LOOMKV: STRUCTURED LOW-RANK KV CACHES FOR VIDEO LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Among the most accurate ways to compress a video language model's KV cache, a low-rank factorization shared across a group of layers must first hold the very cache it compresses. We compute it per input during prefill, keeping a fixed-rank representation rather than a fixed fraction of tokens. Which structure to keep is itself a result: a matched-storage control rejected our first, grid-preserving Tucker design in favour of sharing across layers (47.97 against 59.12), and the grid does not pay for itself after that extraction either. The offline cross-layer factorization, however, consumes the completed prompt cache, which is exactly what cannot be held. LoomKV removes that dependence. As visual blocks arrive it maintains compact coefficients, a shared channel basis and the Gram matrix needed to refresh that basis, and a refresh from $r + S_b$ rows is exact for the whole history the state represents, so the dense multi-layer cache is never materialized. On a 24 GB card the offline factorization fails at 128 frames and a chunked dense control at 160; LoomKV completes 224, per-session resident state is 5.7 to 7.7$\times$ smaller, and nine prefilled sessions stay resident against one. At a matched answer-time cache it outperforms InfiniPot-V by 7.46 points on Qwen2-VL and 13.56 on Qwen2.5-VL on MME-VideoOCR, stays within 3.6 points of the offline ceiling across three model families, and on long Video-MME is indistinguishable from dense while retaining 11.9% of the visual cache. Controlled ablations attribute the gain to cross-layer sharing; refresh frequency, coefficient transport and causal feedback have no additional cost this sample resolves. All primary comparisons are paired, and negative results are reported.

1 INTRODUCTION

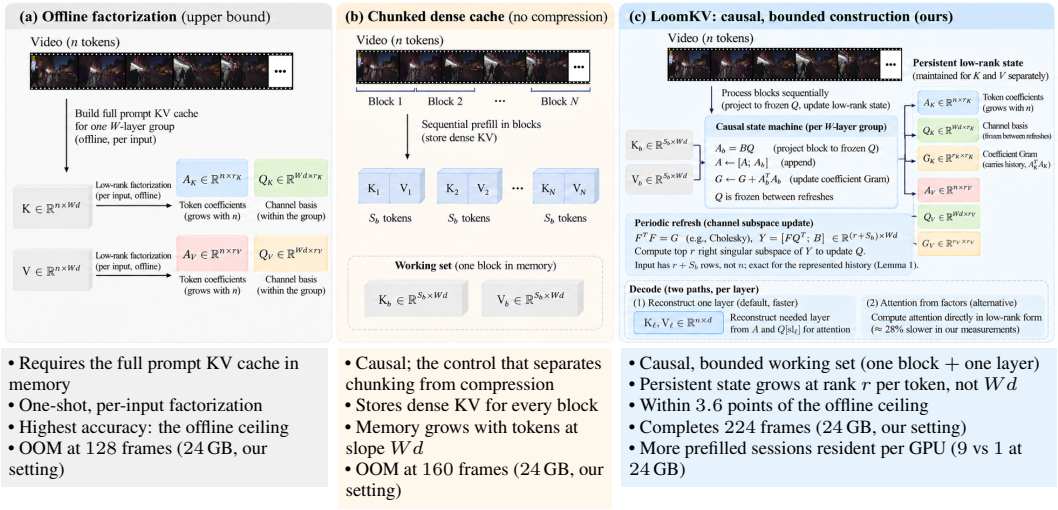
A seven-billion-parameter video language model reading a few hundred high-resolution frames produces a key–value cache of tens of gigabytes. On a 24 GB card, a 128-frame native-resolution prefill of Qwen2-VL-7B runs out of memory before it finishes (Table 2). The cache is the binding constraint, and compressing it is the obvious response. The construction here runs that prefill to 224 frames inside the same 24 GB, nine resident sessions against one.

Cross-layer low-rank compression is the strongest answer currently available. xKV Chang et al. (2026) observes that the caches of neighbouring layers share a dominant subspace, factorizes a group of layers jointly, and reaches about 8$\times$ compression with little accuracy loss. Its construction, however, consumes the completed prompt cache: the group's keys and values for *all* tokens are concatenated and decomposed. When the constraint is that this cache cannot be held, a method that begins by holding it does not apply.

Where to measure matters as much as what. We measure on MME-VideoOCR at native resolution because it is where compression is not free: arms at the same 12.5% budget differ by up to 11 points there (Table 5), whereas on long Video-MME LoomKV at 11.9% of the cache is indistinguishable from dense (Table 8), so that benchmark tests memory rather than accuracy.

This construction has a lineage, and it is a result: **the grid does not pay**. The natural hypothesis is that the frame$\times$space$\times$channel grid is the structure to preserve, and our first design did so with a three-mode Tucker factorization. On MME-VideoOCR at a matched answer-time cache it scores 47.97 against 59.12 for a factorization shared across layers, and widening its spatial subspace makes it worse (45.61; Table 5, Appendix A). What a low-dimensional spatial subspace discards, a rank-188

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.



- Requires the full prompt KV cache in memory
- One-shot, per-input factorization
- Highest accuracy: the offline ceiling
- OOM at 128 frames (24 GB, our setting)

- Causal; the control that separates chunking from compression
- Stores dense KV for every block
- Memory grows with tokens at slope Wd
- OOM at 160 frames (24 GB, our setting)

- Causal, bounded working set (one block + one layer)
- Persistent state grows at rank r per token, not Wd
- Within 3.6 points of the offline ceiling
- Completes 224 frames (24 GB, our setting)
- More prefilled sessions resident per GPU (9 vs 1 at 24 GB)

Figure 1: Three ways to build a cross-layer low-rank cache from the same input. The offline factorization **(a)** needs a layer group's completed prompt cache, the object that cannot be held, and is the ceiling we measure against; a chunked dense prefill **(b)** is causal but compresses nothing; LoomKV **(c)** keeps (A, Q, G) per side, appends projected blocks with no decomposition, and refreshes Q from G and the current block (Lemma 1). Bottom row: measured, Sections 5–7.

factorization of the token axis keeps, and the grid does not pay for itself a second time either: spatially factorizing LoomKV's own coefficient field at matched bytes loses to rank in 12 of 12 cells at the primary rank, and in every row's median (Table 4). That is why this paper preserves cross-layer structure.

This paper asks what to do in that regime. LoomKV builds the same kind of representation—a shared low-rank basis across a group of layers—but builds it *while* the prefill is running, block by block, so that the dense multi-layer cache is never materialized (Figure 1). The visual span is consumed in blocks; each block is projected onto a frozen channel basis and appended to a coefficient matrix; the basis is refreshed on a schedule.

The question this paper answers is which parts of the construction matter. Moving an offline factorization into a causal streaming loop forces several choices that do not exist offline: later blocks' keys and values are computed against an already-compressed history, the basis must be rebuilt on some schedule, old coefficients must be moved into each new basis, and layers must be grouped into some width. We varied all four, and the result is lopsided. **The grouping width dominates, and the online mechanics have no additional cost this sample resolves**, which is the offline width behaviour of Chang et al. (2026) surviving causalization. Giving each layer its own basis instead of sharing one across four costs 10.53 points $[-15.52, -5.53]$ at an identical budget. Against that, refreshing four times less often costs 1.31 points with the interval covering zero, geometry-preserving transport of old coefficients changes nothing we can resolve, and causal feedback is not resolved at n=295: at per-block refresh the construction is within 1.36 points of one-shot factorization and agrees with it on 95.3% of answers.

Our contributions are:

- **LoomKV removes a construction dependency of offline cross-layer factorization** (Section 3): the offline method needs the completed prompt cache before it can compress, whereas LoomKV builds the compressed state causally under a bounded working set, with a rank rule that commits (r_K, r_V) before the tokens arrive.
- **It realizes that construction with periodic low-rank refreshes and a compact sufficient statistic** (Lemma 1): blocks between refreshes enter through their coordinates in the frozen basis, and each refresh is exact for the history the state represents.
- **It measures the resulting quality–memory–latency trade-off** (Sections 5–8): a chunked-uncompressed control separates chunking's share from compression's, and we report peak

and resident memory, accuracy against the offline ceiling and against eviction, and prefill and decode time. Sharing width is the one choice that moves accuracy.

2 BACKGROUND AND RELATED WORK

Low-rank KV compression. Palu Chang et al. (2025) factorizes each layer's cache independently. xKV Chang et al. (2026) observes that neighbouring layers share a dominant subspace and factorizes a group of W layers jointly against a shared token basis, the representation this paper builds. Its construction is offline with respect to the prompt: "we horizontally concatenate the caches of all layers $\ell \in \mathcal{W}_k$ and compute a rank-r low-rank factorization using SVD". The decomposition runs inside prefill rather than as a separate calibration pass, but it consumes the group's completed keys and values. xKV also introduces Selective Reconstruction, a decode-time mechanism that scores queries against per-chunk landmarks and reconstructs only the selected rows; we return to it in Section 7. OjaKV Zhu et al. (2026) adapts a subspace online during decoding, initialized from the complete prompt cache. PuzzleKV Wang et al. (2026) factorizes page-wise and reports logical cache footprint by execution stage, so the observation that construction-time memory differs from stored-cache size is not new with us.

Eviction and token selection. StreamingLLM Xiao et al. (2024), H2O Zhang et al. (2023) and, for video, InfiniPot-V Kim et al. (2025) keep a bounded set of tokens and discard the rest. These methods are bounded-memory by construction and are the natural comparison for ours at a matched cache budget. StreamKV Chen et al. (2026) is the closest neighbour in setting rather than mechanism: it segments a video stream, retrieves segments by summary vector and compresses the rest, retaining selected keys and values where we keep the full support at reduced fidelity. Eviction of either kind discards information irreversibly, and Section 5 measures what that is worth.

Chunked prefill and incremental decomposition. Processing a long prompt in chunks is standard, and KV-Fold Nadali et al. (2026) carries an uncompressed cache across chunks as the accumulator of a left fold. Chunked prefill is therefore not a contribution of this paper, and Section 6 is explicit about which memory effect it accounts for. The incremental update of a truncated decomposition is classical Brand (2006): a new block is split into its component in the current subspace and its orthogonal residual, and a small matrix carrying both is re-decomposed. Our refresh applies that update periodically, every k blocks, with the small matrix formed at refresh time from the Gram matrix of the coordinates accumulated since the last one and the current block; what this periodicity costs is measured rather than assumed, and is not resolved at n=306 (Appendix I). Zhang (2022) instead defers the reorthogonalization of an already-updated basis. Covariance sketching is classical too: Frequent Directions Liberty (2013); Ghashami et al. (2016) keeps a small B with $B^\top B \approx X^\top X$, and Oja's rule Oja (1982) tracks a subspace stochastically, the route OjaKV takes. These are useful primitives, but none of them addresses the constraint studied here: maintaining a cross-layer compressed state through a video prefill without first materializing the completed prompt cache. Our Gram matrix is the exact second moment of the coefficients the state represents rather than an approximation of the dense input's, so the refresh is an equivalence rather than a bound. The orthogonal Procrustes problem used in one ablation is classical too Schönemann (1966), and PagedAttention Kwon et al. (2023) frames how we report capacity.

Quantization. KIVI Liu et al. (2024) keeps a full-precision recent window beside a quantized cache. xKV notes that low-rank compression composes with existing quantization but reports none of its own factors; we include an ablation that quantizes the coefficient matrix directly. At its default 2 bits with a 32-element group sharing a 16-bit scale and zero, KIVI stores about 3 bits per element, about $5\times$ before its full-precision residual, short of this paper's $8\times$ element budget, and about 2.25 bits ($7\times$) even at 128-element groups; it attacks precision where we attack rank, and the two compose.

3 METHOD

Setting. A video language model embeds T temporal patches of S tokens each into a prompt whose visual span occupies positions $[\ell_0, \ell_1)$. Writing L for the number of decoder layers, H_{kv} for the

number of key–value heads and D for the head dimension, the dense visual cache is $2L(\ell_1-\ell_0)H_{kv}D$ elements. On the settings in this paper that is 1.3 to 6.6 GB in bfloat16, on top of 16 GB of weights.

Representation. Following xKV, layers are partitioned into groups of W. For a group g and side $s \in \{K, V\}$ the tokens' cache rows are concatenated across the group's layers into $X \in \mathbb{R}^{n\times Wd}$ with $d = H_{kv}D$, and approximated as $X \approx AQ^\top$ with $A \in \mathbb{R}^{n\times r}$ the coefficients and $Q \in \mathbb{R}^{Wd\times r}$ an orthonormal channel basis. Layer ℓ of the group reads its own slice $Q[\mathrm{sl}_\ell]$, so reconstructing one layer's history costs $n \times d$ and no more.

Construction. The prefix text is kept verbatim; the visual span is consumed in blocks of S_b tokens. For each block $B \in \mathbb{R}^{S_b\times Wd}$ an *append* projects and stacks, $Z = BQ$, $A \leftarrow [A; Z]$, $G \leftarrow G + Z^\top Z$; a *refresh* rebuilds the basis from

$$F^\top F = G, \quad Y = \begin{bmatrix} FQ^\top \\ B \end{bmatrix}, \quad Q' \leftarrow \text{top-}r \text{ right singular vectors of } Y, \quad A \leftarrow \begin{bmatrix} A(Q^\top Q') \\ BQ' \end{bmatrix}, \tag{1}$$

after which $G \leftarrow A^\top A$. An append costs one projection and no decomposition. The refresh is the construction's central step, and it is exact in the following sense.

Lemma 1 (Exact refresh of the represented history)**.** *Let the stored state be $\widehat{X} = AQ^\top$ with $Q^\top Q = I$ and $G = A^\top A$, and let $F^\top F = G$. Then $\widehat{X}^\top \widehat{X} = QGQ^\top = (FQ^\top)^\top (FQ^\top)$, so $Y = [\,FQ^\top; B\,]$ and $[\,\widehat{X}; B\,]$ share a right Gram matrix and therefore a top-r right singular subspace.*

Refreshing from Y is therefore equivalent to refreshing from the entire history the state represents, while the $n \times Wd$ matrix is never formed: Y has $r + S_b$ rows against the history's n. The carried Gram matrix buys this, and is why blocks appended since the last refresh keep their weight. Read as a periodic incremental update Brand (2006), blocks arriving between refreshes are represented in the current subspace until the next basis update, while the block that triggers a refresh also contributes its out-of-subspace component; Lemma 1 is what makes the refresh itself exact for the accumulated represented history. We take the top-r right singular vectors from an eigendecomposition of the smaller Gram matrix rather than a direct SVD; the subspaces agree to a principal cosine of 1.000000, and the route is $4.5\times$ faster on our refreshes and $8.8\times$ on the offline method's single decomposition, so both are timed with it.

Attention during construction. At layer ℓ of a block, attention sees the verbatim prefix, the reconstructed history $A\,Q[\mathrm{sl}_\ell]^\top$ and the current block; the reconstruction lives for that layer's forward and is discarded, so the largest dense object built is one layer's history. With compression disabled the construction reproduces one-shot prefill's predictions on a gate sample of two to nineteen questions, at a maximum absolute logit difference of 0.25 (Appendix C).

What the state costs. The offline factorization needs its input, of order nWd per group, before it produces anything; LoomKV holds $nr + Wdr + r^2$ plus one block's workspace. Both are linear in tokens; the coefficient on the history term falls from Wd to r.

Rank selection is forced to be causal. An offline method can choose r after seeing the whole spectrum; xKV fixes a 1:1.5 ratio between the key and value ranks and does not specify a selection rule. A streaming construction must choose before the tokens arrive. We solve (r_K, r_V) per input from the answer-time budget and the token count alone,

$$n_{\text{groups}}(r_K + r_V)(n + Wd) \;\le\; \rho \cdot 2LnH_{kv}D, \qquad r_V/r_K = 1.5,$$

with $\rho = 0.125$ throughout. The rule uses no answer, no accuracy and no competitor output, and it reproduces the ratio xKV fixes by hand. It does need n: this is a bounded-memory construction over a span whose length is known when it starts, not an open-ended stream. Every arm in every table spends the same number of elements.

4 Experimental setup

Models and benchmarks. Three families: Qwen2-VL-7B-Instruct Wang et al. (2024), Qwen2.5-VL-7B-Instruct Bai et al. (2025), and InternVL2.5-8B Chen et al. (2024), whose language model

is InternLM2.5 and whose visual front end and position encoding differ from the Qwen pair. Two benchmarks: MME-VideoOCR Shi et al. (2025) at native resolution, a reading task on which arms at the same budget differ by up to 11 points (Table 5), and the long split of Video-MME Fu et al. (2025) at 768 frames, on which compression is nearly free (Table 8). Its accuracy runs cap resolution (`max_pixels` 100,352, about 51.8k visual tokens per question); the memory runs of Section 6 use native resolution, about 115k tokens at 768 frames (Table 6). On MME-VideoOCR the loader yields 300 questions of which 295 decode on the Qwen pair; the InternVL recipe samples 64 frames at 256 tokens each and yields 394. Video-MME long is 298 questions. All arms within a table answer the same questions in the same order.

Budget parity. Every compressed arm holds the same answer-time cache, 12.5% of the dense visual key–value tensor in elements. Ranks are solved per input by the rule in Section 3, giving $(188, 282)$ for the Qwen pair and $(329, 494)$ for InternVL. Measured stored fractions land within 0.2% of the cap on every arm's mean, worst single question $+0.89\%$; integer rank rounding causes the spread and affects every arm alike. The comparison is in elements, not bytes, so a dtype change cannot buy budget; the one arm that changes dtype is reported separately in bytes.

Baselines. *InfiniPot-V* Kim et al. (2025), streaming eviction, at the same answer-time cache, is the bounded-memory competitor. *xKV* $W{=}1$ Chang et al. (2026) is per-layer low-rank at the same budget, which isolates what cross-layer sharing buys. *Dense* is uncompressed one-shot prefill. The *chunked exact* control is our own prefill loop with keys and values kept verbatim, which applies no compression and is therefore not an accuracy competitor; it exists to attribute memory effects. *Offline xKV* $W{=}4$ builds the dense cache and then factorizes it. It cannot run in this paper's memory regime, so it appears as the unbounded-memory ceiling in its own column, and what we report against it is the gap to that ceiling.

Statistics. Differences are paired on the same questions and reported as a point estimate with a 95% Wald interval and an exact McNemar p, plus the video-clustered interval where questions share a video; we report both whenever they disagree. Comparisons that carry a verdict are at $n \geq 295$; the two smaller samples we keep, $n{=}19$ and $n{=}12$, are labelled as pilots wherever they appear (Appendix C).

Hardware, and a caution about it. Memory, capacity and latency are measured on a single NVIDIA A6000, a discrete 48 GB card, with the 24 GB condition enforced by `set_per_process_memory_fraction` so that the 16 GB of weights counts against the cap as it would on a smaller card. Reported peaks are PyTorch allocator peaks. Accuracy runs used that card and a GB10 development machine, and *which card a run used changes about* 2.5% *of its predictions*, measured on an uncompressed forward pass with identical code, weights and inputs (Appendix G). It is noise rather than bias, but it is the size of effects this literature reports, so every paired comparison here is within one card unless we say otherwise.

5 Accuracy: near the ceiling, far above eviction

Against the methods that run under the same cap. Table 1 gives the main comparison. **Against the only other method here that never builds the dense cache, LoomKV at refresh-4 beats streaming eviction by** 7.46 **points** (95% CI $[+2.23, +12.69]$, McNemar $p{=}0.007$). It beats per-layer low-rank at the same budget by the same 7.46 points ($[+2.95, +11.96]$, $p{=}0.002$), though that arm does build the dense cache first and is included to isolate what cross-layer sharing buys. On Qwen2.5-VL the margin over eviction is 13.56 points ($[+7.54, +19.58]$). That is the strongest of the bounded-memory methods compared here. Cross-layer sharing is where the margin comes from: at an identical budget $W{=}1$ and eviction score the same number correct, 143 of 295, on different questions.

Against the unbounded-memory ceiling, where it exists. The ceiling must hold the cache it factorizes, so the gap to it is defined only where that cache fits. Table 8 in the appendix reports it across four such settings; **it is within** 3.6 **points everywhere**, and two of the four intervals exclude zero. On Qwen2-VL it is -3.39, Wald $[-6.78, -0.002]$ and video-clustered $[-6.64, -0.34]$; on

Table 1: MME-VideoOCR, Qwen2-VL-7B, 295 questions, every compressed arm at the same $8\times$ answer-time cache. The last row builds the dense cache first and is the unbounded-memory ceiling, not a competitor under this paper's memory cap. The LoomKV rows and the dense row were run on one machine and the three reference rows on another; see Appendix G. Storing the factors in bfloat16 instead of float32 moves this arm by $+0.68$ $[-1.81, +3.16]$, and the memory and width tables use the bfloat16 arm.

Method	Bounded memory	Kind	Correct	Accuracy
dense (no compression)	–	–	173/295	58.64
InfiniPot-V (eviction)	yes	evict	143/295	48.47
xKV $W{=}1$ (needs dense prefill)	no	low-rank	143/295	48.47
LoomKV (refresh 4, float32 factors)	yes	low-rank	165/295	55.93
LoomKV, K frozen	yes	low-rank	158/295	53.56
offline xKV $W{=}4$ (unbounded-memory ceiling)	no	low-rank	175/295	59.32

InternVL -3.55, $[-6.78, -0.33]$ and $[-6.75, -0.50]$. Both exclude zero under both procedures, which we say rather than rounding the Wald endpoint to zero, and both are cross-card, which widens them by about a point (Appendix G). On Qwen2.5-VL and long Video-MME the construction is at or above the ceiling, and on long Video-MME it is indistinguishable from dense, $+0.00$ $[-3.60, +3.60]$, 15 answers lost and 15 gained, holding 11.9% of the visual cache. Where the dense cache fits, the offline method is the right choice, up to three points better on one setting; LoomKV is for where it does not. We do not decompose that gap; the per-block and refresh terms of Section 8 are measured against different references, and adding them is not licensed by anything we ran. Re-running refresh-4 under the chunked vision path the memory runs use gives 55.93 again, $+0.00$ $[-2.49, +2.49]$, so both tables describe the same method.

The construction tracks the ceiling until rank runs out, and this is a limitation. Table 13 in the appendix sweeps the compression ratio. The gap to the unbounded-memory reference is -0.34 $[-3.07, +2.39]$ at $4\times$, -3.04 $[-6.35, +0.27]$ at $8\times$ and -5.41 $[-9.90, -0.91]$ at $16\times$: $16\times$ **is the only one whose interval excludes zero.** The rule fixed in advance separated a capacity reading from a limitation reading by whether the gap widens by more than 2.5 points from $8\times$ to $16\times$; it widens by 2.36 here and 2.70 under the other decomposition route, so it turns on a nuisance term and we claim neither (Appendix G). What holds under both routes: at $16\times$ the shared low-rank representation is near its own capacity, the offline reference falling 6.42 points and LoomKV 8.78, and LoomKV gives up more. Repeated causal approximation is close to free while rank is ample and costs when rank is tight; what degrades is not the refresh identity of Lemma 1 but the rank available to carry the state it refreshes from.

The refresh schedule is a cost dial, not a tuning opportunity. Table 10 in the appendix sweeps it within one card, superseding an earlier cross-card sweep. Refreshing every block instead of every four quadruples the decompositions and buys $+1.31$ points $[-1.56, +4.17]$; every two buys $+0.76$ $[-1.51, +3.03]$, each row paired against the every-four arm run under its own decomposition route (Appendix G). Both intervals cover zero, and the same sweep on the other machine agrees in sign and also covers zero. What the schedule reliably changes is time: 13.57, 10.71 and 9.05 seconds of language-model prefill, against 4.31 for an uncompressed one. A deployment should therefore take the cheapest schedule it can tolerate rather than tune this axis; we use every four throughout. Freezing the key basis outright costs 2.37 points, but that interval $[-6.30, +1.56]$ covers zero as well, so we have found no refresh setting whose accuracy cost this sample can resolve, the extreme one included.

6 Memory: compression bounds the persistent state

The capability gap against the published baselines. Table 2 gives the 24 GB cap; on 48 GB (Table 6, appendix) offline xKV fails at 512 frames and dense at 768, and LoomKV completes both under 24 GB of peak.

The control that reattributes most of that gap. How much of this is compression and how much is simply not holding the whole prompt in flight? A control running the identical prefill loop and

Table 2: Peak memory (GB) during prefill at a 24 GB cap, Qwen2-VL-7B, native resolution; "–" is a cell not run because a smaller one failed. The prior methods' first step is the step that does not fit. The 48 GB grid is Table 6.

Frames	Tokens	dense	offline xKV	chunked exact	**LoomKV**
64	23040	22.4	23.2	19.7	18.8
128	46080	**OOM**	**OOM**	22.6	20.6
160	57600	–	–	**OOM**	21.6
192	69120	–	–	–	22.4
224	80640	–	–	–	23.4
256	92160	–	–	**OOM**	**OOM**

vision path, keeping keys and values verbatim, completes every cell LoomKV completes. Chunked prefill is not our contribution, and this control says which effect it owns: **chunking bounds the working set; compression bounds the persistent state.** Without it a chunked method can claim a compression result it did not earn.

What compression owns, measured against that control. With bfloat16 coefficients, each layer's reconstruction built in the model dtype and the refresh intermediates freed, LoomKV's peak is below the exact control at every completed cell, by 0.85 GB at 64 frames rising to 5.53 at 768. Both arms stay linear in visual tokens; what differs is the slope of the persistent term, r against d. Over 256 to 768 frames our peak is nearly flat (24.26, 23.99, 23.95) against the control's (28.57, 29.20, 29.48): a slope difference, not a different order of growth, and flat over this range rather than by construction.

A capability separation against the control exists, and we located it. The two share the prefill loop and the vision path; the only difference is what is kept, and Table 2 shows where that difference starts to decide whether a cell runs at all.

Does the construction hold up in the regime it opens? A pre-registered probe runs LoomKV alone on Video-MME long at native resolution at 64, 128 and 224 frames over the same 58 questions, scoring 46.55, 50.00 and 50.00. We resolve neither a length-induced degradation (224 against 64, $+3.45$ $[-7.20, +14.10]$) nor, at 224 frames and up to 80,640 visual tokens, a gap to an uncompressed arm measured on the same input in the same process ($+3.45$ $[-4.78, +11.68]$, post hoc)—the comparison this table's 24 GB cap forbids. That arm is itself flat between the same endpoints ($+0.00$), so the benchmark offers no frame headroom to gain here (Appendix B).

Resident-session capacity. Peak during prefill and resident state after it are different quantities, and the second is what a card holding several prefilled conversations pays for. Table 3 retains every session's cache until an allocation fails. Per-session resident state is 5.7 to 7.7 times smaller in every completed cell, short of the $8\times$ element budget because the resident state also carries the verbatim text cache and keeps the channel basis and the Gram matrix in float32. **The session counts follow: 1 against 9 on a 24 GB card at 128 frames**, 4 against 32 on 48 GB at 256. Per-session state is measured in the steady state: the first session's delta is the allocator reclaiming the prefill workspace, large and negative for the compressed arm and small for the control, so averaging it in would flatter us. This is a single-process proxy, not a serving benchmark, so the number to quote is the per-session resident state.

A second family. InternVL2.5-8B gives the same shape (Table 7, appendix): dense prefill fails at 64 frames on 24 GB where LoomKV completes at 18.7, the peak sits below the exact control everywhere, and capacity is 1 against 10.

7 Latency of the bounded-memory construction

Latency is what the construction costs where the alternatives exist. Both references are timed at 64 frames, the largest span on which they run under this paper's cap. Table 9 in the appendix times each stage alone, with CUDA synchronisation, over five videos and three repetitions. Both sides are

Table 3: Resident-session capacity: prefilled sessions retained before an allocation fails, and steady-state resident state per session. A "+" marks a cell that reached the script's own session limit without failing, so both arms are lower bounds there.

Model	Cap (GB)	Frames	chunked exact		**LoomKV**		ratio
			sessions	GB each	sessions	GB each	GB each
Qwen2-VL	24	128	1	2.016	9	0.355	5.7×
Qwen2-VL	24	256	0	–	0	–	–
Qwen2-VL	48	128	10	2.649	64+	0.355	7.5×
Qwen2-VL	48	256	4	5.292	32	0.684	7.7×
InternVL2.5	24	64	1	2.261	10	0.396	5.7×
InternVL2.5	48	64	11	2.261	24+	0.396	5.7×
InternVL2.5	48	128	6	4.467	24+	0.680	6.6×

given their fastest decomposition route, which matters: taking the top-r right singular vectors from the Gram eigendecomposition drops the offline method's single large decomposition from 6.75 to 0.77 seconds and our 112 small refreshes from 8.5 to 1.9. The route therefore helps the baseline far more than it helps us, and we report the comparison after giving it to both. **LoomKV at refresh-4 reaches a ready cache in $1.51\times$ the offline method's time** and $1.56\times$ InfiniPot-V's. That is the price of never materializing the dense cache, paid where the alternative can still be built; we make no speed claim.

Where the cost sits. Splitting the construction gives 2.7 seconds of block-wise forward overhead, which any chunked prefill pays and the exact control pays identically, 1.9 seconds of refreshes and 0.2 of history reconstruction; the refresh cost is the schedule's, and batching each group's decompositions reproduces the fold exactly without being faster at these shapes.

Does decoding give the saving back? No. Table 12 in the appendix generates 32 greedy tokens on twelve questions after the same construction. Reconstructing one layer's history per step, which is xKV's default dense-reconstruction decode path, costs 30% more per token than a full dense cache while reading a resident state $8\times$ smaller. Attention on the factors without reconstruction costs a further 28% and buys a transient below the dense cache's own, 11.2 against 23.8 MB, while still reading that $8\times$ smaller state. All paths gave the same answers on the twelve questions, which bounds the answer-change rate only below about 22%, so the agreement is weak evidence; the distinction that needs no sample is structural, since attention on the factors reads the same compressed state as dense reconstruction and is exact relative to it, the two differing only by a reassociation of matrix products, verified to 9×10^{-7} against a wrong-basis control.

Selective Reconstruction, the closer baseline. xKV's Selective Reconstruction scores the query against per-chunk landmarks and reconstructs only the top-k chunks; on our cache a landmark costs r values per chunk. At our shapes it delivers on memory and not on time: reconstructing 8.9% of the history cuts the transient from 94.7 to 39.2 MB, but no setting beats dense reconstruction on time (58.9 to 65.4 ms against 41.8), because the reconstruction it avoids is one bfloat16 matmul. It changed no answer on these twelve questions. This is bounded to our shapes and does not refute the $3.5\times$ they report at 122K text tokens, where the avoided work grows with context and the selection overhead does not.

8 What carries over online: cross-layer width

The construction forces four choices an offline factorization does not have: group width, refresh frequency, the transport rule, and later blocks seeing a compressed history. We varied all four.

Group width is the factor that matters, as it does offline. That accuracy improves with sharing width and then saturates is xKV's result, obtained offline on text at matched average rank per layer Chang et al. (2026); $W{=}4$ is their default and we froze it before any of this data existed. The question is whether it survives later blocks seeing a compressed history, and what it costs in time. At a matched budget, accuracy and prefill time flatten at the same width (Table 11, appendix). Giving each

layer its own basis is a double loss, 10.53 points $[-15.52, -5.53]$ and 76% more prefill time, because a layer alone affords rank $(50, 75)$ where four layers pooling afford $(188, 282)$ that all four use, and because the card prefers seven large decompositions to twenty-eight small ones. Two mechanisms, reported as two: the accuracy plateau belongs to the representation, the time plateau to hardware utilization on this card. At $W{=}1$ the folds do 33% *fewer* arithmetic operations than at $W{=}4$ yet cost $4.5\times$ the wall time; a better batched solver would shrink the timing half only. Two-layer groups lose 3.29 points $[-6.58, -0.00]$, directional at $p{=}0.076$. Widening from four to seven moves neither axis, $+0.33$ $[-2.48, +3.14]$ and $0.99\times$ the time. **So** $W{=}4$ **is where both axes stop moving** on this benchmark and backbone.

Causal feedback. A middle rung, the same factorization built incrementally from the *true* keys and values, separates block-wise construction from feedback. On 295 questions the first costs -1.69 $[-4.27, +0.88]$ and the second $+0.34$ $[-2.23, +2.91]$, totalling -1.36 $[-3.46, +0.75]$ with 95.3% of answers identical to one shot; neither is detectable here, which is not the same as costing zero.

Refresh frequency. Section 5 gives the sweep; a cheap trigger does not beat a fixed schedule because drift is smooth (Figure 2, appendix; a probe over 21 videos on plain dense prefill, so it is a property of the data). The residual against a frozen basis rises monotonically and never jumps, consecutive blocks sharing a channel subspace at a principal cosine of 0.78 to 0.82, so there is no event for a threshold to catch. Nor can one threshold serve both sides: at $\tau{=}0.02$ it decomposes on every value block, 32 of 32, while a fixed schedule of eight per group is cheaper and more accurate, and keys are not uniformly stable either, group 3's firing 30 times.

Coefficient transport. Old coefficients should be projected; the loss the alternatives target is not present. Repeated projection does not shrink old tokens: the oldest quarter of the history, transported through every refresh, sits at the same stored-to-true norm ratio as the newest quarter, transported through none (0.9419 against 0.9382 on the worst slab). Orthogonal Procrustes raises the error on every slab, as the projection bound predicts, and norm restoration scores -0.34 $[-1.82, +1.14]$ against projection.

Three further controls are in Appendix M: a reconstruction-space criterion for the schedule, fixed in advance, that picks a setting the sweep cannot distinguish from the others; a nineteen-question pilot of a verbatim recent window; and per-row int8 coefficients at $12.7\times$ byte-level compression, which stay an ablation.

9 Conclusion and limitations

LoomKV targets the regime where offline cross-layer factorization cannot form its own input: the carried Gram matrix makes the same representation causally constructible without materializing the completed prompt cache, and Sections 5 to 8 measure what that buys and what it costs. The costs follow.

Latency is a cost where the alternatives exist: a ready cache takes $1.51\times$ the offline method's time and $1.56\times$ InfiniPot-V's, each on its fastest route, and nothing here is a speed result. **Accuracy against the published references is measured where the dense cache fits**, because they need it; beyond it we have one pre-registered probe at $n{=}58$, which resolves neither a length-induced degradation nor a gap to an uncompressed arm, on a benchmark with no frame headroom over that range. **Memory covers two families**: InternVL's accuracy replication uses a shim without the bounded-memory property and its memory numbers come from a separate implementation, and Qwen2.5-VL contributes accuracy only. **Capacity is a single-process proxy** with no decoding, batching or scheduling; nine resident sessions at 1.3 to $1.7\times$ per-token decode cost is not a throughput statement. **Width was varied on one benchmark. The negative results are underpowered rather than settled**: the refresh question needs about $n{=}3300$ against our power of 0.15. **The Selective Reconstruction comparison is at our shapes**, and **StreamKV was not run**. **Scope**: two video benchmarks, three families of 7–8B parameters, greedy decoding, and **a compression sweep** of three ratios on one benchmark and backbone whose InfiniPot-V arm was not run.

REPRODUCIBILITY STATEMENT

Every number in this paper is produced by a script from a record file written when its run finished. The records, the pre-registrations and the scripts that read them are released together, with a map from each claim to the file it came from. Each experiment was pre-registered before the data existed, naming the arms, the sample, the statistic and the decision rule; the one post-hoc contrast is labelled where it appears (Appendix B), and the appendix lists the claims we withdrew when a rule or a later measurement went against us. Budget parity is enforced in elements rather than bytes and the measured fractions are reported. Three gates run before every arm and each has failed at least once on a real bug. Comparisons are paired within one card, because we measured that the card changes about 2.5% of predictions on an uncompressed forward pass, and the three comparisons that cross cards are named. The code, the pre-registrations and the record files will be released.

USE OF AI ASSISTANCE

An AI assistant (Claude, Anthropic) was used throughout this work under the authors' direction: to implement and debug the experiment harness and the cache construction, to write the scripts that generate every table from the committed run records, to run pre-registered analyses on cached tensors, to check claims in this manuscript against those records, and to draft and edit prose, including this section. All experimental designs, pre-registration rules, verdicts and decisions about what the paper claims were made by the authors, who read and are responsible for every sentence and every number. No experimental result was produced or altered by an AI system without a committed record.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Matthew Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20–30, 2006.

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: KV-cache compression with low-rank projection. In *International Conference on Learning Representations (ICLR)*, 2025.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Hung-Yueh Chiang, Yash Akhauri, Xilai Dai, Huiqiang Jiang, Yucheng Li, Luis Ceze, Kai-Chiang Wu, and Mohamed S. Abdelfattah. xKV: Cross-layer KV-cache compression via aligned singular vector extraction. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2503.18893.

Yilong Chen, Xiang Bai, Zhibin Wang, Chengyu Bai, Yuhan Dai, Ming Lu, and Shanghang Zhang. StreamKV: Streaming video question-answering with segment-based KV cache retrieval and compression. In *AAAI Conference on Artificial Intelligence*, 2026. arXiv:2511.07278.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-MME: The first-ever comprehensive evaluation benchmark of multi-modal LLMs in video analysis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. arXiv:2405.21075.

Mina Ghashami, Edo Liberty, Jeff M. Phillips, and David P. Woodruff. Frequent directions: Simple and deterministic matrix sketching. *SIAM Journal on Computing*, 45(5):1762–1792, 2016.

Minsoo Kim, Kyuhong Shim, Jungwook Choi, and Simyung Chang. InfiniPot-V: Memory-constrained KV cache compression for streaming video understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2506.15745.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, 2023.

Edo Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 581–588, 2013.

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. In *International Conference on Machine Learning (ICML)*, 2024.

Alireza Nadali, Patrick Cooper, Ashutosh Trivedi, and Alvaro Velasquez. KV-Fold: One-step KV-cache recurrence for long-context inference, 2026. arXiv:2605.12471.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, 1982.

Peter H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

Yang Shi, Huanqian Wang, Wulin Xie, Huanyao Zhang, Lijie Zhao, Yi-Fan Zhang, Xinfeng Li, Chaoyou Fu, Zhuoer Wen, Wenting Liu, Zhuoran Zhang, Xinlong Chen, Bohan Zeng, Sihan Yang, Yushuo Guan, Zhang Zhang, Liang Wang, Haoxuan Li, Zhouchen Lin, Yuanxing Zhang, Pengfei Wan, Haotian Wang, and Wenjing Yang. MME-VideoOCR: Evaluating OCR-based capabilities of multimodal LLMs in video scenarios. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2505.21333.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution, 2024.

Zizhong Wang, Jieying Wang, Zhao Zhang, and Jiajia Li. PuzzleKV: Page-wise low-rank decomposition for KV cache compression, 2026. arXiv:2608.23843.

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

Yangwen Zhang. An improved incremental singular value decomposition and new error bounds. *arXiv preprint arXiv:2204.05398*, 2022.

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. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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. arXiv:2509.21623.

Appendix roadmap. A is the control that chose the representation, and answers why the cache is factorized across layers rather than along the input grid. C is the three gates every run passes first. D–F are the memory grids and the gap to the ceiling in every setting. G is the two nuisance terms we measured and pair within, the card and the decomposition route. H–L are time, refresh, width and decoding. M holds the controls that did not fit the main text, including two that went against us.

A WHICH STRUCTURE SHOULD THE CACHE PRESERVE? GRID AGAINST CROSS-LAYER

Table 5 is the control that chose this paper's representation, on this paper's benchmark. Every arm holds the same answer-time cache of $2{,}880$ token-equivalents on MME-VideoOCR at $S{=}720$ with Qwen2-VL-7B, $n{=}296$, the same question set and skips as the rest of this paper. The paired differences are the record's own, taken against the Tucker cell; per-question rows for that arm were not re-scored here, so no new pairing is claimed and the table is generated from the verdict record by script. Two readings are fixed by it: a per-input factorization of the token axis at rank 188 keeps what a low-dimensional spatial subspace discards, and spending the channel rank to widen that subspace makes it worse, so the gap is the spatial subspace and not the allocation. It is also why the width result of Section 8 is the offline finding surviving causalization rather than a discovery of this paper.

The same ordering holds on the setting where the grid design was first evaluated: pooled over six real-video MVBench tasks at 64 frames, $n{=}1{,}186$ per backbone, the Tucker cell trails the cross-layer factorization by 3.79 points $[-5.72, -1.85]$ ($p{=}0.0002$) on Qwen2-VL-7B and by 5.06 $[-7.09, -3.03]$ ($p{<}10^{-4}$) on Qwen2.5-VL-7B, video-clustered intervals, exact McNemar; even the per-layer factorization without sharing is ahead there.

Nor does the grid pay for itself a second time. LoomKV's coefficient matrix A is itself a $T \times S$ grid of r-dimensional coefficients, and A is where the bytes are (nr against Wdr for the basis, about $11{:}1$ here). Table 4 factorizes it spatially with one basis $U_S \in \mathbb{R}^{S \times r_s}$ shared across frames and a core per frame, spending every saved coefficient byte on a larger cross-layer rank r' so that total bytes match plain LoomKV to 0.1%, and reads attention-output error against dense attention on the same cached tensors, six questions and two layer groups, keys and values separately. The learned spatial basis beats a random one in every cell, so the field has spatial structure; at every spatial rank tried that structure is worth less than the rank it costs, 0 of 12 cells on either side at $4\times$ spatial compression and still worse at $2\times$. Together with Table 5, spatial structure loses in both orders at these shapes: the grid does not pay, in either direction, and that is a finding rather than a design choice. Pre-registered with an identity control at $r_s{=}S$.

Table 4: Spatial Tucker factorization of the coefficient field at matched total bytes, attention-output error against dense attention, median over 12 cells per row. Generated by script from the gate's cells.

side	r_s ($S{=}720$)	r' at matched bytes	plain LoomKV	Tucker-LoomKV	ratio	Tucker-LoomKV wins	random U_S
K	90	943	0.1515	0.4364	2.743	0/12	1.0050
K	180	587	0.1515	0.3138	2.045	0/12	1.0499
K	360	328	0.1515	0.1635	1.030	4/12	1.1005
V	90	1422	0.0884	0.4036	4.666	0/12	0.8779
V	180	889	0.0884	0.2860	3.219	0/12	0.7686
V	360	502	0.0884	0.1453	1.644	0/12	0.5365

Table 5: Structure selection at one answer-time cache: the grid-preserving three-mode Tucker factorization against cross-layer factorizations of the same cache. Generated by script from the verdict record.

arm, all at 2880 token-equivalents, $n{=}296$	accuracy	Tucker − arm [95% CI]	McNemar
dense (no compression)	60.14	–	–
three-mode Tucker, per input ($r_s{=}146$), the predecessor	47.97	–	–
InfiniPot-V (eviction)	48.65	-0.68 $[-5.76, +4.26]$	0.897
cross-layer factorization, $W{=}4$ ($r_k{=}188$)	59.12	-11.15 $[-15.64, -6.73]$	$< 10^{-3}$
cross-layer factorization, $W{=}1$ ($r_k{=}50$)	48.31	-0.34 $[-4.73, +4.10]$	1.000
Tucker, spatial subspace widened ($r_d{=}30$, $r_s{=}282$)	45.61	$+2.36$ $[-1.37, +6.23]$	0.281

B FRAMES AND ACCURACY IN THE REGIME THE CONSTRUCTION OPENS

Qwen2-VL-7B, Video-MME long, native resolution, LoomKV at refresh-4 and the same $\rho = 0.125$ budget, ranks recomputed per input from the token count; all three arms ran on one card under one decomposition route at 0.1248 to 0.1250 of the dense visual cache. The 58 questions are those present in all three arms and are a contiguous prefix of the shared question order: the 224-frame arm was stopped by the clock, not by any question failing, so the subset is not selected. The uncompressed arm is measured on the same input inside the same process, so no hardware term applies to that column; it needs more than 24 GB at 224 frames and appears here only because the probe ran on the full 48 GB card. This is a different benchmark and setting from Table 1 and the accuracies are not comparable to it. The last column repeats because the two arms differ on two questions that cancel, so their discordant counts against 64 are identical. The 224-against-64 contrast was pre-registered before the data existed, with a reporting bar of $n \geq 40$ and a rule that the result is reported whichever way it goes; the contrast against the uncompressed arm was added afterwards on the same records and is post hoc.

frames	visual tokens	LoomKV	uncompressed	LoomKV − uncompressed	LoomKV, vs. 64
64	8,448–23,040	46.55	46.55	$+0.00$ $[-8.28, +8.28]$	—
128	16,896–46,080	50.00	51.72	-1.72 $[-7.56, +4.11]$	$+3.45$ $[-7.20, +14.10]$
224	29,568–80,640	50.00	46.55	$+3.45$ $[-4.78, +11.68]$	$+3.45$ $[-7.20, +14.10]$

C THE GATES

Every arm passes three checks before it is scored, and each has failed at least once on a real bug. *Splice*: predictions obtained through the evaluation harness and through the embedding path must be identical, which catches a mismatched visual span. *Equivalence*: the block-wise construction with compression disabled must reproduce one-shot prefill; it does, at a maximum absolute logit difference of 0.25, and two earlier versions failed it, once on a causal mask sized from an empty cache and once on a dimension mismatch in the exact-history path. *Entry ticket*: the dense arm must reproduce the archived dense run on at least 95% of paired questions, which catches a changed decode path or loader order before any comparison is read. The three checks are run before every arm. An early pilot at n=19 had two algebraically equivalent backends disagree on exactly the three smallest-margin questions, a noise floor of 15.8 points, and every verdict from it was withdrawn; a directional reading of the causal-feedback split from a 95-question pilot did not survive the full 295 and was withdrawn too.

D PEAK MEMORY

Peak allocator memory during prefill across caps and frame counts; the 24 GB block is Table 2 in the main text. A cell marked OOM failed; "–" was not run because a smaller cell had already failed.

E SECOND-FAMILY MEMORY

The same grid on InternVL2.5-8B, whose language model, visual front end and position encoding all differ from the Qwen pair. The shape repeats: dense fails where LoomKV completes, and the peak sits below the uncompressed chunked control everywhere.

F ACROSS SETTINGS

The gap to the unbounded-memory ceiling in every setting we ran it, so that the two settings where the interval excludes zero are read beside the two where it does not, rather than pooled.

Table 6: Peak allocated memory in GB, Qwen2-VL-7B on Video-MME long videos at native resolution, A6000 with the cap enforced per process. **OOM** is a failure at that cell; "–" is a cell not run because a longer or shorter one settles it; "1/2 ok" means one of two videos completed. All four arms use the same chunked vision path.

Cap (GB)	Frames	Tokens	dense	offline xKV	chunked exact	**LoomKV**
24	64	23040	22.4	23.2	19.7	18.8
24	128	46080	**OOM**	**OOM**	22.6	20.6
24	160	57600	–	–	**OOM**	21.6
24	192	69120	–	–	–	22.4
24	224	80640	–	–	–	23.4
24	256	92160	–	–	**OOM**	**OOM**
48	64	23040	23.9	23.9	19.7	18.8
48	128	46080	29.7	30.4	22.6	20.6
48	256	92160	42.7	44.2	28.6	24.3
48	512	107520	1/2 ok	**OOM**	29.2	24.0
48	768	114816	**OOM**	–	29.5	23.9

Table 7: Peak allocated memory in GB, InternVL2.5-8B, MME-VideoOCR at the model's own 448-tile recipe. Same protocol and same caps as Table 6.

Cap (GB)	Frames	Tokens	dense	chunked exact	**LoomKV**
24	64	16827	**OOM**	19.4	18.7
24	128	33659	–	22.4	21.2
24	192	50491	–	**OOM**	**OOM**
48	64	16827	27.1	20.0	19.2
48	128	33659	36.8	22.4	21.2
48	192	50491	1/2	25.4	23.8
48	256	67323	**OOM**	28.3	26.3

G THE HARDWARE TERM

Decomposition route. `top_right_vectors` has two routes, `gesvd` and the Gram eigendecomposition (`eigh`); they return the same subspace and not bitwise the same vectors. The deployed configuration run under both on the same 306 questions differs by -0.33 points $[-1.76, +1.11]$, McNemar p=1.000, with 6 of 306 predictions changing, the same size as the cross-card term above. Every within-sweep comparison in this paper is therefore made within one route: the compression sweep and the width table use the eigh every-four arm, and the refresh table pairs each row against the every-four arm run under that row's route. The main table keeps its own arms. Which GPU a run used changes a small fraction of its predictions, and we measured it because our own arms were executed on two machines. On an *uncompressed* dense forward pass, with identical code, weights, inputs and decoding, the two cards disagree with each other on 8 of 295 questions in one sample and 5 of 198 in another, about 2.5%. A third, smaller check on the compressed arm agrees at 1 of 24. Different kernels accumulate in a different order and a multiple-choice answer decided by a small logit margin lands either way; nothing about compression is involved, which is why we measure it on the dense arm.

The consequence for this paper is that a paired difference computed across the two machines carries roughly ± 1 point of extra noise. Three comparisons in this paper cross cards, and we list them rather than claim there are none. LoomKV against streaming eviction and against per-layer low-rank, both $+7.46$: the point estimates are several times the hardware term, though the eviction margin's lower bound $+2.23$ is only about twice it, so read that interval as wider than printed. LoomKV against the unbounded-memory ceiling, -3.39 with 18 answers lost and 8 gained: 26 discordant answers against a hardware term of about eight, so this one is the least safe of the three and Section 5 says so at the point of use. The group width sweep and the refresh sweep were originally cross-card and have both been re-run on one card for this reason, which is why their tables carry different sample sizes from Table 1. The consequence for the field is larger: differences of one to three points on a few hundred

Table 8: LoomKV across four settings. Δ to ceiling is against offline xKV $W{=}4$, which requires the dense cache and is the unbounded-memory reference rather than a competitor under this paper's memory cap. Intervals are paired 95% Wald.

Setting	n	Δ to ceiling	Δ to dense	Δ to eviction
Qwen2-VL, MME-VideoOCR	295	-3.39 [-6.78, -0.00]	-2.71 [-6.59, +1.16]	+7.46 [+2.23, +12.69]
Qwen2.5-VL, MME-VideoOCR	295	+0.68 [-3.63, +4.98]	-3.73 [-7.98, +0.53]	+13.56 [+7.54, +19.58]
Qwen2-VL, Video-MME long	298	+1.68 [-1.86, +5.22]	+0.00 [-3.60, +3.60]	–
InternVL2.5, MME-VideoOCR	394	-3.55 [-6.78, -0.33]	-5.84 [-9.59, -2.08]	–

questions are reported routinely, and a difference of that size between arms run on different hardware is not safely attributable to the method.

H TIME TO A READY CACHE

Why our decomposition share differs from xKV's. They report 3.9% of prefill at 128K text tokens; here it is 18% at 23K visual tokens on the Gram route. Both are consistent: the decomposition grows about linearly with length while causal attention grows superlinearly, so its share falls with context.

Table 9: Time to a ready cache, Qwen2-VL-7B, 64 frames (23,040 visual tokens), A6000, each stage timed alone with CUDA synchronisation over five videos and three repetitions. Video decoding is excluded because it is common to every method. InfiniPot-V is timed in its own harness, which includes answer decoding.

Path to a ready cache (vision + LM)	Seconds	× offline xKV
dense prefill	6.97	0.90
offline xKV (prefill + one-shot SVD)	7.74	1.00
LoomKV, refresh 4	11.72	1.51
LoomKV, refresh 2	13.37	1.73
InfiniPot-V (own timing, includes decode)	7.50	0.97

I THE REFRESH SCHEDULE

The refresh sweep within one card. Each row is paired against the every-four arm run under its own decomposition route, and both every-four arms are printed, because the route is worth 0.33 points (Appendix G) and a cross-route pairing would spend that on the comparison.

Table 10: The refresh schedule, every arm on one card, 306 common questions, $W{=}4$. Prefill seconds are isolated and CUDA-synchronised, five videos by three repetitions; an uncompressed dense prefill is 4.31 seconds on the same measurement. The every-8 row was timed but not scored, added after the accuracy sweep to see how far the time curve keeps falling once the accuracy curve is known to be flat.

Refresh	Decomps. / group	Accuracy	Δ to every-4, same route	p	Prefill (s)
every block (eigh)	224	57.19	+1.31 [−1.56, +4.17]	0.503	13.57
every 2 blocks (gesvd)	112	57.22	+0.76 [−1.51, +3.03]	0.664	10.71
every 4 blocks (eigh)	56	**55.88**	–	–	**9.05**
every 4 blocks (gesvd)	56	56.46	–	–	9.05
every 8 blocks	28	not run	–	–	8.18

J GROUP WIDTH

The width ablation behind Section 8. $W{=}4$ was frozen from xKV's default before these runs and is kept whatever the sweep says; the sweep is reported because the question is whether the offline width behaviour survives causalization, not which width to adopt.

Table 11: Group width at a matched budget, MME-VideoOCR, Qwen2-VL-7B, the 304 questions common to every width, one card, ranks recomputed per width so every row spends the same elements; paired against $W{=}4$ with a 95% Wald interval and an exact McNemar p. The $W{=}4$ row is the deployed configuration (bfloat16 factors, the eigh route) on the 304 common questions; Table 1's refresh-4 arm is the same construction with float32 factors on 295, and the two differ by a dtype term, $+0.68$ $[-1.81, +3.16]$ for bfloat16 against float32, and a route term, -0.33 $[-1.76, +1.11]$ (Appendix G).

W	Groups	(r_K, r_V)	Budget	Accuracy	Δ to $W{=}4$	p	Prefill (s)
1	28	(50, 75)	0.1248	45.39	-10.53 $[-15.52, -5.53]$	0.000	15.97
2	14	(98, 147)	0.1249	52.63	-3.29 $[-6.58, -0.00]$	0.076	9.84
4	7	(188, 282)	0.1249	**55.92**	–	–	**9.08**
7	4	(310, 465)	0.1249	56.25	$+0.33$ $[-2.48, +3.14]$	1.000	8.99

K THE REFRESH-TRIGGER PROBE

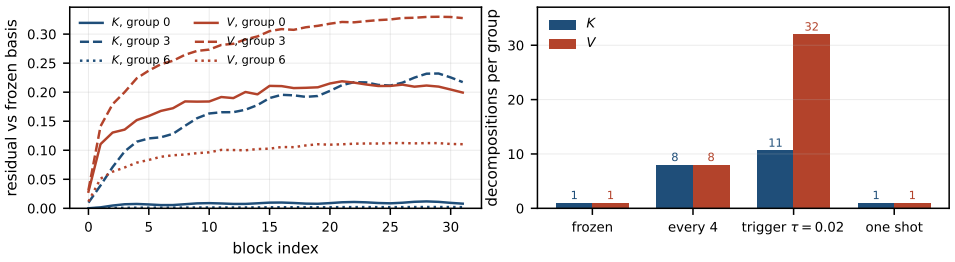


Figure 2: Drift of real visual KV against a frozen channel basis: 21 videos, plain dense prefill with no compression in the model path, 32 blocks of 720 tokens. **Left**: the residual a threshold would trigger on rises smoothly and never jumps. **Right**: decompositions each policy then performs, averaged over groups and videos.

Over 3,906 candidate refresh events on 21 videos, measured on plain dense prefill so the statistic is a property of the data, the cheap residual against the frozen basis correlates with the reconstruction-level benefit of refreshing at Spearman $+0.832$ on keys and $+0.366$ on values, and within (side, group) at $+0.593$, $+0.558$, $+0.366$ on keys and $+0.300$, $+0.368$, $+0.254$ on values, so one value group sits below the pre-registered bar of 0.30. Against distance to the every-block basis, which is what a scheduler wants to minimise, the same residual correlates at -0.03 on keys and -0.14 on values. We report this as licensing a future answer-level test of a budgeted scheduler, not as a result.

L EVERY DECODE SETTING

Selective Reconstruction under two gather strategies. *Per-head* follows the description literally and gathers each key–value head's chosen chunks separately. *Union* gathers the union of the heads' chunks once and reconstructs it for every head, which is a strictly more inclusive approximation; we report both because the per-head version is the one that looks worse and reporting only it would be a strawman. The four key–value heads disagree enough that the union of their top-16 chunks is 22.1% of rows against 8.9% per head, so the union pays the gather once and then attends over a much longer segment.

Table 12: All decode paths, 12 questions, 32 greedy tokens. "Same answer" is agreement with row (b) on the answer token; exact 32-token sequence match is $1/12$ to $7/12$ across these rows and differs only in the length of a run of end-of-turn padding.

Decode path	ms / token	Peak (MB)	History read	Same answer as (b)
dense cache	32.2	23.8	–	–
LoomKV, dense reconstruction (xKV's default)	41.8	94.7	–	–
SR top-16, per-head gather	58.9	39.2	8.9%	12/12
SR top-32, per-head gather	59.3	67.9	17.8%	12/12
SR top-64, per-head gather	59.1	125.4	35.6%	12/12
SR top-16, union gather	64.9	64.6	22.1%	12/12
SR top-32, union gather	65.2	111.1	38.8%	12/12
SR top-64, union gather	65.4	165.6	63.8%	12/12
LoomKV, attention on the factors	53.5	11.2	–	12/12

M TWO CONTROLS IN FULL

Two further controls. A budget-matched nineteen-question pilot of a verbatim recent window is suggestive against the idea that the loss comes from compressing tokens too early, and is not a verdict at that size. Per-row `int8` coefficients raise byte-level compression from $8.0\times$ to $12.7\times$ at -0.68 $[-2.55, +1.20]$, missing a pre-registered -2.0 bound, so it stays an ablation.

A reconstruction-space criterion does not predict downstream behaviour. A pre-registered probe over 3,906 refresh events finds the cheap residual correlates with the reconstruction-level benefit of refreshing yet is nearly uncorrelated with distance to the every-block basis, which is what a scheduler wants (Appendix K). Freezing the key basis, which that residual calls almost free, moves accuracy by -2.37 $[-6.30, +1.56]$, which we also cannot resolve.

A recent verbatim window does not help at a matched budget. Keeping the most recent block in full precision beside the compressed history, and paying for it in rank, is the standard hybrid. On a budget-matched pilot every arm spent between 99.96% and 100.03% of the same cache and a 720-token window changed no prediction at all, while doubling it to $1{,}440$ tokens and cutting rank by 47% flipped three and lost one. The pilot is $n{=}19$, below this paper's noise floor, so we give no accuracy verdict; what it does rule out is the mechanism, because an arm that changes the per-question probability of the correct answer by up to 0.221 and changes no answer is not being limited by when compression happens.

Quantizing the coefficients: an ablation. The coefficient matrix can be stored as per-row `int8` with float32 scales while the basis and the Gram matrix stay in float32. Element counts are unchanged, so budget parity with every other arm holds, and the byte-level compression rises from $8.0\times$ to $12.7\times$. Across 296 paired questions the difference from bfloat16 storage is -0.68 points $[-2.55, +1.20]$, two questions, with 287 of 296 predictions identical. The pre-registration required the lower bound to clear -2.0 and it does not, so this is an ablation row and not a main result; the capacity numbers in Section 6 are the bfloat16 ones and are not restated in bytes.

Table 13: Accuracy against compression ratio, pre-registered before any run. Every arm is the same A6000 box and all six are paired on one question set of $n{=}296$, so G is comparable across ratios. Ranks come from the causal rank rule at each ρ; nothing is hand-set per ratio.

ratio	stored/dense	LoomKV	offline xKV	G = LoomKV − offline	McNemar
$4\times$	0.2499	59.12	59.46	-0.34 $[-3.07, +2.39]$	1.000
$8\times$	0.1249	56.08	59.12	-3.04 $[-6.35, +0.27]$	0.108
$16\times$	0.0625	47.30	52.70	-5.41 $[-9.90, -0.91]$	0.026
dense	1.0000	60.14			