iS-KV: ONLINE LOW-RANK KV CACHE COMPRESSION VIA BLOCK-INCREMENTAL SVD

Anonymous authors
Paper under double-blind review

ABSTRACT

Long chain-of-thought reasoning substantially increases KV-cache memory during autoregressive decoding, as every generated token introduces new key and value states and causes the cache to grow linearly with decoding length. Existing KV-cache compression methods typically control this growth through token eviction, but irreversible deletion can remove historical states that later reasoning may need to revisit. SVD-based low-rank compression provides an alternative by retaining all positions with a more compact representation. However, extending it from a fixed prompt cache to online decoding is non-trivial. Through our investigation, we find that if the basis is updated for new tokens while old tokens keep their coordinates in the old basis, the stored history drifts substantially. Based on this observation, we propose ***iS*-KV**, an online low-rank KV-cache compression method for long-horizon reasoning. *iS*-KV keeps a recent window exact while incrementally folding older states into bounded-rank representations. As the low-rank basis evolves, it synchronizes historical coordinates with the updated basis to maintain representation consistency. On DeepSeek-R1-Distill-Llama-8B, *iS*-KV achieves 82.6% accuracy at 4.06× persistent-KV compression, close to the original model's 83.6%. On Qwen3-8B, it achieves 89.2% accuracy at 5.64× compression. Under matched memory budgets, *iS*-KV consistently outperforms token-eviction baselines.

1 INTRODUCTION

Long chain-of-thought (CoT) reasoning increases the memory needed during generation. Each generated token adds key and value states to the KV cache, so a long reasoning trace can require substantially more KV cache storage than its prompt (Guo et al., 2025; Yang et al., 2025). The challenge is to reduce the KV storage cost during the decoding-phase without losing information that later reasoning may need.

Two observations motivate our design. First, old tokens can become important again. A token that receives little attention at the current step may receive strong attention thousands of tokens later. Once evicted, it cannot be recovered. On Qwen3-8B, for example, R-KV reduces MATH-500 accuracy from 94.4% to 68.8% at about 5.6× compression. Second, KV states have strong low-rank structure, so old tokens can be compressed instead of deleted. Doing this online requires care: when the low-rank basis changes as new tokens arrive, the coordinates of previously compressed tokens must change with it. Otherwise, the stored history no longer represents the same states.

Token-eviction methods reduce cache size by keeping selected positions and removing the rest, typically using attention scores, recency, or redundancy (Zhang et al., 2023; Xiao et al., 2024; Cai et al., 2026). These signals describe which tokens appear important under the queries seen so far, but they cannot predict all future queries. Figure 1(a) illustrates this issue on long reasoning traces from DeepSeek-R1-Distill-Llama-8B and Qwen3-8B on MATH-500, AIME 2024, and AIME 2025. At decoding step 1024, we mark the 75% of tokens with the lowest recent attention, without removing them. As decoding continues, many of these tokens become important again. On AIME, about 30% receive strong attention again within the next 8K tokens, and this fraction grows to about 60% by 30K tokens. At matched memory, actual eviction by R-KV loses 5.4–11.0 points on DeepSeek-R1-Distill-Llama-8B and 23.0–33.6 on Qwen3-8B (Table 1).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

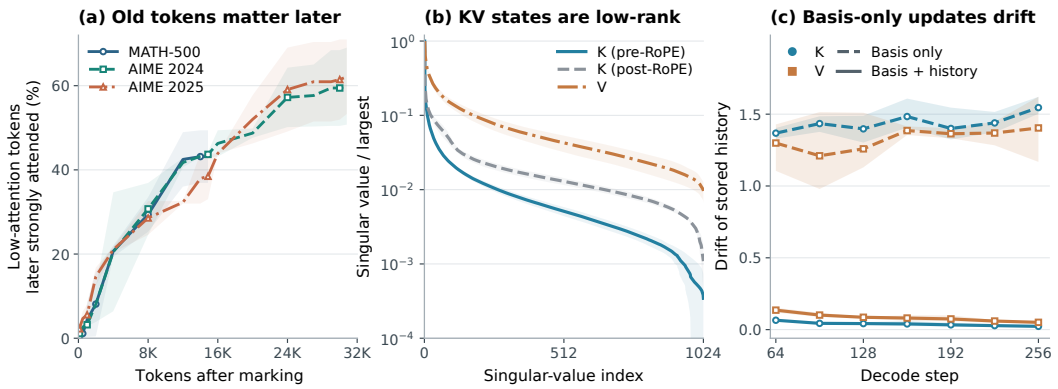

Figure 1: **Keep old positions, compress their KV states, and update them together.** *(a)* At step 1024, we mark the 75% least-attended tokens (none are removed) and plot the share that later receives strong attention. *(b)* Singular values of generated K/V states, divided by the largest. *(c)* Drift of the stored history caused by each update. Dashed: the basis is updated but old coordinates are kept. Solid: the basis and old coordinates are updated jointly, which equals the *iS*-KV update in exact arithmetic. Bands show observed ranges; Appendix A.3 gives all settings.

Low-rank compression offers a different approach: keep every token position, but store each old state more compactly. Each position is represented by a short coefficient vector over a basis shared across positions. Figure 1(b) shows that generated KV states are indeed low-rank. Keys have the fastest singular-value decay before rotary positional embeddings (RoPE), consistent with prior work (Sun et al., 2025), while values decay more slowly and are harder to compress. The main difficulty appears during online decoding. As new states arrive, the low-rank basis must adapt. But the coefficients of old states were computed in the previous basis. If the basis is updated while the old coefficients remain unchanged, the reconstructed historical states change as well. In other words, the same coefficients represent different vectors under a new basis. Figure 1(c) measures this drift. At step 256, updating only the basis gives a drift of 1.55 for keys and 1.40 for values. Updating the old coefficients together with the basis reduces these values to 0.023 and 0.051. This result shows that adapting the basis and preserving old states must be handled together.

We therefore propose *iS*-KV, an online low-rank KV-cache compression method designed for long-horizon reasoning. *iS*-KV keeps recent states at their original precision and folds older states into separate low-rank K and V representations. Using block incremental SVD (Brand, 2006), it updates both the low-rank basis and the coefficients of all previously compressed positions. Keys are compressed before RoPE, where their low-rank structure is stronger. During attention, the reconstructed historical states are combined with the recent exact states. Every token position thus remains available to future queries, although older states are approximate. A fixed rank reduces the storage per old token, so *iS*-KV slows cache growth rather than making it constant in size.

On MATH-500, *iS*-KV achieves 82.6% accuracy at $4.06\times$ persistent-KV compression on DeepSeek-R1-Distill-Llama-8B, close to the original model's 83.6%. On Qwen3-8B, it achieves 89.2% at $5.64\times$ compression, compared with 68.8% for matched-memory R-KV. It outperforms both R-KV and SnapKV at all six matched-memory operating points and remains within 1.0–6.6 accuracy points of the original model. On AIME 2024 at rank 192, *iS*-KV on Qwen3-8B matches the original model with 23 of 30 correct answers, compared with 13 for matched-memory R-KV. Memory matching counts low-rank factors, exact entries, and R-KV's staging buffer at the original model's output lengths. This avoids counting shorter generations as compression gains. Our contributions are:

- **Online K/V compression.** We develop *iS*-KV to retain every token position, keeping recent states exact while storing older states in bounded-rank form.
- **Consistent history updates.** We show that a basis-only update makes the stored history drift, and update the basis and old coefficients jointly via block incremental SVD.

- **Reasoning accuracy at matched memory.** On two 8B models, *iS*-KV outperforms R-KV and SnapKV at all six matched MATH-500 operating points. AIME, RULER, ablation, and latency results further show its strengths and limitations.

2 Related Work

Token selection and eviction. Token-selection methods reduce KV cache storage requirement by keeping a subset of cached positions. H2O retains heavy hitters and recent tokens (Zhang et al., 2023), while StreamingLLM combines attention sinks with a sliding window (Xiao et al., 2024). FastGen selects head-specific policies through attention profiling (Ge et al., 2024). For long prompts, SnapKV selects positions using a trailing observation window (Li et al., 2024), and PyramidKV allocates different cache budgets across layers (Cai et al., 2024). R-KV targets long reasoning with redundancy-aware selection (Cai et al., 2026). These methods preserve selected states at the expense of historical coverage. *iS*-KV instead keeps every position in exact or low-rank form, trading representation precision rather than token availability.

Attention efficiency and cache quantization. Other approaches reduce different parts of the memory cost. FlashAttention reduces attention IO (Dao et al., 2022); PagedAttention reduces cache fragmentation and enables sharing (Kwon et al., 2023); GQA reduces the number of KV heads (Ainslie et al., 2023). Quantization reduces bits per element: KIVI uses per-channel keys and per-token values (Liu et al., 2023), while KVQuant combines pre-RoPE key quantization with separate outlier storage (Hooper et al., 2025). Our focus is instead the number of coordinates stored per historical state, without changing the pretrained attention architecture. We do not evaluate additional low-bit quantization or serving-system integration.

Low-rank KV representations. Low-rank methods exploit redundancy across features without discarding token positions. Palu factorizes projection weights and caches low-dimensional intermediate states (Chang et al., 2024); Eigen Attention performs attention in reduced spaces (Saxena et al., 2024). ShadowKV combines low-rank pre-RoPE keys, value offloading, and sparse retrieval (Sun et al., 2025). More recent work learns bases against decoder-layer output error in StiefAttention (Benfenati et al., 2026), adapts head/block ranks in STAR-KV (Bhatnagar et al., 2026), and benchmarks data-dependent compressibility in KV-CoRE (Chen et al., 2026). These studies inform basis construction and rank choice. Our question is how to incorporate newly decoded states while keeping historical coordinates consistent with an evolving basis.

Online low-rank updates. For newly generated states, *iS*-KV maintains a shared historical factorization for each K/V path and updates its basis and old coordinates together. The update uses Brand's classical incremental SVD (Brand, 2002; 2006), not a new matrix-decomposition primitive. Its small core represents the stored history and incoming block before rank truncation (Proposition 1). Our contribution is this coordinated cache update with exact recent states, evaluated on long reasoning at matched persistent-memory budgets.

3 *iS*-KV

iS-KV compresses the growing KV cache while retaining a representation of every token position. Prefill initializes exact states and a low-rank history; decoding jointly updates the basis and historical coefficients as unprotected rows are incorporated in blocks. Figure 2 illustrates the workflow, with prefill and decoding specified in Algorithms 1 and 2 in Appendix B.

3.1 Objective and Prefill Compression

Consider a single Transformer layer processing a prompt of L tokens, with H_{kv} KV heads of dimension D. Concatenating the KV heads along the feature dimension yields key and value matrices $K, V \in \mathbb{R}^{L\times d}$, where $d = H_{\mathrm{kv}}D$ and each row corresponds to one token. Following Sun et al. (2025), we represent keys before RoPE (pre-RoPE keys). For $X \in \{K, V\}$, we partition the prompt positions $\{1, \ldots, L\}$ into an exact set $\mathcal{E}_X$ and a compressed set $\mathcal{C}_X$. Let $X_{\mathcal{C}_X} := X[\mathcal{C}_X] \in \mathbb{R}^{n_X\times d}$ denote the original submatrix selected for compression, where $n_X = |\mathcal{C}_X|$. *iS*-KV stores $X[\mathcal{E}_X]$

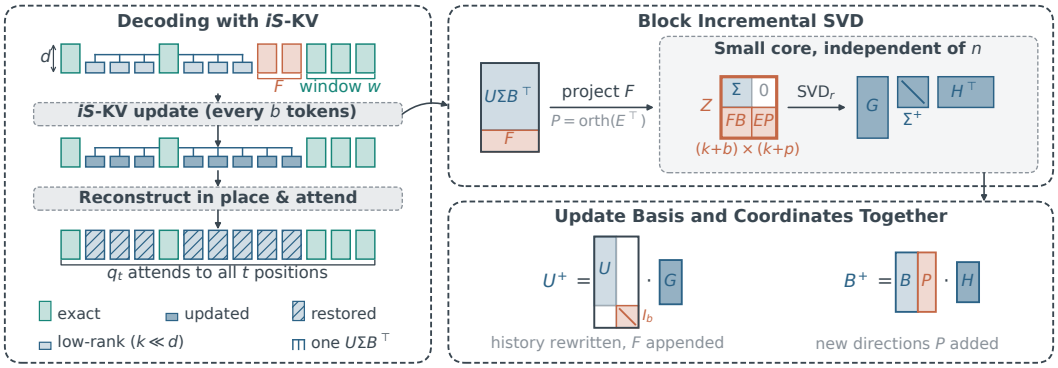

Figure 2: Overview of *iS*-KV. The KV cache grows while *iS*-KV keeps the recent window of w tokens exact (teal) and folds older tokens into a compact low-rank history (blue), one $U\Sigma B^\top$ per block. Every b tokens, the pending block F (orange) is merged in by a Block Incremental SVD that updates basis and coordinates together and solves only a small core, independent of the history length n. Each position can be reconstructed for query q_t attends to all t positions.

unchanged at its original precision and represents $X_{\mathcal{C}_X}$ by a low-rank approximation $\widehat{X}_{\mathcal{C}_X}$ of the same dimensions. We retain the original position indices for reconstruction.

Given this partition and a rank budget r_X, *iS*-KV chooses $\widehat{X}_{\mathcal{C}_X}$ to minimize the squared reconstruction error in the Frobenius norm:

$$\min_{\widehat{X}_{\mathcal{C}_X} \in \mathbb{R}^{n_X \times d}} \quad \|X_{\mathcal{C}_X} - \widehat{X}_{\mathcal{C}_X}\|_F^2 \quad \textit{st.} \quad \mathrm{rank}(\widehat{X}_{\mathcal{C}_X}) \le r_X. \tag{1}$$

The truncated singular value decomposition (SVD), denoted by $(U_X, \Sigma_X, B_X) = \mathrm{SVD}_{r_X}(X_{\mathcal{C}_X})$, retains the leading $k_X = \min(r_X, n_X, d)$ singular components. Here, $U_X \in \mathbb{R}^{n_X \times k_X}$ and $B_X \in \mathbb{R}^{d \times k_X}$ contain the retained left and right singular vectors, with orthonormal columns; $\Sigma_X \in \mathbb{R}^{k_X \times k_X}$ is diagonal with the retained singular values. The cache stores U_X, B_X, and the diagonal of Σ_X; each row of $U_X \Sigma_X$ contains a token's coefficients in the shared basis B_X.

We determine which rows remain exact during prefill, whose attention uses the original keys and values. For keys, we retain a short local suffix of the prompt, including any incomplete chunk, and partition the remaining prefix into fixed-size contiguous chunks. For chunk c and KV head h, let $k_{i,h}^{\mathrm{rot}} \in \mathbb{R}^D$ denote the post-RoPE key at position i, and $\bar{k}_{c,h}^{\mathrm{rot}}$ the mean key in that chunk. Following the outlier selection criterion of Sun et al. (2025), we score each chunk by $s_{c,h} = \min_{i \in c} \cos(k_{i,h}^{\mathrm{rot}}, \bar{k}_{c,h}^{\mathrm{rot}})$. We select up to m lowest-scoring chunks per KV head as outlier chunks; the union of their token positions across heads, together with the local suffix, defines the protected positions $\mathcal{P}_K$. Selection uses post-RoPE keys, but each protected position retains its full pre-RoPE key row across all KV heads.

For values, we define the protected prompt positions $\mathcal{P}_V$ as all prompt positions within ρ tokens of any position in $\mathcal{P}_K$, including $\mathcal{P}_K$ itself. We initialize an exact recent window of size w with the last $\min(w, L)$ prompt values and denote their positions by W_V. The initial exact sets are therefore $\mathcal{E}_K = \mathcal{P}_K$ and $\mathcal{E}_V = \mathcal{P}_V \cup W_V$. We factorize the remaining rows with the truncated SVD above. The protection sets $\mathcal{P}_K, \mathcal{P}_V$ remain fixed during decoding, while the recent window advances with generation. Protection may cover an entire short prompt, leaving an empty low-rank history. The chunk size and protection parameters are specified in Appendix A.

3.2 Block-Incremental Decoding

During decoding, new key and value rows remain exact in recent windows of size w. Unprotected rows leaving these windows remain exact; their positions enter FIFO queues $\mathcal{Q}_K$ for keys and $\mathcal{Q}_V$ for values. For $X \in \{K, V\}$, $\mathcal{Q}_X$ denotes the corresponding queue. Regular updates consume the oldest b pending rows once $|\mathcal{Q}_X| \ge b$, retaining at most r_X singular components. Before regular value updates begin, a one-time refit is performed: once $|\mathcal{Q}_V| \ge r_V$, we append the oldest r_V pending rows to the reconstructed prompt value history and compute a truncated SVD with rank budget r_V. Thus, r_V controls both the value rank budget and the initial refit size.

Each regular update represents the history and new block in a common updated basis. Omitting the subscript X, let $A = U\Sigma B^\top \in \mathbb{R}^{n\times d}$ denote the maintained history with $k \le r$ components and rank budget r, and $F \in \mathbb{R}^{b\times d}$ the next pending block. We seek an approximation $\widehat{M}$ of their concatenation $M = [A; F]$:

$$\min_{\widehat{M}\in\mathbb{R}^{(n+b)\times d}} \quad \|M - \widehat{M}\|_F^2 \quad \textit{st.} \quad \operatorname{rank}(\widehat{M}) \le r. \tag{2}$$

The target M contains the maintained approximation A and exact block F; the original compressed rows are no longer stored. An empty low-rank history is initialized by a truncated SVD of F. Otherwise, block incremental SVD (Brand, 2006) updates the factors without reconstructing A. To incorporate directions outside B, we compute $E = F - (FB)B^\top$ and construct a residual basis $P \in \mathbb{R}^{d\times p}$ by QR and SVD (Appendix A.1). We retain directions whose singular values exceed the numerical threshold, with $p \le \min(b, d-k)$ and $p = 0$ when none are retained. Queue occupancy determines update timing; residual thresholding determines basis expansion. Expressing A and F in the augmented basis $[B\ P]$ yields

$$M \approx \begin{bmatrix} U & 0 \\ 0 & I_b \end{bmatrix} Z[B\ P]^\top, \qquad Z = \begin{bmatrix} \Sigma & 0 \\ FB & FP \end{bmatrix}, \tag{3}$$

where I_b is the $b \times b$ identity matrix. This reduces the SVD to a $(k+b)\times(k+p)$ core, independent of the history length n. Computing $(G, \Sigma^+, H) = \mathrm{SVD}_r(Z)$, with $Z \approx G\Sigma^+H^\top$, gives

$$U^+ = \begin{bmatrix} U & 0 \\ 0 & I_b \end{bmatrix} G, \qquad B^+ = [B\ P]H. \tag{4}$$

The updated factors reconstruct $\widehat{M}^+ = U^+\Sigma^+(B^+)^\top$. The first n and last b rows of $U^+\Sigma^+$ contain the coefficients of A and F, respectively, in the updated basis B^+.

In exact arithmetic, with orthonormal columns in U, B and a complete orthonormal residual basis P, (3) is exact and the core SVD solves (2), with squared error $\sum_{i>r}\sigma_i(Z)^2$. Residual thresholding and finite precision introduce additional error. We periodically reorthogonalize the factors using QR and SVD, preserving their product in exact arithmetic. Numerical settings and proofs are given in Appendix A.

3.3 Reconstruction and Storage Cost

At each decoding step, $\mathcal{E}_X$ and $\mathcal{C}_X$ denote the current exact and compressed position sets, with $\mathcal{E}_X$ containing protected, recent, and pending positions. Let $X_{\mathrm{exact}}[i]$ denote the stored exact row at position i. The reconstructed cache $\widetilde{X}$ is defined in token order by

$$\widetilde{X}[i] = \begin{cases} (U_X\Sigma_X B_X^\top)[j], & i = \mathcal{C}_X[j], \\ X_{\mathrm{exact}}[i], & i \in \mathcal{E}_X. \end{cases} \tag{5}$$

Reconstruction aligns keys and values by their original token positions despite their different compressed position sets. After restoring head dimensions, we apply RoPE to keys at their original positions. For a rotated query q_t^{rot} associated with KV head h, the attention output is

$$o_t = \mathrm{softmax}\Big(q_t^{\mathrm{rot}}(\widetilde{K}_h^{\mathrm{rot}})^\top/\sqrt{D}\Big)\,\widetilde{V}_h. \tag{6}$$

Let $n_X = |\mathcal{C}_X|$ and $e_X = |\mathcal{E}_X|$ denote the numbers of compressed and exact rows. The factors and exact rows use β_f and β_e bytes per scalar, respectively. Excluding metadata and temporary dense reconstruction buffers, the per-layer persistent storage is

$$M_{\textit{iS-KV}} = \beta_f \textstyle\sum_{X\in\{K,V\}} k_X(n_X + d + 1) + \beta_e d \sum_{X\in\{K,V\}} e_X. \tag{7}$$

At equal rank $r < d$ and precision, each compressed key-value pair requires $2r$ coefficients instead of $2d$ scalars, plus shared factors. Storage therefore grows as $O(nr)$ with history length. Updating historical coefficients costs $O(nr(r+b))$ per block; reorthogonalization and reconstruction also depend on n.

4 EXPERIMENTS

We evaluate reasoning accuracy, key/value compression and basis adaptation, long-input retrieval, and generation time and memory.

4.1 LONG-COT REASONING

Models, data, and evaluation. Our main evaluation uses all 500 MATH-500 questions (Hendrycks et al., 2021) with DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025) and Qwen3-8B (Yang et al., 2025). We report final-answer accuracy from one sampled generation per question, using temperature 0.6, top-$p = 0.95$, top-$k = 50$, and a total prompt-plus-generation limit of 16,384 tokens. Prompts follow each model's chat template and request a boxed final answer; empty or non-finite outputs count as incorrect. We compare with the original model and the token-eviction methods R-KV (Cai et al., 2026) and SnapKV (Li et al., 2024). The original model, R-KV, and *iS*-KV use paired questions, prompts, question-specific seeds, and the same answer evaluator. SnapKV results are retained from the existing baseline summaries; Table 1 distinguishes their inferred correct counts from directly counted outcomes.

Compression settings and memory matching. For MATH-500, *iS*-KV uses ranks $r \in \{64, 96, 128\}$, a 128-token exact recent window on each path, and 64-row updates. Persistent factors use FP16, while updates and reconstruction use FP32; further settings are given in Appendix A. We match baselines by persistent KV bytes, not by nominal rank or retained-token count. The accounting includes protected prompt states, exact recent states, low-rank coefficients, singular values, bases, and baseline staging storage. All footprints are evaluated at the original model's output lengths, so different stopping lengths do not change the reference used for matching. These ratios exclude model weights and temporary reconstruction tensors and therefore do not measure peak GPU-memory savings.

MATH-500: higher accuracy at matched memory. Table 1 shows that *iS*-KV outperforms both eviction baselines at all six reported model–budget settings. On DeepSeek-R1-Distill-Llama-8B, rank 128 reaches 82.6% accuracy at $4.06\times$ compression, compared with 83.6% for the original model and 78.2% for matched R-KV. At $4.81\times$ and $5.92\times$ compression, the gains over R-KV are 5.4 and 5.6 percentage points, respectively. The advantage also holds on Qwen3-8B: rank 96 achieves 89.2% at $5.64\times$ compression, compared with 68.8% for R-KV and a reported 73.17% for SnapKV. Even at rank 64, Qwen3-8B retains 87.8% accuracy with $7.12\times$ compression, versus 94.4% for the original model.

Accuracy degrades gradually as the rank decreases. Reducing rank from 128 to 64 lowers *iS*-KV accuracy by 4.4 percentage points on DeepSeek-R1-Distill-Llama-8B and 4.2 points on Qwen3-8B. Over the corresponding matched budgets, R-KV loses 5.6 and 10.6 points, respectively. The advantage persists across ranks, but the growing gap to the original model shows that retaining positions alone cannot compensate for lost representation fidelity.

Comparison with OjaKV. We run the released OjaKV implementation (Zhu et al., 2026) on DeepSeek-R1-Distill-Llama-8B and MATH-500 with the 16K total-token limit used above. Its released checkpoint achieves 62.0% accuracy at approximately $1.30\times$ KV compression, compared with the original model's 83.6% and *iS*-KV's 78.2–82.6% at 4.06–$5.92\times$ compression (Table 2). We also reduce OjaKV's ranks to target the same compression range as *iS*-KV. All 118 generations collected across these three settings fall into repetitive loops and reach the length limit without EOS. We stop these runs early and report the observed failures rather than an unmeasured accuracy.

AIME: the advantage persists at a longer generation budget. We further evaluate Qwen3-8B on AIME 2024 and AIME 2025 with a 32,768-token total limit. Each dataset contains 30 questions, with one sampled generation per question. We match persistent KV memory at the original model's output lengths, including R-KV's 128-token staging buffer; the horizontal axis in Figure 3 gives the corresponding R-KV retained-token budget. These runs store low-rank factors in BF16 to avoid the FP16 overflow observed at this length (Appendix A.2); both formats occupy two bytes per element. As Figure 3 shows, *iS*-KV exceeds R-KV at every tested budget on both datasets. On AIME 2024,

Table 1: **MATH-500 accuracy at matched persistent-KV footprints.** The evaluation uses 500 questions and a 16K total-token limit. KV ratio and compression are measured against the original model at its output lengths, counting R-KV's 128-token staging buffer and SnapKV's 128-token compression interval. Shading marks *iS*-KV; Δ is in percentage points. SnapKV correct counts are inferred by rounding its reported accuracy times 500, not counted from per-question outputs; its reported percentages are not all compatible with single-sample, 500-question accuracy increments.

Method	Setting	KV ratio ↓	Compression ↑	Correct	Accuracy ↑	Δ
DeepSeek-R1-Distill-Llama-8B						
Original	–	100.00%	1.000×	418/500	**83.6**	0.0
R-KV	budget 748	24.63%	4.060×	391/500	78.2	−5.4
SnapKV	$B = 636$	24.64%	4.059×	351/500	70.1	−13.5
iS-KV	$r = 128$	24.63%	4.060×	413/500	82.6	−1.0
R-KV	budget 562	20.78%	4.813×	382/500	76.4	−7.2
SnapKV	$B = 517$	20.80%	4.807×	337/500	67.4	−16.2
iS-KV	$r = 96$	20.79%	4.811×	409/500	81.8	−1.8
R-KV	budget 409	16.89%	5.920×	363/500	72.6	−11.0
SnapKV	$B = 396$	16.90%	5.917×	322/500	64.3	−19.3
iS-KV	$r = 64$	16.90%	5.917×	391/500	78.2	−5.4
Qwen3-8B						
Original	–	100.00%	1.000×	472/500	**94.4**	0.0
R-KV	budget 797	21.35%	4.683×	357/500	71.4	−23.0
SnapKV	$B = 794$	21.36%	4.683×	407/500	81.30	−13.10
iS-KV	$r = 128$	21.36%	4.681×	460/500	92.0	−2.4
R-KV	budget 638	17.73%	5.641×	344/500	68.8	−25.6
SnapKV	$B = 637$	17.72%	5.644×	366/500	73.17	−21.23
iS-KV	$r = 96$	17.72%	5.644×	446/500	89.2	−5.2
R-KV	budget 479	14.06%	7.113×	304/500	60.8	−33.6
SnapKV	$B = 479$	14.06%	7.113×	325/500	65.04	−29.36
iS-KV	$r = 64$	14.05%	7.118×	439/500	87.8	−6.6

Table 2: **OjaKV on DeepSeek-R1-Distill-Llama-8B and MATH-500.** The upper block uses all 500 questions at a 16K total-token limit. OjaKV uses its released checkpoint with error-budget parameter 0.075. The lower block reduces its aggregate rank to target higher compression; these runs were stopped early. "–" means not scored, not zero accuracy.

Method / setting	Compression	Acc. (%)	Observed results
Original model	1.00×	83.6	418/500 correct
OjaKV (released)	≈ 1.30×	62.0	310/500 correct
iS-KV (rank 128)	4.06×	82.6	413/500 correct
iS-KV (rank 96)	4.81×	81.8	409/500 correct
iS-KV (rank 64)	5.92×	78.2	391/500 correct
OjaKV with reduced ranks; compression values are targets			
OjaKV (rank 232)	4.06×	–	30/30 loop to limit
OjaKV (rank 192)	4.81×	–	28/28 loop to limit
OjaKV (rank 152)	5.92×	–	60/60 loop to limit

increasing rank from 64 to 192 raises accuracy from 43.3% to 76.7%, while matched R-KV rises from 13.3% to 43.3%. At rank 192, *iS*-KV answers 23 of 30 questions correctly, as does the original model. AIME 2025 remains more difficult: the same rank achieves 56.7%, above R-KV's 40.0% but below the original model's 63.3%. With one sample per question, each answer changes accuracy by 3.3 percentage points; an equal total does not establish statistical equivalence.

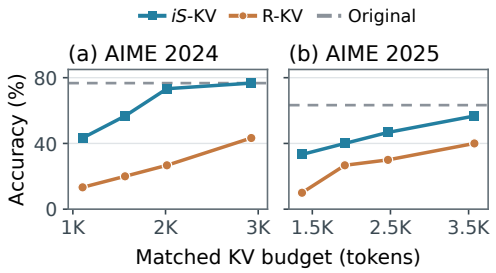

Figure 3: **AIME accuracy on Qwen3-8B.** Ranks 64, 96, 128, and 192 (left to right) use memory budgets matched to R-KV. Dashed lines show the original model.

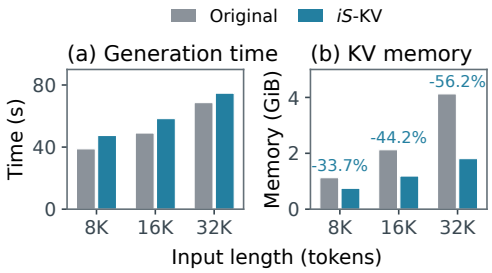

Figure 4: **Generation time and KV memory.** DeepSeek-R1-Distill-Llama-8B on an A40, rank 128: prefill plus 1,024 decode steps. Labels show KV savings over the original model.

4.2 Ablation Study

We ablate key/value compression and basis adaptation on a fixed prefix of 80 MATH-500 questions with DeepSeek-R1-Distill-Llama-8B, paired prompts and seeds, and one generation per question. All compressed variants use rank 96 and the same exact-recent policy. For the frozen-basis variant, both bases are initialized from the prompt and kept fixed during decoding.

Table 3: **Component ablation on a fixed 80-question MATH subset.** Checks/crosses enable/disable each component; frozen bases remain at their prompt initialization. Compressed variants share rank 96 and the exact-recent policy, with one sample per question. Compression is the persistent-KV compression ratio.

Variant	Low-rank K	Low-rank V	Online update	Compression	Correct	Accuracy (%)
Original	×	×	×	1.00×	67/80	83.8
K-only	✓	×	✓	1.64×	**68/80**	**85.0**
V-only	×	✓	✓	1.64×	65/80	81.2
Frozen basis	✓	✓	×	**4.68**×	51/80	63.7
iS-KV	✓	✓	✓	4.61×	65/80	81.2

Key and value compression have different effects. In Table 3, K-only compression answers 68/80 questions correctly, compared with 67/80 for the original model. V-only compression and joint K+V compression each answer 65/80 correctly. Value compression appears more sensitive on this subset, with no further aggregate loss from compressing keys. Equal totals need not imply identical errors, nor does the one-question gain establish an improvement over the original model. This comparison fixes per-path rank, not total memory.

Online basis adaptation matters. Keeping the bases fixed reduces the score from 65/80 to 51/80, a 17.5-percentage-point drop under the same rank and recent-window settings. This supports adapting beyond the prompt-only subspace. The answer-level ablation complements Figure 1(c), which instead measures reconstruction drift from unsynchronized historical coordinates.

4.3 Long Prefill Evaluation

Setup. We evaluate 11 RULER tasks (Hsieh et al., 2024) at 64K context on GLM-4-9B-1M, Llama-3-8B-Instruct-Gradient-1048k, and Llama-3.1-8B, using ten examples per task. *iS*-KV and the original model use identical inputs; we report task-specific percentage scores and their unweighted mean. At initialization, *iS*-KV retains selected prompt chunks and protected neighborhoods at their original precision and factorizes the remaining states, as detailed in Appendix A. With short outputs, this primarily tests prompt-history access under the two configurations in Table 4.

Access to long input histories. Rank 160 matches the original models' 100.0 on S1–S3. Its means are 87.8, 86.3, and 79.0 on GLM-4-9B-1M, Llama-3-8B-Instruct-Gradient-1048k, and Llama-3.1-

Table 4: **Prefill evaluation on 11 RULER tasks at 64K context.** Scores are percentages, with ten examples per task; Avg. is the task mean. S1–S3 denote single-needle retrieval; MK, MQ, and MV denote multi-key, multi-query, and multi-value retrieval; VT, FWE, and QA denote variable tracking, frequent-word extraction, and question answering. Rank 96 uses 128 exact-recent rows and 16-row folds. Rank 160 uses single-row K updates with slack 8 and a 512-row exact-recent V window. These are different configurations, not a rank-only ablation. Comp. is compression ratio.

Model	Method	S1	S2	S3	MK1	MK2	MQ	MV	VT	FWE	QA1	QA2	Avg.	Comp. ↑
GLM-4-9B-1M	Original	100.0	100.0	100.0	100.0	100.0	100.0	98.8	100.0	78.2	72.4	36.5	89.6	1.00×
	iS-KV (rank 160)	100.0	100.0	100.0	100.0	100.0	100.0	85.7	100.0	71.4	72.4	36.5	87.8	2.67×
	iS-KV (rank 96)	100.0	100.0	100.0	100.0	95.2	92.7	82.0	100.0	71.4	72.4	36.5	86.4	3.89×
Llama-3-8B-Instruct-Gradient-1048k	Original	100.0	100.0	100.0	100.0	100.0	94.6	93.6	80.6	80.4	71.1	39.6	87.3	1.00×
	iS-KV (rank 160)	100.0	100.0	100.0	100.0	100.0	88.1	92.1	80.6	80.4	69.0	39.6	86.3	2.29×
	iS-KV (rank 96)	100.0	100.0	100.0	100.0	100.0	62.0	83.8	80.6	85.1	69.0	39.6	83.6	2.54×
Llama-3.1-8B	Original	100.0	100.0	100.0	100.0	61.3	100.0	100.0	94.0	76.6	58.3	51.5	85.6	1.00×
	iS-KV (rank 160)	100.0	100.0	100.0	80.7	62.3	79.7	68.8	94.0	76.0	56.2	51.5	79.0	2.23×
	iS-KV (rank 96)	100.0	100.0	85.7	69.8	62.3	46.6	41.6	92.6	87.4	56.2	46.7	71.7	2.47×

8B, versus 89.6, 87.3, and 85.6 for the original models. Rank 96 retains 100.0 on S1–S3 for GLM-4-9B-1M and Llama-3-8B-Instruct-Gradient-1048k, with means of 86.4, 83.6, and 71.7 across the models. Multi-query and multi-value retrieval show larger differences. MATH-500 and AIME primarily compress generated states because their short prompts normally remain exact. Here, the 64K prompts require compression before decoding, so the results measure retrieval from combined exact and low-rank prompt storage.

4.4 Efficiency

Setup. We use DeepSeek-R1-Distill-Llama-8B on an NVIDIA A40 at rank 128 and batch size one. Both methods receive identical synthetic 8K, 16K, and 32K inputs followed by exactly 1,024 greedy decode steps. We time prefill plus decoding, excluding model loading, extension compilation, and cache inspection, and count KV memory after decoding. Each method–length pair has one fresh-process run, executed serially without inference warmup.

Generation time. In Figure 4(a), *iS*-KV takes 47.45, 58.32, and 74.70 s at 8K, 16K, and 32K, versus 38.83, 49.04, and 68.63 s for the original model. The overhead decreases from 22.2% to 18.9% and 8.8%, including updates and reconstruction. These single-run measurements describe this workload, not a statistically stable latency difference.

Memory. In Figure 4(b), the original model requires 1.125, 2.125, and 4.125 GiB at 8K, 16K, and 32K; *iS*-KV uses 0.746, 1.185, and 1.808 GiB. Savings increase from 33.7% to 44.2% and 56.2% as more history enters the low-rank representation. This includes factors and exact protected, recent, and pending rows, preserving all positions without making memory constant in length. It excludes indices, unused capacity, reconstructed caches, and workspace.

5 Conclusion

Long chain-of-thought reasoning grows the KV cache with every generated token. Eviction methods shrink it by deleting tokens, but a token that looks unimportant now may matter later, and deletion cannot be undone. *iS*-KV instead keeps every token position and stores old states compactly, while recent states stay exact. This lowers the cost of each stored token, though memory still grows with the length of the trace. The difficulty is doing this online: the compact representation adapts as new tokens arrive, and the old states are stored relative to it. If it changes while they stay fixed, those states may not be reconstructed well. *iS*-KV updates both together with block-incremental SVD, so every update preserves the stored history. Our takeaway is that compressing a growing cache is a problem of representing every token faithfully beyond just choosing which tokens to keep.

AI USE STATEMENT

The authors used generative AI tools to assist with language polishing and manuscript drafting, preliminary literature exploration, code assistance, the organization and presentation of experimental data and figures, and feedback on methodology and manuscript drafts. The authors are responsible for reviewing and verifying all AI-assisted outputs, including literature-related suggestions and numerical or visual material. The authors take full responsibility for the study's ideas, methodology, experiments, analysis, claims, and final text.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

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.

Priyansh Bhatnagar, Ashkan Moradifirouzabadi, Se-Hyun Yang, SeungJae Lee, Jungwook Choi, and Mingu Kang. STAR-KV: Low-rank KV cache compression via soft thresholding for adaptive rank control. *arXiv preprint arXiv:2606.08382*, 2026.

M. Brand. Incremental Singular Value Decomposition of Uncertain Data with Missing Values. In *European Conference on Computer Vision (ECCV)*, volume 2350, pp. 707–720, May 2002. URL https://www.merl.com/publications/TR2002-24.

M. Brand. Fast Low-Rank Modifications of the Thin Singular Value Decomposition. *Linear Algebra and Its Applications*, 415(1):20–30, May 2006. URL https://www.merl.com/publications/TR2006-059.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, et al. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Zefan Cai, Wen Xiao, Hanshi Sun, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-Wen Chang, Jiuxiang Gu, Zhen Dong, et al. R-KV: Redundancy-aware KV cache compression for reasoning models. *Advances in neural information processing systems*, 38:60980–61005, 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: Compressing KV-cache with low-rank projection. *arXiv preprint arXiv:2407.21118*, 2024.

Jian Chen, Zhuoran Wang, Jiayu Qin, Ming Li, Meng Wang, Changyou Chen, Yin Chen, Qizhen Weng, and Yirui Liu. KV-CoRE: Benchmarking data-dependent low-rank compressibility of KV-caches in LLMs. *arXiv preprint arXiv:2602.05929*, 2026.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *International Conference on Learning Representations*, volume 2024, pp. 22975–22988, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization, 2025. URL https://arxiv.org/abs/2401.18079.

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

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, 2023. URL https://arxiv.org/abs/2309.06180.

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, 2024. URL https://arxiv.org/abs/2404.14469.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI : Plug-and-play 2bit KV cache quantization with streaming asymmetric quantization, 2023. URL https://www.researchgate.net/doi/10.13140/RG.2.2.28167.37282.

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/.

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, 2025. URL https://arxiv.org/abs/2410.21465.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks, 2024. URL https://arxiv.org/abs/2309.17453.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-hitter oracle for efficient generative inference of large language models, 2023. URL https://arxiv.org/abs/2306.14048.

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/.

A Implementation Details and Reproducibility

This appendix gives the settings behind the reported results. Appendix A.1 describes the method settings, Appendix A.2 the evaluation protocols, Appendix A.3 the probes behind Figure 1, Appendix A.4 the properties of the block update, and Appendix A.5 further numerical diagnostics.

A.1 Method Settings

Main configuration. The key and value caches use exact recent windows of 128 rows, block size 64, and residual tolerance $\tau = 10^{-6}$. Each update keeps at most the given rank budget. MATH-500 uses ranks 64, 96, and 128 on both models, with all other settings unchanged. Reorthogonalization runs after every 256 rows processed by regular block updates; these rows include initialization from an empty history but not the first value refit. The persistent factors U_X, B_X and the diagonal entries of Σ_X are stored in FP16; projection, residual QR, core SVD, reorthogonalization, and reconstruction use FP32. Factors are cast back to the storage precision after each update. Each right basis is stored explicitly, protected prompt keys are stored once, and no additional $\Sigma_X B_X^\top$ product is cached. Optional features of our code that this paper does not use are disabled: prompt sketches, persistent residual corrections, dynamic historical-token scoring, and integer quantization.

Prompt protection. During prefill, we partition post-RoPE keys into eight-token chunks, excluding the last four chunks and any final incomplete chunk as a local suffix. For each KV head, we score a chunk by the minimum cosine similarity between its token keys and their chunk mean, then select the 128 lowest-scoring chunks (or all available chunks for a shorter prompt). We take the union of their token positions across heads and the local suffix. These positions retain exact pre-RoPE keys and are excluded from key factorization; all other prompt keys are factorized. Values within 32 tokens of the protected key positions remain protected throughout decoding. The latest 128 prompt values initialize the exact recent window; values outside both sets are factorized. The 64K RULER evaluation uses this same rule, so its prompt is only partly represented by low-rank factors.

Cache layout. The key cache stores pre-RoPE representations. Protected prompt rows remain exact and are excluded from the compact low-rank coefficient matrix. In the MATH protocol, the chunk-selection rule normally covers the entire short prompt, which therefore remains exact. Decoded keys enter an exact FIFO buffer; once it contains $128+64$ rows, the oldest 64 are incorporated into the low-rank history. The value cache retains protected prompt rows and a recent window of 128 rows exactly. The value rank budget r_V also determines the number of rows used in the initial value refit. Once r_V unprotected value rows have left the recent window, we append them to the reconstructed prompt value history and recompute a truncated SVD with rank budget r_V. Later value updates use 64-row blocks. Incomplete blocks remain uncompressed. Recent and eligible unprotected rows share one exact buffer, with eligibility determined by position. The separate W_X and $\mathcal{Q}_X$ in Algorithms 1 and 2 are logical index lists for these two subsets.

Residual basis construction. The update schedule is determined by the number of eligible rows, independently of the residual threshold. For a block F, we first compute $E = F - (FB)B^\top$ and then replace it by $E - (EB)B^\top$ to reduce residual components along the current basis caused by finite precision. If $\|E\|_F \le \tau$ or $k = d$, the residual basis is empty. Otherwise, we compute a reduced QR factorization $E^\top = Q_E R_E$ and an SVD $R_E = L_E D_E H_E^\top$. Let $\delta_1 \ge \delta_2 \ge \cdots$ denote the singular values on the diagonal of D_E. The retained residual basis is defined by

$$\tau_{\text{eff}} = \max(\tau, \epsilon \max(b,d)\|F\|_F), \quad p = \min(d-k, \#\{i : \delta_i > \tau_{\text{eff}}\}), \quad P = Q_E L_E[:, 1:p],$$

where ϵ is machine epsilon at the computation precision and $\tau = 10^{-6}$ by default. The lower-right block of the core matrix is FP, which equals EP when the residual is exact and $B^\top P = 0$.

A.2 Evaluation Protocols

MATH-500. We evaluate DeepSeek-R1-Distill-Llama-8B and Qwen3-8B in BF16 on all 500 questions. Prompts use each model's chat template and request a boxed final answer. Each question receives one generation with temperature 0.6, top-$p = 0.95$, top-$k = 50$, maximum total length 16,384, and a fixed question-specific seed. The original model, *i*S-KV, and R-KV use the same questions, prompts, seeds, and answer evaluator, and every reported row covers all 500 questions.

Memory accounting. All cache numbers are persistent analytical bytes validated against the factor layout in (7). Temporary reconstruction tensors and model weights are excluded. Compression ratios in the main table use the original model's output lengths. The implementation has no fused factor-aware attention kernel, and no throughput or peak-HBM advantage is claimed.

R-KV matching. R-KV uses its released Hugging Face implementation, divide length 128, and a 128-token staging buffer. On DeepSeek-R1-Distill-Llama-8B, retained budgets 748, 562, and 409 match *iS*-KV ranks 128, 96, and 64 at 4.060×, 4.813×, and 5.920× compression. Their paired accuracy advantages are 4.4, 5.4, and 5.6 points, with question-bootstrap 95% intervals $[1.6, 7.2]$, $[2.0, 8.8]$, and $[2.2, 9.0]$. On Qwen3-8B, budget 638 is matched using the original model's mean total length of 4,317.452 tokens. Rank-96 *iS*-KV and R-KV then give 5.6442× and 5.6410× compression; their relative mismatch is 0.056%. R-KV staging rows are included in every calculation.

OjaKV evaluation. Table 2 reports our evaluation of the released OjaKV implementation on DeepSeek-R1-Distill-Llama-8B and MATH-500. The released checkpoint, configured with error-budget parameter 0.075, retains approximately 77% of the original model's persistent KV storage. It answers 310/500 questions correctly; three empty outputs count as incorrect. For higher compression, we change its aggregate ranks to 232, 192, and 152, targeting $4.06\times$, $4.81\times$, and $5.92\times$ compression. These runs are stopped after 30, 28, and 60 generations, respectively: every collected output repeats until the approximately 16K-token limit, without emitting EOS. These outputs are not scored.

Component ablation. The key/value ablation and the frozen-basis ablation use a fixed prefix of 80 MATH-500 questions on DeepSeek-R1-Distill-Llama-8B, with the same prompts, seeds, evaluator, and single-generation protocol as the main MATH-500 runs. All compressed variants use rank 96, exact recent windows of 128 rows, block size 64, and reorthogonalization every 256 folded rows. The frozen-basis variant initializes both low-rank bases from the prompt, freezes them before the first decode fold, and keeps the same exact recent windows afterwards.

RULER. The rank-96 configuration of the 64K RULER evaluation uses exact recent windows of 128 rows but an earlier update setting: folds contain 16 rows and reorthogonalization runs every 32 folded rows. Every task has ten examples. Table 4 also lists the different update and recent-window settings of the rank-160 configuration.

Factor precision at a 32K budget. The persistent factors are FP16 in the MATH-500 runs. At a 32,768-token generation budget on Qwen3-8B they overflow: entries exceed the FP16 maximum of 65,504, become $\pm\infty$, and the logits turn into NaN. On the same question, we observed this with three versions of our code: the version used for the main table, the latest version, and the latest version with all speed optimizations turned off, at 13K, 19K, and 55K decoded tokens, respectively. FP32 and BF16 factors both completed the same generation, so the failure comes from the representation rather than from one implementation. The original model and R-KV are unaffected because they store raw keys and values in BF16, whose largest finite value is about 3.4×10^{38}, whereas *iS*-KV stores derived factors that are updated incrementally and grow with sequence length.

We therefore store factors in BF16 for all 32K experiments. BF16 occupies the same two bytes as FP16, so the compression ratios are unchanged; FP32 would cut Qwen3-8B rank-128 compression from 6.47× to about 3.8×. BF16 gives up three mantissa bits for a much larger range. Because BF16 and FP16 factors do not give bitwise identical results, we report the 32K results as a separate configuration and do not rerun MATH-500 in BF16. In the one setting run in both formats, DeepSeek-R1-Distill-Llama-8B at rank 128 on AIME 2024, accuracy is the same (56.7%), but BF16 lengthens mean generations from 15,541 to 18,126 tokens and doubles the share of generations that reach the length limit from 13% to 27%. The lower precision therefore still changes when generation stops.

A.3 Probe Settings

Settings of Figure 1. Panel (a) uses the longest trace of DeepSeek-R1-Distill-Llama-8B and Qwen3-8B on each of MATH-500, AIME 2024, and AIME 2025. At decode step 1024, tokens are ranked by their average attention over the preceding 32 steps, and the bottom 75% are marked.

No token is removed, so this panel does not measure the accuracy of eviction; Table 1 does. A marked token counts as strongly attended once its attention at a later step exceeds 25 times the uniform value $1/n$, where n is the number of visible tokens. Attention is measured at layer 15. Panel (b) uses four 8K-token AIME 2024 decode traces per model; singular values are divided by the largest, and medians are taken over layers and then over traces. In panels (a) and (b), lines show two-model means and bands show two-model ranges. Panel (c) uses the update-rule probe below: three DeepSeek traces, layers 0, 15, and 31, rank 32, and 64-row blocks. Drift is the relative error with respect to the represented history before each update, not the original KV cache; it differs from the factor drift of (8). Dashed lines update the basis and keep the old coordinates; solid lines update the basis and all old coordinates jointly, which in exact arithmetic equals the *iS*-KV update (Propositions 1 and 2). Lines show means and bands span layer-averaged traces. All bands are observed ranges, not confidence intervals.

Fidelity probes. Factor drift reconstructs the low-rank state and compares it with a fresh SVD of the same represented history at the same rank. The update-rule probe extracts query, key, and value features from MATH samples 138, 450, and 216 at layers 0, 15, and 31, then compares a stale-coordinate basis update with a synchronized Brand update using rank 32 and a 64-row block. End-to-end probes teacher-force the original model's continuation through rank-96 and rank-64 caches and report centered vocabulary-logit error, top-1 agreement, EOS-logit shift, and teacher-token logit shift.

A.4 Properties of the Block Update

We establish the reconstruction, truncation, and reorthogonalization properties of one block update. The analysis concerns the concatenation of the maintained low-rank history and the arriving block, as defined in Equation (2).

Proposition 1 (Exact block incorporation before truncation)**.** *Assume exact arithmetic and $A = U\Sigma B^\top$, with orthonormal columns in U and B. For a new block F, let P have orthonormal columns spanning the complete row space of $E = F - (FB)B^\top$. If the core Z in Equation* (3) *is factorized without truncation, the update in Equation* (4) *satisfies*

$$U^+\Sigma^+(B^+)^\top = \begin{bmatrix} A \\ F \end{bmatrix}.$$

Proof. Since $B^\top P = 0$ and $E = EPP^\top$, the arriving block satisfies $F = (FB)B^\top + (FP)P^\top$. Hence

$$\begin{bmatrix} A \\ F \end{bmatrix} = \begin{bmatrix} U & 0 \\ 0 & I_b \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ FB & FP \end{bmatrix} [B\ P]^\top.$$

Substituting the untruncated SVD of Z yields Equation (4). □

Proposition 2 (Optimal local truncation)**.** *Under the assumptions of Proposition 1, let $M = [A; F]$ and let $\sigma_1(Z) \geq \cdots \geq \sigma_q(Z)$ be the singular values of the core. Let M_r be the reconstruction obtained by retaining its leading $\min(r, q)$ singular components. Then*

$$\|M - M_r\|_F = \left(\sum_{i=r+1}^{q} \sigma_i(Z)^2 \right)^{1/2} = \min_{\substack{\widehat{M} \in \mathbb{R}^{(n+b)\times d} \\ \operatorname{rank}(\widehat{M}) \leq r}} \|M - \widehat{M}\|_F.$$

Proof. The outer factors in Equation (3) have orthonormal columns, so M and Z have the same nonzero singular values. The result follows from the Eckart–Young–Mirsky theorem. □

Effect of residual thresholding. When P retains only part of the residual row space, define $R_\perp = F - (FB)B^\top - (FP)P^\top$. Assuming exact arithmetic, orthonormal columns in U, B, P, and $B^\top P = 0$, the discarded component is orthogonal to the retained feature subspace. The update therefore satisfies

$$\|M - \widehat{M}^+\|_F^2 = \|R_\perp\|_F^2 + \sum_{i>r} \sigma_i(Z)^2.$$

This follows by decomposing the error into its orthogonal components inside and outside $\mathrm{span}([B\ P])$. The discarded core singular values therefore characterize the full error only when the residual basis spans the complete residual row space. Finite-precision factors introduce further deviations from these exact-arithmetic identities.

Local optimality and factor drift. The optimality in Proposition 2 is local to the maintained history and the arriving block. Since the maintained history already incorporates earlier approximations, successive updates need not recover the rank-r SVD of the original uncompressed history. For a nonzero reference matrix X used to evaluate reconstruction, let $\widehat{X}_{\mathrm{inc}}$ denote the incremental approximation and $\widehat{X}_{\mathrm{fresh}}$ the direct truncated-SVD approximation at the same rank. Their difference in relative reconstruction error is

$$\delta = \frac{\|\widehat{X}_{\mathrm{inc}} - X\|_F}{\|X\|_F} - \frac{\|\widehat{X}_{\mathrm{fresh}} - X\|_F}{\|X\|_F}. \tag{8}$$

This quantity, referred to as factor drift, measures excess relative reconstruction error. Logit probes against the original model and free-generation probes evaluate the effects of cache approximation on model outputs.

Proposition 3 (Product-preserving reorthogonalization)**.** *In exact arithmetic, the periodic reorthogonalization used by* iS-*KV leaves $U\Sigma B^\top$ unchanged.*

Proof. Let $U = Q_U R_U$ and $B = Q_B R_B$ be thin QR factorizations, and let $R_U \Sigma R_B^\top = \widetilde{U}\widetilde{\Sigma}\widetilde{B}^\top$ be an untruncated SVD. The factors $Q_U\widetilde{U}$, $\widetilde{\Sigma}$, and $Q_B\widetilde{B}$ have the same product $U\Sigma B^\top$. In finite precision, the factors are orthonormal up to computation roundoff; casting them to storage precision introduces additional rounding error. □

Proposition 4 (Block-update cost)**.** *For block size b, width d, rank budget r, and n factorized rows, projection and residual computation require $O(bdr)$ work. Residual basis construction factorizes a $d \times b$ matrix, and the core matrix has at most $r + b$ rows and columns. Updating the historical coefficients costs $O(nr(r + b))$. Persistent factor storage is $O((n + d)r)$. Periodic reorthogonalization adds work that depends on the number n of historical rows.*

A.5 Numerical Diagnostics

Residual threshold. These probes test the numerical fidelity of the update, not the gap to the original KV cache or generation quality. With an invalid, oversized residual threshold, value drift reaches 0.961 (step 1,024, layer 0) and 0.906 (step 2,048, layer 15); the stable update gives 0.0001 and 0.0109 at the same points.

Drift over a long trajectory. We measure the drift of (8) on one AIME 2024 trajectory that never emits EOS (8,192 decoded rows, FP16 persistent factors, reorthogonalization every 256 folded rows). Table 5 gives the values at step 8,192. The key path stays within 0.007 excess error and the value path within 0.016–0.021, so the numerical cost of the incremental update is far smaller than truncation itself on keys, and values are the more constrained of the two.

Table 5: **Factor drift at step 8,192.** Excess relative reconstruction error (8) on one AIME 2024 trajectory, by path, rank, and layer.

	$r = 96$			$r = 128$		
Path	**L0**	**L15**	**L31**	**L0**	**L15**	**L31**
key	0.0024	0.0071	0.0046	0.0016	0.0065	0.0043
value	0.0207	0.0176	0.0168	0.0194	0.0185	0.0160

Reorthogonalization. On the same states, reorthogonalization reduces the basis orthogonality error $\|\boldsymbol{U}^\top\boldsymbol{U} - \boldsymbol{I}\|_F$ from 9.5×10^{-3} to 1.1×10^{-3} (layer 15, r=128; 8–10×), while the median relative change of the represented product across reorthogonalization events is 2.4×10^{-4}, consistent with Proposition 3. Within 8K steps, reorthogonalization does not reduce reconstruction error; it keeps the bases orthogonal, which guards against error build-up in longer generations.

Per-depth spectra. For pre-RoPE keys, the rank $r_{99\%}$ that keeps 99% of the spectral energy is not stable across depth: beyond layer 3, its coefficient of variation is 0.19 on DeepSeek-R1-Distill-Llama-8B and 0.26 on Qwen3-8B, where it spans 122–338. The energy kept at the ranks we use is stable: at $r{=}128$ the coefficient of variation across depth is ≤ 0.023 for both key paths and ≤ 0.055 for values. This stability, not a constant $r_{99\%}$, is what supports a single rank setting for all layers.

B ALGORITHMS

The prefill initialization and per-step decoding procedures summarized in Section 3 are specified in Algorithms 1 and 2. The accompanying paragraphs walk through their steps and point to the definitions and equations they instantiate.

Prefill initialization (Algorithm 1). Algorithm 1 realizes the construction of Section 3.1. It first evaluates exact causal attention over the prompt, so the prefill outputs never depend on the compressed representation. It then selects the protected key positions $\mathcal{P}_K$ from the chunk outlier scores $s_{c,h}$ of Section 3, taking the union of the selected chunks with the local suffix and dilating them by ρ positions to form the protected value positions $\mathcal{P}_V$; the last $\min(w, L)$ prompt positions initialize the exact recent windows W_K and W_V. For each of K and V, the exact set $\mathcal{E}_X = \mathcal{P}_X \cup W_X$ is stored verbatim and the remaining positions $\mathcal{C}_X = \mathcal{I} \setminus \mathcal{E}_X$ are factorized by the truncated SVD SVD_{r_X} of Section 3, retaining $k_X = \min(r_X, |\mathcal{C}_X|, d)$ components. The pending queues $\mathcal{Q}_X$ and update counters c_X start empty, and the flag f_V records that the one-time value refit of Section 3.2 is still pending.

Algorithm 1 *iS*-KV: prefill initialization at one layer

Input: Prompt queries Q and keys K before RoPE, values V, positions $\mathcal{I} = \{1, \ldots, L\}$
Parameters: Rank budgets r_K, r_V, window size w, protection settings from Section 3.1
Output: Prefill output O; cache $\mathcal{S}$ with factors, exact rows, position lists, ranks, counters, and flag f_V

$O \leftarrow \mathrm{CausalAttn}(\mathrm{RoPE}_{\mathcal{I}}(Q), \mathrm{RoPE}_{\mathcal{I}}(K), V)$ ▷ Exact prefill attention
$\mathcal{P}_K \leftarrow$ union of outlier-chunk positions across heads and the local suffix ▷ Outlier scores, Section 3
$\mathcal{P}_V \leftarrow \{i \in \mathcal{I} : \exists j \in \mathcal{P}_K,\ |i - j| \leq \rho\}$
$W_K \leftarrow \varnothing$; $W_V \leftarrow$ the last $\min(w, L)$ prompt positions ▷ Initialize recent windows
for $X \in \{K, V\}$ **do**
  $\mathcal{E}_X \leftarrow \mathcal{P}_X \cup W_X$; $\mathcal{C}_X \leftarrow \mathcal{I} \setminus \mathcal{E}_X$, ordered by position
  $X_{\mathrm{exact}}[\mathcal{E}_X] \leftarrow X[\mathcal{E}_X]$
  $(U_X, \Sigma_X, B_X) \leftarrow \mathrm{SVD}_{r_X}(X[\mathcal{C}_X])$
  $k_X \leftarrow \min(r_X, |\mathcal{C}_X|, d)$; $\mathcal{Q}_X \leftarrow \varnothing$; $c_X \leftarrow 0$
$f_V \leftarrow \mathrm{true}$ ▷ Initial value refit is pending

Decoding step (Algorithm 2). Algorithm 2 maintains the cache of Section 3.2 during generation. For each of the key and value streams it stores the arriving row exactly, advances the recent window, and enqueues the unprotected positions that leave it. A fold is triggered once the queue holds a full block: an empty history is initialized by a truncated SVD of the block, the pending value refit appends the oldest r_V rows to the reconstructed value history and recomputes a rank-r_V truncated SVD, and otherwise the BLOCKUPDATE subprocedure applies the synchronized block-incremental update of (3) and (4). Queue occupancy sets the update timing, the tolerance τ controls the residual basis of Appendix A.1, and the factors are reorthogonalized every T folded rows, which preserves their product by Proposition 3. The folded positions move from the queue into the compressed set $\mathcal{C}_X$, and the exact set becomes $\mathcal{E}_X = \mathcal{P}_X \cup W_X \cup \mathcal{Q}_X$. The remaining steps reconstruct the cache in token order by (5) and form the attention output of (6) (Section 3.3).

Algorithm 2 *iS*-KV: one decoding step at one layer

Input: Cache $\mathcal{S}$, pre-RoPE query q_t and key k_t, value v_t, position t
Parameters: Rank budgets r_K, r_V, window w, block size b, tolerance τ, reorthogonalization interval T
Output: Attention output o_t and updated cache $\mathcal{S}$

for $(X, x_t) \in \{(K, k_t), (V, v_t)\}$ **do** ▷ Update the cache state
  Store x_t exactly at position t and advance W_X
  Enqueue positions leaving W_X and outside $\mathcal{P}_X$ in $\mathcal{Q}_X$
  $a \leftarrow r_V$ if $X = V$ and f_V, otherwise b
  if $|\mathcal{Q}_X| \geq a$ **then**
    $J \leftarrow$ the oldest a positions in $\mathcal{Q}_X$; $F \leftarrow X_{\text{exact}}[J]$
    if $X = V$ and f_V **then**
      $(U_V, \Sigma_V, B_V) \leftarrow \mathrm{SVD}_{r_V}([U_V \Sigma_V B_V^\top; F])$; $f_V \leftarrow$ false
    else
      $(U_X, \Sigma_X, B_X) \leftarrow \textsc{BlockUpdate}(U_X, \Sigma_X, B_X, F; r_X, \tau)$; $c_X \leftarrow c_X + b$
      if $T > 0$ and $c_X \geq T$ **then**
        Reorthogonalize (U_X, Σ_X, B_X); $c_X \leftarrow c_X \bmod T$ ▷ Proposition 3
    $k_X \leftarrow$ the number of columns of B_X
    Append J to $\mathcal{C}_X$; remove J from $\mathcal{Q}_X$ and delete the corresponding exact rows
  $\mathcal{E}_X \leftarrow \mathcal{P}_X \cup W_X \cup \mathcal{Q}_X$
▷ *Reconstruct the cache and evaluate attention*
Reconstruct $\widetilde{K}, \widetilde{V}$ in token order by Equation (5)
Restore head dimensions; apply RoPE to keys and q_t at their original positions
Compute o_t by Equation (6)

BlockUpdate: for an empty low-rank history, return $\mathrm{SVD}_{r_X}(F)$. Otherwise, construct P using tolerance τ, truncate the core SVD in Equation (3) to rank budget r_X, and return the updated factors in Equation (4).