AttSVD: Prompt-Adaptive Low-Rank KV Cache Compression via Attention-Guided SVD

Anonymous authors
Paper under double-blind review

Abstract

The key-value (KV) cache of autoregressive transformers grows linearly with context length and dominates memory at long context. Most training-free remedies evict low-importance tokens, an irreversible choice along the sequence axis. We instead keep every token and store it more cheaply along the "*feature*" axis. We therefore propose AttSVD, a new *interpretable* low-rank compression whose basis is derived from each prompt's own attention geometry: an online, per-prompt truncated SVD that keeps only the directions attention actually reads, cutting persistent per-head KV memory in proportion to the retained rank. We propose two decode-time caching strategies, accumulating and streaming, for short and long generation regimes. Furthermore, we propose two refinements that make compression adaptive. A per-matrix energy rule sizes the logit space and the attention mass independently. An attention-aware basis truncates only in the spaces attention actually reads, preserving both the attention logits and the attention output. The same factors also provide free, per-head interpretability insights into the effective rank and the geometry attention consumes. Across multiple models, on both an agentic benchmark and the full LongBench suite AttSVD stays on par with the dense cache while using up to $\sim 50\%$ of the KV-cache memory.

1 Introduction

Training-free KV-cache compression methods fall into two families. *Eviction* methods (e.g. H2O (Zhang et al., 2023), SnapKV (Li et al., 2024), KeyDiff (Park et al., 2025), KVzip (Kim et al., 2025)) keep a budget of tokens and drop the rest entirely; they differ only in how they score importance. Eviction is a hard, irreversible choice along the sequence axis: a wrongly dropped token can never be attended to again. *Reconstruction* methods (e.g. VECTOR (Lin et al., 2026), Palu (Chang et al., 2025)) instead keep all positions but store them approximately.

We propose AttSVD, which is not only a feature-axis compression method but also introduces two decode-time caching strategies for different generation regimes. The core rationale builds on the observed fact that a prompt's keys and values are effectively low-rank, the keys especially (Saxena et al., 2024; Sun et al., 2025); we extend this low-rank observation from the K and V spaces to the attention logit space and the attention mass space. AttSVD computes an online attention-aware factorization in the subspace attention actually reads (rather than by plain spectral energy), per layer and per key/value head, both at prefill (factoring the prompt bulk) and during decode (the streaming scenario), where generated tokens are factored once they age out of a short recent-token window so long generations stay compressed as well (Figure 1). Because the basis is the SVD of the prompt's own activations, it is, by Eckart-Young (Eckart & Young, 1936), the best possible rank-r approximation of that prompt's attention logits and attention mass, obtained with no calibration corpus and no training; no token is discarded, and every position is still attended to, just from reconstructed rank-r factors. Our contributions are:

- A per-prompt, calibration-free low-rank KV cache that retains all token positions and is exact at full rank, with a dynamic per-matrix rank rule that sizes keys and values separately so memory adapts to each prompt instead of a fixed schedule;
- An attention-aware basis that truncates in the subspace attention actually reads, preserving both the attention logits and the attention output;

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.

- Two decode-time caching strategies that both compress at prefill but differ at decode: an accumulating mode for long-prompt, short-decode scenarios (e.g., no-think mode) that keeps generated tokens exact, and a streaming mode for long-prompt, long-decode scenarios (e.g., think mode) that compresses aged-out tokens into thin blocks for a bounded cache.
- Interpretability at no extra cost: because the basis is derived from each prompt's own attention geometry, the retained per-matrix ranks and attention-mass profile also serve as per-head diagnostics, exposing each head's effective-rank compressibility, query-subspace focus, and token saliency without calibration.

2 Related Work

Most low-rank KV-cache methods extract the basis ahead of time and reuse it for every prompt. *Weight decompositions* factor the key/value projection *weights* offline (ASVD (Yuan et al., 2023), Palu (Chang et al., 2025), LoRC (Zhang et al., 2024)); *calibration* methods estimate the basis from a corpus (Eigen Attention (Saxena et al., 2024), ReCalKV (Yan et al., 2025), KQ-SVD (Lesens et al., 2026)); and *cross-layer or trained* methods share or learn it (xKV (Chang et al., 2026), MatryoshkaKV (Lin et al., 2025)). OjaKV (Zhu et al., 2026) is also online and per-context but *approximate* and *variance-greedy*. It keeps a running incremental-PCA basis seeded from a small calibration set, using attention only to pick which tokens stay full-rank rather than to shape its basis.

How `AttSVD` differs. Prior methods either extract or learn a *single* basis reused for every prompt (the offline families above), or track an *approximate* running basis seeded from calibration (OjaKV). `AttSVD` differs on three axes. *(i) Per-prompt and exact.* It computes the Eckart-Young-optimal truncated SVD of *this* prompt's keys and values at prefill, with no calibration corpus, no training, and no running-basis bookkeeping; the basis is then frozen during generation (unlike OjaKV's seeded, incremental estimate). *(ii) Attention-guided, not variance-greedy.* Rather than keeping the directions of largest K, V variance, it truncates in the space attention actually consumes; weighting keys by the query covariance $C_Q{=}Q^\top Q$ (which preserves the logits $QK^\top$) and values by per-token attention mass (which preserves the output PV), so the retained subspace is *functional* rather than merely descriptive. This is a distinct *form* of attention-awareness from KQ-SVD: rather than low-rank-approximating the *products* $KQ^\top$ and VW_O learned offline from calibration, `AttSVD` keeps the ordinary per-matrix factors and merely reweights their SVD metric, so one calibration-free rule aligns *both* keys (via C_Q, preserving the scores) and values (via attention mass, preserving the output) with *this* prompt's own attention. *(iii) Dynamic per-matrix rank.* A spectral-energy rule factors each matrix independently from the data so that memory adapts to each prompt instead of following a fixed schedule. For a comprehensive related work discussion, see Appendix A.

3 Method

Motivation: Existing KV-cache methods save memory by *evicting* tokens, permanently discarding their keys and values. We instead retain *every* token in a cheaper form, re-expanded transiently at inference time, with no eviction or training. Unlike other low-rank reconstruction methods, we aim at a low-rank compression that is *per-matrix adaptive*, *calibration-free*, *prompt-adaptive*, robust across *prefill & decode length scenarios*, and *attention-importance aware*.

3.1 Proposed AttSVD

Setup and notation. Let $K, V \in \mathbb{R}^{S\times d}$ denote the keys and values for a prompt of length S, where d is the head dimension. Standard attention computes $P = \mathrm{softmax}\left(QK^\top/\sqrt{d} + M\right)$ and outputs PV, where M is the causal mask and $Q \in \mathbb{R}^{S\times d}$ the queries. A single head stores $2Sd$ entries (K and V), for a total KV-cache footprint of $2SdLH$ across L layers and H key/value heads. In case of grouped-query attention (GQA), H is the number of *key/value* heads, typically a small multiple fewer than the number of query heads H_q; this matters for the attention-aware variant, where a key head is shared by several query heads. Throughout the paper, K denotes the

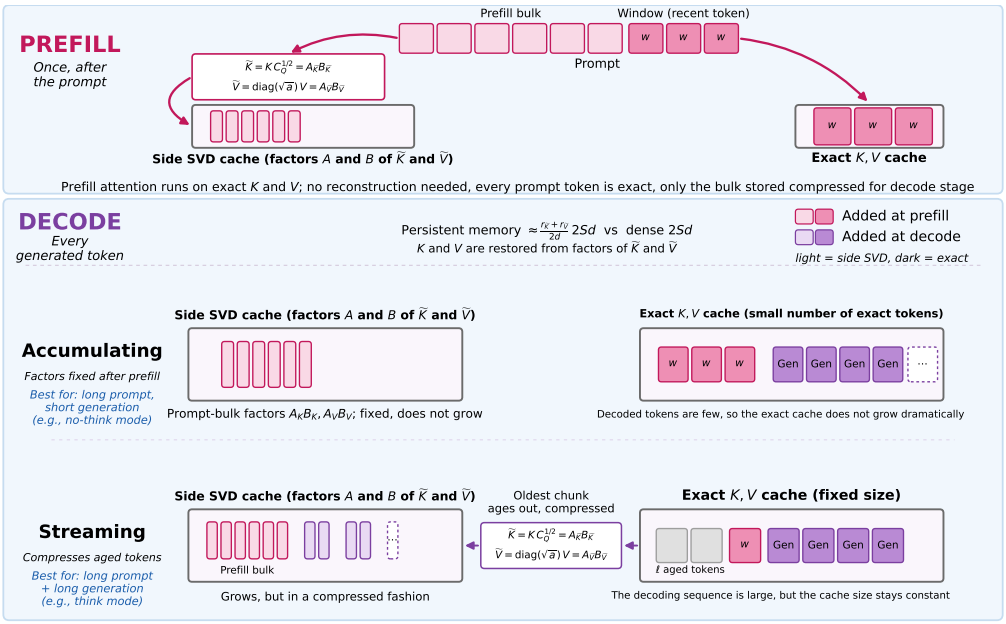


Figure 1: `AttSVD` keeps a thin, compressed side cache of attention-aware factors A, B plus a small exact cache of recent tokens, cutting persistent memory to a fraction of dense while retaining all token positions. At decode, *accumulating* keeps every token exact (small decoded tokens); *streaming* compresses tokens leaving a sliding window, keeping the exact buffer bounded.

post-RoPE keys as they sit in the cache,[1] so our attention-weighted factorization operates on the rotated keys; this differs from ShadowKV (Sun et al., 2025), which compresses the *pre-RoPE* keys because the position-dependent rotation spreads the keys across more singular directions, so the pre-RoPE keys are lower-rank and compress better, at the price of storing and replaying the rotation at reconstruction. This pre-RoPE trick does not apply to us, because we never run an SVD on raw K or V; we factor in the attention-logit and attention-mass geometry instead. The quantity our key basis must preserve is the logit inner product $QK^\top$, which is exact only when the keys carry the *same* rotation as the queries, so operating post-RoPE is required and a pre-RoPE key basis would be misaligned with the post-RoPE query metric that our attention-aware truncation uses. The modestly higher $r_{\widetilde{K}}$ this positional spreading induces is in turn absorbed by the per-prompt adaptive rank.

Standard SVD on weighted inputs. `AttSVD` introduces no new decomposition. It runs the *same* standard truncated SVD as SVD on raw KV, only on attention-weighted K and V rather than on K and V directly, so it is standard SVD on weighted inputs, not a separate weighted-SVD routine.

3.2 Per-prompt and attention-aware low-rank compression

In general, for an $S \times d$ matrix Z with $S > d$ the (full) singular value decomposition (SVD) is $Z = U\Sigma\mathcal{V}^\top$, where *at full rank* $U \in \mathbb{R}^{S\times d}$, $\Sigma = \mathrm{diag}(\sigma_1 \geq \cdots \geq \sigma_d \geq 0) \in \mathbb{R}^{d\times d}$, and $\mathcal{V} \in \mathbb{R}^{d\times d}$ (so Z has at most d nonzero singular values). The rank-r truncation ($r \leq d$)

$$Z_r \;=\; U_{:,1:r}\,\Sigma_{1:r}\,\mathcal{V}_{:,1:r}^\top \;=\; \operatorname*{arg\,min}_{\mathrm{rank}(X)\leq r} \|Z - X\|_F \tag{1}$$

is, by the Eckart-Young theorem, the *optimal* rank-r approximation of Z, with error $\|Z - Z_r\|_F = \left(\sum_{i>r}\sigma_i^2\right)^{1/2}$, where we store the *factors*

$$A \;=\; U_{:,1:r}\,\Sigma_{1:r} \in \mathbb{R}^{S\times r}, \qquad B \;=\; \mathcal{V}_{:,1:r}^\top \in \mathbb{R}^{r\times d}, \qquad \widetilde{Z} \;=\; AB. \tag{2}$$

[1] Rotary embeddings rotate Q and K but never V.

AttSVD is attention-aware, i.e. the SVD is taken in the score and output spaces: it preserves the attention logits $QK^\top$ by weighting the keys in the query geometry $C_Q = Q^\top Q$, and the output $O = PV$ by weighting the values by their per-key attention mass, changing only the basis, not the factor shapes. SVD on raw K and V instead minimizes $\|K - K_r\|_F$ and $\|V - V_r\|_F$, so it can spend rank on directions attention never queries; we truncate in the spaces attention uses, keeping the decomposed storage of Eq. (2) unchanged. *Either way the decomposition is the same standard truncated SVD; only its input differs, raw versus attention-weighted K and V.*

Weighted keys via the query geometry. Replacing K by K_r perturbs the logits by $Q(K - K_r)^\top/\sqrt{d}$, so the relevant objective is

$$K_r^\star = \operatorname*{arg\,min}_{\operatorname{rank}(X)\le r} \left\|Q(K-X)^\top\right\|_F = \operatorname*{arg\,min}_{\operatorname{rank}(X)\le r} \left\|(K-X)\,C_Q^{1/2}\right\|_F, \qquad C_Q = Q^\top Q, \tag{3}$$

the optimal rank-r approximation of K in the query-covariance metric C_Q (the two norms are equal since both are $\operatorname{tr}\big((K-X)\,C_Q\,(K-X)^\top\big)$). By the change of variables $\widetilde{K} = K\,C_Q^{1/2}$ the generalized Eckart-Young solution is the ordinary truncated SVD of $\widetilde{K}$ mapped back: factor $\widetilde{K} = U\Sigma\mathcal{V}^\top$ and store $A_{\widetilde{K}} = U_{:,1:r}\Sigma_{1:r}$ and $B_{\widetilde{K}} = \mathcal{V}_{:,1:r}^\top C_Q^{-1/2}$, so that $A_{\widetilde{K}}B_{\widetilde{K}} = K_r^\star$ in the original coordinates. Only the basis changes; the cache layout and decode reconstruction are identical.

Weighted values via the attention mass. Replacing V by V_r perturbs the output by $P(V - V_r)$. The exact minimizer of $\|P(V - V_r)\|_F$ needs the $S \times S$ Gram $P^\top P$; we use the diagonal (per-key mass) approximation $a_i = \sum_q P_{qi}$, the total attention received by key i, giving

$$V_r^\star = \operatorname*{arg\,min}_{\operatorname{rank}(X)\le r} \sum_{i=1}^{S} a_i\,\|V_i - X_i\|_2^2 = \operatorname*{arg\,min}_{\operatorname{rank}(X)\le r} \left\|\operatorname{diag}(\sqrt{a})\,(V-X)\right\|_F, \tag{4}$$

solved by a change of variables: form the row-weighted values $\widetilde{V} = \operatorname{diag}(\sqrt{a})\,V = U\Sigma\mathcal{V}^\top$, whose best rank-$r$ fit is the standard truncated SVD $\widetilde{V}_r = U_{:,1:r}\Sigma_{1:r}\mathcal{V}_{:,1:r}^\top$ (Eq. 1); un-weighting its rows by $\operatorname{diag}(1/\sqrt{a})$ recovers the store $A_{\widetilde{V}} = \operatorname{diag}(1/\sqrt{a})\,U_{:,1:r}\Sigma_{1:r}$ and $B_{\widetilde{V}} = \mathcal{V}_{:,1:r}^\top$, so that $A_{\widetilde{V}}B_{\widetilde{V}} = \operatorname{diag}(1/\sqrt{a})\,\widetilde{V}_r = V_r^\star$. In both cases the kept directions are exactly the ones attention reads; full proofs of both truncations are given in Appendix B.

Obtaining Q and a. Both are available at prefill: we capture the *post-RoPE* queries with lightweight forward hooks (a short observation window per layer) and read the value mass a from one attention-shaped pass, with GQA expansion and small numerical guards for the un-weighting. This composes with any rank rule of Section 3.4: the weighting sets the basis, the rule sets the rank.

3.3 Two KV-cache compression regimes tailored to prefill and decode length

AttSVD always compresses the prompt at *prefill*; the two regimes differ only in how the cache is maintained during *decode*. We first describe the shared prefill phase, then the two decode variants: an *accumulating* and a *streaming* cache.

Prefill. Given the prompt's $\widetilde{K}, \widetilde{V}$ for a layer, split off a small *recent window* of the newest w tokens, kept exact; call the remaining bulk $K^{(b)}, V^{(b)} \in \mathbb{R}^{(S-w)\times d}$. AttentionSVD each via Eq. (2) into $(A_{\widetilde{K}}, B_{\widetilde{K}})$ and $(A_{\widetilde{V}}, B_{\widetilde{V}})$ and store *only the factors* in a compressed side cache, writing just the exact tail to the exact cache. The prefill forward itself is served with the *exact* K, V that are already in hand; reconstruction at prefill would only inject a lossy step we do not need, so prefill output is bit-identical to the dense cache. The compressed factors are a lossy approximation, consulted only when those tokens are later reconstructed at decode.

Exact recent-window tail. The newest w tokens are stored exactly rather than decomposed. They carry the highest-impact local context, and a handful of tokens cannot be usefully compressed by a rank-r SVD (and a single decode token is appended one at a time, where factoring is counter-productive). Setting $w = 0$ recovers a fully decomposed prefill.

Decode. At each generation step the decomposed bulk is re-materialized transiently as $\widehat{K}^{(b)} = A_{\widetilde{K}} B_{\widetilde{K}} \in \mathbb{R}^{(S-w)\times d}$ and $\widehat{V}^{(b)} = A_{\widetilde{V}} B_{\widetilde{V}}$ (the stored B factors carry the reverse map, so these products approximate the *original* $K^{(b)}, V^{(b)}$, not $\widetilde{K}, \widetilde{V}$), concatenated with the exact tail and consumed by attention, which forms P from $QK^\top$ and $O = PV$ on the recovered K, V; these tensors exist for one layer at a time and are freed on return, so the transient peak is $\mathcal{O}(Sd)$, never the whole cache. Only the low-rank factors persist across steps. The two strategies differ in decode-time caching:

Decode strategy 1: accumulating cache. The prefill factors are fixed and each new token is appended *exactly* to the cache, never compressed; no decode-time SVD is performed and the decode-factor side cache stays empty, so the cache grows with generation length. This is best when the prompt is long but generation is short (e.g. multiple-choice tasks or *no-think* mode), where the exact generated tokens stay a small fraction of the cache.

Decode strategy 2: streaming cache (sliding window). Streaming `AttSVD` bounds the decode-time footprint by compressing *during* decode (Figure 1, bottom), holding the cache at a *fixed* size regardless of output length, which suits long *think*-mode generations. It maintains a *sliding* exact window of the most recent w tokens: each generated token is appended to the window, and once the window overflows by a full chunk of ℓ tokens (the `streaming` branch of Algorithm 1, App. C), the oldest ℓ tokens (still held *exactly*) are factored with the same decomposition into a per-layer list of frozen *decode blocks*, then dropped from the window. Decode attention runs on the full sequence, formed by concatenation ($\oplus$) of prefill bulk $\oplus$ decode blocks $\oplus$ exact window in position order. Two properties keep it sound: aged-out tokens are factored from their *exact* values (never by compressing the already-lossy prefill bulk), so error does not compound across the generation, and each block is frozen, so a token is compressed at most once. The scheme composes with any rank rule of Section 3.4 and the attention-aware basis of Section 3.2.

3.4 Choosing the rank: compression-ratio or per-matrix energy

`AttSVD` offers two strategies for setting the rank, each suited to a different use case: a *(1) fixed compression ratio* c when a specific memory budget is required, and a *(2) per-matrix energy rule* when factor denoising and minimizing reconstruction error matter more than hitting a target budget.

Compression-ratio rank (unified). If an absolute rank r or a compression ratio c is supplied, both K and V use the same r (with $r = \mathrm{round}((1-c)d)$ for a ratio), recovering the basic decomposition of Eq. (2). This *uniform* setting trades away per-matrix adaptivity for an exact, predictable budget; note that c selects a *rank* (a fraction of the d feature directions), not a fraction of *tokens*, so all S tokens are retained. This is the operating point used for the matched-memory settings in Section D.3 (e.g. c=0.5 $\Rightarrow$ r=64). The two ranks may also be set individually as explicit $r_{\widetilde{K}}$ and $r_{\widetilde{V}}$, overriding the shared value; this exposes the per-matrix budget as a direct hyperparameter and enables controlled study of the asymmetric $r_{\widetilde{K}} \ll r_{\widetilde{V}}$ regime.

Per-matrix energy rank (dynamic). Keys and values are not equally compressible. Building on the pre-RoPE-key observation of ShadowKV (Sun et al., 2025) that keys and values differ in their intrinsic compression capacity, the attention-weighted matrices we factor, $\widetilde{K}$ and $\widetilde{V}$, likewise likely differ, so a single shared rank over-spends on $\widetilde{K}$ and under-serves $\widetilde{V}$ (or vice versa) and the two may need different ranks. Here `AttSVD` lets each matrix choose its own rank from its own spectrum. Ordering a matrix M's singular values $\sigma_1 \geq \sigma_2 \geq \cdots$, we keep the fewest leading directions that retain a fraction τ of its energy $\|M\|_F^2 = \sum_i \sigma_i^2$,

$$r_M(\tau) \;=\; \min\Big\{ r : \; \textstyle\sum_{i\leq r} \sigma_i^2(M) \;\geq\; \tau \sum_i \sigma_i^2(M) \Big\}, \qquad M \in \{\widetilde{K}, \widetilde{V}\}, \tag{5}$$

the *effective rank* of M at tolerance τ (Roy & Vetterli, 2007) (default 0.99): a fast-decaying spectrum reaches τ in a few directions, a flat one needs many. Applying the *same* τ to both matrices yields per-matrix ranks $r_{\widetilde{K}}$ and $r_{\widetilde{V}}$ that adapt to each prompt's own spectrum rather than being hand-set, and may differ from one another. Equation (5) gives one rank per (batch, head); since the cache stores a single rank per matrix, these are reduced by a configurable aggregator: `max` (the default; the largest, so *every* head retains at least τ), `mean`, or a percentile `p`N (allowing a few low-energy

heads to fall below τ to save memory). An optional `rank_max` caps the result; passing an explicit τ selects this dynamic mode even when a rank or ratio is otherwise set.

3.5 MEMORY COST

The saving is explicit. A dense head stores $K, V \in \mathbb{R}^{S\times d}$, i.e. $2Sd$ entries; `AttSVD` instead keeps the factors that reconstruct them. For the prompt bulk it stores the factors $A_{\widetilde{K}} \in \mathbb{R}^{(S-w)\times r_{\widetilde{K}}}, B_{\widetilde{K}} \in \mathbb{R}^{r_{\widetilde{K}}\times d}$ for the keys and $A_{\widetilde{V}} \in \mathbb{R}^{(S-w)\times r_{\widetilde{V}}}, B_{\widetilde{V}} \in \mathbb{R}^{r_{\widetilde{V}}\times d}$ for the values. A rank-r factorization of an $(S-w)\times d$ matrix costs $r\,(S-w+d)$ entries, so the two factor pairs together use $(r_{\widetilde{K}}+r_{\widetilde{V}})(S-w+d)$ entries, plus a short exact tail of the w most recent tokens ($2wd$ entries). The tail is $\mathcal{O}(wd)$ with w fixed and $w \ll S$, so it is negligible against the bulk; dropping it and using $S-w \approx S$ leaves a persistent factor cost of $(r_{\widetilde{K}}+r_{\widetilde{V}})(S+d)$. Dividing by the dense $2Sd$ and expanding,

$$\frac{(r_{\widetilde{K}}+r_{\widetilde{V}})(S+d)}{2Sd} \;=\; \frac{r_{\widetilde{K}}+r_{\widetilde{V}}}{2d}\Big(1+\frac{d}{S}\Big) \;=\; \frac{r_{\widetilde{K}}+r_{\widetilde{V}}}{2d} + \frac{r_{\widetilde{K}}+r_{\widetilde{V}}}{2S} \;\xrightarrow[S\gg d]{}\; \frac{r_{\widetilde{K}}+r_{\widetilde{V}}}{2d}. \tag{6}$$

The residual $(r_{\widetilde{K}}+r_{\widetilde{V}})/(2S)$ term vanishes once the context is long ($S \gg d$), leaving the compression ratio $(r_{\widetilde{K}}+r_{\widetilde{V}})/(2d)$. The asymmetric regime $r_{\widetilde{K}} \ll r_{\widetilde{V}}$ shrinks this further, while a shared rank $r_{\widetilde{K}}{=}r_{\widetilde{V}}{=}r$ reduces it to simply r/d. The recent window adds a further $\mathcal{O}(w/S)$ term, and as context grows the measured ratio tightens to r/d. At decode, the *accumulating* cache keeps every generated token exact, so its exact part is the recent tail plus all g generated tokens, $\mathcal{O}\big((w{+}g)d\big)$; the *streaming* cache instead bounds it, oscillating in $[w,\, w{+}\ell]$ independent of g, while the compressed part grows by one $\mathcal{O}(\ell r + rd)$ block per ℓ tokens ($\approx r/d$ of dense per token once $\ell \gg d$, the same asymptotic ratio as the prefill bulk). Streaming thus trades a small fixed exact buffer plus extra *decode-time* SVDs (one per compressed chunk) for its $\mathcal{O}(g)$ tail, winning when g is large. The chunk ℓ sets the granularity. It must exceed the rank for the compression to be lossy, and larger ℓ relative to d tightens the storage toward its r/d asymptote; taking $w \to \infty$ recovers the accumulating cache (further details in Table 1). We use $\ell{=}128$ in our experiments, above every rank we use so each aged-out chunk is genuinely truncated, and a larger ℓ is equally valid.

3.6 INTERPRETABILITY: WHAT THE RETAINED BASIS EXPOSES

Insight. By compressing in the score and output space, weighting K by the query covariance $C_Q{=}Q^\top Q$ and V by the per-token attention mass, `AttSVD` retains a head's *functional* subspace. Because this basis is derived from the prompt's own attention geometry rather than from fixed weights or calibration data, these same quantities also serve as free, per-head interpretability insights: they reveal how each head geometrically uses its context, rather than what it attends to semantically.

(i) Effective rank. Roy & Vetterli (2007) observe that the effective rank of a matrix such as K or V measures its *intrinsic dimensionality*. Because our compression happens in the score and output geometry, the per-matrix ranks $r_{\widetilde{K}}, r_{\widetilde{V}}$ of Eq. (5) report the intrinsic dimensionality of the scores P (through the query-weighted keys $\widetilde{K}$) and of the output O (through the mass-weighted values $\widetilde{V}$). The per-matrix ranks can be read as the intrinsic compression capacity of each matrix, and the per-(layer, head) rank map shows which heads carry low-rank, highly compressible context. We empirically observe that, on the models we test, the query- and mass-weightings redistribute singular-value energy yet still leave a systematic gap $r_{\widetilde{K}} \ll r_{\widetilde{V}}$ (Appendix D, Figures 5 and 6).

(ii) Query subspace. The covariance $C_Q = Q^\top Q$ is the subspace the head's queries actually probe; the decay of its spectrum quantifies how *focused* (few-directional) vs. diffuse a head's attention is.

(iii) Token saliency. The per-key mass $a_i = \sum_q P_{qi}$ is a token-importance profile (the same heavy-hitter signal exploited by eviction (Zhang et al., 2023)) emerged here freely, so even an eviction-only pipeline can read off which tokens to keep and which to drop.

We emphasize that this is *geometric*, not *semantic*. The basis vectors are singular directions, not named concepts, they mix position and content (post-RoPE), and they describe each prompt and head rather than explain a prediction. Within those limits, the effective-rank map, query-subspace concentration, and attention-mass profile are concrete, reusable quantities the compression produces at no cost (interpretability analysis in Appendix D).

Table 1: Measured KV-cache memory on Phi-4-14B (40 layers, 40 query / 10 KV heads) and Qwen3-8B (36 layers, 32 query / 8 KV heads); d=128, bf16.

		Phi-4-14B			Qwen3-8B		
	Rank r	KV @32K (GiB)	Measured ratio	KV @9.4K (GiB)	KV @32K (GiB)	Measured ratio	KV @9.4K (GiB)
Dense cache	128	6.250	1.000	1.804	4.500	1.000	1.299
AttSVD (c=0.50)	64	3.139	0.502	0.916	2.260	0.502	0.660
AttSVD (c=0.75)	32	1.571	0.251	0.459	1.131	0.251	0.330
AttSVD (c=0.875)	16	0.787	0.126	0.231	0.567	0.126	0.166

Table 2: At 50% cache on LongBench, AttSVD (Accum/Stream) matches the dense cache average on both Phi-4-14B and Qwen3-8B and leads every reconstruction and eviction baseline.

	Phi-4-14B							Qwen3-8B						
		Reconstruction			Eviction				Reconstruction			Eviction		
LongBench task	Dense	AttSVD-Accum (ours)	AttSVD-Stream (ours)	SVD on raw KV	KeyDiff	KVzip	H2O	Dense	AttSVD-Accum (ours)	AttSVD-Stream (ours)	SVD on raw KV	KeyDiff	KVzip	H2O
PassageRetrieval-en	98.5	98.5	98.5	99.0	98.0	98.5	96.0	100.0	100.0	100.0	78.0	100.0	100.0	99.5
Qasper	27.9	26.9	28.1	21.8	23.4	28.1	26.7	41.0	41.4	41.4	37.0	30.7	41.2	37.5
MultiFieldQA-en	32.9	33.0	31.3	23.4	32.4	29.6	32.3	53.8	53.8	53.8	51.3	44.9	54.1	51.7
HotpotQA	21.4	20.7	20.8	14.8	20.8	20.5	20.4	59.4	59.1	59.1	53.5	56.8	58.9	58.3
2WikiMQA	36.5	35.6	35.8	21.9	27.2	33.3	33.8	43.0	43.3	43.3	38.1	40.9	44.0	43.0
GovReport	14.3	14.1	14.2	13.8	14.0	14.4	14.6	18.5	18.6	18.9	16.8	17.1	18.6	18.6
TriviaQA	28.6	26.6	26.8	20.3	26.8	27.6	27.3	88.5	89.0	89.0	87.1	85.2	88.5	88.5
TREC	52.5	53.3	51.9	52.3	47.8	58.5	51.2	59.5	57.7	57.7	25.8	35.1	62.1	61.1
PassageCount	17.5	15.5	14.0	12.0	11.0	8.0	12.0	2.5	3.5	3.5	2.5	6.0	0.0	2.5
average	36.7	**36.0**	35.7	31.0	33.5	35.4	34.9	51.8	51.8	**51.9**	43.3	46.3	51.9	51.2

4 EXPERIMENTS

Baselines. We compare AttSVD against three reference points at a token budget matched to AttSVD's memory at each compression setting, so every method shares the same model, prompts, decoding, and cache size. The *dense cache* is the uncompressed upper bound. We compare against *eviction* methods: **H2O** (heavy-hitter attention mass), **KeyDiff** (attention-free cosine similarity), and **KVzip** (query-agnostic reconstruction scoring); and *low-rank compression* methods: **SVD on raw KV** (standard top-r truncated SVD applied directly to K and V, without AttSVD's attention weighting), **KQ-SVD** (attention-aware but from an *offline calibration* set), and **OjaKV** (an *online* Oja's-rule basis seeded from calibration).

Datasets and benchmarks. We evaluate on *two complementary benchmarks*.

(i) Agentic benchmark: 184 prompts spanning 13 common scenarios across three difficulty tiers, each annotated with the correct action to take (from ∼77 candidates) or *no* action when none applies, examining action *selection* and *necessity*. An LLM judge scores each output 1 to 5; we report the mean judge score and pass@4 (fraction ≥ 4). Prompts are short but the reasoning-and-response generations are long (median ∼800 tokens, up to ∼10K).

(ii) LongBench (Bai et al., 2024): the full public long-context suite (1,750 examples over 9 tasks), each scored by its native metric; we report per-task accuracy and the average (0 to 100). Its long-context inputs and outputs stress the cache at real context length and exercise the streaming schedule's sliding window. The two examine different axes: the agentic eval isolates answer *quality*, LongBench tests *generality* at long context, so agreement across both guards against over-reliance on either. Memory sums the *actual* stored tensor bytes (factors plus exact tail) from one prefill forward, cross-checked against the allocator's peak (full setup and model choice in Appendix D).

4.1 ATTSVD VS. EVICTION METHODS

On the agentic benchmark at the aggressive half-rank (50% cache) point (Figure 2), attention-aware AttSVD-Stream reaches 4.07 mean judge / 62.5% pass@4, edging out the strongest eviction baselines (H2O 4.04/59.0%, KVzip 3.95/55.4%, KeyDiff 3.90/51.1%), while its AttSVD-Accum cache (4.01/59.2%) is on par with H2O at half the memory. On Qwen3-8B (Figure 2) the mean-judge gaps among Full, AttSVD (Accum/Streaming) and the eviction baselines H2O/KVzip all fall within the reported 95% CI, they are statistically tied with full attention, and only KeyDiff differs. The ordering is stable across the full rank range (Figure 3). The same holds for long context: on LongBench at the same 50% cache point (Table 2), AttSVD-Accum leads every eviction baseline on Phi-4-14B (36.0 vs. KVzip 35.4, H2O 34.9, KeyDiff 33.5) and ties the strongest eviction methods on Qwen3-8B (KVzip 51.9, H2O 51.2), while staying within 0.7 of the dense cache.

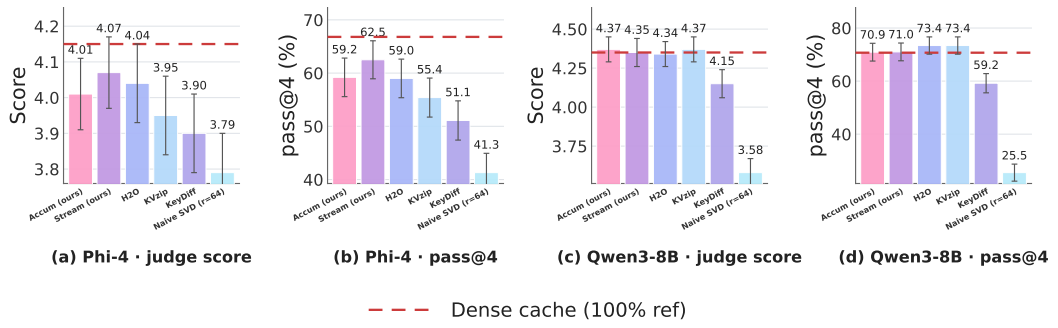


Figure 2: At the aggressive half-rank (50% cache) point, AttSVD stays within noise of the dense cache and matches or beats the strongest baselines on both Phi-4-14B and Qwen3-8B.

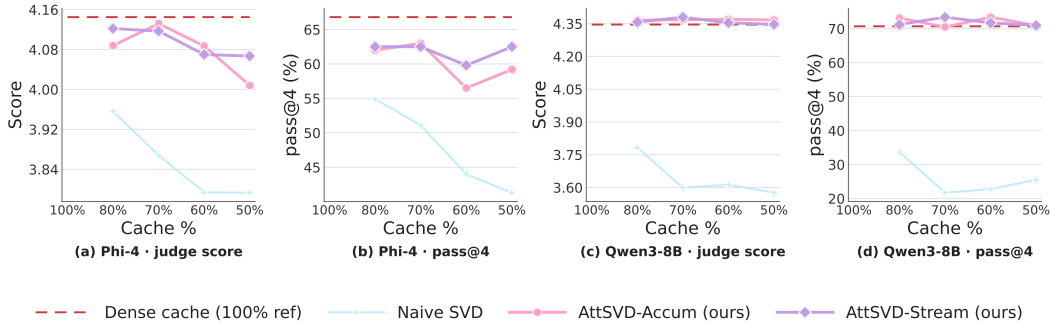


Figure 3: **Agentic benchmark** (n=184): mean judge score (a, c) and pass@4 (b, d) as the cache is reduced from 100% down to 50%, on Phi-4-14B (a, b) and Qwen3-8B (c, d). AttSVD (Accum/Stream) follows the dense cache far more closely than SVD on raw KV across the rank range.

4.2 AttSVD vs. other low-rank compression methods

Against other low-rank / SVD compressors the *basis* is what matters, not just the rank. SVD on raw KV truncation trails even the eviction family at r=64 on the agentic benchmark (3.79/41.3%, Figure 2), reflecting that a softmax-pretrained model has not adapted to low-rank K/V; attention-aware AttSVD closes this gap outright, adding +0.13 to 0.29 mean judge and +7 to 18 points pass@4 over SVD on raw KV at matched memory across the rank range (Figure 3). The same holds on LongBench, where AttSVD (36.0) is the strongest reconstruction compressor and SVD on raw KV trails (31.0; Table 2). We further test two *calibrated* SVD baselines on a third backbone, Llama-3.1-8B-Instruct,[2] (Table 3): KQ-SVD and OjaKV. We run each with its official implementation and released settings, matched only in cache budget, so the numbers reflect the methods as released. AttSVD-Accum matches the dense cache almost exactly on the average (48.5 vs. 48.5 at 80% cache; 47.8 vs. 48.5 at 60%) and leads KQ-SVD (47.4 and 45.2) by 1.1 to 2.6 points, with the largest gaps on the QA tasks (HotpotQA, MultiFieldQA-en) where a fixed calibration basis transfers least well across prompts. Both AttSVD schedules lead the OjaKV baseline (42.9 at 80%), which collapses to 30.5 at 60%. These gaps show that AttSVD's per-prompt, attention-aware basis generalizes better than both KQ-SVD and OjaKV, widening its advantage as the cache tightens.

4.3 Dynamic energy-rule rank

Section 3.4 sets $r_{\widetilde{K}}, r_{\widetilde{V}}$ per prompt from each matrix's own spectrum at a single tolerance τ (Eq. (5)), with no fixed rank or ratio. On the agentic benchmark (Figure 4) this dynamic mode matches the dense cache within 0.05 mean judge by τ=0.999 (4.098/64.7% vs. 4.145/66.8%), picking $r_{\widetilde{K}}, r_{\widetilde{V}}$

[2]We use Llama-3.1-8B-Instruct here because both baselines ship official code for it. KQ-SVD evaluates on LLaMA/Mistral, and OjaKV uses Llama-3.1-8B as its reference model.

Table 3: On Llama-3.1-8B-Instruct, AttSVD matches the dense cache average at 80% (48.5 vs. 48.5) and stays within 0.7 at 60%, beating KQ-SVD and OjaKV at both matched cache points.

	80% cache (c=0.2, r≈102)					60% cache (c=0.4, r≈77)				
LongBench task	Dense	AttSVD-Accum (ours)	AttSVD-Stream (ours)	KQ-SVD	OjaKV	Dense	AttSVD-Accum (ours)	AttSVD-Stream (ours)	KQ-SVD	OjaKV
MultiFieldQA-en	53.5	53.3	53.3	50.5	48.3	53.5	51.2	51.2	48.1	45.6
HotpotQA	54.2	54.3	54.3	50.9	35.8	54.2	53.5	53.5	48.2	26.9
2WikiMQA	44.6	45.3	45.3	40.9	40.1	44.6	45.4	45.4	40.4	31.8
TREC	7.5	7.0	7.0	12.0	8.2	7.5	6.0	6.0	7.8	3.0
PassageCount	6.2	6.8	6.8	7.2	3.0	6.2	7.2	7.2	7.6	4.5
PassageRetrieval-en	99.5	99.5	99.5	99.5	89.0	99.5	99.5	99.5	99.5	36.2
Qasper	47.4	47.6	47.6	46.5	45.1	47.4	46.3	46.3	45.3	37.7
GovReport	34.7	33.8	33.8	32.2	32.6	34.7	31.3	31.3	26.4	26.3
TriviaQA	89.1	89.2	89.2	86.6	84.1	89.1	89.5	89.5	83.5	62.2
average	48.5	**48.5**	**48.5**	47.4	42.9	48.5	**47.8**	**47.8**	45.2	30.5

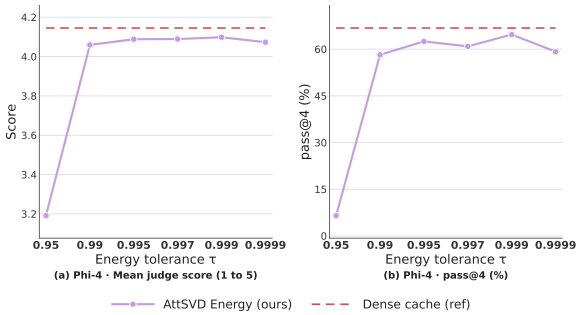


Figure 4: Under the dynamic energy rule, quality saturates by τ=0.99 and matches the dense cache from there on (agentic benchmark, n=184).

Table 4: The dynamic per-matrix energy rule picks $r_{\widetilde{K}}, r_{\widetilde{V}}$ from each prompt's own spectrum at tolerance τ and reaches near-exact LongBench quality on Phi-4-14B and Qwen3-8B.

	Phi-4-14B			Qwen3-8B		
LongBench task	Dense	Energy τ=0.997	Energy τ=0.995	Dense	Energy τ=0.997	Energy τ=0.995
PassageRetrieval-en	98.5	99.5	98.5	100.0	100.0	100.0
Qasper	27.9	27.0	28.7	41.0	40.5	41.2
MultiFieldQA-en	32.9	33.1	31.2	53.8	53.8	53.2
HotpotQA	21.4	21.4	22.7	59.4	59.3	59.3
2WikiMQA	36.5	35.8	35.6	43.0	43.5	43.3
GovReport	14.3	14.4	14.5	18.5	19.2	19.2
TriviaQA	28.6	28.3	28.4	88.5	88.5	88.5
TREC	52.5	53.9	52.9	59.5	60.1	61.4
PassageCount	17.5	15.5	15.0	2.5	3.0	3.5
average	36.7	**36.5**	36.4	51.8	52.0	**52.2**

per prompt with neither a rank nor a ratio preset. Running the same rule on the full LongBench suite (Phi-4-14B, attention-aware basis; Table 4), at τ=0.997 it reads $r_{\widetilde{K}} \approx 65$ and $r_{\widetilde{V}} \approx 127$ off the data, keys collapse to about half rank while values stay near-full, exactly the $r_{\widetilde{K}} \ll r_{\widetilde{V}}$ asymmetry of Section 3.4. At the operating point the data itself selects (≈75% cache), the average (36.5) lands within 0.2 of the dense cache (36.7); the tolerance τ alone recovers near-exact quality while each prompt sets its own rank. For a detailed interpretability analysis on real models, see Appendix D.

5 Conclusion

AttSVD shows that the KV cache can be compressed without discarding tokens: it retains every token yet stores each in an instance-optimal rank-r subspace obtained from an *online, per-prompt* SVD and reconstructed lazily at decode, so the persistent cache shrinks to $\approx r/d$ of dense. Its contributions are threefold: an online, calibration-free basis that is provably optimal (Eckart-Young) for the prompt at hand; adaptive per-matrix ranks and an attention-aware objective that spend the budget where attention actually concentrates; and accumulating and streaming decode modes for different generation scenarios. Because the factors are exact, they are also directly *interpretable*: the effective-rank map exposes the intrinsic compressibility of the query-weighted keys $\widetilde{K}$ and mass-weighted values $\widetilde{V}$, the query subspace shows each head's focus, and the attention-mass profile flags salient tokens, all without calibration. Empirically, across several models, AttSVD matches the dense cache on the agentic benchmark and LongBench at roughly half the KV-cache memory, positioning instance-optimal reconstruction as a strong, training-free alternative to eviction.

USE OF AI TOOLS

All ideas presented in this paper are original to the authors, or build upon the related work cited herein. AI-based tools were used only in an assistive capacity: to improve the clarity of the authors' initial statements, to help draft sections of the paper in LaTeX, to improve the quality of figures and tables, to correct grammatical errors, to make sentences more salient, and to serve as a coding assistant that helps implement the authors' ideas. All technical content, claims, and conclusions were conceived, verified, and approved by the authors.

ETHICS STATEMENT

The authors have reviewed the ICLR Code of Ethics and believe this work is fully consistent with it. `AttSVD` is a training-free, inference-time KV-cache compression method. Because the method operates purely on the key and value caches of a frozen pretrained model, it introduces no new training signal; any bias or fairness limitations already present in the underlying language models are inherited unchanged, and `AttSVD` is designed to alter neither their behavior nor their outputs beyond compressing the cache. The authors report no competing interests and no external funding that could have influenced the findings.

REPRODUCIBILITY STATEMENT

We have taken concrete steps to make each reported result reproducible. The method, including the attention-guided per-prompt truncated SVD, the energy-based rank rule, and the accumulating and streaming decode strategies, is specified in Section 3, with the full procedures given as explicit pseudocode in Appendix C (Algorithms 1 and 2) and the attention-aware truncation derived with a complete proof in Appendix B. The experimental setup, including implementation details, baselines, models, and extended studies, is documented in Appendix D. To further support exact reproduction, we release the core code of `AttSVD`, the model-independent dynamic cache class (a `DynamicCache` subclass that implements Algorithms 1 and 2) together with its unit tests, as anonymized supplementary material, and defer the release of the full repository to acceptance.

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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Luca Benfenati, Matteo Risso, Andrea Vannozzi, Ahmet Caner Yüzügüler, Lukas Cavigelli, Enrico Macii, Daniele Jahier Pagliari, and Alessio Burrello. Don't be so stief! learning kv cache low-rank approximation over the stiefel manifold. *arXiv preprint arXiv:2601.21686*, 2026.

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 *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=LWMS4pk2vK.

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 *Forty-third International Conference on Machine Learning*, 2026.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Jitai Hao, Qiang Huang, Yaowei Wang, Min Zhang, and Jun Yu. DeltaKV: Residual-based KV cache compression via long-range similarity. *arXiv preprint arXiv:2602.08005*, 2026.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. KVzip: Query-agnostic KV cache compression with context reconstruction. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=JFygzwx8SJ.

Jianlong Lei and Shashikant Ilager. ARKV: Adaptive and resource-efficient KV cache management under limited memory budget for long-context inference in LLMs. *arXiv preprint arXiv:2603.08727*, 2026.

Damien Lesens, Beheshteh T. Rakhshan, and Guillaume Rabusseau. KQ-SVD: Compressing the KV cache with provable guarantees on attention fidelity. In *The 29th International Conference on Artificial Intelligence and Statistics*, 2026. URL https://openreview.net/forum?id=6PltpCBNmO.

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. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=poE54GOq2l.

Bokai Lin, Zihao Zeng, Zipeng Xiao, Siqi Kou, TianQi Hou, Xiaofeng Gao, Hao Zhang, and Zhijie Deng. MatryoshkaKV:adaptive KV compression via trainable orthogonal projection. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=BQwsRy1h3U.

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.

Leon Mirsky. Symmetric gauge functions and unitarily invariant norms. *The Quarterly Journal of Mathematics*, 11(1):50–59, 1960.

Junyoung Park, Dalton Jones, Matthew J Morse, Raghavv Goel, Mingu Lee, and Christopher Lott. Keydiff: Key similarity-based KV cache eviction for long-context LLM inference in resource-constrained environments. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=uBaFH7aQnC.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *European Signal Processing Conference (EUSIPCO)*, pp. 606–610, 2007.

Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. Eigen attention: Attention in low-rank space for KV cache compression. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 15332–15344, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.899. URL https://aclanthology.org/2024.findings-emnlp.899/.

Alina Shutova, Vladimir Malinovskii, Vage Egiazarian, Denis Kuznedelev, Denis Mazur, Surkov Nikita, Ivan Ermakov, and Dan Alistarh. Cache me if you must: Adaptive key-value quantization for large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=COowwJOAZi.

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 *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=oa7MYAO6h6.

Zhongwei Wan, Xinjian Wu, Yu Zhang, Yi Xin, Chaofan Tao, Zhihong Zhu, Xin Wang, Siqi Luo, Jing Xiong, Longyue Wang, and Mi Zhang. D_2O: Dynamic discriminative operations for efficient long-context inference of large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=HzBfoUdjHt.

Yuhui Xu, Zhanming Jie, Hanze Dong, Lei Wang, Xudong Lu, Aojun Zhou, Amrita Saha, Caiming Xiong, and Doyen Sahoo. Think: Thinner key cache by query-driven pruning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=n0OtGl6VGb.

Xianglong Yan, Zhiteng Li, Tianao Zhang, Linghe Kong, Yulun Zhang, and Xiaokang Yang. ReCalKV: Low-rank KV cache compression via head reordering and offline calibration. *arXiv preprint arXiv:2505.24357*, 2025.

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*, 2023.

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 Re, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=RkRrPp7GKO.

Yuhao Zhou, Sirui Song, Boyang Liu, Zhiheng Xi, Senjie Jin, Xiaoran Fan, Zhihao Zhang, Wei Li, and Xuanjing Huang. EliteKV: Scalable KV cache compression via RoPE frequency selection and joint low-rank projection. *arXiv preprint arXiv:2503.01586*, 2025.

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 Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 10161–10178, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.494. URL https://aclanthology.org/2026.findings-acl.494/.

Adam Zweiger, Xinghong Fu, Han Guo, and Yoon Kim. Fast KV compaction via attention matching. *arXiv preprint arXiv:2602.16284*, 2026.

A Extended related work

Eviction methods. *Eviction* methods take an orthogonal route, keeping a token budget and dropping the rest by scoring tokens on accumulated heavy-hitter attention mass (H2O (Zhang et al., 2023)), attention-free key cosine similarity (KeyDiff (Park et al., 2025)), or query-agnostic reconstruction importance (KVzip (Kim et al., 2025)); we compare against all three. A parallel line keeps *eviction* as the backbone but relaxes its binary keep/drop decision into a non-binary retain / approximate / evict allocation (D2O (Wan et al., 2025), ARKV (Lei & Ilager, 2026)).

Compression and reconstruction methods. Most methods extract the basis ahead of time and reuse it for every prompt; Table 5 groups them by basis source. *Weight decompositions* apply an offline SVD to the key/value projection *weight matrices* (ASVD (Yuan et al., 2023), Palu (Chang et al., 2025), LoRC (Zhang et al., 2024)). *Calibration* methods estimate the basis from a corpus (Eigen Attention (Saxena et al., 2024), ReCalKV (Yan et al., 2025)); KQ-SVD (Lesens et al., 2026) is closest in spirit, factoring the *query-key product* $KQ^\top$ to target the attention scores rather than K alone, but still from an *offline* calibration set of queries fixed across prompts. *Cross-layer or trained* methods share or learn the basis (xKV (Chang et al., 2026), MatryoshkaKV (Lin et al., 2025), Stiefel-manifold projections (Benfenati et al., 2026), EliteKV (Zhou et al., 2025)). All of these tie the basis to weights, a calibration distribution, or a learning phase. ShadowKV (Sun et al., 2025) is per-prompt, applying an online SVD to the pre-RoPE keys with no calibration, but it compresses keys only and offloads the full values to CPU. ThinK (Xu et al., 2025) is likewise per-prompt, pruning key channels by query-driven scoring with no training (keys only). OjaKV (Zhu et al., 2026) is also online and per-context, i.e., it maintains a *running* incremental-PCA basis (Oja's rule), seeded from a small calibration set and re-orthonormalized during decoding, and uses attention only to choose which tokens stay full-rank rather than to shape the basis, trading exactness for within-generation adaptation. A separate cache-style line predicts or residual-codes values from keys via a fixed map $V \approx KW$ fit offline (DeltaKV (Hao et al., 2026)), keeping keys and approximating values; these share our "keep all positions" philosophy but with a *fixed, generic* map. In the same reconstruction-and-compression vein, VECTOR (Lin et al., 2026) reconstructs evicted values from retained keys by least squares, and Attention Matching (Zweiger et al., 2026) fits the compressed cache to match the original *attention output* rather than the raw K, V; orthogonally, adaptive KV *quantization* lowers the bit-width of the stored cache rather than its rank (AQUA-KV (Shutova et al., 2025)). These are complementary to `AttSVD`, which instead sets the per-prompt low-rank basis itself from this prompt's own attention.

Table 5: SVD / low-rank KV-cache methods by basis source. †ShadowKV low-rank-compresses keys and offloads full values to CPU.

Method	AttentionSVDs	Basis source	Calib./train?	K&V
ASVD (Yuan et al., 2023)	K/V proj. weights	offline, activation-aware	calib.	weights
Palu (Chang et al., 2025)	K/V proj. weights	offline, one-time	calib.	K & V
LoRC (Zhang et al., 2024)	KV weight matrices	offline, layer-progressive	calib.	K & V
Eigen Attn (Saxena et al., 2024)	K/Q/V activations	offline calibration	calib.	K/Q/V
KQ-SVD (Lesens et al., 2026)	*score* $KQ^\top/VW_O$	offline calibration (queries)	calib.	K & V
ReCalKV (Yan et al., 2025)	K (grouped) / V	offline calibration	calib.	K & V
ShadowKV (Sun et al., 2025)	pre-RoPE *keys*	per-prompt SVD (keys only)	none	K only†
xKV (Chang et al., 2026)	KV across layers	post-hoc, shared	none	K & V
MatryoshkaKV (Lin et al., 2025)	KV feature dim	*trained* projections	train	K & V
ThinK (Xu et al., 2025)	key *channels*	query-aware (axis-aligned)	none	K only
OjaKV (Zhu et al., 2026)	K/V activations	*online* Oja PCA (seeded)	seed	K & V
`AttSVD` (ours)	$KC_Q^{1/2}$, $\mathrm{diag}(\sqrt{a})V$, post-RoPE, per prompt	*online* exact per-prompt SVD	**none**	K & V

B Proof of Eq. (3) (attention-aware key truncation)

We show that the query-metric key objective of Eq. (3) is (i) identical to a plain Frobenius objective in the metric C_Q, and (ii) solved exactly by an ordinary truncated SVD after a change of variables, yielding the factors $A_{\widetilde{K}}, B_{\widetilde{K}}$ stored in Algorithm 2. Throughout, $Q \in \mathbb{R}^{m\times d}$ are the (post-RoPE) queries, $K \in \mathbb{R}^{n\times d}$ the keys, $C_Q = Q^\top Q \in \mathbb{R}^{d\times d}$, and $E = K - X$ for a candidate rank-r approximation X. As $C_Q \succeq 0$ is symmetric positive semidefinite it admits a unique symmetric

PSD square root $C_Q^{1/2}$; with the ridge $C_Q = Q^\top Q + \epsilon\,\bar{c}\,I_d \succ 0$ (Hoerl & Kennard, 1970) used in Algorithm 2, $C_Q^{1/2}$ is invertible with inverse $C_Q^{-1/2}$.

(i) Norm equivalence. Using $\|M\|_F^2 = \mathrm{tr}(MM^\top)$ and the cyclic invariance of the trace,

$$\big\|QE^\top\big\|_F^2 = \mathrm{tr}\big(QE^\top(QE^\top)^\top\big) = \mathrm{tr}\big(QE^\top E\,Q^\top\big) = \mathrm{tr}\big(E^\top E\,Q^\top Q\big) = \mathrm{tr}\big(E^\top E\,C_Q\big),$$
$$\big\|E\,C_Q^{1/2}\big\|_F^2 = \mathrm{tr}\big(E\,C_Q^{1/2}C_Q^{1/2}E^\top\big) = \mathrm{tr}\big(E\,C_Q\,E^\top\big) = \mathrm{tr}\big(E^\top E\,C_Q\big),$$

where both chains use cyclicity of the trace and the second uses $C_Q^{1/2}C_Q^{1/2} = C_Q$. Hence $\|Q(K - X)^\top\|_F = \|(K - X)\,C_Q^{1/2}\|_F$ for every X, i.e. the two objectives in Eq. (3) coincide; both equal $\mathrm{tr}\big((K - X)\,C_Q\,(K - X)^\top\big)^{1/2}$. Up to the constant $1/\sqrt{d}$ this is precisely the perturbation of the logits $Q(K-X)^\top/\sqrt{d}$, so minimizing the reconstruction error of K in the C_Q metric minimizes the logit error. Weighting the factorization by the query-side second moment $C_Q = Q^\top Q$, rather than by a fixed calibration basis, is the metric analogue of learning a low-rank attention subspace (Saxena et al., 2024).

(ii) Generalized Eckart-Young. Introduce $\widetilde{K} = K\,C_Q^{1/2}$ and $\widetilde{X} = X\,C_Q^{1/2}$. Since $C_Q^{1/2}$ is invertible, $\mathrm{rank}(\widetilde{X}) = \mathrm{rank}(X)$, so $X \mapsto \widetilde{X}$ is a bijection on the set of rank-$\le r$ matrices, and $(K - X)C_Q^{1/2} = \widetilde{K} - \widetilde{X}$. The problem of Eq. (3) thus reduces to the *ordinary* rank-constrained Frobenius problem

$$\min_{\mathrm{rank}(\widetilde{X})\le r} \big\|\widetilde{K} - \widetilde{X}\big\|_F.$$

By the Eckart-Young theorem (Eckart & Young, 1936; Mirsky, 1960) (Eq. (1)) its minimizer is the rank-r truncated SVD of $\widetilde{K}$: writing $\widetilde{K} = U\Sigma\mathcal{V}^\top$ with singular values nonincreasing, $\widetilde{X}^\star = U_{:,1:r}\Sigma_{1:r}\mathcal{V}_{:,1:r}^\top$. Mapping back through $C_Q^{-1/2}$,

$$X^\star = \widetilde{X}^\star C_Q^{-1/2} = \underbrace{U_{:,1:r}\Sigma_{1:r}}_{A_{\widetilde{K}}}\ \underbrace{\mathcal{V}_{:,1:r}^\top C_Q^{-1/2}}_{B_{\widetilde{K}}},$$

which is exactly the decomposed store $A_{\widetilde{K}}B_{\widetilde{K}} = K_r^\star$ of Eq. (3) and Algorithm 2. Because the change of variables preserves the objective value and is a bijection, $X^\star$ is a global minimizer of the original query-metric problem; it is unique whenever $\sigma_r(\widetilde{K}) > \sigma_{r+1}(\widetilde{K})$. □

Remark (values, Eq. (4)). The value objective is the same statement with a left (row) metric $\mathrm{diag}(a)$ replacing the right (column) metric C_Q: for $E = V - X$, $\sum_i a_i\|E_i\|_2^2 = \|\mathrm{diag}(\sqrt{a})\,E\|_F^2$, so by the identical change-of-variables argument the minimizer is the truncated SVD of $\mathrm{diag}(\sqrt{a})\,V$ with the left factor rescaled by $\mathrm{diag}(1/\sqrt{a})$, giving $A_{\widetilde{V}}, B_{\widetilde{V}}$ as in Algorithm 2.

Why the diagonal (per-key mass) metric. We deliberately replace the exact row metric $P^\top P$ by $\mathrm{diag}(a)$, $a_i = \sum_q P_{qi}$, for four reasons; all stem from $P^\top P$ being an $S \times S$ object that grows with the context length, whereas the key metric C_Q stays a fixed $d \times d$ gram. *(i) Cost.* $P^\top P$ is quadratic in S to form and its symmetric square root is an $\mathcal{O}(S^3)$ eigendecomposition, per layer and forward pass; at S=32K that is a $\sim$4 GB, $\sim 10^{13}$-flop object per head, against $\mathcal{O}(S)$ to form, store, and invert a. A metric that costs more than the low-rank cache saves would defeat the method. *(ii) No materialized P.* $P^\top P$ needs the dense $S \times S$ attention matrix, which memory-efficient kernels (FlashAttention) never form; the mass $a = \sum_q P_{q,:}$ is one attention-shaped reduction that accumulates incrementally as tokens arrive, so it stays compatible with both the memory goal and standard kernels. *(iii) Conditioning.* $P^\top P$ has rank at most the number of query rows and is singular whenever queries are fewer than keys, so $(P^\top P)^{-1/2}$ is ill-defined on its null space; $\mathrm{diag}(a)$ with the floor $a \leftarrow a + 10^{-3}\bar{a}$ is strictly positive and trivially invertible. *(iv) First-order sufficiency.* Attention mass concentrates on a few heavy tokens (the first-token sink alone carries a large share; Fig. 6), so the diagonal already encodes which rows the output reads, while the off-diagonal terms of $P^\top P$ are second-order. Empirically this diagonal weighting alone lowers the value output error $\|P(V - V_r)\|_F/\|PV\|_F$ from 0.154 to 0.012 at 50% cache, most of the attainable gain.

C ALGORITHMS

Algorithm 1 ATTSVD; online per-prompt low-rank KV cache (one layer, one key/value head). A single DECODE serves both schedules: the `streaming` flag switches on decode-time compression (branch-specific steps for both schedules are highlighted in blue).

Input: keys K, values $V \in \mathbb{R}^{S\times d}$; post-RoPE queries Q (only for the attention-aware basis); recent window w; `streaming` flag with chunk ℓ

Output: factors $(A_{\widetilde{K}}, B_{\widetilde{K}}), (A_{\widetilde{V}}, B_{\widetilde{V}})$; exact tail (T_K, T_V) (a sliding window when `streaming`); decode blocks $(\mathcal{D}_K, \mathcal{D}_V)$

Rank rule (the one knob, fixed once): *fixed* $r{=}\mathrm{round}\big((1{-}c)d\big)$ for both K, V, *or per-matrix energy* τ (set $r{=}$NONE, so r_K, r_V are read from each matrix's own spectrum, giving $r_K \ll r_V$). The prompt bulk is factored attention-aware (ATTENTIONSVD, Alg. 2); streaming decode chunks, which carry no queries, use the unweighted TRUNCSVD.

procedure PREFILL(K, V, Q) — *once, after the prompt is encoded*
  $r \leftarrow \mathrm{round}\big((1-c)\,d\big)$ **if** ratio c is given **else** NONE — *fixed unified rank, else per-matrix energy rule*
  $K^{(b)}, T_K \leftarrow K_{1:S-w},\ K_{S-w+1:S};\quad V^{(b)}, T_V \leftarrow V_{1:S-w},\ V_{S-w+1:S}$ — *split bulk / exact tail*
  $(A_{\widetilde{K}}, B_{\widetilde{K}}), (A_{\widetilde{V}}, B_{\widetilde{V}}) \leftarrow$ ATTENTIONSVD$(K^{(b)}, V^{(b)}, Q, r)$ — *attention-aware factors (Alg. 2); discard the bulk*
  $\mathcal{D}_K, \mathcal{D}_V \leftarrow (\,), (\,)$ — *decode-block lists; populated only when* `streaming`
  return K, V — *prefill served exactly from the original* K, V
end procedure

procedure DECODE(k, v) — *every generated token*
  $T_K \leftarrow [\,T_K;\ k\,],\quad T_V \leftarrow [\,T_V;\ v\,]$ — *append the new token exactly*
  if `streaming` **then** — ***STREAMING**: factor aged-out tokens during decode*
    while $\mathrm{rows}(T_K) - w \geq \ell$ **do** — *exact window overflowed by a full chunk*
      $C_K, C_V \leftarrow$ oldest ℓ rows of T_K, T_V — *still exact values*
      $(a_K, b_K) \leftarrow$ TRUNCSVD(C_K, r), $(a_V, b_V) \leftarrow$ TRUNCSVD(C_V, r); append to $\mathcal{D}_K, \mathcal{D}_V$ — *aged-out tokens carry no queries ⇒ unweighted SVD; freeze*
      drop those oldest ℓ rows from T_K, T_V — *window stays in* $[w,\ w{+}\ell]$
    end while
  else — ***STATIC**: no decode-time compression*
    keep T_K, T_V exact — *the tail grows by this token; prefill factors fixed, $\mathcal{D}_K, \mathcal{D}_V$ stay empty*
  end if
  $K_{\mathrm{full}} \leftarrow [\,A_{\widetilde{K}} B_{\widetilde{K}};\ \mathrm{RECON}(\mathcal{D}_K);\ T_K\,];\quad V_{\mathrm{full}} \leftarrow [\,A_{\widetilde{V}} B_{\widetilde{V}};\ \mathrm{RECON}(\mathcal{D}_V);\ T_V\,]$ — *bulk ⊕ decode blocks ⊕ exact tail/window*
  return K_{full}, V_{full} — *accumulating: $\mathcal{D}$ empty ⇒ $[A_{\widetilde{K}} B_{\widetilde{K}}; T_K]$; freed after this layer*
end procedure

RECON($\mathcal{D}$): concatenate AB over all blocks $(A, B) \in \mathcal{D}$ (oldest first); empty in accumulating mode. ATTENTIONSVD is the attention-aware factorization of Alg. 2.

Algorithm 2 ATTENTIONSVD: the attention-aware low-rank factorization for the prompt bulk (one layer/head). Keys are truncated in the query geometry C_Q and values in the attention-mass metric, so the kept directions are exactly the ones attention reads. TRUNCSVD is the rank-r thin SVD (rank rule).

function ATTENTIONSVD(K, V, Q, r) — *attention-aware basis (our method); r fixed, or NONE⇒ energy rule*
  if Q unavailable **then** — *rare: capture hook did not fire ⇒ no weighting available*
    return TRUNCSVD(K, r), TRUNCSVD(V, r) — *degenerate unweighted fallback only (Eq. 2)*
  end if
  KEYS *truncate in the query geometry C_Q, preserving the logits $QK^\top$*
  $C_Q \leftarrow Q^\top Q + \epsilon\,\bar{c}\,I_d,\ \ \bar{c}{=}\frac{1}{d}\mathrm{tr}(Q^\top Q);\ \ W, W^{-1} \leftarrow C_Q^{1/2}, C_Q^{-1/2}$ — *$d{\times}d$ gram; ridge ϵ (def. 10^{-3})*
  $(A_{\widetilde{K}}, B_{\widetilde{K}}) \leftarrow$ TRUNCSVD$(KW, r);\quad B_{\widetilde{K}} \leftarrow B_{\widetilde{K}} W^{-1}$ — *$A_{\widetilde{K}} B_{\widetilde{K}} = K_r^\star$ (Eq. 3)*
  VALUES *weight rows by attention mass a, preserving the output PV*
  $a \leftarrow \sum_q \mathrm{softmax}(QK^\top/\sqrt{d})_{q,:} + 10^{-3}\,\bar{a}$ — *per-key mass (GQA-expanded), floored*
  $(A_{\widetilde{V}}, B_{\widetilde{V}}) \leftarrow$ TRUNCSVD$(\mathrm{diag}(\sqrt{a})\,V, r);\quad A_{\widetilde{V}} \leftarrow \mathrm{diag}(1/\sqrt{a})\,A_{\widetilde{V}}$ — *$A_{\widetilde{V}} B_{\widetilde{V}} = V_r^\star$ (Eq. 4)*
  return $(A_{\widetilde{K}}, B_{\widetilde{K}}), (A_{\widetilde{V}}, B_{\widetilde{V}})$
end function

function TRUNCSVD(X, r) — *rank-r thin SVD with the rank rule (linear-algebra primitive)*
  $U, \Sigma, \mathcal{V} \leftarrow \mathrm{SVD}\ (\mathrm{fp32}(X))$ — *singular values Σ in descending order*
  if $r =$ NONE **then** — *dynamic per-matrix rank from X's own spectrum*
    $r \leftarrow \min\{\rho :\ \sum_{i\leq\rho} \Sigma_{ii}^2 \ \geq\ \tau \sum_i \Sigma_{ii}^2\}$ — *energy rule (Eq. 5); default $\tau{=}0.99$*
  end if
  return $\big(\,U_{:,1:r}\Sigma_{1:r}\ ,\ \ \mathcal{V}_{:,1:r}^\top\,\big)$ — *$\in \mathbb{R}^{n\times r} \times \mathbb{R}^{r\times d}$; Eckart-Young (Eq. 1)*
end function

Table 6: Streaming vs. accumulating decode on LongBench (Phi-4-14B, full suite, n=1,750; average, Δ the streaming-minus-accumulating gap), showing that streaming stays within run-to-run noise of the accumulating cache at every budget and lets the growing $\mathcal{O}(g)$ exact decode tail be bounded to a fixed window at no quality cost.

			Phi-4-14B				Qwen3-8B			
Cache	c	r	SVD on raw KV	AttSVD-Accum (decode exact) (ours)	AttSVD-Stream (w=64) (ours)	Δ	SVD on raw KV	AttSVD-Accum (ours)	AttSVD-Stream (ours)	Δ
100%	–	Dense cache	–	36.7		–	–	51.8		–
80%	0.2	102	33.83	36.21	36.45	+0.24	48.84	51.75	51.75	+0.00
70%	0.3	90	33.13	37.21	36.93	−0.28	48.33	52.10	52.09	−0.01
60%	0.4	77	33.16	36.64	36.65	+0.01	46.88	52.55	52.54	−0.01
50%	0.5	64	31.0	36.02	35.73	−0.29	43.34	51.81	51.85	+0.04

D Experimental setup and extended experiments

On a small model we verify that full-rank (r=d) decomposition reproduces the dense-attention output *bit-for-bit*; the reconstruction error then decreases monotonically as r increases, consistent with Eq. (1). Table 1 reports the KV-cache footprint on Phi-4-14B: the measured ratio matches the theoretical r/d to the decimal at 32K context; at shorter context the fixed rd and exact-tail terms make the ratio slightly larger, exactly as Eq. (6) predicts.

D.1 Streaming vs. accumulating decode

The accumulating and streaming schedules differ only in the decode tail. Accumulating keeps every earlier decode token exact (an $\mathcal{O}(g)$ tail that grows with the generation), while streaming factors aged-out tokens into frozen rank-r blocks so the exact buffer stays bounded. The agentic benchmark generations are mostly too short to overflow the w-token window, so streaming rarely evicts; to stress it where it genuinely compresses decode history we repeat the comparison on the full LongBench suite (n=1,750), whose long-context outputs routinely overflow the window and trigger repeated chunk compression (Table 6). Across all ranks, streaming stays within 0.3 points of the accumulating (decode-exact) cache on average ($\Delta \in [-0.29, +0.24]$), a gap well inside run-to-run noise, and it follows the dense cache (36.7) down to the 50% budget. This holds *even though* streaming repeatedly factors aged-out tokens into frozen low-rank blocks rather than keeping them exact. Across both a short-output agentic eval (Figure 3) and a long-output academic suite, then, bounding the $\mathcal{O}(g)$ decode tail to a fixed window costs no measurable quality. Per task, the two schedules are in fact identical on 8 of 9 LongBench tasks and differ only on GovReport, the sole task with a long (512-token) generation budget, where streaming compression fires once the generation overflows the w=64 window; the remaining tasks generate ≤ 128 tokens (mostly ≤ 64) and end early under greedy decoding, so the streaming path never fires and reduces exactly to accumulating. At the aggressive r=64 the streaming cache holds 62.5% pass@4 with the w=64 window, vs. 56.0% at w=16 in an earlier run; enlarging the exact window removes the only point where compressing decode history had cost quality. The result is decisive in one direction: streaming *loses nothing* relative to the accumulating cache, while replacing the accumulating cache's $\mathcal{O}(g)$ exact tail (which grows with every one of the g generated tokens) with a *bounded* one; a large memory-efficiency gain on long generations at no quality cost.

D.2 Streaming-window ablation on a long-generation task

Table 7: Streaming-window ablation on GovReport (Llama-3.1-8B-Instruct, c=0.2) for different exact decode-tail windows w, where ROUGE-L stays flat as w shrinks from ∞ down to 16 so even a small exact window preserves quality.

Decode schedule	Exact decode window w	GovReport (ROUGE-L)
FullKV (parity anchor)	n/a	34.66
AttSVD-Accum (decode exact)	∞	33.79
AttSVD-Stream	256	33.80
AttSVD-Stream	128	33.81
AttSVD-Stream	64	33.77
AttSVD-Stream	32	33.88
AttSVD-Stream	16	33.74

The streaming schedule bounds the exact *decode* tail to a sliding window of w tokens, factoring aged-out tokens into frozen low-rank blocks (§3.3). On standard LongBench QA this path rarely fires, because most tasks generate ≤ 128 tokens and never overflow the window, so streaming reduces to the accumulating (decode-exact) cache (§4; Table 3). To isolate the effect of w where the schedule *does* fire, we run a single-job ablation on GovReport, the LongBench task with the longest generation budget (512 tokens), at a fixed cache ratio (c=0.2, i.e. 80%), varying the window w and including the accumulating cache as the w=∞ limit. All variants share the same attention-aware prefill factorization and are scored together against the FullKV anchor. The result is a flat curve where ROUGE-L varies by under 0.15 across w from 256 down to 16 and stays level with the w=∞ accumulating cache, so bounding the exact decode window to as few as 16 tokens costs no measurable quality.

D.3 Interpretability analysis

D.3.1 Adversarial case on a controlled synthetic head

We begin with a controlled synthetic head that isolates the core effect. Table 8 is built so that K's query-relevant directions are deliberately misaligned with its dominant singular directions, the adversarial case for energy-ordered SVD. At matched rank we compare SVD on raw KV, which minimizes $\|K - K_r\|_F$, against the *attention-aware* truncation of Eqs. (3) and (4). On the keys, attention-aware truncation cuts the relative logit error $\|Q(K - K_r)^\top\|_F / \|QK^\top\|_F$ from 0.98 to 0.27 at rank r=4, a 3.6× reduction on the quantity attention actually consumes, even though its raw $\|K - K_r\|_F$ is *larger*. On the values, mass-weighted truncation lowers the output error $\|P(V - V_r)\|_F / \|PV\|_F$ from 0.92 to 0.64 at r=8 and from 0.61 to 0.00 at r=16, recovering the exact attention output once the rank spans the mass-weighted subspace. The retained *basis*, not just the rank, governs quality at a fixed memory budget.

D.3.2 Real-world interpretability analysis

We then move from the synthetic head to a real one. Figures 7, 5, and 6 inspect a single layer/head of SmolLM2-135M[3] under both weightings. Figure 7 contrasts the attention-aware and raw-KV caches directly, while Figures 5 and 6 give a full decomposition of K and V on that head; each reports four panels: *(a)* the attention map, *(b)* the singular spectrum under raw vs. attention-aware weighting, *(c)* the per-token factor weights $|U|$, and *(d)* the top-contributing tokens of each factor. The three quantities of §3.6 map onto this head as follows.

(i) Effective rank. Panel (b) of each factor figure reports the per-matrix compression capacity as the decay rate of its energy curve. The energy rule of Eq. (5) measures this decay on the attention-aware matrices $\widetilde{K}$ and $\widetilde{V}$, not the raw K and V, and sets each its own rank accordingly. $\widetilde{K}$'s spectrum is steep, with ∼67% of the energy in a single leading factor and 99.5% in the top-32 (a 50% cache), whereas $\widetilde{V}$'s spectrum is markedly flatter, with only ∼36% in its leading factor. The raw K and V spectra (red in panel (b)) track these attention-aware curves closely in overall decay, so the steep versus flat gap is intrinsic to this head rather than an artifact of the weighting; the query-weighted $\widetilde{K}$ curve is only somewhat sharper, concentrating on the query-relevant directions that item (ii) examines. That asymmetry is the concrete per-head cause of the $r_{\widetilde{K}} \ll r_{\widetilde{V}}$ allocation that Eq. (5) exploits, with $\widetilde{V}$ genuinely less compressible than $\widetilde{K}$.

(ii) Query subspace. Panel (b) also shows that the attention-aware basis re-ranks factors by query relevance rather than raw energy, keeping a *different* rank-r subspace than SVD on raw KV; the gap between the raw-energy (red) and attention-weighted (blue) curves makes visible that ordering factors by energy is not the same as ordering them by what the queries consume. Figure 7 confirms on the real head the same effect Table 8 showed on the synthetic one: at 50% cache the attention-aware keys (blue, panel (a)) sit almost exactly on the full-attention distribution (black) while SVD

[3] We use SmolLM2-135M for the factor-level and real-head visualizations because its small hidden size and few heads keep a single head's attention map, singular spectrum, and per-token factor weights legible in one figure, and the full per-head SVD runs on CPU; the same attention-aware effect holds on the larger models used in our main quantitative experiments.

Table 8: SVD on raw KV vs. attention-aware truncation at matched rank on a synthetic head whose query-relevant directions are misaligned with K's dominant singular directions. At equal rank the attention-aware basis drives the attention-relevant error far below SVD on raw KV for both K and V, and reaches it exactly once the retained rank spans the weighted subspace.

Attention-relevant error (matched rank)	Rank r	SVD on raw KV	Attention-aware
$\|Q(K-K_r)^\top\|_F \,/\, \|QK^\top\|_F$	4	0.976	**0.267**
$\|P(V-V_r)\|_F \,/\, \|PV\|_F$	8	0.923	**0.643**
$\|P(V-V_r)\|_F \,/\, \|PV\|_F$	16	0.608	**0.000**

on raw KV (red) distorts it, and the query-relevant logit error stays more than $10\times$ lower across budgets (0.013 vs. 0.141 at 50%, panel (b)).

(iii) Token saliency. Panels (c) and (d) tie the factors back to tokens. The per-token factor weights $|U|$ in panel (c) group the leading K factors into roughly semantic clusters, for example a colour factor {color, purple, green} identified from these weights, and panel (d) lists the top-contributing tokens of each factor with their energy shares. Row-weighting each token by its attention mass $a_i = \sum_q P_{qi}$ reconstructs the heavily attended tokens, such as the first-token sink, most faithfully, lowering $\|P(V-V_r)\|_F / \|PV\|_F$ from 0.154 to 0.012 at 50% cache, roughly $13\times$. End-to-end on Phi-4-14B agentic benchmark (Figure 3) the same attention-aware basis adds $+0.13$ to 0.29 mean judge and $+7$ to 18 points pass@4 over SVD on raw KV, reaching 4.067 / 62.5% at 50% cache, within 0.08 of dense.

Taken together, the three items justify all of `AttSVD`'s design choices at once: low-rank caching (both spectra decay), asymmetric per-matrix ranks ($r_{\widetilde{K}} \ll r_{\widetilde{V}}$), and the attention-aware basis (blue $\neq$ red).

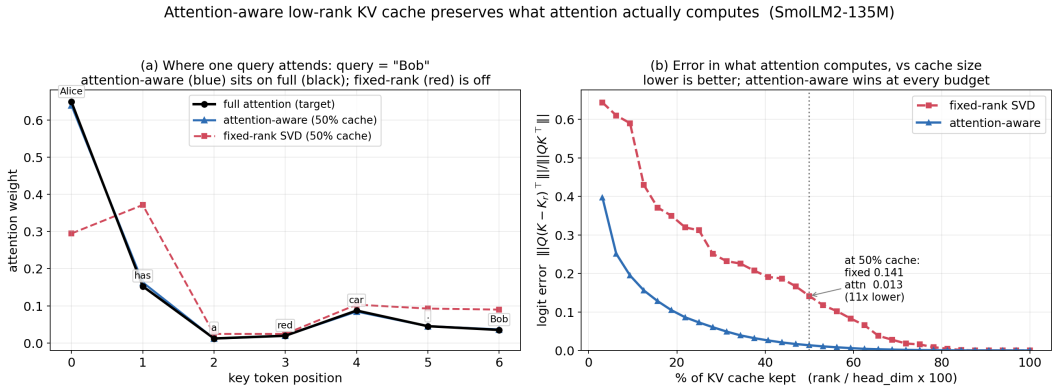


Figure 7: The attention-aware cache preserves what attention computes (a real layer/head of SmolLM2-135M). *(a)* At 50% cache the attention-aware keys (blue) lie almost on the full-attention target (black), while SVD on raw KV (red) distorts it. *(b)* Query-relevant logit error vs. cache size: attention-aware stays below SVD on raw KV at every budget (0.013 vs. 0.141 at 50%). Real-model analogue of Table 8.

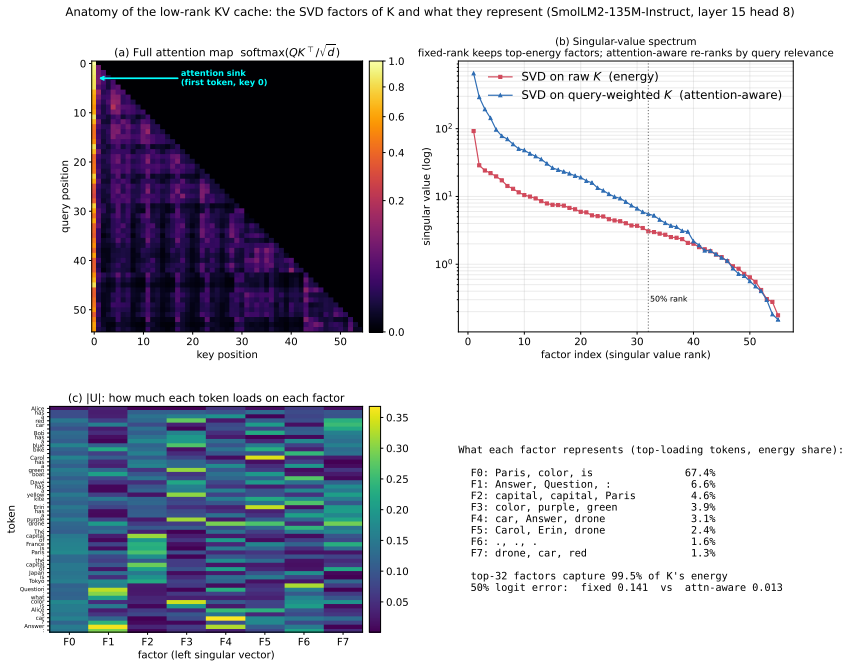


Figure 5: SVD factors of the low-rank KV cache, K (SmolLM2-135M, one layer/head). *(a)* attention map; *(b)* K's singular spectrum, SVD on raw KV (energy) vs. attention-aware (query-weighted); *(c)* per-token factor weights $|U|$; *(d)* top tokens per factor with energy shares.

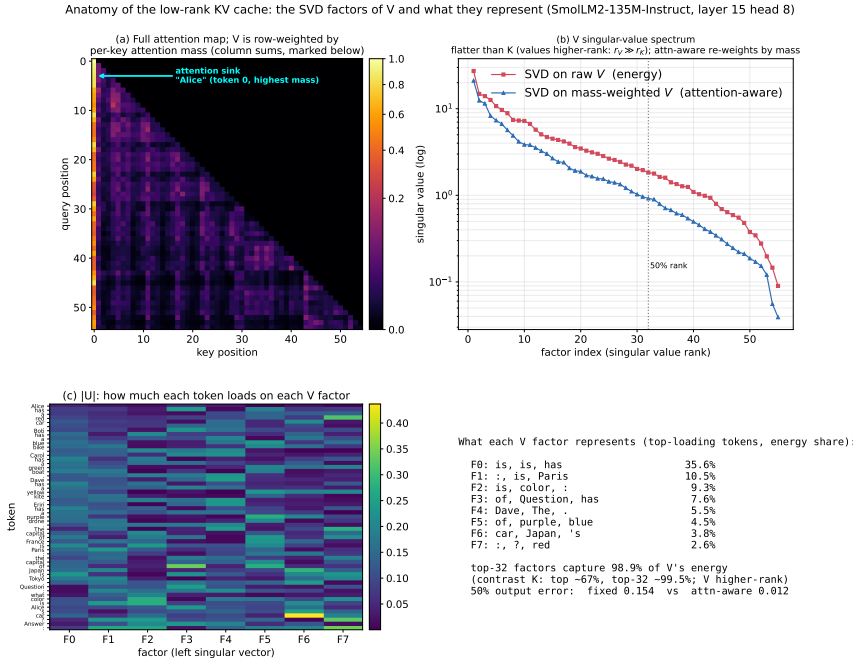


Figure 6: SVD factors of the low-rank KV cache, V (same layer/head and prompt as Figure 5). *(a)* attention map, with the per-key attention mass that weights V; *(b)* V's singular spectrum, SVD on raw KV vs. attention-aware (mass-weighted); *(c)* per-token factor weights $|U|$; *(d)* top tokens per factor with energy shares.