
StreamingLLM	LLC	6144	2048	256	82.51
StreamingLLM	LLC	6144	2048	512	88.01
StreamingLLM	VEC	6144	2048	–	84.55
StreamingLLM	–	8192	0	–	95.94
SnapKV	–	2048	0	–	81.85
SnapKV	LLC	2048	2048	256	78.68
SnapKV	LLC	2048	2048	512	79.49
SnapKV	VEC	2048	2048	–	82.61
SnapKV	LLC	2048	4096	256	81.47
SnapKV	LLC	2048	4096	512	83.46
SnapKV	VEC	2048	4096	–	81.59
SnapKV	LLC	2048	6144	256	90.47
SnapKV	LLC	2048	6144	512	94.41
SnapKV	VEC	2048	6144	–	93.55
SnapKV	LLC	4096	2048	256	89.29
SnapKV	LLC	4096	2048	512	89.24
SnapKV	VEC	4096	2048	–	88.76
SnapKV	LLC	4096	4096	256	95.10
SnapKV	LLC	4096	4096	512	95.64
SnapKV	VEC	4096	4096	–	95.99
SnapKV	LLC	6144	2048	256	95.78
SnapKV	LLC	6144	2048	512	95.73
SnapKV	VEC	6144	2048	–	95.73
SnapKV	–	8192	0	–	95.94

Table 19: Gemma2-9B-it, RULER-8K, post-RoPE. Dense score: 86.40.

Eviction policy π	Codec	B_e	B_c	r	RULER-8K
Dense					86.40
KD	–	1024	0	–	84.66
KD	LLC	1024	2048	256	84.84
KD	LLC	1024	2048	512	85.67
KD	VEC	1024	2048	–	84.48
KD	LLC	1024	4096	256	85.96
KD	LLC	1024	4096	512	86.38
KD	VEC	1024	4096	–	85.35
KD	LLC	1024	6144	256	85.86
KD	LLC	1024	6144	512	86.25
KD	VEC	1024	6144	–	86.62
KD	LLC	2048	2048	256	86.08
KD	LLC	2048	2048	512	85.97
KD	VEC	2048	2048	–	85.35
KD	LLC	2048	4096	256	86.72
KD	LLC	2048	4096	512	86.57
KD	VEC	2048	4096	–	85.99
KD	LLC	2048	6144	256	86.12
KD	LLC	2048	6144	512	86.57
KD	VEC	2048	6144	–	86.58
KD	LLC	4096	2048	256	86.15

continued on next page

Eviction policy π	Codec	B_{e}	B_{c}	r	RULER-8K
KD	LLC	4096	2048	512	86.82
KD	VEC	4096	2048	–	86.39
KD	LLC	4096	4096	256	86.12
KD	LLC	4096	4096	512	86.80
KD	VEC	4096	4096	–	86.15
KD	LLC	6144	2048	256	86.57
KD	LLC	6144	2048	512	86.46
KD	VEC	6144	2048	–	86.51
KD	–	8192	0	–	86.36
VD	–	1024	0	–	85.12
VD	LLC	1024	2048	256	84.49
VD	LLC	1024	2048	512	84.74
VD	VEC	1024	2048	–	84.60
VD	LLC	1024	4096	256	85.93
VD	LLC	1024	4096	512	86.65
VD	VEC	1024	4096	–	85.65
VD	LLC	1024	6144	256	86.74
VD	LLC	1024	6144	512	87.15
VD	VEC	1024	6144	–	82.73
VD	–	2048	0	–	85.97
VD	LLC	2048	2048	256	85.86
VD	LLC	2048	2048	512	85.47
VD	VEC	2048	2048	–	84.57
VD	LLC	2048	4096	256	85.59
VD	LLC	2048	4096	512	86.97
VD	VEC	2048	4096	–	84.86
VD	LLC	2048	6144	256	88.87
VD	LLC	2048	6144	512	88.85
VD	VEC	2048	6144	–	85.62
VD	–	4096	0	–	85.84
VD	LLC	4096	2048	256	87.08
VD	LLC	4096	2048	512	85.94
VD	VEC	4096	2048	–	85.22
VD	LLC	4096	4096	256	89.16
VD	LLC	4096	4096	512	88.89
VD	VEC	4096	4096	–	86.93
VD	LLC	6144	2048	256	88.61
VD	LLC	6144	2048	512	88.06
VD	VEC	6144	2048	–	86.84
VD	–	8192	0	–	86.38
StreamingLLM	–	2048	0	–	45.25
StreamingLLM	LLC	2048	2048	256	45.84
StreamingLLM	LLC	2048	2048	512	45.70
StreamingLLM	VEC	2048	2048	–	45.72
StreamingLLM	LLC	2048	4096	256	45.74
StreamingLLM	LLC	2048	4096	512	45.29
StreamingLLM	VEC	2048	4096	–	45.67
StreamingLLM	LLC	2048	6144	256	58.50
StreamingLLM	LLC	2048	6144	512	86.27
StreamingLLM	VEC	2048	6144	–	52.89
StreamingLLM	LLC	4096	2048	256	59.30
StreamingLLM	LLC	4096	2048	512	58.79
StreamingLLM	VEC	4096	2048	–	58.58
StreamingLLM	LLC	4096	4096	256	64.68
StreamingLLM	LLC	4096	4096	512	86.17
StreamingLLM	VEC	4096	4096	–	58.28
StreamingLLM	LLC	6144	2048	256	79.65
StreamingLLM	LLC	6144	2048	512	86.67
StreamingLLM	VEC	6144	2048	–	74.09
StreamingLLM	–	8192	0	–	86.36

continued on next page

Eviction policy π	Codec	B_e	B_c	r	RULER-8K
SnapKV	–	2048	0	–	81.10
SnapKV	LLC	2048	2048	256	83.11
SnapKV	LLC	2048	2048	512	82.95
SnapKV	VEC	2048	2048	–	82.55
SnapKV	LLC	2048	4096	256	83.88
SnapKV	LLC	2048	4096	512	83.96
SnapKV	VEC	2048	4096	–	83.22
SnapKV	LLC	2048	6144	256	85.63
SnapKV	LLC	2048	6144	512	87.77
SnapKV	VEC	2048	6144	–	85.81
SnapKV	LLC	4096	2048	256	86.85
SnapKV	LLC	4096	2048	512	86.48
SnapKV	VEC	4096	2048	–	86.38
SnapKV	LLC	4096	4096	256	86.73
SnapKV	LLC	4096	4096	512	87.19
SnapKV	VEC	4096	4096	–	87.23
SnapKV	LLC	6144	2048	256	86.48
SnapKV	LLC	6144	2048	512	86.83
SnapKV	VEC	6144	2048	–	86.26
SnapKV	–	8192	0	–	86.36

Table 20: Gemma-3-4B-it, RULER-8K, post-RoPE. Dense score: 91.64.

Eviction policy π	Codec	B_e	B_c	r	RULER-8K
Dense					91.64
KD	–	1024	0	–	72.68
KD	LLC	1024	2048	256	83.34
KD	LLC	1024	2048	512	86.08
KD	VEC	1024	2048	–	84.53
KD	LLC	1024	4096	256	87.61
KD	LLC	1024	4096	512	89.27
KD	VEC	1024	4096	–	85.84
KD	LLC	1024	6144	256	89.57
KD	LLC	1024	6144	512	90.99
KD	VEC	1024	6144	–	84.74
KD	LLC	2048	2048	256	85.52
KD	LLC	2048	2048	512	87.15
KD	VEC	2048	2048	–	86.04
KD	LLC	2048	4096	256	89.36
KD	LLC	2048	4096	512	91.22
KD	VEC	2048	4096	–	87.72
KD	LLC	2048	6144	256	91.30
KD	LLC	2048	6144	512	91.93
KD	VEC	2048	6144	–	82.89
KD	LLC	4096	2048	256	89.80
KD	LLC	4096	2048	512	90.85
KD	VEC	4096	2048	–	88.98
KD	LLC	4096	4096	256	91.74
KD	LLC	4096	4096	512	91.68
KD	VEC	4096	4096	–	87.97
KD	LLC	6144	2048	256	91.07
KD	LLC	6144	2048	512	91.80
KD	VEC	6144	2048	–	89.71
KD	–	8192	0	–	91.59
VD	–	1024	0	–	77.09
VD	LLC	1024	2048	256	86.25
VD	LLC	1024	2048	512	87.17
VD	VEC	1024	2048	–	85.29

continued on next page

Eviction policy π	Codec	B_{e}	B_{c}	r	RULER-8K
VD	LLC	1024	4096	256	87.71
VD	LLC	1024	4096	512	90.14
VD	VEC	1024	4096	–	87.23
VD	LLC	1024	6144	256	89.42
VD	LLC	1024	6144	512	91.30
VD	VEC	1024	6144	–	87.71
VD	–	2048	0	–	83.68
VD	LLC	2048	2048	256	87.45
VD	LLC	2048	2048	512	87.16
VD	VEC	2048	2048	–	87.84
VD	LLC	2048	4096	256	89.83
VD	LLC	2048	4096	512	91.29
VD	VEC	2048	4096	–	89.84
VD	LLC	2048	6144	256	90.79
VD	LLC	2048	6144	512	91.43
VD	VEC	2048	6144	–	81.20
VD	–	4096	0	–	87.79
VD	LLC	4096	2048	256	90.73
VD	LLC	4096	2048	512	90.92
VD	VEC	4096	2048	–	90.74
VD	LLC	4096	4096	256	91.64
VD	LLC	4096	4096	512	91.78
VD	VEC	4096	4096	–	85.17
VD	LLC	6144	2048	256	91.35
VD	LLC	6144	2048	512	91.86
VD	VEC	6144	2048	–	87.80
VD	–	8192	0	–	91.60
StreamingLLM	–	2048	0	–	35.99
StreamingLLM	–	4096	0	–	59.92
StreamingLLM	LLC	2048	2048	256	45.02
StreamingLLM	LLC	2048	2048	512	50.60
StreamingLLM	VEC	2048	2048	–	40.35
StreamingLLM	LLC	2048	4096	256	57.32
StreamingLLM	LLC	2048	4096	512	67.39
StreamingLLM	VEC	2048	4096	–	47.01
StreamingLLM	LLC	2048	6144	256	80.46
StreamingLLM	LLC	2048	6144	512	90.31
StreamingLLM	VEC	2048	6144	–	54.58
StreamingLLM	LLC	4096	2048	256	63.28
StreamingLLM	LLC	4096	2048	512	68.71
StreamingLLM	VEC	4096	2048	–	59.28
StreamingLLM	LLC	4096	4096	256	87.07
StreamingLLM	LLC	4096	4096	512	91.02
StreamingLLM	VEC	4096	4096	–	71.31
StreamingLLM	LLC	6144	2048	256	89.76
StreamingLLM	LLC	6144	2048	512	91.72
StreamingLLM	VEC	6144	2048	–	83.73
StreamingLLM	–	8192	0	–	91.60
SnapKV	–	2048	0	–	66.11
SnapKV	LLC	2048	2048	256	75.02
SnapKV	LLC	2048	2048	512	74.62
SnapKV	VEC	2048	2048	–	72.89
SnapKV	LLC	2048	4096	256	77.34
SnapKV	LLC	2048	4096	512	79.54
SnapKV	VEC	2048	4096	–	78.50
SnapKV	LLC	2048	6144	256	91.66
SnapKV	LLC	2048	6144	512	91.41
SnapKV	VEC	2048	6144	–	86.12
SnapKV	–	4096	0	–	80.97
SnapKV	LLC	4096	2048	256	82.87
SnapKV	LLC	4096	2048	512	82.79

continued on next page

Eviction policy π	Codec	B_e	B_c	r	RULER-8K
SnapKV	VEC	4096	2048	–	81.83
SnapKV	LLC	4096	4096	256	91.62
SnapKV	LLC	4096	4096	512	91.54
SnapKV	VEC	4096	4096	–	89.46
SnapKV	LLC	6144	2048	256	91.60
SnapKV	LLC	6144	2048	512	92.05
SnapKV	VEC	6144	2048	–	90.69
SnapKV	–	8192	0	–	91.59

Table 21: Qwen3.5-4B, RULER-8K, post-RoPE. The full-context exact result (95.52 across all four policies) is used as a dense-equivalent reference.

Eviction policy π	Codec	B_e	B_c	r	RULER-8K
Dense (floor-equivalent)					95.52
KD	–	1024	0	–	84.97
KD	LLC	1024	2048	256	88.70
KD	LLC	1024	2048	512	91.78
KD	VEC	1024	2048	–	90.54
KD	LLC	1024	4096	256	90.54
KD	LLC	1024	4096	512	94.66
KD	VEC	1024	4096	–	93.67
KD	LLC	1024	6144	256	92.61
KD	LLC	1024	6144	512	94.74
KD	VEC	1024	6144	–	95.15
KD	LLC	2048	2048	256	90.89
KD	LLC	2048	2048	512	93.37
KD	VEC	2048	2048	–	92.50
KD	LLC	2048	4096	256	93.20
KD	LLC	2048	4096	512	94.67
KD	VEC	2048	4096	–	94.66
KD	LLC	2048	6144	256	93.91
KD	LLC	2048	6144	512	95.44
KD	VEC	2048	6144	–	95.44
KD	LLC	4096	2048	256	93.27
KD	LLC	4096	2048	512	94.94
KD	VEC	4096	2048	–	93.90
KD	LLC	4096	4096	256	94.77
KD	LLC	4096	4096	512	95.30
KD	VEC	4096	4096	–	95.16
KD	LLC	6144	2048	256	95.32
KD	LLC	6144	2048	512	95.63
KD	VEC	6144	2048	–	95.32
KD	–	8192	0	–	95.52
VD	–	1024	0	–	89.47
VD	LLC	1024	2048	256	91.75
VD	LLC	1024	2048	512	94.30
VD	VEC	1024	2048	–	92.63
VD	LLC	1024	4096	256	94.21
VD	LLC	1024	4096	512	94.96
VD	VEC	1024	4096	–	95.01
VD	LLC	1024	6144	256	94.53
VD	LLC	1024	6144	512	95.16
VD	VEC	1024	6144	–	95.16
VD	–	2048	0	–	92.36
VD	LLC	2048	2048	256	94.32
VD	LLC	2048	2048	512	95.13
VD	VEC	2048	2048	–	94.56
VD	LLC	2048	4096	256	94.60

continued on next page

Eviction policy π	Codec	B_{e}	B_{c}	r	RULER-8K
VD	LLC	2048	4096	512	95.30
VD	VEC	2048	4096	–	95.61
VD	LLC	2048	6144	256	95.28
VD	LLC	2048	6144	512	95.33
VD	VEC	2048	6144	–	95.47
VD	–	4096	0	–	94.58
VD	LLC	4096	2048	256	94.96
VD	LLC	4096	2048	512	95.41
VD	VEC	4096	2048	–	95.47
VD	LLC	4096	4096	256	95.33
VD	LLC	4096	4096	512	95.47
VD	VEC	4096	4096	–	95.32
VD	LLC	6144	2048	256	95.16
VD	LLC	6144	2048	512	95.68
VD	VEC	6144	2048	–	95.68
VD	–	8192	0	–	95.52
StreamingLLM	–	2048	0	–	41.07
StreamingLLM	–	4096	0	–	61.38
StreamingLLM	LLC	2048	2048	256	55.22
StreamingLLM	LLC	2048	2048	512	54.49
StreamingLLM	VEC	2048	2048	–	42.94
StreamingLLM	LLC	2048	4096	256	57.52
StreamingLLM	LLC	2048	4096	512	87.33
StreamingLLM	VEC	2048	4096	–	59.66
StreamingLLM	LLC	2048	6144	256	92.45
StreamingLLM	LLC	2048	6144	512	94.91
StreamingLLM	VEC	2048	6144	–	95.16
StreamingLLM	LLC	4096	2048	256	67.94
StreamingLLM	LLC	4096	2048	512	78.61
StreamingLLM	VEC	4096	2048	–	63.29
StreamingLLM	LLC	4096	4096	256	93.40
StreamingLLM	LLC	4096	4096	512	94.65
StreamingLLM	VEC	4096	4096	–	94.99
StreamingLLM	LLC	6144	2048	256	93.93
StreamingLLM	LLC	6144	2048	512	94.81
StreamingLLM	VEC	6144	2048	–	95.03
StreamingLLM	–	8192	0	–	95.52
SnapKV	–	2048	0	–	71.85
SnapKV	LLC	2048	2048	256	79.04
SnapKV	LLC	2048	2048	512	81.76
SnapKV	VEC	2048	2048	–	77.96
SnapKV	LLC	2048	4096	256	86.48
SnapKV	LLC	2048	4096	512	93.41
SnapKV	VEC	2048	4096	–	82.91
SnapKV	LLC	2048	6144	256	94.63
SnapKV	LLC	2048	6144	512	95.47
SnapKV	VEC	2048	6144	–	95.49
SnapKV	–	4096	0	–	80.88
SnapKV	LLC	4096	2048	256	90.02
SnapKV	LLC	4096	2048	512	93.69
SnapKV	VEC	4096	2048	–	85.85
SnapKV	LLC	4096	4096	256	95.45
SnapKV	LLC	4096	4096	512	95.15
SnapKV	VEC	4096	4096	–	95.23
SnapKV	LLC	6144	2048	256	95.83
SnapKV	LLC	6144	2048	512	95.47
SnapKV	VEC	6144	2048	–	95.68
SnapKV	–	8192	0	–	95.52

Table 22: Qwen3.5-9B, RULER-8K, post-RoPE. The full-context exact result is used as a dense-equivalent reference: 97.47 under ValueDiff and 97.27 under KeyDiff, StreamingLLM, and SnapKV.

Eviction policy π	Codec	B_e	B_c	r	RULER-8K
Dense (floor-equivalent)					97.47
KD	–	1024	0	–	84.29
KD	LLC	1024	2048	256	88.62
KD	LLC	1024	2048	512	92.95
KD	VEC	1024	2048	–	90.35
KD	LLC	1024	4096	256	90.76
KD	LLC	1024	4096	512	95.14
KD	VEC	1024	4096	–	93.74
KD	LLC	1024	6144	256	93.47
KD	LLC	1024	6144	512	96.20
KD	VEC	1024	6144	–	95.89
KD	LLC	2048	2048	256	92.18
KD	LLC	2048	2048	512	95.31
KD	VEC	2048	2048	–	94.04
KD	LLC	2048	4096	256	94.28
KD	LLC	2048	4096	512	96.23
KD	VEC	2048	4096	–	95.42
KD	LLC	2048	6144	256	95.02
KD	LLC	2048	6144	512	96.79
KD	VEC	2048	6144	–	97.39
KD	LLC	4096	2048	256	95.82
KD	LLC	4096	2048	512	96.74
KD	VEC	4096	2048	–	96.36
KD	LLC	4096	4096	256	96.28
KD	LLC	4096	4096	512	96.77
KD	VEC	4096	4096	–	96.91
KD	LLC	6144	2048	256	96.70
KD	LLC	6144	2048	512	96.96
KD	VEC	6144	2048	–	97.11
KD	–	8192	0	–	97.27
VD	–	1024	0	–	89.59
VD	LLC	1024	2048	256	92.70
VD	LLC	1024	2048	512	95.05
VD	VEC	1024	2048	–	94.67
VD	LLC	1024	4096	256	94.78
VD	LLC	1024	4096	512	96.13
VD	VEC	1024	4096	–	96.16
VD	LLC	1024	6144	256	95.57
VD	LLC	1024	6144	512	96.54
VD	VEC	1024	6144	–	96.74
VD	–	2048	0	–	93.23
VD	LLC	2048	2048	256	96.00
VD	LLC	2048	2048	512	96.19
VD	VEC	2048	2048	–	96.46
VD	LLC	2048	4096	256	95.92
VD	LLC	2048	4096	512	96.32
VD	VEC	2048	4096	–	96.69
VD	LLC	2048	6144	256	95.97
VD	LLC	2048	6144	512	96.50
VD	VEC	2048	6144	–	96.74
VD	–	4096	0	–	95.68
VD	LLC	4096	2048	256	96.87
VD	LLC	4096	2048	512	96.96
VD	VEC	4096	2048	–	96.91
VD	LLC	4096	4096	256	96.35
VD	LLC	4096	4096	512	96.94
VD	VEC	4096	4096	–	97.11

continued on next page

Eviction policy π	Codec	B_e	B_c	r	RULER-8K
VD	LLC	6144	2048	256	96.81
VD	LLC	6144	2048	512	97.11
VD	VEC	6144	2048	–	97.27
VD	–	8192	0	–	97.47
StreamingLLM	–	2048	0	–	42.71
StreamingLLM	LLC	2048	2048	256	42.78
StreamingLLM	LLC	2048	2048	512	49.92
StreamingLLM	VEC	2048	2048	–	43.78
StreamingLLM	LLC	2048	4096	256	48.88
StreamingLLM	LLC	2048	4096	512	85.84
StreamingLLM	VEC	2048	4096	–	51.58
StreamingLLM	LLC	2048	6144	256	81.71
StreamingLLM	LLC	2048	6144	512	96.02
StreamingLLM	VEC	2048	6144	–	96.87
StreamingLLM	LLC	4096	2048	256	60.34
StreamingLLM	LLC	4096	2048	512	72.91
StreamingLLM	VEC	4096	2048	–	63.02
StreamingLLM	LLC	4096	4096	256	86.72
StreamingLLM	LLC	4096	4096	512	96.27
StreamingLLM	VEC	4096	4096	–	96.96
StreamingLLM	LLC	6144	2048	256	92.96
StreamingLLM	LLC	6144	2048	512	96.82
StreamingLLM	VEC	6144	2048	–	96.78
StreamingLLM	–	8192	0	–	97.27
SnapKV	–	2048	0	–	75.02
SnapKV	LLC	2048	2048	256	80.05
SnapKV	LLC	2048	2048	512	84.79
SnapKV	VEC	2048	2048	–	81.18
SnapKV	LLC	2048	4096	256	85.26
SnapKV	LLC	2048	4096	512	95.18
SnapKV	VEC	2048	4096	–	86.98
SnapKV	LLC	2048	6144	256	93.80
SnapKV	LLC	2048	6144	512	96.44
SnapKV	VEC	2048	6144	–	96.64
SnapKV	LLC	4096	2048	256	90.02
SnapKV	LLC	4096	2048	512	95.63
SnapKV	VEC	4096	2048	–	91.53
SnapKV	LLC	4096	4096	256	95.43
SnapKV	LLC	4096	4096	512	96.75
SnapKV	VEC	4096	4096	–	97.28
SnapKV	LLC	6144	2048	256	96.65
SnapKV	LLC	6144	2048	512	97.27
SnapKV	VEC	6144	2048	–	97.27
SnapKV	–	8192	0	–	97.27

D.2 RULER-16K: LONGER-CONTEXT GRID

Tables 23 to 25 extend the RULER accuracy-vs-memory sweep of Section 4.1 from context 8192 to context 16384 for Qwen3-4B-Instruct, Gemma-3-4B-it, and Qwen3.5-4B under all four eviction policies (KeyDiff, ValueDiff, StreamingLLM, and SnapKV; the latter two per the eviction-policy ablation of Section D.4). All cells are post-RoPE, $r \in \{256, 512\}$ for LLC, 50 samples/task, 13-task RULER-16K mean, using the *same* calibrated codec as the 8192-context experiments, without re-fitting for the longer context. The fixed six-pair budget sweep contains every combination of $B_e \in \{4096, 8192\}$ and $B_c \in \{4096, 6144, 8192\}$, chosen independently of the $B_e{+}B_c{\leq}8192$ constraint used at context 8192. Each table also reports the $B_c{=}0$ (no-compressed-cache) floor at each B_e, following the same convention used throughout our cache-budget comparisons.

The RULER-8K recovery pattern persists: LLC improves the corresponding no-codec floor throughout the KeyDiff/ValueDiff grid and in nearly every configuration across all four policies. Rank-512 LLC also usually exceeds VECTOR at matched token budgets. Most importantly, the bases calibrated

on sequences capped at 8k remain close to the length-specific dense references at 16k without refitting. The tables retain the less favorable and non-monotonic cells rather than reporting only the best operating points.

Table 23: Qwen3-4B-Instruct, RULER-16K, post-RoPE, all four eviction policies. Dense score: 95.47.

Eviction policy π	Codec	B_e	B_c	r	RULER-16K
Dense					95.47
KD	–	4096	0	–	86.37
KD	LLC	4096	4096	256	91.57
KD	LLC	4096	4096	512	93.30
KD	VEC	4096	4096	–	91.43
KD	LLC	4096	6144	256	92.53
KD	LLC	4096	6144	512	93.67
KD	VEC	4096	6144	–	91.12
KD	LLC	4096	8192	256	92.04
KD	LLC	4096	8192	512	93.46
KD	VEC	4096	8192	–	90.60
KD	–	8192	0	–	91.98
KD	LLC	8192	4096	256	93.83
KD	LLC	8192	4096	512	94.52
KD	VEC	8192	4096	–	93.38
KD	LLC	8192	6144	256	93.98
KD	LLC	8192	6144	512	94.67
KD	VEC	8192	6144	–	93.23
KD	LLC	8192	8192	256	94.86
KD	LLC	8192	8192	512	94.93
KD	VEC	8192	8192	–	94.72
VD	–	4096	0	–	92.88
VD	LLC	4096	4096	256	93.68
VD	LLC	4096	4096	512	93.96
VD	VEC	4096	4096	–	93.41
VD	LLC	4096	6144	256	93.77
VD	LLC	4096	6144	512	94.28
VD	VEC	4096	6144	–	93.75
VD	LLC	4096	8192	256	93.86
VD	LLC	4096	8192	512	94.44
VD	VEC	4096	8192	–	93.73
VD	–	8192	0	–	94.24
VD	LLC	8192	4096	256	94.55
VD	LLC	8192	4096	512	94.51
VD	VEC	8192	4096	–	94.43
VD	LLC	8192	6144	256	94.60
VD	LLC	8192	6144	512	94.56
VD	VEC	8192	6144	–	94.32
VD	LLC	8192	8192	256	95.63
VD	LLC	8192	8192	512	95.54
VD	VEC	8192	8192	–	95.81
StreamingLLM	–	4096	0	–	43.18
StreamingLLM	LLC	4096	4096	256	38.41
StreamingLLM	LLC	4096	4096	512	42.78
StreamingLLM	VEC	4096	4096	–	41.55
StreamingLLM	LLC	4096	6144	256	40.69
StreamingLLM	LLC	4096	6144	512	48.36
StreamingLLM	VEC	4096	6144	–	49.28
StreamingLLM	LLC	4096	8192	256	54.93
StreamingLLM	LLC	4096	8192	512	69.34
StreamingLLM	VEC	4096	8192	–	60.32
StreamingLLM	–	8192	0	–	61.47

continued on next page

Eviction policy π	Codec	B_e	B_c	r	RULER-16K
StreamingLLM	LLC	8192	4096	256	58.26
StreamingLLM	LLC	8192	4096	512	62.80
StreamingLLM	VEC	8192	4096	–	62.92
StreamingLLM	LLC	8192	6144	256	62.34
StreamingLLM	LLC	8192	6144	512	67.84
StreamingLLM	VEC	8192	6144	–	70.05
StreamingLLM	LLC	8192	8192	256	91.69
StreamingLLM	LLC	8192	8192	512	93.33
StreamingLLM	VEC	8192	8192	–	90.65
SnapKV	–	4096	0	–	78.65
SnapKV	LLC	4096	4096	256	79.09
SnapKV	LLC	4096	4096	512	82.17
SnapKV	VEC	4096	4096	–	82.10
SnapKV	LLC	4096	6144	256	80.49
SnapKV	LLC	4096	6144	512	83.49
SnapKV	VEC	4096	6144	–	83.86
SnapKV	LLC	4096	8192	256	84.97
SnapKV	LLC	4096	8192	512	89.88
SnapKV	VEC	4096	8192	–	88.66
SnapKV	–	8192	0	–	88.06
SnapKV	LLC	8192	4096	256	91.76
SnapKV	LLC	8192	4096	512	91.85
SnapKV	VEC	8192	4096	–	91.44
SnapKV	LLC	8192	6144	256	93.07
SnapKV	LLC	8192	6144	512	93.12
SnapKV	VEC	8192	6144	–	93.47
SnapKV	LLC	8192	8192	256	94.59
SnapKV	LLC	8192	8192	512	94.36
SnapKV	VEC	8192	8192	–	94.84

Table 24: Gemma-3-4B-it, RULER-16K, post-RoPE, all four eviction policies. Dense score: 89.61. †VECTOR collapses at the heaviest budget (B_e=B_c=8192) under KeyDiff and ValueDiff; StreamingLLM (64.09) and SnapKV (85.66) do not reproduce this collapse at the same budget, see discussion above.

Eviction policy π	Codec	B_e	B_c	r	RULER-16K
Dense					89.61
KD	–	4096	0	–	75.05
KD	LLC	4096	4096	256	83.04
KD	LLC	4096	4096	512	84.82
KD	VEC	4096	4096	–	82.61
KD	LLC	4096	6144	256	84.93
KD	LLC	4096	6144	512	85.89
KD	VEC	4096	6144	–	81.37
KD	LLC	4096	8192	256	86.90
KD	LLC	4096	8192	512	87.59
KD	VEC	4096	8192	–	80.38
KD	–	8192	0	–	83.17
KD	LLC	8192	4096	256	87.57
KD	LLC	8192	4096	512	87.78
KD	VEC	8192	4096	–	86.09
KD	LLC	8192	6144	256	88.20
KD	LLC	8192	6144	512	88.76
KD	VEC	8192	6144	–	84.71
KD	LLC	8192	8192	256	88.63
KD	LLC	8192	8192	512	89.42
KD	VEC	8192	8192	–	77.40†

continued on next page

Eviction policy π	Codec	B_{e}	B_{c}	r	RULER-16K
VD	–	4096	0	–	83.47
VD	LLC	4096	4096	256	84.89
VD	LLC	4096	4096	512	85.01
VD	VEC	4096	4096	–	85.06
VD	LLC	4096	6144	256	86.06
VD	LLC	4096	6144	512	86.68
VD	VEC	4096	6144	–	83.92
VD	LLC	4096	8192	256	87.78
VD	LLC	4096	8192	512	87.76
VD	VEC	4096	8192	–	82.04
VD	–	8192	0	–	85.51
VD	LLC	8192	4096	256	87.71
VD	LLC	8192	4096	512	87.61
VD	VEC	8192	4096	–	84.85
VD	LLC	8192	6144	256	89.26
VD	LLC	8192	6144	512	89.36
VD	VEC	8192	6144	–	84.04
VD	LLC	8192	8192	256	89.86
VD	LLC	8192	8192	512	89.44
VD	VEC	8192	8192	–	$67.64^{\dagger}$
StreamingLLM	–	4096	0	–	37.75
StreamingLLM	LLC	4096	4096	256	48.23
StreamingLLM	LLC	4096	4096	512	54.84
StreamingLLM	VEC	4096	4096	–	42.62
StreamingLLM	LLC	4096	6144	256	53.50
StreamingLLM	LLC	4096	6144	512	62.97
StreamingLLM	VEC	4096	6144	–	48.16
StreamingLLM	LLC	4096	8192	256	60.29
StreamingLLM	LLC	4096	8192	512	74.66
StreamingLLM	VEC	4096	8192	–	46.95
StreamingLLM	–	8192	0	–	56.11
StreamingLLM	LLC	8192	4096	256	64.53
StreamingLLM	LLC	8192	4096	512	71.14
StreamingLLM	VEC	8192	4096	–	60.15
StreamingLLM	LLC	8192	6144	256	71.50
StreamingLLM	LLC	8192	6144	512	79.26
StreamingLLM	VEC	8192	6144	–	65.84
StreamingLLM	LLC	8192	8192	256	83.09
StreamingLLM	LLC	8192	8192	512	89.20
StreamingLLM	VEC	8192	8192	–	64.09
SnapKV	–	4096	0	–	68.16
SnapKV	LLC	4096	4096	256	72.14
SnapKV	LLC	4096	4096	512	70.95
SnapKV	VEC	4096	4096	–	74.37
SnapKV	LLC	4096	6144	256	75.45
SnapKV	LLC	4096	6144	512	75.87
SnapKV	VEC	4096	6144	–	75.53
SnapKV	LLC	4096	8192	256	78.90
SnapKV	LLC	4096	8192	512	85.46
SnapKV	VEC	4096	8192	–	79.74
SnapKV	–	8192	0	–	75.77
SnapKV	LLC	8192	4096	256	83.15
SnapKV	LLC	8192	4096	512	82.55
SnapKV	VEC	8192	4096	–	84.14
SnapKV	LLC	8192	6144	256	86.03
SnapKV	LLC	8192	6144	512	87.69
SnapKV	VEC	8192	6144	–	88.19
SnapKV	LLC	8192	8192	256	89.27
SnapKV	LLC	8192	8192	512	89.51
SnapKV	VEC	8192	8192	–	85.66

Table 25: Qwen3.5-4B, RULER-16K, post-RoPE, all four eviction policies. Dense score: 95.88.

Eviction policy π	Codec	B_e	B_c	r	RULER-16K
Dense					95.88
KD	–	4096	0	–	90.45
KD	LLC	4096	4096	256	93.93
KD	LLC	4096	4096	512	95.53
KD	VEC	4096	4096	–	94.70
KD	LLC	4096	6144	256	93.95
KD	LLC	4096	6144	512	95.63
KD	VEC	4096	6144	–	95.20
KD	LLC	4096	8192	256	94.84
KD	LLC	4096	8192	512	95.83
KD	VEC	4096	8192	–	95.14
KD	–	8192	0	–	93.30
KD	LLC	8192	4096	256	94.70
KD	LLC	8192	4096	512	95.36
KD	VEC	8192	4096	–	95.21
KD	LLC	8192	6144	256	95.16
KD	LLC	8192	6144	512	95.52
KD	VEC	8192	6144	–	95.52
KD	LLC	8192	8192	256	95.52
KD	LLC	8192	8192	512	95.67
KD	VEC	8192	8192	–	95.67
VD	–	4096	0	–	92.60
VD	LLC	4096	4096	256	95.44
VD	LLC	4096	4096	512	95.21
VD	VEC	4096	4096	–	95.66
VD	LLC	4096	6144	256	95.30
VD	LLC	4096	6144	512	95.49
VD	VEC	4096	6144	–	95.83
VD	LLC	4096	8192	256	95.30
VD	LLC	4096	8192	512	95.67
VD	VEC	4096	8192	–	95.52
VD	–	8192	0	–	95.15
VD	LLC	8192	4096	256	95.31
VD	LLC	8192	4096	512	95.36
VD	VEC	8192	4096	–	95.52
VD	LLC	8192	6144	256	95.36
VD	LLC	8192	6144	512	95.36
VD	VEC	8192	6144	–	95.83
VD	LLC	8192	8192	256	95.67
VD	LLC	8192	8192	512	95.67
VD	VEC	8192	8192	–	96.03
StreamingLLM	–	4096	0	–	41.59
StreamingLLM	LLC	4096	4096	256	53.22
StreamingLLM	LLC	4096	4096	512	67.74
StreamingLLM	VEC	4096	4096	–	47.64
StreamingLLM	LLC	4096	6144	256	55.73
StreamingLLM	LLC	4096	6144	512	87.64
StreamingLLM	VEC	4096	6144	–	62.23
StreamingLLM	LLC	4096	8192	256	71.64
StreamingLLM	LLC	4096	8192	512	93.93
StreamingLLM	VEC	4096	8192	–	84.05
StreamingLLM	–	8192	0	–	60.54
StreamingLLM	LLC	8192	4096	256	67.85
StreamingLLM	LLC	8192	4096	512	83.96
StreamingLLM	VEC	8192	4096	–	68.14
StreamingLLM	LLC	8192	6144	256	85.37
StreamingLLM	LLC	8192	6144	512	93.76
StreamingLLM	VEC	8192	6144	–	82.80

continued on next page

Eviction policy π	Codec	B_e	B_c	r	RULER-16K
StreamingLLM	LLC	8192	8192	256	93.00
StreamingLLM	LLC	8192	8192	512	95.45
StreamingLLM	VEC	8192	8192	–	95.58
SnapKV	–	4096	0	–	78.76
SnapKV	LLC	4096	4096	256	86.96
SnapKV	LLC	4096	4096	512	89.98
SnapKV	VEC	4096	4096	–	83.65
SnapKV	LLC	4096	6144	256	91.58
SnapKV	LLC	4096	6144	512	93.80
SnapKV	VEC	4096	6144	–	89.74
SnapKV	LLC	4096	8192	256	93.23
SnapKV	LLC	4096	8192	512	95.42
SnapKV	VEC	4096	8192	–	94.58
SnapKV	–	8192	0	–	82.05
SnapKV	LLC	8192	4096	256	93.77
SnapKV	LLC	8192	4096	512	95.14
SnapKV	VEC	8192	4096	–	92.59
SnapKV	LLC	8192	6144	256	94.95
SnapKV	LLC	8192	6144	512	95.61
SnapKV	VEC	8192	6144	–	95.77
SnapKV	LLC	8192	8192	256	94.90
SnapKV	LLC	8192	8192	512	95.76
SnapKV	VEC	8192	8192	–	95.67

D.3 RULER-32K RESULTS

The 16k grid's clearest result is VECTOR's collapse under both eviction policies at the heaviest budget (Table 24). Pushing to context 32768 (the ValueDiff eviction policy, same calibrated codecs, no re-fitting) on Qwen3-4B-Instruct and Llama-3.1-8B reproduces this pattern in a starker form, and additionally shows it is not confined to one split.

Table 26: RULER-32K pre- vs. post-RoPE accuracy under ValueDiff at three compressed-heavy cache splits. LLC uses r=256 and Δ = post − pre.

Model	Codec	Split (B_e+B_c)	Pre	Post	Δ
Qwen3-4B-Instruct	LLC	4096+12288	92.06	92.42	+0.36
	VECTOR	4096+12288	93.16	91.52	−1.64
	LLC	8192+8192	92.99	93.37	+0.38
	VECTOR	8192+8192	93.04	93.09	+0.05
	LLC	8192+16384	79.23	93.72	+14.49
	VECTOR	8192+16384	93.95	78.23	−15.72
Llama-3.1-8B	LLC	4096+12288	68.82	71.84	+3.02
	VECTOR	4096+12288	82.08	**0.31**	**−81.77**
	LLC	8192+8192	72.27	76.25	+3.98
	VECTOR	8192+8192	84.60	45.87	−38.73
	LLC	8192+16384	76.03	78.98	+2.95
	VECTOR	8192+16384	81.01	**0.00**	**−81.01**

Both models fail under VECTOR at 32k given enough compressed-tier tokens. Llama-3.1-8B collapses to near-zero at two of three splits, generating degenerate repetitive output rather than an on-task answer, and Qwen3-4B-Instruct, robust everywhere at 16k, breaks too at the 75% split. At a fixed 16k-token budget (50% total retention), Llama-3.1-8B's penalty grows from −38.73 at 8192+8192 to −81.77 at 4096+12288 as the allocation shifts from exact to compressed states. Thus total budget or retention level alone does not explain the failure. LLC is positive in every cell in Table 26.

This pattern is not universal across every split, however. At the lighter 2048+6144 split (one exact token per three compressed tokens, Qwen3-4B-Instruct, ValueDiff), both codecs stay post-RoPE-*favorable* at both 16k and 32k:

Table 27: Qwen3-4B-Instruct, 2048+6144 split, the ValueDiff eviction policy, at 16384 and 32768 context. At 32k, VECTOR's post-minus-pre sign is opposite to its negative changes at 4096+12288 and 8192+16384 in Table 26.

Context	LLC pre	LLC post	Δ	VECTOR pre	VECTOR post	Δ
16384	91.78	93.50	+1.72	92.25	92.44	+0.19
32768	87.50	90.87	+3.37	87.83	90.73	+2.90

At 32k, this split has the same exact-to-compressed ratio (B_e:B_c = 1:3) as the 4096+12288 row in Table 26, but half the capacity in each tier. Their opposite VECTOR pre/post behavior therefore shows that retention ratio alone does not explain the collapse. Isolating whether absolute exact capacity, absolute compressed capacity, or their interaction is responsible would require a dedicated sweep holding one tier fixed while varying the other; we have not run that ablation. We report both regimes rather than extrapolate a mechanism from the heavy-split grid alone.

D.4 EVICTION-POLICY COMPARISON

The four policies in Figure 2 draw on distinct signals. StreamingLLM retains attention sinks and recent tokens using a positional rule (Xiao et al., 2024); SnapKV (Li et al., 2024) uses attention scores; KeyDiff and ValueDiff use key- and value-space diversity. All four feed the same two-tier cache, so matched comparisons isolate the eviction signal rather than the codec or budget.

At RULER-8K, SnapKV exceeds StreamingLLM in 107 of 108 matched codec–budget pairs. The gap is largest at small exact-cache allocations, where the positional StreamingLLM rule leaves more recovery work to the compressed tier. Nevertheless, increasing the compressed capacity substantially improves both policies. The full six-model breakdown is reported in Section D.1.

At RULER-16K, the same ordering holds at lighter budgets but narrows at the largest cache allocation:

Table 28: Best RULER-16K result per model for each policy family, excluding B_c=0, at the shared maximum budget B_e=B_c=8192.

Model	StreamingLLM best	SnapKV best	KD/VD best	Dense
Qwen3-4B-Instruct	93.33	94.84	95.81	95.47
Gemma-3-4B-it	89.20	89.51	89.86	89.61
Qwen3.5-4B	95.58	95.76	96.03	95.88

At this largest budget, every policy family approaches its dense reference; KeyDiff/ValueDiff reaches the highest cell for each model. Gemma-3-4B also shows that the isolated VECTOR collapse under KeyDiff/ValueDiff is not a general consequence of increasing compressed capacity: the corresponding StreamingLLM and SnapKV results remain near dense. Thus the base policy matters most when exact capacity is scarce, while a sufficiently large, high-fidelity second tier narrows the policy gap.