COFEATKV: COALESCED-FEATURE LOW-RANK COMPRESSION FOR POST-ROPE KV CACHES

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have achieved strong performance across many applications, but long-context inference is increasingly limited by the KV cache, whose size grows linearly with context length. Low-rank approximation is a widely used approach for KV-cache compression, but it faces a trade-off between compression quality and decoding cost. Pre-RoPE keys retain stronger low-rank structure, enabling more accurate SVD compression, but require reconstruction and Rotary position encoding (RoPE) before attention. Post-RoPE keys can be used during decoding and avoid this repeated computation, but positional rotation weakens their low-rank structure and increases SVD approximation error. However, substantial redundancy remains across feature dimensions. We introduce CoFeatKV, a training-free method for post-RoPE key compression. CoFeatKV exploits the remaining redundancy across feature dimensions by grouping key features with similar structure and applying a low-rank projection within each group. Unlike prior methods that rely on dense feature-to-latent projections, CoFeatKV uses structured group-wise low-rank projections. The compressed cache stores only the group representations, feature coefficients, and group assignments. During decoding, queries are mapped directly into the compressed group space to compute attention scores, avoiding full reconstruction of historical keys and repeated RoPE on cached keys. Extensive experiments with popular LLMs on RULER, LongBench, and WikiText show that CoFeatKV maintains near-baseline quality while reducing Key-cache storage by up to 45.31% and increasing decoding throughput by up to 26.49%.

1 INTRODUCTION

Recent advances in LLMs have enabled their use in complex and demanding applications, which in turn require larger models and longer context lengths. To support these growing demands, modern LLMs have continued to scale in both parameter size and context length. Longer contexts allow models to process more complex inputs and preserve information over extended sequences, but this capability also increases the cost of inference. During decoding, previously computed keys and values are stored in the key–value (KV) cache so that they can be reused at each decoding step without recomputing the entire history. However, as the context becomes longer, more keys and values must be retained, causing the KV cache to grow linearly with sequence length. As a result, the KV cache can become a dominant source of GPU memory consumption during long-context inference. This growing memory cost has motivated extensive research on reducing the storage and access overhead of the KV cache.

Existing KV-cache optimization methods mainly follow three directions: token eviction, quantization, and low-rank compression. Token eviction reduces memory and computation by keeping only a subset of past tokens, but discarded tokens may become useful at later decoding steps. Quantization lowers the precision of cached keys and values, reducing storage and memory traffic while preserving the original cache structure. However, because the representation dimension remains unchanged, quantization does not directly reduce attention computation. Low-rank compression instead exploits redundancy in the feature space of key representations and stores them in a lower-dimensional form, reducing the storage cost of the KV cache.

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.

Within low-rank compression, an important distinction is whether keys are compressed before or after RoPE (Su et al., 2021). Pre-RoPE keys exhibit stronger low-rank structure, but require reconstruction and RoPE before attention. SALS (Mu et al., 2025) reduces this cost by first selecting important tokens, so that only their keys need to be reconstructed and rotated. However, this selection requires an additional attention computation over the compressed keys. Post-RoPE methods avoid repeated RoPE and operate directly on position-aware cached keys. PuzzleKV (Wang et al., 2026) further adopts page-based SVD to avoid repeated global updates and reduce approximation error. However, its factorization is still performed on post-RoPE keys, where the low-rank structure is weaker, and the page-based projections introduce additional projection-matrix storage overhead. Therefore, achieving high compression quality without key reconstruction or repeated RoPE remains a key challenge for low-rank KV-cache compression.

To address these limitations, we introduce **CoFeatKV**, a training-free method for post-RoPE key compression. To preserve more attention-relevant information under the weakened low-rank structure after RoPE, CoFeatKV groups features according to their importance and similarity. Important features are kept as separate group representatives, while similar features share the same group direction. Instead of mapping all d feature dimensions to all m latent dimensions, each feature is assigned to only one group. This structured assignment replaces the dense $d \times m$ feature-to-latent projection with d feature coefficients and compact group indices, reducing per-page projection storage. During decoding, queries use the same feature-to-group mapping to interact directly with the stored group directions. Attention is computed in the compressed space without reconstructing historical keys or reapplying RoPE. Pages are compressed only after they are full, while the active page remains dense. CoFeatKV therefore preserves more attention-relevant information in post-RoPE keys while reducing projection storage and enabling reconstruction-free attention.

Our contributions are:

- We propose CoFeatKV, which forms groups around important key features and assigns the remaining features by correlation. It applies low-rank approximation within each group to reduce the low-rank approximation error after RoPE.
- We replace dense projection matrices with one coefficient and one group index per feature, reducing storage and computing attention without reconstructing cached keys or repeating RoPE.
- Extensive experiments on RULER, LongBench, and WikiText show near-baseline quality across models up to 32B and contexts up to 32K.

2 Related Work

Selecting cache entries and reducing precision. One line of work asks which cached information to retain. H2O and Scissorhands use attention history to identify important tokens (Zhang et al., 2023; Liu et al., 2023); StreamingLLM retains initial attention sinks and a recent window for streaming generation (Xiao et al., 2023). SnapKV selects prompt positions using an observation window (Li et al., 2024), while PyramidKV and Ada-KV adapt cache budgets across layers and heads, respectively (Cai et al., 2024; Feng et al., 2024). DuoAttention distinguishes heads that need full-context retrieval from heads that mainly use recent tokens and sinks (Xiao et al., 2024). Quest instead selects pages for each query to reduce cache reads; this is different from permanently evicting those pages (Tang et al., 2024). Another line reduces the cost of each retained entry. KIVI uses per-channel key quantization and per-token value quantization (Liu et al., 2024b); KVQuant handles activation distributions and outliers (Hooper et al., 2024); GEAR adds low-rank and sparse corrections to quantization (Kang et al., 2024). Feature redundancy is also explored by ThinK, which prunes key channels according to their contribution to attention (Xu et al., 2024). CoFeatKV shares the goal of reducing feature storage, but represents feature columns through factors or shared prototypes rather than discarding token positions. Keeping every position does not make the approximation lossless, and its benefit must be tested at a fixed storage budget.

Low-rank and shared representations. Low-rank compression asks how much information can be retained in fewer coordinates. ASVD accounts for activation distributions when factorizing model weights and also applies reduced projections to KV storage (Yuan et al., 2023); SVD-LLM studies

truncation-aware weight compression (Wang et al., 2024). These works motivate attention to activation error, although weight compression and online cache compression are different problems. Eigen Attention computes attention in a reduced space (Saxena et al., 2024), whereas Palu factorizes projection layers, stores compressed intermediate states, and reconstructs keys and values as needed (Chang et al., 2024). Sharing can also occur across layers: MiniCache merges similar adjacent-layer states, and xKV jointly factorizes grouped-layer caches (Liu et al., 2024a; Chang et al., 2025). ShadowKV combines low-rank key storage with sparse reconstruction and value offloading (Sun et al., 2024). The closest page-local comparison is PuzzleKV, which independently factorizes completed pages and computes attention over dense and factorized pages (Wang et al., 2026). We therefore do not claim paging or reconstruction-free attention alone as new. CoFeatKV instead changes the representation inside each post-RoPE key page: all feature columns are assigned to shared page directions, and each column stores only one coefficient and one group index. This one-sparse feature-to-group mapping avoids the dense projection factor used by conventional latent representations. The distinction calls for comparisons against page-wise SVD at matched bytes, including representative and assignment metadata, rather than at the same nominal rank.

3 BACKGROUND

3.1 DECODER ATTENTION AND THE KV CACHE

For one attention head, let $Q \in \mathbb{R}^{T_q \times d}$, $K \in \mathbb{R}^{T_k \times d}$, and $V \in \mathbb{R}^{T_k \times d_v}$. Scaled dot-product attention (Vaswani et al., 2017) is computed as

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V. \tag{1}$$

During decoding, historical keys and values are stored in the KV cache and reused in subsequent attention steps. For low-rank Key compression, an important choice is whether compression is applied before or after RoPE. We next analyze the role of RoPE and its effect on Key representations

3.2 THE IMPORTANCE OF ROPE

During inference, RoPE is applied to queries and keys to incorporate positional information into the query–key interaction, allowing attention to distinguish tokens at different positions. For a token at position t, let $\bar{Q}_t$ and $\bar{K}_t$ denote the representations before RoPE:

$$Q_t = \bar{Q}_t R_t, \qquad K_t = \bar{K}_t R_t, \tag{2}$$

where R_t is the position-dependent rotation matrix used by RoPE. Since R_t varies across token positions, a single rotated low-rank basis cannot be shared across historical keys. Applying RoPE in the compressed representation would therefore require a different basis transformation for each token position. Therefore, pre-RoPE methods reconstruct the required keys and apply RoPE at their original positions, while post-RoPE compression stores the rotated keys and requires no repeated RoPE during decoding.

3.3 POST-ROPE COMPRESSION AND LOW-RANK DEGRADATION

Post-RoPE compression stores the rotated keys used by attention, so Key reconstruction and repeated RoPE are not required during decoding. However, applying RoPE also makes low-rank compression more difficult.

For a pre-RoPE Key matrix $\bar{K} \in \mathbb{R}^{S \times d}$, a rank-$r$ truncated SVD gives

$$\bar{K} \approx ZB^\top, \qquad Z = \bar{K}B, \tag{3}$$

where $B \in \mathbb{R}^{d \times r}$ is the top-r right singular vectors and $Z \in \mathbb{R}^{S \times r}$ is the low-dimensional Key representation. For token t, with z_t denoting the t-th row of Z,

$$\widehat{k}_t = z_t B^\top R_t. \tag{4}$$

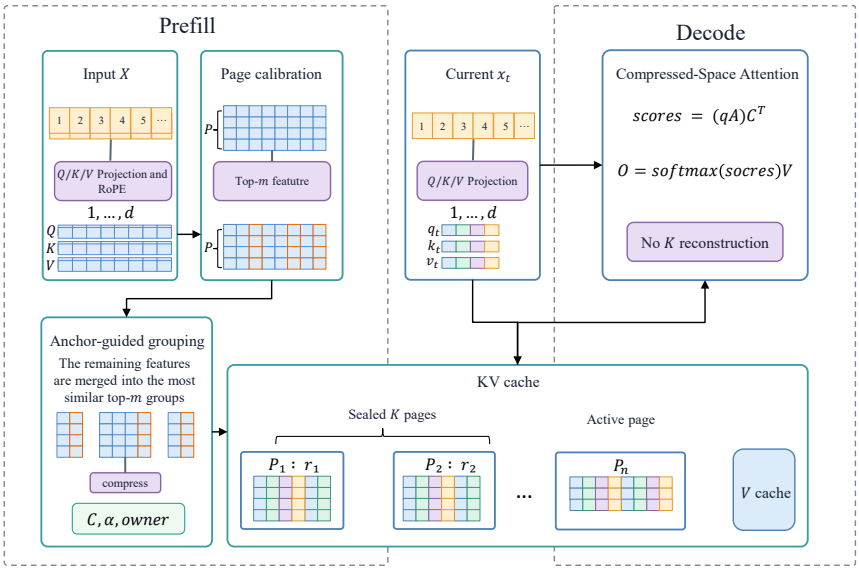


Figure 1: Overview of CoFeatKV. During prefill, Q/K/V projection and RoPE produce post-RoPE key pages. For each completed page, CoFeatKV selects m anchor features, assigns the remaining features to the most similar anchors, and stores page directions C, coefficients α, and owner indices. During decoding, the query uses the same sparse feature-to-group mapping to compute attention in the compressed space. The active page remains dense, while sealed pages avoid key reconstruction and repeated RoPE.

Since R_t varies across token positions, the effective basis $B^\top R_t$ also varies, weakening the shared low-rank structure after RoPE. This effect is reflected in the equal-rank experiment in Section 5, where post-RoPE SVD shows larger quality degradation than pre-RoPE SVD at the same rank.

4 Method

4.1 Overview

CoFeatKV improves the compression quality and storage efficiency of existing post-RoPE low-rank methods while retaining their efficient decoding. First, we select the top-m key feature dimensions by attention importance as group representatives, then assign each remaining key feature dimension to the most similar representative. Second, we store each key feature dimension with one group index and one scalar coefficient, reducing the storage overhead of the dense projection matrices used in conventional low-rank methods. Third, during decoding, we compute attention directly from the compressed keys, without key reconstruction or repeated RoPE computation.

4.2 Attention-Aware Feature Coalescing

To reduce the information loss caused by the weakened low-rank structure after RoPE, CoFeatKV compresses each completed post-RoPE key page by grouping key feature dimensions according to their attention importance and similarity. We first identify the most important key feature dimensions and use each of them to initialize a separate group. The remaining feature dimensions are then assigned to the group with the most similar key structure. This prevents high-importance features from being merged with one another, while allowing similar remaining features to share the same group representation.

For each KV head, we partition the post-RoPE keys along the token dimension into fixed-size pages of P tokens. Let $K \in \mathbb{R}^{P\times d}$ denote one completed page, and let $k_j = K_{:,j} \in \mathbb{R}^P$ denote feature dimension j across the page. Since the contribution of a feature to the attention score depends on

both its key representation and the corresponding query dimension, we estimate feature importance using

$$w_j = \frac{1}{GP} \sum_{g,t} Q_{\text{page}}[g,t,j]^2, \qquad s_j = w_j \|k_j\|_2^2, \tag{5}$$

where w_j measures the query importance of feature j, and s_j is the resulting importance score after combining w_j with the squared ℓ_2 norm of the corresponding key feature. A larger s_j indicates a more important feature for attention.

Given a group budget $m < d$, we select the top-m features according to s_j as group representatives. For each remaining feature ℓ, we measure the information loss caused by merging it with representative h. Specifically, we consider the query-weighted pair $\left[\sqrt{w_h}k_h, \sqrt{w_\ell}k_\ell\right]$ and use the residual error of its best rank-one approximation as the merge cost:

$$\Delta(h,\ell) = \frac{1}{2}\left(a_h + b_\ell - \sqrt{(a_h - b_\ell)^2 + 4c_{h\ell} + \varepsilon}\right), \tag{6}$$

where $a_h = w_h\|k_h\|_2^2$, $b_\ell = w_\ell\|k_\ell\|_2^2$, and $c_{h\ell} = w_h w_\ell (k_h^\top k_\ell)^2$. A smaller $\Delta(h,\ell)$ means that the two feature dimensions can be represented by one shared direction with less information loss. We therefore assign

$$g(\ell) = \arg\min_h \Delta(h,\ell). \tag{7}$$

The closed form above is the residual of the best rank-one approximation of the weighted two-feature matrix; its derivation is given in Appendix A.1. The assignment is a greedy, page-local criterion rather than a claim of globally optimal clustering. In particular, w_j is computed once from the queries observed while the page is being formed; future queries do not trigger regrouping. The effect of this calibration choice is covered by the conditional attention-error bound in Appendix A.3.

4.3 Structured Shared-Feature Representation

To reduce the projection storage overhead of conventional low-rank methods, CoFeatKV uses one shared group direction and one scalar coefficient for each key feature dimension, reducing the storage required for projection coefficients from dm to d.

For group r, let $\mathcal{G}_r$ denote the set of key feature dimensions assigned to this group:

$$\mathcal{G}_r = \{j : g(j) = r\}. \tag{8}$$

Then, we compute one shared direction to represent all features in $\mathcal{G}_r$. Specifically,

$$S_r = \sum_{j \in \mathcal{G}_r} w_j k_j k_j^\top, \qquad c_r = \text{principal_eigenvector}(S_r). \tag{9}$$

Here, S_r summarizes the query-weighted structure of the features in group r, and $c_r \in \mathbb{R}^P$ is their shared direction.

Although features in the same group share a similar direction, their magnitudes along this direction can be different. Therefore we store one scalar coefficient for each feature:

$$\alpha_j = c_{g(j)}^\top k_j, \qquad \widehat{K}_{:,j} = \alpha_j c_{g(j)}. \tag{10}$$

For a fixed unit-norm group direction, α_j is the projection coefficient of feature j onto that direction.

Collecting the m group directions gives

$$C = [c_1, \ldots, c_m] \in \mathbb{R}^{P \times m}. \tag{11}$$

Collecting the approximations of all key feature dimensions gives

$$\widehat{K} = CA^\top, \qquad A_{j,r} = \begin{cases} \alpha_j, & r = g(j), \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

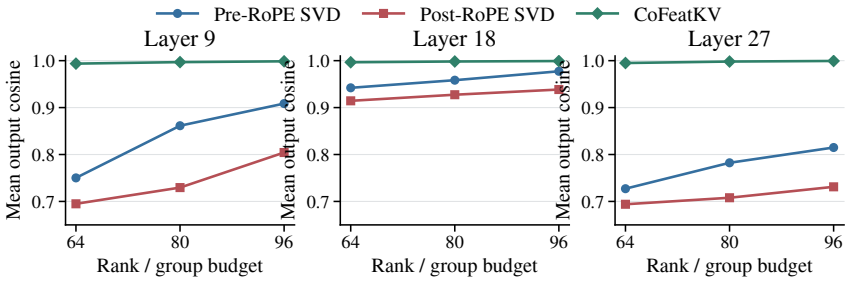


Figure 2: Layer-wise attention-output fidelity on Qwen3-8B. Each panel is a zero-based layer and reports mean output cosine similarity to Full. Values are averaged over all query heads and tokens in one 4096-token WikiText-2 window.

Since each key feature dimension belongs to one group, CoFeatKV does not need to store the dense coefficient matrix A. Instead, for each page it stores only the group directions

$$C \in \mathbb{R}^{P\times m}, \qquad \alpha \in \mathbb{R}^{d}, \qquad \text{owner} \in \{1,\dots,m\}^{d}, \tag{13}$$

where $\text{owner}_j = g(j)$ is the group assignment of feature j, and α_j is its scalar coefficient. In contrast, a conventional dense approximation stores $d \times m$ projection coefficients. CoFeatKV reduces this to one coefficient and one group index per feature, so the number of projection coefficients decreases from dm to d. Correlated features reuse the same group direction, which reduces the per-page projection storage.

4.4 Reconstruction-Free Grouped Attention

During decoding, reconstructing the full historical Key matrix before attention introduces additional computation. To reduce this cost, CoFeatKV computes attention from the grouped representation. For a post-RoPE query $q \in \mathbb{R}^{d}$,

$$q\widehat{K}^{\top} = qAC^{\top} = (qA)C^{\top}. \tag{14}$$

Since both the query and the compressed Key representation are in the post-RoPE space, attention can be computed as $(qA)C^{\top}$. The query uses the feature groups determined by the Key page to form qA, which is then multiplied by $C^{\top}$ to obtain the attention scores. This computation requires neither Key reconstruction nor additional RoPE.

Table 1: LongBench results on Meta-Llama-3.1-8B-Instruct over four question-answering tasks. Avg. is the unweighted mean.

Method	Qasper	MFQA	HotpotQA	2WikiMQA	Avg.
FullKV	46.05	53.68	52.25	39.35	47.83
PaLu (Chang et al., 2024)	29.15	38.61	25.07	36.41	32.31
OjaKV (Zhu et al., 2025)	35.65	46.77	38.94	34.28	38.91
ReCalKV (Yan et al., 2025)	24.47	40.25	8.82	6.22	19.94
CoFeatKV ($m = 80$)	45.76	54.29	52.25	42.68	48.75

5 Experiments

5.1 Experimental setup

Models and benchmarks. Our experiments use Meta-Llama-3.1-8B-Instruct (Dubey et al., 2024), Qwen3-8B, Qwen3-32B, and Llama-2-7B. 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. We use FullKV with the original KV cache as the uncompressed reference. We compare CoFeatKV with four low-rank KV-cache compression methods covering different designs: PaLu (Chang et al., 2024) performs offline factorization of projection weights, ReCalKV (Yan et al., 2025) determines ranks using calibration data, OjaKV (Zhu et al., 2025) updates the low-rank subspace during inference, and PuzzleKV (Wang et al., 2026) applies decomposition at the page level. For methods with released implementations, we run the baselines in our local evaluation environment.

CoFeatKV configuration. Runtime and attention-output experiments use 32-token pages. The page-local representation study uses 128-token pages to expose the effect of the group budget. Unless stated otherwise, we test $m \in \{64, 80, 96\}$.

Table 2: RULER results on Llama-3.1-8B-Instruct and Qwen3-8B at 16K and 32K context lengths.

Model	Length	Method	S1	S2	S3	MK1	MK2	MV	MQ	Avg.
Llama-3.1-8B Instruct	16K	FullKV	100.00	100.00	100.00	99.00	100.00	99.50	99.25	99.68
		PaLu (Chang et al., 2024)	**100.00**	99.00	**100.00**	98.00	95.00	96.25	96.50	97.82
		OjaKV (Zhu et al., 2025)	97.00	**100.00**	98.00	97.00	41.00	97.75	84.25	87.86
		ReCalKV (Yan et al., 2025)	**100.00**	**100.00**	**100.00**	95.00	99.00	94.00	96.50	97.79
		PuzzleKV (Wang et al., 2026)	98.80	95.80	82.00	98.20	**100.00**	91.10	98.85	94.96
		CoFeatKV ($m = 64$)	**100.00**	**100.00**	**100.00**	99.20	**100.00**	**98.05**	99.85	99.59
		CoFeatKV ($m = 80$)	**100.00**	**100.00**	**100.00**	**100.00**	99.80	97.90	**99.95**	**99.66**
Llama-3.1-8B Instruct	32K	FullKV	100.00	100.00	99.00	97.00	100.00	99.50	99.75	99.32
		PaLu (Chang et al., 2024)	**100.00**	**100.00**	99.00	97.00	97.00	95.50	97.00	97.93
		OjaKV (Zhu et al., 2025)	**100.00**	99.00	98.00	94.00	24.00	97.50	94.25	86.68
		ReCalKV (Yan et al., 2025)	**100.00**	**100.00**	96.00	96.00	96.00	87.25	95.25	95.79
		PuzzleKV (Wang et al., 2026)	99.80	97.60	83.00	96.60	99.80	91.40	98.85	95.29
		CoFeatKV ($m = 64$)	**100.00**	**100.00**	**100.00**	99.00	**100.00**	**99.25**	**99.75**	99.71
		CoFeatKV ($m = 80$)	**100.00**	**100.00**	**100.00**	**99.80**	99.80	99.15	99.70	**99.78**
Qwen3-8B	16K	FullKV	100.00	100.00	100.00	100.00	100.00	99.25	99.25	99.79
		PaLu (Chang et al., 2024)	**100.00**	98.00	99.00	98.00	96.00	96.00	92.25	97.04
		OjaKV (Zhu et al., 2025)	84.00	81.00	60.00	89.00	60.00	32.75	69.25	68.00
		ReCalKV (Yan et al., 2025)	**100.00**	**100.00**	**100.00**	**100.00**	92.00	87.25	69.00	92.61
		PuzzleKV (Wang et al., 2026)	**100.00**	**100.00**	99.00	99.60	**99.80**	**99.90**	**99.90**	99.74
		CoFeatKV ($m = 64$)	**100.00**	**100.00**	**100.00**	**100.00**	99.59	99.65	99.83	99.87
		CoFeatKV ($m = 80$)	**100.00**	**100.00**	**100.00**	**100.00**	99.60	99.70	99.85	**99.88**
Qwen3-8B	32K	FullKV	100.00	100.00	100.00	99.00	99.00	97.50	99.50	99.29
		PaLu (Chang et al., 2024)	**100.00**	98.00	99.00	99.00	59.00	90.00	98.25	91.89
		OjaKV (Zhu et al., 2025)	78.00	66.00	55.00	70.00	28.00	34.75	56.25	55.43
		ReCalKV (Yan et al., 2025)	**100.00**	99.00	**100.00**	98.00	85.00	95.75	93.00	95.82
		PuzzleKV (Wang et al., 2026)	**100.00**	99.80	98.60	**99.80**	97.00	98.35	99.60	99.02
		CoFeatKV ($m = 64$)	**100.00**	**100.00**	**100.00**	99.20	96.80	99.55	99.60	99.31
		CoFeatKV ($m = 80$)	**100.00**	**100.00**	**100.00**	99.60	**97.60**	**99.75**	**99.75**	**99.53**

5.2 Compression before and after RoPE

Figure 2 shows the mean cosine similarity between the attention outputs produced by compressed keys and those produced by the original keys. As the rank or group budget increases from 64 to 96, both pre-RoPE SVD and post-RoPE SVD improve, while post-RoPE SVD remains lower than pre-RoPE SVD. In contrast, CoFeatKV maintains cosine similarity close to 1.0 across all three layers and all tested budgets. This shows that CoFeatKV preserves more attention-relevant information when compressing post-RoPE keys and keeps the resulting attention outputs close to those of the original keys.

5.3 LONG-CONTEXT COMPARISON ON 8B MODELS

We next evaluate long-context understanding on Meta-Llama-3.1-8B-Instruct using four LongBench question-answering tasks. The table reports the official LongBench task metrics, with FullKV and baseline methods evaluated on the same models for comparison.

As shown in Table 1, we compare CoFeatKV with FullKV and three low-rank baselines on four LongBench tasks. CoFeatKV achieves the highest average score of 48.75, compared with 47.83 for FullKV. It improves MFQA, and 2WikiMQA, and matches FullKV on HotpotQA. Compared with the low-rank baselines, CoFeatKV preserves important key features and shares representations only among similar features, reducing the information loss of post-RoPE compression. CoFeatKV exceeds FullKV on several tasks because compression changes attention scores and can lead to different generations; since LongBench measures task-level answer quality rather than agreement with FullKV, some of these changes receive higher scores.

Table 2 compares CoFeatKV with FullKV and four low-rank baselines on Llama-3.1-8B-Instruct and Qwen3-8B at 16K and 32K context lengths. CoFeatKV matches FullKV across both models and context lengths. With $m = 80$, it reaches 99.66 and 99.78 on Llama-3.1-8B-Instruct, compared with 99.68 and 99.32 for FullKV. On Qwen3-8B, CoFeatKV achieves 99.88 and 99.53, compared with 99.79 and 99.29 for FullKV. It also outperforms the other compressed methods in all four settings. These results show that CoFeatKV preserves long-context performance under post-RoPE key compression.

5.4 QWEN3-32B LONG-CONTEXT EVALUATION

we evaluate CoFeatKV on Qwen3-32B to test whether the compressed representation remains effective at a larger model scale and longer context. Since the baseline methods do not report results on 32B models, we compare CoFeatKV with the standard FullKV setting on RULER. We use the first 100 examples from each of 11 tasks and evaluate both 16K and 32K context lengths using task accuracy.

Table 3: Per-task Qwen3-32B RULER accuracy. S1–S3 achieve 100.0 in all settings and are included in the average.

Context	Method	NIAH-MK1	NIAH-MK2	NIAH-MV	NIAH-MQ	VT	CWE	FWE	QA-1	Avg.
16K	Full	100.0	100.0	100.0	100.0	100.0	49.30	91.00	86.00	93.30
16K	$m = 64$	100.0	100.0	100.0	100.0	100.0	71.30	92.00	84.00	95.21
16K	$m = 80$	100.0	100.0	100.0	100.0	100.0	63.20	92.33	86.00	94.68
32K	Full	100.0	100.0	100.0	100.0	100.0	51.90	92.67	82.00	93.32
32K	$m = 64$	100.0	100.0	100.0	100.0	100.0	60.50	85.00	81.00	93.32
32K	$m = 80$	100.0	100.0	100.0	100.0	100.0	58.00	91.67	85.00	94.06

Table 3 reports the RULER results on Qwen3-32B. CoFeatKV preserves 100% accuracy on all NIAH tasks and VT at both 16K and 32K context lengths. At 16K, $m = 64$ achieves an average score of 95.21, compared with 93.30 for FullKV, with the largest improvement on CWE. At 32K, $m = 80$ reaches 94.06, compared with 93.32 for FullKV, while improving CWE and QA-1. The variation mainly comes from CWE, FWE, and QA-1, whereas the retrieval tasks remain unchanged. These results show that CoFeatKV maintains long-context performance when scaling from 8B to Qwen3-32B.

5.5 ABLATION STUDIES

We examine how feature merging changes storage and prediction quality, and how the group budget affects Key approximation, perplexity, and decode throughput. The short end-to-end grouping study, the page-local budget sweep, and the archived runtime comparison use different settings and are reported separately.

5.5.1 EFFECT OF THE GROUP BUDGET.

We vary m from 8 to 80 on 32 WikiText-2 sequences in the Qwen3-8B page diagnostic, with 128 tokens per page. We measure weighted Key approximation error at layers 9, 18, and 27, using zero-based layer indices. This metric measures Key reconstruction under the compression objective, not task accuracy or attention-output error.

Table 4: Group-budget sensitivity of weighted Key approximation error. Results average over 32 WikiText-2 sequences with page size 128. Lower is better; bold marks the smallest error in each layer.

m	Layer 9	Layer 18	Layer 27
8	0.3258	0.2287	0.2518
16	0.2185	0.1525	0.1695
24	0.1567	0.1136	0.1250
32	0.1196	0.0900	0.0989
48	0.0833	0.0643	0.0718
64	0.0703	0.0525	0.0605
80	**0.0665**	**0.0479**	**0.0566**

As shown in Table 4, increasing m consistently reduces the approximation error across all three layers, with smaller gains at larger budgets. In Layer 9, increasing m from 32 to 64 reduces the error from 0.1196 to 0.0703, while increasing m from 64 to 80 further reduces it to 0.0665. Layers 18 and 27 show the same trend.

5.5.2 EFFECT OF GROUP BUDGET ON COMPRESSION, THROUGHPUT, AND QUALITY

We evaluate Qwen3-8B results on WikiText-2 with $m \in \{64, 80, 96\}$ to measure how the group budget affects KV-cache compression, decoding throughput, and model quality. All runs use BF16 on a single RTX 3090 with batch size 1, a 128-token prefill, followed by 2048 autoregressive decoding steps.

Table 5: Compression, throughput, and perplexity under different m. † denotes K savings for the compact storage format, not measured allocation savings in these archived timing runs. FullKV PPL is 6.226532.

		Decode throughput ↑			Perplexity ↓		
m	K saving† (%)	FullKV (tok/s)	CoFeatKV (tok/s)	Gain (%)	CoFeatKV	Δ	Relative Δ (%)
64	45.31	27.41	33.63	+22.69	6.254589	+0.028057	+0.451
80	32.81	27.70	35.04	+26.49	6.190962	-0.035570	-0.571
96	20.31	27.35	33.79	+23.53	6.225549	-0.000983	-0.016

Table 5 shows that CoFeatKV improves decoding throughput by 22.69–26.49% across different m, while keeping perplexity close to FullKV. Smaller m provides higher Key-cache saving, reaching 45.31% at $m = 64$.

6 CONCLUSION

CoFeatKV groups post-RoPE key features by importance and correlation, then approximates each group with a shared direction. The compressed cache stores one coefficient and one group index per feature and supports attention without reconstructing historical keys or repeating RoPE. Experiments on RULER, LongBench, and WikiText show near-baseline quality, with long-context tests covering models up to 32B and contexts up to 32K. The compact format reduces sealed-key storage by up to 45.31%, while separate runtime tests show higher decode throughput than native dense attention.

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

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV Cache Compression based on Pyramidal Information Funneling. *arXiv preprint arXiv:2406.02069*, 2024. doi: 10.48550/arXiv.2406.02069.

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. doi: 10.48550/arXiv.2407.21118.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Hung-Yueh Chiang, Yash Akhauri, Xilai Dai, Huiqiang Jiang, Yucheng Li, Luis Ceze, Kai-Chiang Wu, and Mohamed S. Abdelfattah. xKV: Cross-Layer KV-Cache Compression via Aligned Singular Vector Extraction. *arXiv preprint arXiv:2503.18893*, 2025. doi: 10.48550/arXiv.2503.18893.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-KV: Optimizing KV Cache Eviction by Adaptive Budget Allocation for Efficient LLM Inference. *arXiv preprint arXiv:2407.11550*, 2024. doi: 10.48550/arXiv.2407.11550.

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. *arXiv preprint arXiv:2401.18079*, 2024. doi: 10.48550/arXiv.2401.18079.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. GEAR: An Efficient KV Cache Compression Recipe for Near-Lossless Generative Inference of LLM. *arXiv preprint arXiv:2403.05527*, 2024. doi: 10.48550/arXiv.2403.05527.

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. *arXiv preprint arXiv:2404.14469*, 2024. doi: 10.48550/arXiv.2404.14469.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. MiniCache: KV Cache Compression in Depth Dimension for Large Language Models. *arXiv preprint arXiv:2405.14366*, 2024a. doi: 10.48550/arXiv.2405.14366.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the Persistence of Importance Hypothesis for LLM KV Cache Compression at Test Time. *arXiv preprint arXiv:2305.17118*, 2023. doi: 10.48550/arXiv.2305.17118.

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*, 2024b. doi: 10.48550/arXiv.2402.02750.

Junlin Mu, Hantao Huang, Jihang Zhang, Minghui Yu, Tao Wang, and Yidong Li. SALS: Sparse Attention in Latent Space for KV cache Compression. *arXiv preprint arXiv:2510.24273*, 2025. doi: 10.48550/arXiv.2510.24273.

Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. Eigen Attention: Attention in Low-Rank Space for KV Cache Compression. *arXiv preprint arXiv:2408.05646*, 2024. doi: 10.48550/arXiv.2408.05646.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding. *arXiv preprint arXiv:2104.09864*, 2021. doi: 10.48550/arXiv.2104.09864.

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. *arXiv preprint arXiv:2410.21465*, 2024. doi: 10.48550/arXiv.2410.21465.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-Aware Sparsity for Efficient Long-Context LLM Inference. *arXiv preprint arXiv:2406.10774*, 2024. doi: 10.48550/arXiv.2406.10774.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv preprint arXiv:1706.03762*, 2017. doi: 10.48550/arXiv.1706.03762.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware Singular Value Decomposition for Large Language Model Compression. *arXiv preprint arXiv:2403.07378*, 2024. doi: 10.48550/arXiv.2403.07378.

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. doi: 10.48550/arXiv.2608.23843.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient Streaming Language Models with Attention Sinks. *arXiv preprint arXiv:2309.17453*, 2023. doi: 10.48550/arXiv.2309.17453.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. DuoAttention: Efficient Long-Context LLM Inference with Retrieval and Streaming Heads. *arXiv preprint arXiv:2410.10819*, 2024. doi: 10.48550/arXiv.2410.10819.

Yuhui Xu, Zhanming Jie, Hanze Dong, Lei Wang, Xudong Lu, Aojun Zhou, Amrita Saha, Caiming Xiong, and Doyen Sahoo. ThinK: Thinner Key Cache by Query-Driven Pruning. *arXiv preprint arXiv:2407.21018*, 2024. doi: 10.48550/arXiv.2407.21018.

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. doi: 10.48550/arXiv.2505.24357.

Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: Activation-aware Singular Value Decomposition for Compressing Large Language Models. *arXiv preprint arXiv:2312.05821*, 2023. doi: 10.48550/arXiv.2312.05821.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. *arXiv preprint arXiv:2306.14048*, 2023. doi: 10.48550/arXiv.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 with Oja's Rule. *arXiv preprint arXiv:2509.21623*, 2025. doi: 10.48550/arXiv.2509.21623.

A Additional Derivations and Error Analysis

This appendix gives the derivation of the feature-merging objective, the optimal shared direction used within each group, and an error bound for the resulting reconstruction-free attention computation. The bounds are stated for one KV head and one completed page; the same argument applies independently to every layer, KV head, and page.

A.1 Derivation of the pairwise merge cost

Let $k_h, k_\ell \in \mathbb{R}^P$ be two feature columns in a completed post-RoPE page. Given nonnegative query-energy weights w_h, w_ℓ, define

$$X_{h\ell} = [\sqrt{w_h}k_h, \sqrt{w_\ell}k_\ell] \in \mathbb{R}^{P\times 2}. \tag{15}$$

Merging the two features into one shared page direction is equivalent to approximating $X_{h\ell}$ with a rank-one matrix. By the Eckart–Young theorem, the minimum squared residual is

$$\min_{\mathrm{rank}(Y)\le 1} \|X_{h\ell} - Y\|_F^2 = \|X_{h\ell}\|_F^2 - \sigma_1(X_{h\ell})^2. \tag{16}$$

The two nonzero eigenvalues of $X_{h\ell}X_{h\ell}^\top$ are the eigenvalues of its 2×2 Gram matrix

$$G_{h\ell} = X_{h\ell}^\top X_{h\ell} = \begin{bmatrix} a_h & \sqrt{w_h w_\ell}\, k_h^\top k_\ell \\ \sqrt{w_h w_\ell}\, k_h^\top k_\ell & b_\ell \end{bmatrix}, \tag{17}$$

where

$$a_h = w_h\|k_h\|_2^2, \qquad b_\ell = w_\ell\|k_\ell\|_2^2, \qquad c_{h\ell} = w_h w_\ell (k_h^\top k_\ell)^2. \tag{18}$$

The largest eigenvalue is

$$\lambda_1(G_{h\ell}) = \frac{1}{2}\left(a_h + b_\ell + \sqrt{(a_h - b_\ell)^2 + 4c_{h\ell}}\right). \tag{19}$$

Since $\|X_{h\ell}\|_F^2 = a_h + b_\ell$, substituting this eigenvalue into Eq. equation 16 gives

$$\Delta(h,\ell) = \frac{1}{2}\left(a_h + b_\ell - \sqrt{(a_h - b_\ell)^2 + 4c_{h\ell}}\right). \tag{20}$$

The implementation adds a small ε inside the square root only for numerical stability. Thus, the cost used by CoFeatKV is the exact minimum weighted rank-one residual for a pair of feature dimensions, up to this numerical stabilizer.

The importance score used to choose representatives is the weighted energy

$$s_j = w_j\|k_j\|_2^2. \tag{21}$$

Selecting the top-m scores reserves one group direction for features with the largest contribution to the estimated attention energy of the page. Each remaining feature is then assigned by

$$g(\ell) = \arg\min_{h\in\mathcal{H}} \Delta(h,\ell), \qquad |\mathcal{H}| = m. \tag{22}$$

This is a greedy assignment rule. It does not solve the combinatorial global clustering problem, but it is page-local and requires only the feature statistics available when the page is sealed.

A.2 Optimal shared direction within a group

For a resulting group $\mathcal{G}_r$, CoFeatKV solves the weighted rank-one problem

$$\min_{\|c_r\|_2=1,\{\alpha_j\}} \sum_{j\in\mathcal{G}_r} w_j\|k_j - \alpha_j c_r\|_2^2. \tag{23}$$

For a fixed unit direction c_r, the minimizing coefficient is

$$\alpha_j = c_r^\top k_j. \tag{24}$$

Substituting this coefficient into the objective shows that the optimal direction is the principal eigenvector of

$$S_r = \sum_{j \in \mathcal{G}_r} w_j k_j k_j^\top, \qquad c_r = \mathrm{eigvec}_{\max}(S_r). \tag{25}$$

The minimized group residual is therefore

$$\mathcal{E}_r^2 = \sum_{j \in \mathcal{G}_r} w_j \|k_j\|_2^2 - \lambda_{\max}(S_r), \qquad \mathcal{E}_{\text{page}}^2 = \sum_{r=1}^{m} \mathcal{E}_r^2. \tag{26}$$

The pairwise cost is used to construct groups, whereas the group eigenvector is computed after all assignments are complete. Thus, Eq. equation 20 is an assignment loss; it is not claimed to be the final residual of a group containing more than two features.

A.3 Attention computation and error bound

Let $K \in \mathbb{R}^{P \times d}$ be the original post-RoPE page and let $\widehat{K}$ be its CoFeatKV approximation. Define $E = K - \widehat{K}$. For a query $q \in \mathbb{R}^d$, the unnormalized attention logits are

$$z = \frac{Kq}{\sqrt{d}}, \qquad \widehat{z} = \frac{\widehat{K}q}{\sqrt{d}}. \tag{27}$$

The logit error satisfies

$$\|z - \widehat{z}\|_2 = \frac{\|Eq\|_2}{\sqrt{d}} \le \frac{\|E\|_F \|q\|_2}{\sqrt{d}}. \tag{28}$$

This bound is conditional on the realized page approximation E. It does not require future queries to have the same distribution as the queries used to estimate w_j. A distribution shift can increase $\|Eq\|$, but its effect is explicitly controlled by the residual in the realized query direction.

Let $p = \mathrm{softmax}(z)$ and $\widehat{p} = \mathrm{softmax}(\widehat{z})$. Since the softmax Jacobian has Euclidean operator norm at most $1/2$,

$$\|p - \widehat{p}\|_2 \le \frac{\|E\|_F \|q\|_2}{2\sqrt{d}}. \tag{29}$$

For a value matrix $V \in \mathbb{R}^{P \times d_v}$, the attention outputs $y = p^\top V$ and $\widehat{y} = \widehat{p}^\top V$ satisfy

$$\|y - \widehat{y}\|_2 \le \frac{\|E\|_F \|q\|_2 \|V\|_F}{2\sqrt{d}}. \tag{30}$$

For multiple pages, the same result applies after concatenating the page errors, or more conservatively by summing the per-page bounds. The result does not assert exactness; it identifies the quantities governing the error: the page residual, query norm, attention scaling, and value magnitude.

A.4 DIRECT GROUPED ATTENTION WITHOUT RECONSTRUCTION

Let $C = [c_1, \ldots, c_m] \in \mathbb{R}^{P \times m}$ and define the sparse feature-to-group matrix $A \in \mathbb{R}^{d \times m}$ by

$$A_{j,r} = \begin{cases} \alpha_j, & r = \mathrm{owner}_j, \\ 0, & \text{otherwise.} \end{cases} \tag{31}$$

Then $\widehat{K} = CA^\top$ and

$$q\widehat{K}^\top = qAC^\top = (qA)C^\top, \qquad (qA)_r = \sum_{j:\mathrm{owner}_j = r} q_j \alpha_j. \tag{32}$$

The dense matrix A and the reconstructed historical Key matrix are never materialized. The implementation stores only C, one coefficient per feature, and one group index per feature. Since C is built from post-RoPE keys, no additional RoPE operation is applied to sealed pages during decode.