PASVD: ADAPTIVE CROSS-HEAD KEY SELECTION FROM LOW-RANK KV CACHES

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) are widely used in interactive assistants, long-document understanding, and complex reasoning. However, as context length increases, the key–value (KV) cache becomes a major memory bottleneck. Many low-rank methods construct projection spaces offline from model weights or calibration data. These fixed projections cannot adapt to the changing token distribution of each sequence. Recent online page-based methods divide the KV cache into fixed-size token blocks and apply SVD to the post-RoPE keys of each completed page. However, RoPE can weaken the low-rank structure of keys, while fixed ranks cannot adapt to the varying redundancy across pages. To address these limitations, we introduce PASVD, a training- and calibration-free method for page-adaptive low-rank key compression. PASVD first concatenates the pre-RoPE keys across all KV heads, then partitions them into token pages and selects a page-specific rank according to the singular-value spectrum of each page. To reduce decoding overhead, PASVD scores historical keys directly in the compressed space and reconstructs only the selected keys for RoPE and final attention, avoiding full-cache reconstruction. Across the reported LongBench, MMLU, and RULER settings, PASVD's average score is at most 0.16 points below FullKV, while delivering 2.37$\times$ higher generation-stage throughput at batch size 1.

1 INTRODUCTION

In recent years, LLMsMeta (2024); Cho et al. (2025); Singh et al. (2025) have become an essential foundation for a wide range of applications, providing capabilities that were difficult to attain with conventional machine-learning systems. To continuously push the performance boundaries in increasingly complex tasks such as reasoning and multimodal interaction, scaling up model parameters has become an inevitable direction in the evolution of LLMs. However, the rapid growth in both model parameters and context length has led to increasingly severe inference latency issues. Currently, the KV cache, which stores intermediate attention keys and values to avoid expensive recomputation during token generation, has become a standard technique for accelerating autoregressive decoding in LLMs. Although KV caching significantly improves inference efficiency, the associated GPU memory consumption has gradually emerged as a new bottleneck for LLM deployment. Consequently, how to efficiently compress the KV cache while preserving model performance has become a major research focus in both academia and industry.

Existing approaches to the KV-cache memory bottleneck can be divided into four categories: quantization, offloading, token eviction, and low-rank approximation. Quantization represents cached keys and values at lower numerical precision (Liu et al., 2024), whereas offloading moves part of the cache from GPU memory to CPU memory or external storage. Token-eviction methods estimate the importance of historical tokens from observed attention statistics and discard entries with low importance scores (Zhang et al., 2023). These techniques reduce KV-cache memory by lowering numerical precision, moving data across memory tiers, or retaining fewer tokens, but they do not exploit structural redundancy in the KV feature space. As shown in the left panel of Figure 1, the singular values of the key and value matrices decay rapidly, with a small number of singular directions capturing most of the spectral energy. This spectral decay suggests substantial redundancy across feature dimensions, making low-rank approximation a natural choice for KV-cache compression. Motivated by this observation, low-rank methods exploit this structure through matrix factorization

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.

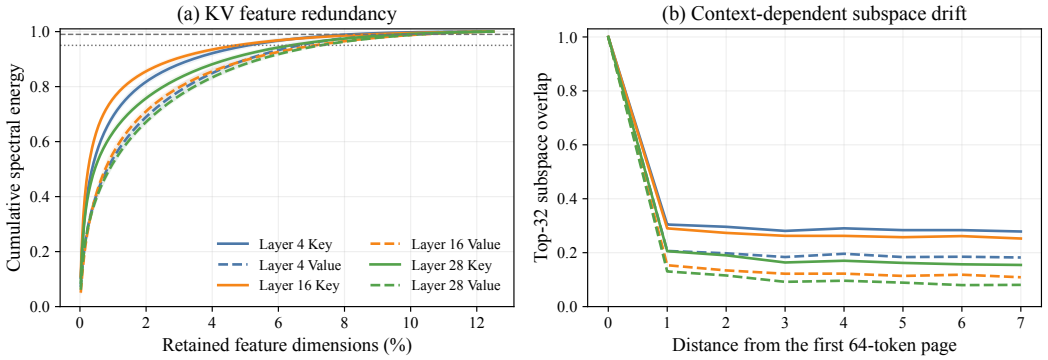


Figure 1: Feature redundancy and contextual subspace variation in Llama-2-7B. We extract pre-RoPE keys and values from Layers 4, 16, and 28 over eight 512-token WikiText-2 training sequences. **Left:** Cumulative spectral energy versus the number of retained dimensions; solid and dashed curves denote keys and values, respectively. **Right:** Overlap between the top-32 principal subspace of each 64-token page and that of the first page within the same sequence. Values are included as a diagnostic reference; PASVD compresses and selects keys.

or learned projections, retaining the principal information in a compact representation and thereby reducing the cache footprint along the hidden dimension.

Among existing low-rank KV-cache compression methods, CUR-based approaches rely on sampling and projection approximations, which introduce additional approximation error, making SVD a common choice for accurate low-rank compression. Existing SVD-based methods can be broadly divided into offline and online approaches. Offline methods construct the low-rank projection space before inference: Palu applies SVD to the key and value projection weights (Chang et al., 2025), while Eigen Attention constructs the projection space from calibration data (Saxena et al., 2024). Although their factorization targets differ, both keep the projection space fixed throughout decoding. As new tokens are generated, the KV distribution evolves with the context, making a fixed projection difficult to adapt to the current sequence. Online methods address this limitation by adapting the low-rank space during decoding. OjaKV updates the projection basis using Oja's rule (Zhu et al., 2026) to track changes in the KV distribution. Since recomputing the global SVD whenever new tokens are generated is computationally expensive, OjaKV uses approximate incremental updates to reduce this cost. However, these approximations introduce additional error and degrade compression quality. PuzzleKV (Wang et al., 2026) avoids repeated global updates by restricting SVD to completed token pages, making exact decomposition practical at the page level. However, it factorizes post-RoPE keys, where positional rotations weaken the low-rank structure and increase approximation error. In addition, using the same rank for all pages ignores differences in their low-rank structure, leading to a suboptimal trade-off between inference accuracy and memory saving.

To address these limitations, we introduce PASVD, a training- and calibration-free framework for page-adaptive low-rank key-cache compression. PASVD first applies page-based SVD to pre-RoPE keys and jointly factorizes the keys across all KV heads within each page, avoiding the degradation of low-rank structure caused by positional rotation. However, pre-RoPE compression requires historical keys to be reconstructed and rotated before final attention, introducing additional decoding overhead. To reduce this cost, PASVD scores historical keys directly in the compressed space and reconstructs only the selected keys. RoPE is then applied to these reconstructed keys at their original token positions before final attention. In addition, using a uniform rank across pages cannot adapt to their different levels of redundancy. PASVD therefore evaluates a set of candidate ranks according to the residual spectral energy of each page and selects the smallest rank that satisfies a given error threshold, allowing different pages to use different compression ratios. The resulting variable-rank factors are stored in a compact quantized layout, while fused kernels further reduce reconstruction and memory-access overhead during decoding. Our contributions are:

- We propose a pre-RoPE cross-head paged SVD scheme that jointly factorizes all KV heads within each page, reducing repeated SVD operations and exploiting shared redundancy across KV heads.
- We develop page-adaptive rank selection based on residual spectral energy, allowing each page to select the smallest supported rank under an error constraint and avoiding the memory waste of fixed-rank allocation.
- We perform key selection from the compressed factors and reconstruct only the selected keys for RoPE-aware attention. Combined with INT8 basis storage and fused kernels, this enables efficient compressed attention with low memory and execution overhead.

2 Related Work

2.1 KV Cache Quantization and Eviction

Existing KV-cache compression techniques primarily reduce numerical precision, move cached states across the memory hierarchy, or retain only a subset of tokens. Quantization methods reduce the bit width of cached states: Atom jointly quantizes model weights and the KV cache, whereas KIVI and KVQuant exploit the distinct distributions and outlier patterns of keys and values to support low-bit cache representations (Zhao et al., 2024; Liu et al., 2024; Hooper et al., 2024). Offloading systems such as FlexGen place attention states in CPU memory or secondary storage, trading GPU capacity for additional data-transfer overhead (Sheng et al., 2023). Token-eviction methods instead exploit nonuniform token importance. H_2O preserves accumulated heavy hitters, StreamingLLM retains attention sinks together with a recent window, and Keyformer and FastGen apply task- or model-dependent token selection policies (Zhang et al., 2023; Xiao et al., 2024; Adnan et al., 2024; Ge et al., 2024). SnapKV selects clustered prompt positions from an observation window, while PyramidKV allocates progressively smaller cache budgets across layers and CurDKV uses approximate CUR leverage scores for value-aware token selection (Li et al., 2024; Cai et al., 2024; Sengupta et al., 2025). These approaches are effective and complementary, but they mainly operate on storage precision, placement, or the token dimension rather than directly modeling redundancy along the hidden feature dimension.

2.2 Low-Rank KV Cache Compression

Low-rank approaches compress the hidden feature dimension by projecting keys and values into smaller subspaces. Palu factorizes the key and value projection weights offline and caches their low-dimensional intermediate representations (Chang et al., 2025), while Eigen Attention estimates a static low-rank attention space from calibration data (Saxena et al., 2024). MatryoshkaKV learns nested orthogonal projections through distillation and then allocates different compression rates across layers and heads (Lin et al., 2024). ReCalKV further combines head reordering and grouped SVD for keys with offline calibration and matrix fusion for values (Yan et al., 2025). Although these methods provide effective feature-dimension compression, their subspaces are learned or calibrated before deployment and remain fixed for future contexts. OjaKV instead introduces online subspace tracking through Oja's rule, enabling its bases to follow an evolving context at the cost of periodic approximate updates (Zhu et al., 2026).

3 Background

Large language models typically consist of multiple Transformer layers. In each layer, the input hidden states X are projected into query, key, and value representations using the corresponding projection matrices W_Q, W_K, and W_V. The query, key, and value representations are given by

$$Q = XW_Q, \qquad K = XW_K, \qquad V = XW_V. \tag{1}$$

Using these representations, standard self-attention computes scaled query–key similarities and uses the normalized scores to form a weighted sum of the values:

$$O = \mathrm{Softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V, \tag{2}$$

where d denotes the feature dimension of each head.

During autoregressive decoding, each newly generated token attends to the keys and values of all preceding tokens. Consequently, the key and value of previous tokens must be retained across decoding steps, resulting in the growing memory footprint of the KV cache. At decoding step t, the newly generated key and value are appended as

$$K_{1:t} = \mathrm{Concat}(K_{1:t-1}, k_t), \qquad V_{1:t} = \mathrm{Concat}(V_{1:t-1}, v_t),$$

and the attention output is computed as

$$o_t = \mathrm{Softmax}\left(\frac{q_t K_{1:t}^\top}{\sqrt{d}}\right) V_{1:t}.$$

Since one key and one value must be retained for every token at every KV head and Transformer layer, the cache grows linearly with the sequence length. For an LLM with L layers, H_{KV} KV heads per layer, sequence length N, and feature dimension d, the KV cache contains $2LH_{\mathrm{KV}}Nd$ elements, corresponding to $2LH_{\mathrm{KV}}Ndb$ bytes when each element occupies b bytes. Consequently, KV-cache storage and memory access become increasingly expensive as the context length grows.

These observations motivate a KV-cache compression method that reduces storage and memory-access overhead while preserving attention quality. We next introduce PASVD, a page-wise adaptive low-rank framework designed for efficient long-context decoding.

4 METHOD

4.1 OVERVIEW

PASVD concatenates the pre-RoPE keys from all KV heads and partitions them into fixed-size token pages. The current page receiving newly generated tokens is kept uncompressed until it becomes full. PASVD then performs a joint SVD over the completed page and selects its rank from the resulting singular-value spectrum. During decoding, historical keys are scored in the compressed space, and only the selected keys are reconstructed and rotated by RoPE for final attention. Since different pages may use different ranks, their factors are stored according to the selected ranks, with the bases quantized to INT8 to reduce memory overhead.

Figure 2 summarizes the resulting selection-before-reconstruction pipeline and separates page construction from the operations performed at every decoding step.

4.2 CROSS-HEAD PAGE FACTORIZATION

As shown in Figure 1, the low-rank projection directions learned from the first page have limited overlap with the dominant directions of later pages. This suggests that a single low-rank basis cannot accurately represent the entire sequence, motivating page-wise factorization. However, performing a separate SVD for each KV head within every page would introduce repeated decomposition overhead. PASVD concatenates the keys from all KV heads and performs a single joint SVD for each page, reducing the number of SVD calls per page from H_{KV} to one.

For a Transformer layer with H_{KV} KV heads and head dimension d, let $D = H_{\mathrm{KV}}d$. We concatenate the pre-RoPE keys of token t across all heads as $x_t = \mathrm{vec}(k_t) \in \mathbb{R}^D$. For a sealed page p containing S tokens,

$$X_p = [x_{(p-1)S+1}, \ldots, x_{pS}]^\top \in \mathbb{R}^{S\times D}. \tag{3}$$

PASVD performs a truncated SVD, $X_p^\top = U_p \Sigma_p W_p^\top$, and retains the leading r_p components:

$$B_p = U_p(:, 1 : r_p), \qquad Z_p = X_p B_p, \qquad X_p \approx Z_p B_p^\top. \tag{4}$$

Here, $B_p \in \mathbb{R}^{D\times r_p}$ captures the dominant low-rank subspace of the page, while $Z_p \in \mathbb{R}^{S\times r_p}$ contains the compressed representation of each token. After factorization, PASVD retains only the cross-head basis and the shared latent matrix, requiring $r_p(D + S)$ elements per page.

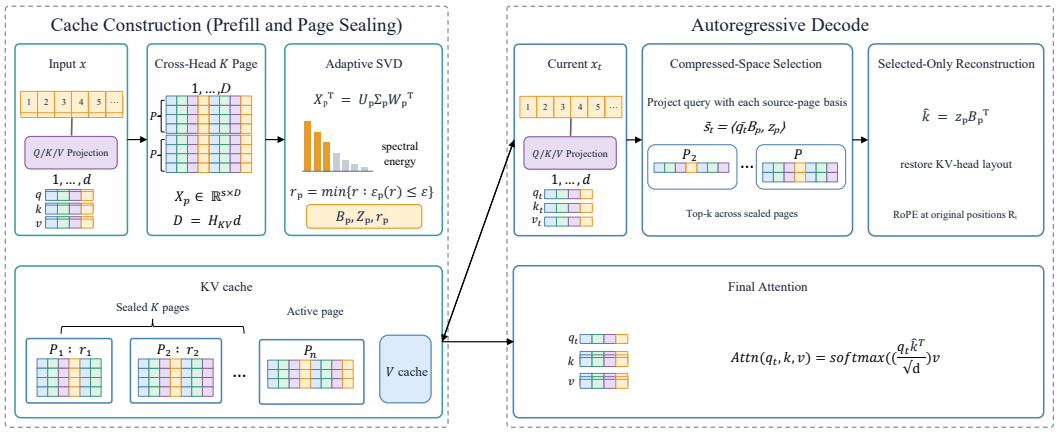


Figure 2: Overview of PASVD. During cache construction, pre-RoPE keys from all KV heads within each sealed token page are concatenated and jointly factorized. PASVD selects the smallest supported rank that satisfies the spectral-error target and stores each page according to its selected rank. During decoding, the current query is projected into each page's low-rank space to score historical keys. The top-k candidates are combined with sink, recent, and active-page tokens before reconstruction. Only the selected historical keys are reconstructed, split back into KV heads, and rotated at their original token positions for final attention with the corresponding values.

Since X_p concatenates the keys of all KV heads along the feature dimension, the rank r_p is shared by the entire cross-head page. The full key page is reconstructed as

$$\widehat{X}_p \approx Z_p B_p^\top \in \mathbb{R}^{S \times D}. \tag{5}$$

Following the same feature layout as X_p, the reconstructed matrix $\widehat{X}_p$ can be split into H_{KV} KV heads.

4.3 Content-Adaptive Rank Selection

The factorization above produces a singular-value spectrum for each sealed page. Since different pages exhibit different spectral decay rates, using a fixed rank may either retain unnecessary components or discard important ones. Let $\sigma_{p,1} \geq \cdots \geq \sigma_{p,m}$, where $m = \min(S, D)$. We measure the fraction of discarded spectral energy after retaining rank r as

$$\epsilon_p(r) = \frac{\sum_{j=r+1}^{m} \sigma_{p,j}^2}{\sum_{j=1}^{m} \sigma_{p,j}^2}. \tag{6}$$

Given an error threshold ϵ and a supported rank set $\mathcal{R} = \{r_{\min}, r_{\min} + \Delta r, \ldots, r_{\max}\}$, PASVD selects

$$r_p = \min\{r \in \mathcal{R} : \epsilon_p(r) \leq \epsilon\}. \tag{7}$$

Here, $\epsilon_p(r)$ measures the fraction of spectral energy discarded after retaining rank r. PASVD selects the smallest supported rank that meets the error target, falling back to $r_{\max}$ otherwise. Since the selected rank varies across pages, the resulting low-rank factors also have different sizes.

4.4 Quantized Heterogeneous-Rank Execution

To store these variable-rank factors efficiently, PASVD packs each page according to its selected rank rather than padding all pages to $r_{\max}$. The basis B_p is further quantized to INT8:

$$a_{p,j} = \frac{\max_i |B_p[i,j]|}{127}, \qquad \overline{B}_p[i,j] = \mathrm{clip}\left(\mathrm{round}\,\frac{B_p[i,j]}{a_{p,j}}, -127, 127\right). \tag{8}$$

Let $o_p = \sum_{\ell<p} r_\ell$ denote the rank offset of page p. The quantized basis, scales, and latent factors are packed as

$$\mathcal{B}[:, o_p : o_p + r_p] = \overline{B}_p, \quad \mathcal{A}[o_p : o_p + r_p] = a_p, \quad \mathcal{Z}[:, o_p : o_p + r_p] = Z_p. \tag{9}$$

Each page therefore stores only (o_p, r_p) as metadata, with memory cost

$$M_p^K = r_p(Db_{\mathcal{B}} + Sb_{\mathcal{Z}} + b_{\mathcal{A}}) + \mathcal{O}(1), \tag{10}$$

where $b_{\mathcal{B}}$, $b_{\mathcal{Z}}$, and $b_{\mathcal{A}}$ denote the bytes per basis, latent, and scale element, respectively. Thus, physical storage follows the actual rank r_p rather than $r_{\max}$.

At decoding time, fused kernels recover the dequantized basis $\widehat{B}_p[:, j] = a_{p,j}\overline{B}_p[:, j]$ from (o_p, r_p) and reconstruct only the selected keys.

4.5 Compressed Selection and Selective Reconstruction

Pre-RoPE compression introduces additional reconstruction and RoPE computation during decoding. To reduce this overhead, PASVD avoids reconstructing all historical keys and instead first selects candidate keys directly in the compressed space.

For GQA, PASVD averages the $G = H_Q/H_{\mathrm{KV}}$ query heads associated with each KV head and concatenates the results: $\bar{q}_t = \mathrm{vec}(\bar{q}_t^{(1)}, \ldots, \bar{q}_t^{(H_{\mathrm{KV}})}) \in \mathbb{R}^D$, where $\bar{q}_t^{(h)} = G^{-1} \sum_{g=1}^{G} q_t^{(h,g)}$. It is then projected into the low-rank space of page p:

$$\widetilde{q}_{t,p} = \bar{q}_t \widehat{B}_p, \qquad \widetilde{s}_{t,i} = \langle \widetilde{q}_{t,p}, z_{p,i} \rangle . \tag{11}$$

PASVD retains the top-k keys together with sink, recent, and active tokens. For a selected token i in page p, PASVD reconstructs

$$\widehat{x}_i = z_{p,i} \widehat{B}_p^\top . \tag{12}$$

RoPE is then applied at the original positions for final attention; the proxy's retained attention mass is analyzed in Appendix A.3.

5 Experiments

5.1 Experimental Setup

Models and benchmarks. Our experiments use Meta-Llama-3.1-8B-Instruct (Dubey et al., 2024), Qwen3-8B, and Llama-2-7B in FP16. We evaluate on LongBench (Bai et al., 2023), RULER (Hsieh et al., 2024), MMLU (Hendrycks et al., 2021), and WikiText-2 (Merity et al., 2017).

Baselines. FullKV stores the uncompressed FP16 cache and serves as the quality reference. We compare against four representative low-rank compressors: PaLu (Chang et al., 2025), which factorizes projection weights offline; ReCalKV (Yan et al., 2025), which uses calibrated rank allocation; and OjaKV (Zhu et al., 2026), which updates the subspace online; and PuzzleKV (Wang et al., 2026), which performs page-wise decomposition. We locally rerun PaLu, ReCalKV, and OjaKV using their available implementations. Because PuzzleKV does not provide public source code, its RULER results are transcribed from the paper and marked with †.

Testbed. Experiments are run on a server with two NVIDIA GeForce RTX 5090 GPUs (32GB each), two Intel Xeon Gold 6226R CPUs (32 physical cores and 64 logical threads in total), and 128GB of host memory, running Ubuntu 22.04.5 LTS. Each evaluation process uses one GPU. Runtime measurements use a warm-up run and are averaged over three repetitions; the batch size is reported with each experiment.

PASVD configuration. Unless otherwise specified, PASVD uses 96-token pages and selects the smallest rank in $\{80, 96\}$ that retains at least 0.997 of the spectral energy, with 96 as the maximum rank. The K bases and V caches use the same symmetric INT8 quantization rule, while all remaining model states retain their original precision. All experiments use $k = 64$, together with 16 sink tokens, 64 recent tokens, and the active page.

5.2 LONG-CONTEXT GENERATION QUALITY

Table 1: LongBench results on Meta-Llama-3.1-8B-Instruct.

Method	Qasper	MFQA	HotpotQA	2WikiMQA	GovReport	QMSum	RepoBench-P	Avg.
FullKV	46.05	53.68	52.25	39.35	34.89	25.23	12.62	37.72
PaLu	29.14	38.61	25.07	36.41	26.53	20.43	**14.35**	27.22
OjaKV	35.65	46.77	38.94	34.28	27.65	23.80	11.63	31.25
ReCalKV	24.47	40.25	8.82	6.22	20.51	20.80	13.63	19.24
PASVD	**45.56**	**53.82**	**52.25**	**40.00**	**34.62**	**25.18**	11.80	**37.60**

Table 2: RULER results on eight synthetic tasks at 16K and 32K. † denotes PuzzleKV results transcribed from its paper because its source code is not publicly available; its averages are recomputed over the eight tasks shown here.

Model	Length	Method	S1	S2	S3	MK1	MK2	MV	MQ	FWE	Avg.
Llama-3.1-8B Instruct	16K	FullKV	100.00	100.00	100.00	99.00	100.00	99.50	99.25	95.00	99.09
		PaLu	**100.00**	99.00	**100.00**	98.00	95.00	96.25	96.50	89.67	96.80
		OjaKV	97.00	**100.00**	98.00	97.00	41.00	97.75	84.25	52.67	83.46
		ReCalKV	**100.00**	**100.00**	**100.00**	95.00	99.00	94.00	96.50	90.00	96.81
		PuzzleKV†	98.80	95.80	82.00	98.20	**100.00**	91.10	98.85	88.40	94.14
		PASVD	**100.00**	**100.00**	**100.00**	**99.00**	99.00	**99.25**	**99.50**	**94.67**	**98.93**
	32K	FullKV	100.00	100.00	100.00	99.00	100.00	99.50	99.75	82.00	97.53
		PaLu	**100.00**	**100.00**	99.00	97.00	97.00	95.50	97.00	76.67	95.27
		OjaKV	**100.00**	99.00	98.00	94.00	24.00	97.50	94.25	48.67	81.93
		ReCalKV	**100.00**	**100.00**	96.00	96.00	96.00	87.25	95.25	81.33	93.98
		PuzzleKV†	99.80	97.60	83.00	**99.60**	99.80	91.40	98.85	**93.13**	95.40
		PASVD	**100.00**	**100.00**	**100.00**	99.00	**100.00**	**99.75**	**99.50**	80.67	**97.37**
Qwen3-8B	16K	FullKV	100.00	100.00	100.00	100.00	100.00	99.25	99.25	91.67	98.77
		PaLu	**100.00**	98.00	99.00	98.00	96.00	96.00	92.25	90.33	96.20
		OjaKV	84.00	81.00	60.00	89.00	60.00	32.75	69.25	58.67	66.83
		ReCalKV	**100.00**	**100.00**	**100.00**	**100.00**	92.00	87.25	69.00	90.00	92.28
		PuzzleKV†	**100.00**	**100.00**	99.00	99.60	99.80	**99.60**	**99.90**	86.67	98.07
		PASVD	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	99.50	99.25	**92.00**	**98.84**
	32K	FullKV	100.00	100.00	100.00	99.00	99.00	97.50	99.50	83.67	97.33
		PaLu	**100.00**	98.00	99.00	99.00	59.00	89.50	98.25	75.67	89.80
		OjaKV	78.00	66.00	55.00	70.00	28.00	34.75	56.25	48.00	54.50
		ReCalKV	**100.00**	99.00	**100.00**	98.00	85.00	95.75	93.00	52.67	90.43
		PuzzleKV†	**100.00**	99.80	98.60	**99.80**	97.00	98.35	**99.60**	**95.93**	**98.64**
		PASVD	**100.00**	**100.00**	**100.00**	99.00	**99.00**	**98.36**	**99.60**	83.67	97.45

Tables 1 and 2 evaluate PASVD on real-world and synthetic long-context tasks. On LongBench, PASVD achieves an average score of 37.60, only 0.12 points below FullKV, and obtains the best performance among compressed methods on six of the seven tasks. The exception is RepoBench-P because code completion is more sensitive to the removal of historical keys. On RULER, PASVD reaches 98.93 and 97.37 at 16K and 32K on Llama-3.1-8B, remaining close to FullKV at 99.09 and 97.53. On Qwen3-8B, PASVD achieves 98.84 and 97.45, slightly exceeding FullKV at both context lengths. It leads the compressed methods in three of the four model–length settings; at Qwen3-8B 32K, the 98.64 reported by PuzzleKV is 1.19 points higher. The largest PASVD drop appears on FWE, which requires aggregating information from many tokens throughout the context and is therefore more sensitive to key selection. Overall, PASVD maintains stable long-context performance across both models as the context grows from 16K to 32K.

5.3 General Knowledge and Language Modeling

We further evaluate MMLU to measure whether KV-cache compression preserves the model's general knowledge and reasoning ability. Table 3 shows that PASVD retains 68.32% accuracy at a 31.43% KV-cache reduction, only 0.12 points below FullKV. The compressed methods use similar budgets: 31.25% for PaLu, 29.72% for OjaKV, 31.25% for ReCalKV, and a 31.43% page-adaptive average for PASVD. While PaLu's fixed offline projection drops by 12.65 points, PASVD builds page-specific representations from the current sequence and better preserves general knowledge and reasoning. The full Qwen3-8B evaluation on the same 14,042 examples is reported in Appendix Table A2, where PASVD reaches 72.94% accuracy, within 0.04 points of FullKV.

Table 3: Zero-shot MMLU accuracy on Llama-3.1-8B-Instruct using all 14,042 test examples. Δ is the absolute accuracy-point change from FullKV. Bold denotes the best compressed method.

Model	Method	KV reduction (%)	Accuracy (%)	Δ
Llama-3.1-8B Instruct	FullKV	0.00	68.44	0.00
	PaLu	31.25	55.79	−12.65
	OjaKV	29.72	64.98	−3.46
	ReCalKV	31.25	58.81	−9.63
	PASVD	31.43	**68.32**	−0.12

Table 4: WikiText-2 perplexity on Llama-2-7B and Qwen3-8B, with KV-cache reduction reported for Llama-2-7B. Lower PPL is better.

Method	Llama KV reduction (%)	PPL ↓	
		Llama-2-7B	Qwen3-8B
FullKV	0.00	5.47	8.90
PaLu	37.50	6.57	10.29
OjaKV	24.65	6.33	10.27
ReCalKV	50.20	5.86	9.56
PASVD	42.33	**5.52**	**8.95**

As shown in Table 4, PASVD maintains perplexity close to FullKV across both models and achieves the lowest PPL among the compressed methods. It also provides substantial KV-cache reduction. These results show that PASVD achieves a good balance between model quality and KV-cache compression.

5.4 Ablation

5.4.1 Analysis of Adaptive Rank Selection

We evaluate PASVD on RULER and WikiText-2 using Qwen3-8B to analyze how adaptive rank selection affects model quality and storage efficiency, and whether the selected ranks change across different input sequences.

Table 5: Adaptive rank ablation on four representative RULER-16K tasks. Adaptive rank selection chooses from $\{64, 72, 80, 88, 96\}$ with an energy threshold of 0.997. "Saved" denotes the reduction in K-side rank allocation relative to fixed rank 96.

Rank policy	NIAH-MK3	CWE	FWE	QA-1	Avg.	Avg. rank	Saved (%)
Fixed 64	**100.00**	84.50	88.33	**70.00**	85.71	64.00	33.33
Adaptive 64–96	**100.00**	91.50	92.00	**70.00**	88.38	67.68	29.50
Fixed 96	**100.00**	**93.00**	**93.33**	**70.00**	**89.08**	96.00	0.00

Table 5 compares fixed ranks of 64 and 96 with adaptive rank selection over $\{64, 72, 80, 88, 96\}$. Fixed rank 96 achieves the highest average score of 89.08. Rank 64 saves 33.33% of the rank allocation compared with rank 96, but its average score drops by 3.37 points to 85.71. In contrast, adaptive rank selection reaches the same performance as rank 96 on NIAH-MK3 and QA-1, with only small drops on CWE and FWE. With an average rank of 67.68, it saves 29.50% of the K-cache storage while achieving an average score of 88.38, only 0.70 points below fixed rank 96. This result shows that page-adaptive rank selection retains most of the accuracy of a high fixed rank while approaching the storage cost of a much lower rank.

Table 6: Adaptive-rank statistics on 166 WikiText-2 segments. The table shows the selected ranks, the reduction in K-side rank allocation relative to fixed rank 96, the corresponding K-cache storage reduction, and the total KV-cache reduction including both compressed keys and values.

Statistic	Mean	Range across segments
Selected page rank	90.24	81.3–94.6
Logical K-rank reduction vs. rank 96 (%)	6.00	1.51–15.27
Allocated K rank-pool reduction (%)	5.85	1.6–18.2
Total resident-KV reduction vs. FP16 (%)	42.33	41.1–45.8

To examine whether PASVD adapts its rank selection to different input data, we evaluate the selected ranks over 166 WikiText-2 segments in Table 6. Different segments receive different rank allocations. The overall average rank is 90.24, while the average rank across all pages within each segment ranges from 81.3 to 94.6. Compared with fixed rank 96, adaptive rank selection reduces the K-side rank allocation by 6.00% and the corresponding K-cache storage by 5.85% on average. These results show that PASVD adjusts its rank allocation to the input data.

5.4.2 Batch-Scaling Efficiency

Table 7: Decode-only throughput on Qwen3-8B at 32K on one RTX 5090. Results are three-run means; FullKV uses the Hugging Face SDPA dynamic cache.

Batch	FullKV (tokens/s)	PASVD (tokens/s)	Speedup (×)	Throughput increase (%)
1	20.96	**49.58**	2.37	136.56
2	24.51	**73.59**	3.00	200.31
3	11.68	**88.11**	7.54	654.31

Table 7 evaluates how execution scales with batch size. PASVD improves aggregate generation-stage throughput by 2.37× at batch size 1 and up to 7.54× at batch size 3, where FullKV operates near GPU-memory capacity.

5.4.3 Ablation on Selective Reconstruction

Compared with full reconstruction, PASVD's selective reconstruction improves throughput from 3.08–3.47 to 68.29–74.21 tokens/s on Qwen3-8B with a 32K context length on an NVIDIA RTX 5090 GPU, achieving an average $22.01\times$ speedup. This confirms the efficiency of selection before reconstruction.

6 Conclusion

PASVD jointly factorizes pre-RoPE keys across heads, assigns page-specific ranks, and reconstructs only compressed-space-selected keys. Across generation, retrieval, and reasoning tasks, it remains close to FullKV while reducing KV-cache storage. These results establish page-adaptive low-rank selection as an effective quality–memory trade-off for long-context inference.

AI USE STATEMENT

Generative AI tools were used to assist with language editing. The authors reviewed and verified all resulting content and take full responsibility for the paper.

REFERENCES

Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant J Nair, Ilya Soloveychik, and Purushotham Kamath. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. *Proceedings of Machine Learning and Systems*, 6:114–127, 2024.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. LongBench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

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.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S. Abdelfattah, and Kai-Chiang Wu. Palu: KV-cache compression with low-rank projection. In *International Conference on Learning Representations*, 2025.

Kyungsuk Cho, Yunji Park, Jiyun Kim, Byeongjun Kim, and Doowon Jeong. Conversational ai forensics: A case study on chatgpt, gemini, copilot, and claude. *Forensic Science International: Digital Investigation*, 52:301855, 2025.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *Conference on Language Modeling*, 2024.

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

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. *arXiv preprint arXiv:2410.14731*, 2024.

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

AI Meta. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI*, 2(5): 6, 2024.

Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. Eigen attention: Attention in low-rank space for KV cache compression. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024.

Ayan Sengupta, Siddhant Chaudhary, and Tanmoy Chakraborty. Value-guided KV compression for LLMs via approximated CUR decomposition. In *Advances in Neural Information Processing Systems*, 2025.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. FlexGen: High-throughput generative inference of large language models with a single GPU. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31094–31116, 2023.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

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

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

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.

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. *arXiv preprint arXiv:2306.14048*, 2023.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems*, 6:196–209, 2024.

Yuxuan Zhu, David H. Yang, Mohammad Mohammadi Amiri, Keerthiram Murugesan, Tejaswini Pedapati, and Pin-Yu Chen. OjaKV: Context-aware online low-rank KV cache compression. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026.

A Additional Experimental Results

A.1 LongBench Rank-Policy Ablation

We additionally evaluate the same rank policies on Qwen3-8B over eight LongBench tasks. Table A1 shows that the adaptive policy attains an average score of 34.34 with an average rank of 75.58, reducing K-side rank allocation by 21.27% relative to fixed rank 96. Its average is within 0.23 points of fixed rank 96, although the preferred policy varies by task. This complements the RULER analysis by showing that content-dependent rank allocation remains competitive on heterogeneous real-world inputs.

Table A1: Rank-policy ablation on Qwen3-8B over eight LongBench tasks. "Saved" is the reduction in K-side rank allocation relative to fixed rank 96.

Rank policy	Qasper	MFQA	Hotpot QA	2Wiki MQA	Gov Report	QMSum	TREC	RepoBench-P	Avg.	Avg. rank	Saved (%)
Fixed 64	**46.44**	53.17	50.20	38.99	34.45	**26.27**	**7.00**	**21.06**	**34.70**	64.00	33.33
Adaptive 64–96	45.38	**53.63**	**52.25**	**39.95**	34.69	24.79	5.00	19.00	34.34	75.58	21.27
Fixed 96	46.07	53.62	**52.25**	39.35	**34.75**	25.53	4.00	20.98	34.57	96.00	0.00

A.2 Full MMLU Evaluation on Qwen3-8B

We evaluate all five methods on the complete official MMLU test split in the zero-shot setting. Each result in Table A2 covers all 14,042 examples. PASVD reaches 72.94% accuracy, 0.04 points below FullKV and 4.71 points above the strongest evaluated compressed baseline.

Table A2: Zero-shot MMLU accuracy on Qwen3-8B using all 14,042 official test examples. Δ is the absolute accuracy-point change from FullKV; bold denotes the best compressed method.

Method	Accuracy (%)	Δ
FullKV	72.98	0.00
PaLu	63.92	-9.06
OjaKV	67.48	-5.50
ReCalKV	68.23	-4.75
PASVD	**72.94**	-0.04

A.3 Pre-RoPE Selection Diagnostic

PASVD uses pre-RoPE scoring as a content-retrieval proxy, not as an estimate of the exact position-aware ranking. We evaluate this proxy on eight 2,048-token WikiText-2 segments using Qwen3-8B, covering layers 4, 16, and 28 at four query positions per segment. The diagnostic follows the evaluated PASVD configuration: 96-token pages, page-adaptive ranks in $\{80, 96\}$ at a 0.997 energy threshold, and an INT8 basis. Within the selectable historical pool, we measure the post-RoPE attention mass captured by pre-RoPE top-64 and normalize it by the maximum mass attainable by an equal-budget post-RoPE oracle. Sink, recent, and active tokens are excluded because they are retained independently of selection.

Table A3: Post-RoPE attention mass (%) retained by PASVD's pre-RoPE top-64 proxy on Qwen3-8B. Oracle uses the same selection budget.

Selection	Captured mass	Oracle mass	Normalized mass
Pre-RoPE top-64	52.76	60.74	84.11

As shown in Table A3, the proxy captures 84.11% of the attention mass retained by an equal-budget oracle. Selected keys are subsequently reconstructed and evaluated with their exact positional

rotations, while the protected token sets provide a complementary path for position-sensitive local context.

A.4 VALUE-CACHE QUANTIZATION

The value cache uses the same symmetric max-absolute INT8 quantizer as the key basis. For the value vector $v_{p,i}^{(h)} \in \mathbb{R}^d$ of token i and KV head h in page p, PASVD stores

$$c_{p,i,h} = \frac{\max_j |v_{p,i,j}^{(h)}|}{127}, \qquad \overline{v}_{p,i,j}^{(h)} = \mathrm{clip}\left(\mathrm{round}\, \frac{v_{p,i,j}^{(h)}}{c_{p,i,h}}, -127, 127\right). \tag{13}$$

The scale $c_{p,i,h}$ is stored in FP16, and the selected values are dequantized to the attention-compute precision after gathering.