

MOB-KV: Mixture-of-Basis for Key-Value Cache Compression

Jinseop Yeom^{1,*} Taehyeon Kim^{2,*} Eunhyeok Park¹

¹Graduate School of Artificial Intelligence

²Department of Computer Science and Engineering

Pohang University of Science and Technology (POSTECH)

{jsyeom, taehyeonkim, eh.park}@postech.ac.kr

Abstract

In this paper, we present **MOB-KV**, a **Mixture-of-Basis** KV cache compression method that makes adaptive low-rank key compression practical without runtime decomposition. We first analyze KV cache compression through attention output error and reveal a clear asymmetry between keys and values: pre-RoPE keys benefit from preserving dominant low-rank structure while quantizing low-energy residuals, whereas values are better compressed by direct quantization. This analysis motivates an asymmetric design that applies low-rank projection with residual quantization to keys and direct low-bit quantization to values. However, obtaining accurate input-adaptive low-rank bases with online SVD is prohibitively expensive for serving. MOB-KV addresses this by replacing online SVD with a reusable dictionary of offline-constructed key bases, transforming adaptive compression into lightweight basis selection and projection. This design preserves the accuracy benefit of local low-rank subspaces while avoiding the latency bottleneck of runtime decomposition. Across reasoning and long-context benchmarks, MOB-KV substantially reduces KV cache memory, preserves near full-precision accuracy, and establishes a stronger memory–accuracy–latency Pareto frontier than existing quantization-only, low-rank-only, and online SVD-based baselines.

1 Introduction

LLM serving is rapidly shifting toward long-context generation, multi-turn dialogue, retrieval-augmented generation, and agentic workloads (Kwon et al., 2023; Zheng et al., 2024; Lewis et al., 2020; Yao et al., 2022). In these settings, inference efficiency is increasingly limited not only by model parameters, but also by the key-value (KV) cache. Since the KV cache grows linearly with sequence length, batch size, and the

number of layers, it becomes a major memory bottleneck for long-context LLM serving (Zhang et al., 2023; Xiao et al., 2024; Brandon et al., 2024).

Recent KV cache compression methods mainly rely on quantization or low-rank approximation. Quantization reduces memory by lowering the precision of cached keys and values, often using asymmetric schemes to handle distributional differences or outliers (Liu et al., 2024b; Kang et al., 2024). Low-rank methods instead exploit hidden-dimensional or cross-layer redundancy to represent KV states in compact subspaces (Chang et al., 2025b,a). These approaches are effective, but they leave two important questions unresolved: how should quantization and low-rank approximation be combined under a fixed memory budget, and can low-rank compression remain adaptive without incurring expensive online decomposition?

The second question is particularly important for practical serving. The strength of low-rank KV compression comes from adapting the subspace to the current key distribution. A natural solution is to compute an input-dependent basis using online SVD. However, this turns compression itself into a costly inference-time operation, introducing substantial latency during prefill and decoding. Thus, the central challenge is not simply to find a compact subspace, but to obtain the benefit of adaptive low-rank compression within affordable cost.

In this paper, we address this challenge by first analyzing KV cache compression through attention output error. We compare quantization, low-rank approximation, and their hybrid variants under a unified error–compression ratio landscape. This analysis reveals a clear asymmetry between pre-RoPE keys and values: keys benefit from preserving dominant low-rank structure while quantizing low-energy residual components, whereas values are better compressed by direct quantization. This suggests that effective KV cache compression

*Equal contribution.

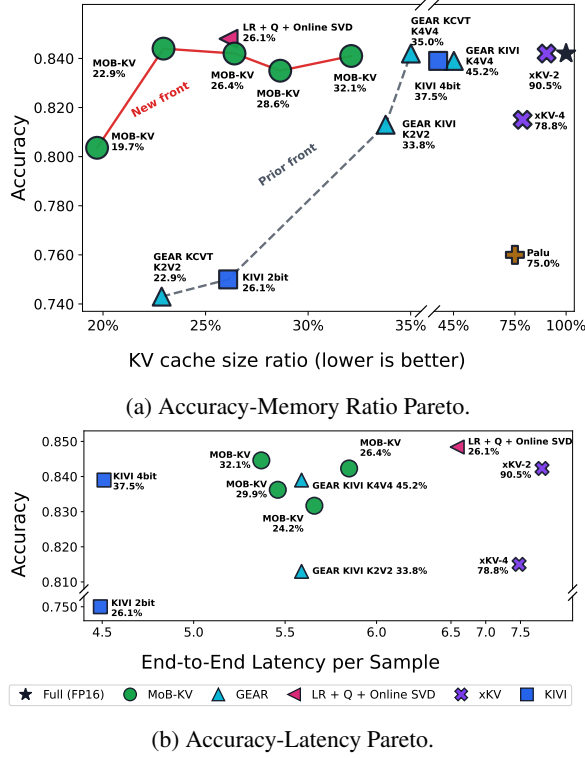


Figure 1: (a) Pareto frontier between accuracy and compression ratio. MOB-KV achieves higher robustness at lower memory ratios. (b) Pareto frontier between accuracy and end-to-end latency. MOB-KV preserves high accuracy with significantly lower latency than online-SVD-based methods. Results are measured on GSM8K-CoT. LR + Q + Online SVD denotes an online per-prompt SVD upper-bound variant of MOB-KV. Detailed MOB-KV configurations are provided in Appendix K.

should be asymmetric, and that low-rank compression should be applied where it matters most.

Motivated by this observation, we propose MOB-KV, a Mixture-of-Basis KV cache compression method that replaces online SVD with reusable offline bases. Instead of recomputing a new basis for every input, MOB-KV constructs a small dictionary of cluster-specific key bases from calibration data and selects an appropriate basis during inference. This reframes adaptive low-rank compression as a lightweight basis-selection problem rather than an expensive decomposition problem. As a result, MOB-KV preserves the main advantage of input-adaptive low-rank compression while avoiding the runtime bottleneck of online SVD.

This design leads to a stronger compression–accuracy–latency trade-off. MOB-KV preserves dominant key subspaces, compresses residual key information through quantization, and applies direct quantization to values, while relying only on offline bases and GPU-friendly projections during in-

ference. As shown in Figures 1a and 1b, MOB-KV establishes a new Pareto frontier over quantization-only, low-rank-only, and online SVD-based baselines, substantially reducing KV cache memory while preserving near full-precision performance with practical latency.

Our contributions are summarized as follows:

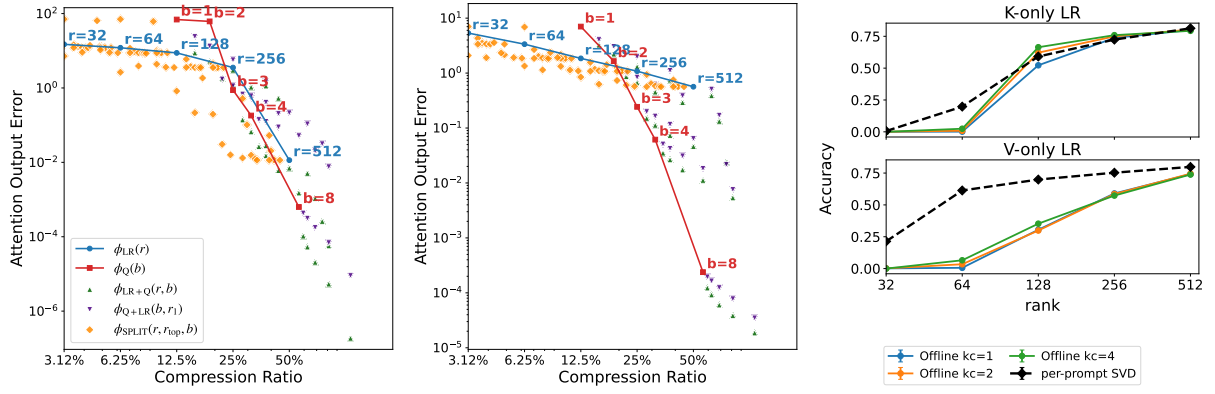
- We analyze quantization and low-rank approximation for KV cache through attention output error, revealing how their errors should be allocated under aggressive compression.
- We identify an asymmetric compression principle: pre-RoPE keys benefit from low-rank projection with quantized residuals, whereas values are better served by direct quantization.
- We propose MOB-KV, an online-SVD-free adaptive KV cache compression method that replaces runtime decomposition with offline mixture-of-basis selection.
- MOB-KV achieves a superior compression–accuracy–latency trade-off across reasoning and long-context benchmarks.

2 Related Work

KV Cache Quantization. KV cache quantization stores key/value states in low-bit formats while retaining all tokens. Prior methods address key–value distributional differences, activation outliers, and quantization residuals through asymmetric, outlier-aware, mixed-precision, or progressive quantization (Liu et al., 2024b; Kang et al., 2024; Hooper et al., 2024; Yang et al., 2024; Su et al., 2025; Liu et al., 2025). However, aggressive low-bit quantization can still degrade attention quality.

KV Cache Low-Rank Approximation. Low-rank methods exploit hidden-dimensional or cross-layer redundancy to compactly represent KV caches or projections (Chang et al., 2025b,a; Saxena et al., 2024; Liu et al., 2024a; Zhang et al., 2024). While effective, they may discard residual information and introduce reconstruction overhead.

Existing KV cache compression methods typically use quantization or low-rank approximation, but their separate and joint effects on attention output error remain underexplored. Moreover, hybrid methods often rely on online SVD, limiting practical serving efficiency. We address this by analyzing quantization–low-rank interactions and deriving an



(a) Err-Compression Ratio Trend(Key). (b) Err-Compression Ratio Trend(Value). (c) Offline vs Online Low-Rank Approx.

Figure 2: (a) Error-compression landscape for pre-RoPE keys. (b) Error-compression landscape for values. (c) GSM8K-CoT accuracy of offline and online low-rank approximation methods under varying numbers of clusters.

online-SVD-free compression design. Full related work is provided in Appendix L.

3 Error-Compression-Ratio Landscape of the KV Cache

Existing KV cache compression methods mainly use quantization, low-rank projection, or their combination, but it remains unclear which strategy best minimizes attention output error under different compression ratios. We compare these approaches under a unified metric and show that pre-RoPE keys and values exhibit distinct compression trends, motivating the asymmetric design of MOB-KV.

Setup. We measure the attention output error E induced when the pre-RoPE key/value cache is replaced with its compressed counterpart:

$$E = \left\| \mathcal{S}\left(\frac{qK^\top}{\sqrt{d_h}}\right)V - \mathcal{S}\left(\frac{q\hat{K}^\top}{\sqrt{d_h}}\right)\hat{V} \right\|_2^2 \quad (1)$$

Here, q denotes the query at the first decoding step, K and V denote the original key and value matrices, respectively, $\hat{K}$ and $\hat{V}$ denote the compressed key and value caches, d_h is the per-head dimension, and $\mathcal{S}$ denotes the softmax function. We define the compression ratio as the storage ratio between the original KV cache $C := (K, V)$ and the compressed KV cache $\hat{C} := (\hat{K}, \hat{V})$, i.e., $\rho := |\hat{C}|/|C|$. Keys and values are compressed independently so that the measured error can be attributed solely to the corresponding key or value component. Under this metric, we compare the five compression operators listed in Table 1 using Llama-3.1-8B-Instruct on samples from the GSM8K-CoT task.¹

¹Configurations: rank $r \in \{32, 64, 128, 256, 512\}$, bits

Table 1: Evaluated Compression operators.

Operator	Definition
$\phi_{LR}(r)$	rank- r projection
$\phi_Q(b)$	b -bit quantization
$\phi_{Q+LR}(b, r_1)$	quant. + rank- r_1 residual
$\phi_{LR+Q}(r, b)$	rank- r proj. + quant. residual
$\phi_{SPLIT}(r, r_{fp}, b)$ (ours)	rank- r proj. + $(r - r_{fp})$ rank b -bit quant.

3.1 Key: Low-Rank Top-Bottom Splitting Establishes a New Pareto Frontier

Figure 2a compares the E - ρ trade-off of the five compression operators in Table 1 when applied to pre-RoPE keys. The operators separate the effects of low-rank projection, quantization, and their combination. Low-rank projection exploits the strong spectral structure of pre-RoPE keys, but loses residual information, whereas low-bit quantization preserves all dimensions but can perturb attention-critical directions. Hybrid residual-compensation operators, ϕ_{Q+LR} (purple) and ϕ_{LR+Q} (green), reduce this error by modeling the residual left by one compression mechanism with the other (Kang et al., 2024), but the additional residual storage shifts them toward larger ρ . The split operator ϕ_{SPLIT} (orange) (Yankun et al., 2025) avoids this trade-off by applying quantization selectively in the spectral domain. After rank- r projection, it keeps the leading r_{fp} rank coefficients in full precision and quantizes only the low-energy bottom $r - r_{fp}$ rank coefficients. Because pre-RoPE keys are strongly low-rank (Saxena et al., 2024), even low-bit bottom rank quantization introduces little

$b \in \{1, 2, 3, 4, 8\}$, group size $g = 32$. Low-rank approximation uses `torch.svd_lowrank`. We use per-channel quantization for Key and per-token quantization for Value.

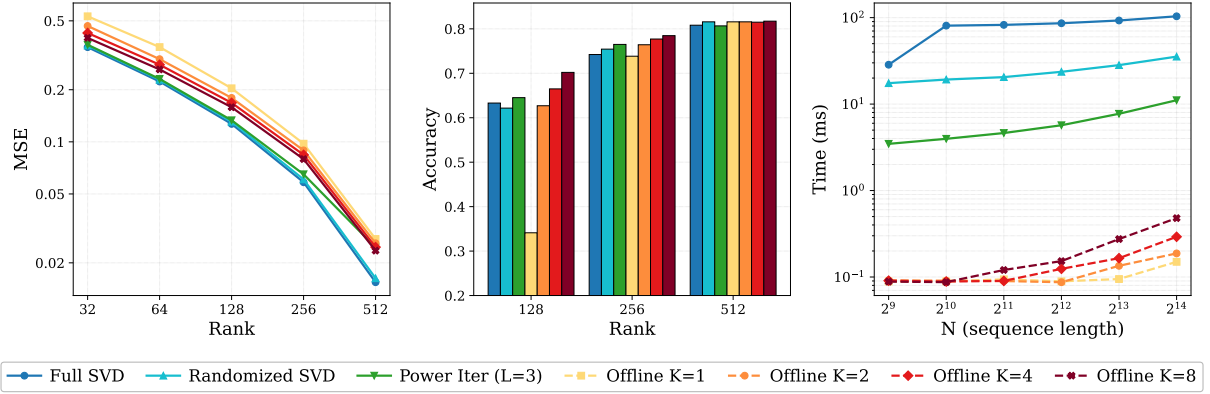


Figure 3: Comparison of online SVD variants and offline clustering-based basis selection. (Left) Reconstruction Error(MSE), (Middle) Accuracy on GSM8K(Prefill-only compressed), (Right) Compression Latency(ms).

attention output error while substantially reducing memory. As shown by the orange points, this strategy forms a new Pareto frontier in the low- ρ regime, motivating the MOB-KV design in Section 5.

3.2 Value: Quantization Efficiently Reduces Error

Figure 2b presents the corresponding analysis for values, which exhibits a qualitatively different trend from pre-RoPE keys. The low-rank-based operators, ϕ_{LR} (blue) and ϕ_{SPLIT} (orange), achieve lower E than quantization in the aggressive compression regime, i.e., at small ρ . However, even as ρ increases, their error does not decrease below a certain floor. This is because values do not exhibit a strong low-rank structure, as also observed in (Saxena et al., 2024). The residual-compensation hybrid operators, ϕ_{Q+LR} (purple) and ϕ_{LR+Q} (green), can reduce E through the residual term, but the additional stored representation increases the storage cost, eventually placing them above the quantization-only curve ϕ_Q (red). In other words, for values, increasing the bit-width of quantization alone effectively reduces E , making quantization the method that defines the Pareto frontier in the practically relevant low-error regime.

4 Offline Clustering-based Subspace Construction

As shown in Section 3, pre-RoPE keys have strong low-rank structure, and low-rank projection with bottom-rank quantization provides a Pareto-optimal compression strategy. However, practical deployment requires efficient construction of projection bases. Existing low-rank KV cache methods such as xKV often rely on online SVD, which adapts to the current token distribution but intro-

duces substantial compression latency during inference.

Offline Basis Construction. Motivated by this limitation, we examine whether representative bases can be constructed offline from large-scale pre-RoPE key activations. Our intuition is that tokens with similar activation patterns share similar dominant spectral structures, allowing them to be approximated by a common low-rank basis. This can retain the adaptivity of online low-rank approximation while avoiding online SVD overhead. The detailed construction procedure is described in Section 5.2.

Reconstruction Error Analysis. To validate this hypothesis, we compare exact SVD (torch.svd), randomized SVD (torch.svd_lowrank), power-iteration-based SVD, and offline clustering-based basis selection under the same pre-RoPE key compression setting. Figure 3 (left) shows that the offline clustering-based basis incurs slightly higher MSE than online SVD variants but follows a similar trajectory, indicating that offline basis dictionaries capture much of the dominant spectral structure of pre-RoPE keys. Figure 3 (middle) further shows nearly comparable downstream accuracy, while increasing the number of clusters consistently reduces MSE and narrows the gap to online SVD.

Latency Analysis. Finally, from the latency perspective, offline clustering-based basis selection is significantly more efficient than online SVD-based methods. While online SVD must perform low-rank decomposition during inference, the offline approach only needs to select an appropriate subspace from the precomputed bases and apply projection. As shown in Figure 3(right), this difference substantially reduces compression latency, demonstrating

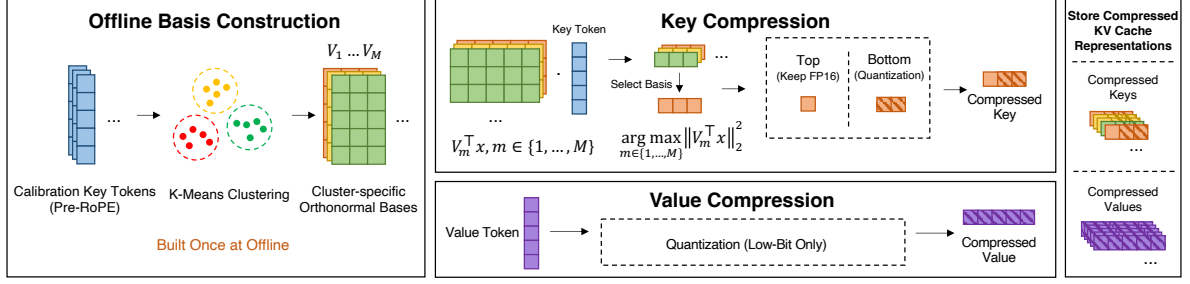


Figure 4: Overview of MOB-KV key and value compression. MOB-KV compresses keys by projecting each key token onto offline-constructed orthonormal bases, followed by bottom-rank quantization. In contrast, values are directly quantized without low-rank projection.

that offline clustering-based subspace construction can serve as a practical alternative to online SVD.

Offline Basis Transfers for Key but Not Value.

Figure 2c compares offline cluster bases with $M \in \{1, 2, 4\}$ against per-prompt online SVD over ranks $r \in \{32, \dots, 512\}$ for pre-RoPE keys and values. For pre-RoPE keys, offline bases closely match online SVD, with the gap decreasing as M increases, indicating that calibration-derived cluster bases capture the dominant low-rank structure across inputs. In contrast, for values, offline bases remain far below online SVD even with more clusters, suggesting that their dominant directions are highly input-dependent. Thus, values are poorly suited to offline low-rank compression, motivating our use of direct quantization instead.

Key-Value Asymmetry Design. Overall, these results reveal a clear asymmetry between pre-RoPE keys and values: keys are well suited to offline cluster-basis low-rank projection with bottom-rank quantization, whereas values are better compressed by direct per-token quantization. This motivates the asymmetric design of MOB-KV in Section 5.

5 Method: MOB-KV

5.1 Overview

Based on the observations in Sections 3 and 4, we design **MOB-KV** (Mixture-of-Basis KV cache compression), illustrated in Figure 4. MOB-KV approximates adaptive online low-rank compression using reusable offline local subspaces.

MOB-KV consists of three stages: (i) offline cluster-specific basis construction, (ii) online per-token basis routing, and (iii) FP16-leading-quantized-tail rank split quantization. We denote the total key coefficient rank by r , the leading rank retained in FP16 by r_{fp} , and the quantized tail rank by $r_q = r - r_{\text{fp}}$. Each incoming pre-RoPE key to-

ken is routed to the basis that maximizes preserved projection energy, compressed into a low-rank representation, and further compressed through the FP16-leading-quantized-tail split. For the value cache, following observations in Section 3 and Section 4, MOB-KV applies quantization-only compression. Compared with online SVD-based methods, MOB-KV replaces repeated decomposition during inference with routing over a reusable basis pool and hardware-accelerated GEMM operations, thereby substantially improving end-to-end latency.

5.2 Offline Cluster-Specific Basis Construction

MOB-KV constructs an independent pre-RoPE key basis pool for each transformer layer. We partition the calibration matrix $X_{\text{calib}}^{(\ell)} \in \mathbb{R}^{N_{\text{calib}} \times d_K}$ of layer ℓ into M clusters using K-means:

$$\mathcal{C}^{(\ell)} = \{C_1^{(\ell)}, \dots, C_M^{(\ell)}\} = \text{KMeans}(X_{\text{calib}}^{(\ell)}; M), \quad (2)$$

Here, $C_m^{(\ell)}$ denotes the index set of calibration tokens assigned to cluster m at layer ℓ .

For each cluster, we compute a cluster-specific rank- r orthonormal basis that maximizes the preserved projection energy:

$$V_m^{(\ell)} = \arg \max_{V \in \mathbb{R}^{d_K \times r}, V^\top V = I_r} \|X_{C_m^{(\ell)}}^{(\ell)} V\|_F^2. \quad (3)$$

The solution to Eq. (3) is given by the top- r right singular vectors of $X_{C_m^{(\ell)}}^{(\ell)}$. The basis pool $\mathcal{V}^{(\ell)} = \{V_1^{(\ell)}, \dots, V_M^{(\ell)}\}$ at each layer jointly covers all key heads (Appendix C.1).

5.3 Online Per-Token Basis Routing

During inference, each pre-RoPE key token $x^{(\ell)} \in \mathbb{R}^{d_K}$ at layer ℓ is compressed using one basis selected from the layer-specific pool $\mathcal{V}^{(\ell)}$.

MOB-KV routes each token to the basis that maximizes the preserved projection energy, which is equivalent to minimizing the projection reconstruction error as shown in Appendix B:

$$m_\ell^*(x^{(\ell)}) = \arg \max_{m \in \{1, \dots, M\}} \|(V_m^{(\ell)})^\top x^{(\ell)}\|_2^2. \quad (4)$$

After routing, the coefficient is computed as $c^{(\ell)}(x) = (V_{m_\ell^*(x^{(\ell)})}^{(\ell)})^\top x^{(\ell)} \in \mathbb{R}^r$. MOB-KV stores the routing index $m_\ell^*(x^{(\ell)})$ and coefficient $c^{(\ell)}(x)$, and reconstructs the key as $\hat{x}^{(\ell)} = V_{m_\ell^*(x^{(\ell)})}^{(\ell)} c^{(\ell)}(x)$. For brevity, we omit the superscript (ℓ) when referring to a fixed layer hereafter.

Unlike online SVD-based methods, MOB-KV does not recompute adaptive bases during prefill or decoding. The additional inference overhead is limited to evaluating projection energies over a small set of reusable basis candidates.

5.4 FP16-Leading-Quantized-Tail Rank Split Quantization

The compressed coefficient $c(x)$ lies on the spectral axes of the selected basis. Since basis vectors are ordered by singular values, the leading coefficient dimensions correspond to dominant spectral components, whereas the trailing dimensions mainly contain low-energy residual information.

Following the analysis in Section 3, we split the coefficient vector into an FP16 leading component and a quantized tail component:

$$c(x) = [c_{\text{fp}}(x); c_{\text{q}}(x)], \quad (5)$$

where $c_{\text{fp}}(x) \in \mathbb{R}^{r_{\text{fp}}}$ and $c_{\text{q}}(x) \in \mathbb{R}^{r_{\text{q}}}$.

The leading component is retained in FP16, while the tail component is compressed using low-bit quantization:

$$\tilde{c}(x) = [c_{\text{fp}}(x); Q_{b_k}(c_{\text{q}}(x))], \quad (6)$$

Here, $Q_{b_k}(\cdot)$ denotes the b_k -bit quantizer applied to the tail coefficients. Detailed settings for asymmetric min-max quantization, token-dimension grouping, metadata precision, bit-packed storage, and the recent full-precision window are provided in Appendix C.2.

The reconstructed key vector is computed as $\hat{x} = V_{m^*(x)} \tilde{c}(x)$. This design preserves the dominant spectral information while aggressively compressing only the low-energy residual components.

Keys are stored in pre-RoPE space, and RoPE is applied after reconstruction. The complete pre-fill and decoding procedure is provided in Appendix C.3.

5.5 Value Quantization

For the value cache, we directly group-quantize each value token vector into b_v -bit integers:

$$\tilde{v} = Q_{b_v}(v), \quad (7)$$

Here, $Q_{b_v}(\cdot)$ denotes the b_v -bit quantizer applied directly to the original value tensor. Detailed settings for asymmetric min-max quantization, channel-dimension grouping, scale/zero-point metadata, and bit-packed storage are provided in Appendix C.2.

6 Experiments

6.1 Experimental Setup

Models. We evaluate MOB-KV-based KV cache compression on four models that use Grouped-Query Attention (GQA) (Ainslie et al., 2023): LLaMA3.1-8B-Instruct (Grattafiori et al., 2024), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), LLaMA3.2-3B-Instruct (Grattafiori et al., 2024), and Qwen-2.5-7B-Instruct-1M (Yang et al., 2025).

Datasets. We evaluate reasoning performance on GSM8K (Cobbe et al., 2021), BBH (Suzgun et al., 2022), and MMLU (Hendrycks et al., 2021) using Chain-of-Thought prompting (Wei et al., 2022), which induces longer generations. For long-context evaluation, we use LongBench (Bai et al., 2024), RULER (Hsieh et al., 2024), and AgentLongBench (Fang et al., 2026). LongBench covers 16 English tasks including multi-document question answering, summarization, and code completion; RULER evaluates synthetic long-context retrieval and reasoning as context length increases; and AgentLongBench measures long-context agent-style workloads. We report both task-level and average scores. All main experiments are conducted on NVIDIA A100-80GB GPUs, while AgentLongBench is evaluated separately on an NVIDIA RTX 3090 to reflect consumer-grade local-first agent deployment scenarios such as OpenClaw (OpenClaw Contributors, 2026).

Baselines. We compare MOB-KV with five KV cache compression baselines: **KIVI** (Liu et al., 2024b), a quantization-only method using asymmetric key/value quantization; **GEAR-KCVT** (Kang et al., 2024), which combines sparse

Table 2: Main results across three models on four benchmarks. “KV” (shaded) denotes KV cache size relative to the full-precision baseline. The KV size ratio includes quantization metadata, routing indices, and the recent full-precision window; therefore, it varies with the average sequence length of each benchmark.

Model	Method	GSM8K-CoT		BBH-CoT		MMLU-CoT		LongBench	
		KV	Acc.	KV	Acc.	KV	Acc.	KV	Score
LLaMA-3.1-8B-Instruct	Full (FP16)	100%	0.842	100%	0.693	100%	0.724	100%	0.431
	Palu	75.00%	0.760	75.00%	0.628	75.00%	0.665	75.00%	0.418
	KIVI (K2V2)	26.10%	0.750	24.69%	0.471	42.17%	0.627	19.26%	0.340
	GEAR-KCVT (K2V2)	33.78%	0.813	32.26%	0.611	51.07%	0.712	26.42%	0.429
	xKV	78.76%	0.815	66.08%	0.598	56.03%	0.559	16.37%	0.435
	MOB-KV ($r=512, r_{fp}=128, r_q=384, K2V4$)	32.10%	0.841	30.82%	0.682	46.68%	0.723	25.86%	0.431
	MOB-KV ($r=512, r_{fp}=128, r_q=384, K2V2$)	26.41%	0.842	25.02%	0.669	42.20%	0.722	19.65%	0.427
	MOB-KV ($r=256, r_{fp}=32, r_q=224, K2V2$)	20.79%	0.827	19.31%	0.603	37.72%	0.715	13.53%	0.427
Mistral-7B-Instruct-v0.3	Full (FP16)	100%	0.523	100%	0.531	100%	0.596	100%	0.385
	Palu	75.00%	0.490	75.00%	0.487	75.00%	0.586	75.00%	0.332
	KIVI (K2V2)	26.10%	0.466	24.69%	0.442	42.17%	0.493	19.26%	0.349
	GEAR-KCVT (K2V2)	33.78%	0.503	32.26%	0.481	51.07%	0.582	26.42%	0.381
	xKV	78.76%	0.478	66.08%	0.518	56.03%	0.553	16.37%	0.379
	MOB-KV ($r=512, r_{fp}=128, r_q=384, K2V4$)	32.10%	0.529	30.82%	0.514	46.68%	0.595	25.86%	0.385
	MOB-KV ($r=512, r_{fp}=128, r_q=384, K2V2$)	26.41%	0.513	25.02%	0.503	42.20%	0.594	19.65%	0.384
	MOB-KV ($r=256, r_{fp}=32, r_q=224, K2V2$)	20.79%	0.506	19.31%	0.473	37.72%	0.576	13.53%	0.384
LLaMA-3.2-3B-Instruct	Full (FP16)	100%	0.766	100%	0.548	100%	0.646	100%	0.331
	Palu	75.00%	0.694	75.00%	0.519	75.00%	0.593	75.00%	0.295
	KIVI (K2V2)	26.10%	0.637	24.69%	0.247	42.17%	0.549	19.26%	0.226
	GEAR-KCVT (K2V2)	33.78%	0.753	32.26%	0.486	51.07%	0.641	26.42%	0.331
	xKV	78.76%	0.707	66.08%	0.502	56.03%	0.541	16.37%	0.327
	MOB-KV ($r=512, r_{fp}=128, r_q=384, K2V4$)	32.10%	0.766	30.82%	0.525	46.68%	0.640	25.86%	0.331
	MOB-KV ($r=512, r_{fp}=128, r_q=384, K2V2$)	26.41%	0.758	25.02%	0.502	42.20%	0.636	19.65%	0.329
	MOB-KV ($r=256, r_{fp}=32, r_q=224, K2V2$)	20.79%	0.754	19.31%	0.440	37.72%	0.633	13.53%	0.317

outlier separation with low-rank residual approximation; **xKV** (Chang et al., 2025a), which performs online SVD-based cross-layer low-rank compression during prefill; and **Palu** (Chang et al., 2025b), which applies Fisher Information-based grouped low-rank decomposition to W_K and W_V . We also compare with **TurboQuant** (Zandieh et al., 2025), a quantization-only method based on PolarQuant and 1-bit QJL residual correction, but report it separately in Appendix D because no official implementation is available.

Detailed configurations of MOB-KV and all baseline methods are provided in Appendix C.

Offline Calibration Dataset. We construct the calibration set by collecting 10K tokens from each of 13 diverse tasks. Further details are provided in Appendix E.

6.2 Results of Reasoning Tasks

Table 2 reports performance on three reasoning benchmarks: GSM8K-CoT, BBH-CoT, and MMLU-CoT. Overall, MOB-KV substantially reduces the KV cache size while preserving reasoning performance close to the full-precision baseline.

Across LLaMA-3.1-8B-Instruct, Mistral-7B-Instruct-v0.3, and LLaMA-3.2-3B-Instruct, MOB-KV consistently preserves CoT reasoning performance under substantial KV cache compression. On GSM8K-CoT and MMLU-CoT, MOB-KV achieves scores nearly identical to the full-precision baseline. Although BBH-CoT exhibits a modest degradation in some cases, the overall performance drop remains small and gradual across model scales and architectures. In contrast, prior methods such as KIVI, Palu, and xKV suffer substantial accuracy degradation at comparable compression ratios. These results indicate that MOB-KV is not specific to a single model family, but provides robust and generalizable KV cache compression while largely retaining reasoning capability.

Taken together, the reasoning benchmark results demonstrate that MOB-KV provides a favorable trade-off between compression ratio and accuracy. While quantization-only and low-rank-only baselines suffer from substantial reasoning degradation under aggressive compression, MOB-KV more stably preserves attention behavior by jointly leveraging low-rank projection and bottom rank coefficient quantization. These results support the importance

Table 3: Calibration robustness under distribution shift and leave-one-dataset-out calibration. For this experiment, BBH is added to the default calibration mixture. Distribution-shift settings replace this mixture with general-domain text, multilingual data, or code data; leave-one-dataset-out settings remove calibration data corresponding to GSM8K, BBH, MMLU, or LongBench.

Calibration setting	GSM8K-CoT	BBH-CoT	MMLU-CoT	LongBench
Default mixture	0.841	0.683	0.723	0.431
WikiText only	0.835	0.673	0.723	0.431
Multilingual only	0.837	0.673	0.723	0.431
Code generation only	0.831	0.677	0.725	0.430
Without GSM8K	0.841	—	—	—
Without BBH	—	0.682	—	—
Without MMLU	—	—	0.723	—
Without LongBench-related datasets	—	—	—	0.430

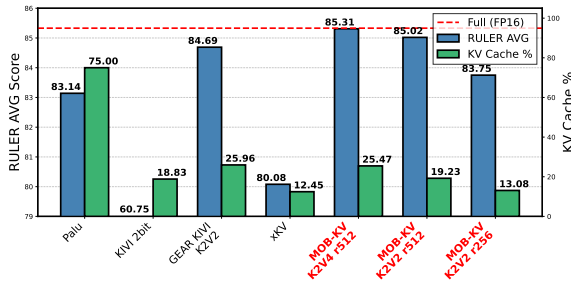


Figure 5: Performance on the RULER benchmark and the retained KV cache size for each method.

of jointly controlling low-rank approximation error and quantization error when compressing KV caches for reasoning-intensive workloads.

6.3 Results of Long-context Tasks

The LongBench column in Table 2 reports the average over 16 tasks. Across LLaMA-3.1-8B-Instruct, Mistral-7B-Instruct-v0.3, and LLaMA-3.2-3B-Instruct, MOB-KV maintains full-precision-level long-context performance while substantially reducing KV cache memory. It matches the full-precision scores on LLaMA-3.1-8B-Instruct and Mistral-7B-Instruct-v0.3 with 0.431 and 0.385, respectively, using only 25.86% of the original KV cache for LLaMA-3.1, and achieves 0.331 on LLaMA-3.2-3B-Instruct. This shows a stable compression–accuracy trade-off across long-context tasks such as multi-document question answering, summarization, and code completion. Compared with prior methods, KIVI and Palu show noticeable degradation on some models, while GEAR-KCVT and xKV are competitive only in certain settings; MOB-KV remains consistently close to full precision with a smaller retained KV cache. Figure 5 further confirms this trend on RULER: MOB-KV achieves 85.31 at 32.10% KV cache and 85.02 at 26.41%, nearly matching the full-precision baseline of 85.33, whereas Palu, KIVI (K2V2), and

GEAR-KCVT (K2V2) obtain 83.14, 60.75, and 84.69, respectively. All RULER methods use the same configurations as in Table 2.

6.4 Robustness to Calibration Distribution

The default calibration set of MOB-KV includes GSM8K, MMLU, and several datasets related to LongBench. For the experiments in Table 3 only, we additionally include BBH in this mixture. We then examine whether the reported performance depends on calibration data drawn from distributions similar to the evaluation benchmarks. All experiments use LLaMA-3.1-8B-Instruct with $M=4$ clusters, total key coefficient rank $r=512$, FP16 leading rank $r_{fp}=128$, quantized tail rank $r_q=384$, K2V4, quantization group size 32, and a recent full-precision window of 64 tokens.

Table 3 first replaces the original calibration mixture entirely with three substantially different distributions: general-domain text from WikiText (Merity et al., 2016), multilingual data from XNLI (Conneau et al., 2018), XCOPA (Ponti et al., 2020), and XQuAD (Artetxe et al., 2020), and code data from HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), and LCC (Guo et al., 2023). Despite these substantial distribution shifts, MMLU-CoT and LongBench performance remains nearly unchanged, while the reductions on GSM8K-CoT and BBH-CoT are limited to at most 0.010. LongBench retains a score of 0.430 even when only code data are used for calibration.

The same table also reports leave-one-dataset-out results for GSM8K, BBH, MMLU, and LongBench. Excluding GSM8K or MMLU leaves performance on the corresponding benchmark unchanged, while excluding BBH or LongBench-related datasets changes the respective score by only 0.001. Thus, the reported gains do not result from calibration on the same benchmark or

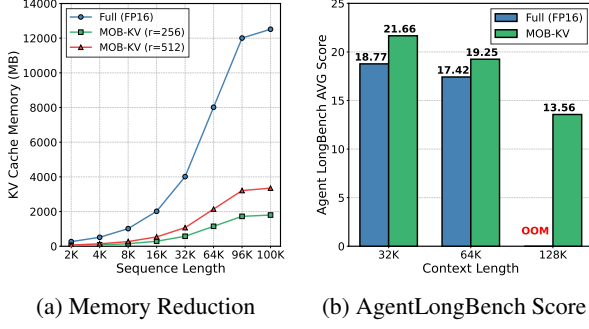


Figure 6: (a) Memory reduction across prefill sequence lengths on LLaMA-3.1-8B-Instruct. (b) Average score on AgentLongBench using Qwen-2.5-7B-Instruct-1M.

a closely matched distribution. To examine this robustness more directly, we also measure the similarity of the dominant pre-RoPE key subspaces obtained from different calibration distributions. At rank 512, the mean subspace overlap across 32 layers remains above 0.87 for every alternative distribution; the definition and layer-wise results are provided in Appendix F.

6.5 Inference Efficiency

Figure 6a shows that MOB-KV greatly reduces KV cache memory on LLaMA-3.1-8B-Instruct, with larger absolute savings at longer sequence lengths. Both $r=256$, $r_{fp}=64$, $r_q=192$ with K2V2 and $r=512$, $r_{fp}=128$, $r_q=384$ with K2V4 yield substantial reductions, demonstrating practical benefits for long-context inference. To assess feasibility under limited VRAM, we evaluate Qwen-2.5-7B-Instruct-1M on AgentLongBench at 32K, 64K, and 128K using a single 24GB RTX 3090, reflecting consumer-grade local deployment scenarios such as OpenClaw. We use $r=128$, $r_{fp}=64$, $r_q=64$, $M=4$, and 2-bit quantization for both key bottom rank coefficients and value caches, over eight Knowledge-Intensive, Concise-Response tasks. As shown in Figure 6b, MOB-KV matches full-precision performance at 32K and 64K; at 128K, the full-precision baseline OOMs, while MOB-KV runs successfully. Thus, MOB-KV preserves long-context agent performance while extending feasible context length on memory-constrained GPUs.

Figure 7 reports the runtime overhead of MOB-KV under two configurations, ($r=256$, $r_{fp}=64$, $r_q=192$, K2V2, $M=4$) and ($r=512$, $r_{fp}=128$, $r_q=384$, K2V4, $M=4$), compared with xKV and KIVI (K2V2). By avoiding online SVD and reconstructing keys with matrix multiplications over offline bases, MOB-KV keeps TTFT close to the no-compression baseline,

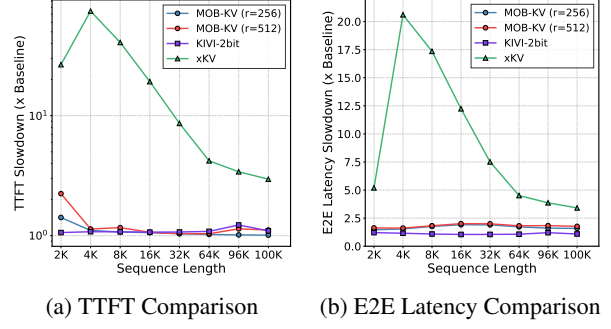


Figure 7: (a) Time-to-first-token (TTFT) across different sequence lengths on LLaMA-3.1-8B-Instruct. (b) End-to-end latency including prefill and 128-token gen.

around $1.02\text{--}1.05\times$ across sequence lengths. Although basis-based key recovery adds decoding cost, the overhead remains bounded and becomes less dominant at longer contexts, where prefill and memory-related costs increase. In contrast, xKV incurs impractically high runtime overhead from online SVD. Overall, MOB-KV achieves a practical memory-latency trade-off, enabling long-context inference with substantial KV cache reduction and modest decoding overhead.

7 Conclusion

We proposed MOB-KV, an asymmetric KV cache compression method that applies offline mixture-of-basis low-rank approximation with quantization to pre-RoPE keys, and direct low-bit quantization to values. Our error-compression analysis shows that keys can reuse calibration-derived cluster bases with token-wise routing and tail quantization, avoiding online SVD, whereas values are better compressed by quantization due to weak offline low-rank transferability. Experiments show that MOB-KV substantially reduces KV cache memory while preserving near full-precision performance with practical inference overhead, making it effective for long-context LLM serving.

Limitations

MOB-KV requires an offline calibration stage before deployment to construct its layer-wise basis pools. This introduces an additional setup cost, as calibration examples must be collected and processed from a sufficiently diverse set of datasets before inference begins. Our calibration-robustness results partially mitigate this limitation: as shown in Appendix F, even calibration mixtures containing only a few datasets from a limited domain can cover a large portion of the dominant pre-RoPE key

space and retain stable downstream performance when a sufficient number of calibration tokens is collected. This suggests that exhaustive coverage of all potential deployment tasks is not necessary. Nevertheless, calibration remains a prerequisite, and performance may degrade when the collected samples do not adequately represent the key distribution of a specialized deployment workload. Calibration-free basis construction or lightweight online adaptation therefore remains an important direction for future work.

Acknowledgements

This work was supported by IITP and NRF grant funded by the Korea government(MSIT) (RS-2025-02218259, RS-2024-00396013, RS-2019-II191906)

References

- Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901.
- Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4623–4637.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.
- Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. Longbench: A bilingual, multi-task benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.
- William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. 2024. Reducing transformer key-value cache size with cross-layer attention. *Advances in Neural Information Processing Systems*, 37:86927–86957.
- Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and 1 others. 2024. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*.
- Chi-Chih Chang, Chien-Yu Lin, Yash Akhauri, Wei-Cheng Lin, Kai-Chiang Wu, Luis Ceze, and Mohamed S Abdelfattah. 2025a. xkv: Cross-layer svd for kv-cache compression. *arXiv preprint arXiv:2503.18893*.
- Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed Abdelfattah, and Kai-Chiang Wu. 2025b. Palu: Kv-cache compression with low-rank projection. In *International Conference on Learning Representations*, volume 2025, pages 50222–50249.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2475–2485.
- Shicheng Fang, Yuxin Wang, Xiaoran Liu, Jiahao Lu, Chuanyuan Tan, Xinchu Chen, Yining Zheng, Xuanjing Huang, and Xipeng Qiu. 2026. Agentlongbench: A controllable long benchmark for long-contexts agents via environment rollouts. *arXiv preprint arXiv:2601.20730*.
- Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. 2026. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *Advances in Neural Information Processing Systems*, 38:113152–113188.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Daya Guo, Canwen Xu, Nan Duan, Jian Yin, and Julian McAuley. 2023. Longcoder: A long-range pre-trained language model for code completion. In *International Conference on Machine Learning*, pages 12098–12107. PMLR.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

- Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt Keutzer, and Amir Gholami. 2024. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303.
- Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekeshe, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What’s the real context size of your long-context language models? In *First Conference on Language Modeling*.
- Albert Q. Jiang and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.
- Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. 2024. Gear: An efficient kv cache compression recipe for near-lossless generative inference of llm. *arXiv preprint arXiv:2403.05527*.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.
- Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970.
- Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. 2024a. Minicache: Kv cache compression in depth dimension for large language models. *Advances in Neural Information Processing Systems*, 37:139997–140031.
- Tengxuan Liu, Shiyao Li, Jiayi Yang, Tianchen Zhao, Feng Zhou, Xiaohui Song, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. 2025. Pm-kvq: Progressive mixed-precision kv cache quantization for long-cot llms. *arXiv preprint arXiv:2505.18610*.
- Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2023. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364.
- Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024b. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.
- OpenClaw Contributors. 2026. [OpenClaw: Personal ai assistant](#). GitHub repository. Accessed: 2026-05-25.
- Edoardo Maria Ponti, Goran Glavaš, Olga Majewska, Qianchu Liu, Ivan Vulić, and Anna Korhonen. 2020. Xcopa: A multilingual dataset for causal common-sense reasoning. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 2362–2376.
- Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. 2024. Eigen attention: Attention in low-rank space for kv cache compression. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15332–15344.
- Zunhai Su, Zhe Chen, Wang Shen, Hanyu Wei, Ling Li, Huangqi Yu, and Kehong Yuan. 2025. Rotatekv: Accurate and robust 2-bit kv cache quantization for llms via outlier-aware adaptive rotations. *arXiv preprint arXiv:2501.16383*.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.
- Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pages 21875–21895.
- An Yang and 1 others. 2025. Qwen2.5-1m technical report. *arXiv preprint arXiv:2501.15383*.
- June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. 2024. No token left behind: Reliable kv cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*.
- Hong Yankun, Li Xing, Zhen Hui-Ling, Yu Xianzhi, Liu Wulong, and Yuan Mingxuan. 2025. Svdq: 1.25-bit and 410x key cache compression for llm attention. *arXiv preprint arXiv:2502.15304*.

- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.
- Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. 2025. Turboquant: Online vector quantization with near-optimal distortion rate. *arXiv preprint arXiv:2504.19874*.
- Rongzhi Zhang, Kuang Wang, Liyuan Liu, Shuohang Wang, Hao Cheng, Chao Zhang, and Yelong Shen. 2024. Lorc: Low-rank compression for llms kv cache with a progressive compression strategy. *arXiv preprint arXiv:2410.03111*.
- Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuan-dong Tian, Christopher Ré, Clark Barrett, and 1 others. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.
- Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, and 1 others. 2024. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583.

Appendix

A Effect of Clustering Granularity at Decoding Stage

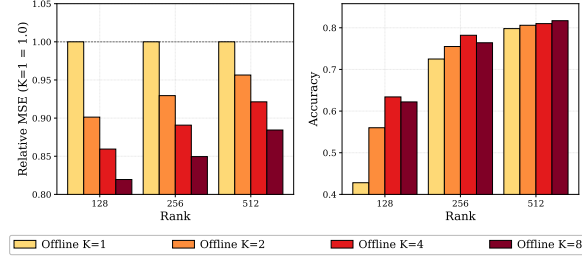


Figure 8: Effect of clustering granularity on reconstruction error and downstream accuracy. (Left) Reconstruction Error(MSE), (Right) GSM8K-CoT Accuracy(Prefill and Decode Compressed.)

Figure 8 demonstrates the effect of using cluster-specific bases compared with using a single global basis. Here, a single global basis does not refer to one basis shared across all model layers; it denotes one basis per layer constructed without clustering. As the number of clusters increases, the reconstruction error decreases effectively, and downstream accuracy tends to improve over the single-basis baseline. This suggests that the pre-RoPE key distribution is better approximated as a mixture of multiple local subspaces rather than as a single global subspace. In other words, increasing the clustering granularity allows each cluster-specific basis to capture more fine-grained local spectral structure, thereby reducing the approximation gap relative to online SVD.

B Derivation of Projection-Energy-Based Routing

This routing rule directly corresponds to minimizing the projection reconstruction error. Since each basis is orthonormal, we have In this derivation, we consider a fixed layer and omit the layer superscript (ℓ).

$$\|x - V_m V_m^\top x\|_2^2 = \|x\|_2^2 - \|V_m^\top x\|_2^2. \quad (8)$$

Therefore, maximizing the projection energy is equivalent to minimizing the reconstruction error.

C Baseline and MOB-KV Configurations

Table 4 summarizes the configurations used in our experiments.

For KIVI, GEAR-KCVT, and MOB-KV, we use a quantization group size of $g = 32$ and keep a

recent full-precision window of $w_{fp} = 64$ tokens; compression is applied once this window is filled. KIVI and GEAR-KCVT are evaluated under K2V2 quantization. For GEAR-KCVT, we further set the low-rank residual rank to $r_{res} = 2$, the sparse outlier ratio to $\rho_{sp} = 2\%$, and the number of power-iteration loops to $n_{pi} = 3$ for both prefill and decoding. For xKV, we use a layer group size of $G = 4$ and key/value ranks of $(r_K, r_V) = (384, 576)$ based on a per-layer dimension of 1024. Since MMLU-CoT has an average prompt length of 226, ranks 384/576 are infeasible for this task; therefore, we use $(r_K, r_V) = (96, 144)$ instead. For Palu, we use the configuration that retains 75% of the full KV projection rank. For MOB-KV, we use $M = 4$, total key coefficient rank $r \in \{512, 256\}$, FP16 leading rank $r_{fp} \in \{128, 32\}$, quantized tail rank $r_q = r - r_{fp}$, and key-tail quantization bit-width $b_k = 2$, and value quantization bit-width $b_v \in \{2, 4\}$.

C.1 Basis Granularity

MOB-KV constructs an independent basis pool $\mathcal{V}^{(\ell)} = \{V_1^{(\ell)}, \dots, V_M^{(\ell)}\}$ for the pre-RoPE keys of each layer. Let H_{KV} and d_h denote the number of key heads and the head dimension, respectively. Each basis is constructed over the dimension $d_K = H_{KV}d_h$ formed by concatenating all key heads. Thus, bases are neither shared across layers nor constructed separately for individual heads.

During calibration, we flatten only the batch and token axes of the pre-RoPE key activations at each layer while retaining the concatenated key-head projection dimension d_K , yielding $X_{calib}^{(\ell)} \in \mathbb{R}^{N_{calib} \times d_K}$. Clustering and rank- r basis construction are then performed independently for each layer. During inference, a token at layer ℓ is routed only among the bases in $\mathcal{V}^{(\ell)}$, selecting the one with the highest projection energy.

C.2 Key/Value Quantization and Recent-Window Handling

Each key-cache token is projected onto its selected basis and represented by a coefficient vector $c(x) \in \mathbb{R}^r$. The leading coefficients $c_{fp}(x) \in \mathbb{R}^{r_{fp}}$ are stored in FP16, while the tail coefficients $c_q(x) \in \mathbb{R}^{r_q}$ are compressed using asymmetric min-max quantization. The tail-coefficient tensor is grouped along the token dimension in groups of $g_k=32$ tokens, with an independent scale and zero point for each coefficient channel within each token group. Scales and zero points are stored in FP16,

Table 4: Configurations of baseline methods and MOB-KV. Here, g denotes the quantization group size, w_{fp} the full-precision buffer capacity, M the number of clusters, K2V2 denotes 2-bit key and 2-bit value quantization, r_{res} the low-rank residual rank, ρ_{sp} the sparse outlier ratio, n_{pi} the number of power-iteration loops, G the layer group size, r the total key coefficient rank, r_{fp} the leading rank retained in FP16, $r_q = r - r_{\text{fp}}$ the quantized tail rank, and b_k the key-tail quantization bit-width, and b_v the value quantization bit-width.

Method	g	w_{fp}	M	Rank / Ratio	Bits	Extra
KIVI (K2V2)	32	64	—	—	K2V2	—
GEAR-KCVT (K2V2)	32	64	—	$r_{\text{res}} = 2$	K2V2	$\rho_{\text{sp}} = 2\%$, $n_{\text{pi}} = 3$
xKV	—	—	—	$(r_K, r_V) = (384, 576)$	—	$G = 4$
xKV (MMLU-CoT)	—	—	—	$(r_K, r_V) = (96, 144)$	—	$G = 4$
Palu	—	—	—	75%	—	—
MOB-KV	32	64	4	$r \in \{512, 256\}$, $r_{\text{fp}} \in \{128, 32\}$, $r_q = r - r_{\text{fp}}$	$b_k = 2$, $b_v \in \{2, 4\}$	—

and the quantized tail coefficients are stored in a b_k -bit packed integer representation. The compressed key representation therefore consists of routing indices, an FP16 leading-coefficient buffer, a packed tail-coefficient buffer, and FP16 scale/zero-point metadata.

We do not apply low-rank projection to the value cache; instead, we compress the original value tensor directly using asymmetric min-max quantization. For each token, value channels are partitioned along the channel dimension into groups of $g_v=32$, and an independent scale and zero point are computed for each token-channel group. Scales and zero points are stored in FP16, while quantized values are stored in a b_v -bit packed integer representation. The compressed value representation therefore consists of a packed value buffer and FP16 scale/zero-point metadata for each token-channel group.

Each layer maintains a full-precision key/value buffer for tokens awaiting compression, with a default capacity of $w_{\text{fp}}=64$. When the buffer reaches 64 tokens, the entire buffer is flushed to the compressed cache. Before flushing, keys are stored as pre-RoPE FP16 tensors and values at their exact precision. During a flush, keys are processed by basis routing, coefficient projection, the FP16-leading/quantized-tail split, tail asymmetric quantization, and bit packing, in that order; values are quantized directly in a separate operation. The reported KV-cache retention ratios include the compressed tensors, packed integer storage, FP16 scale/zero-point metadata, and the full-precision buffer.

C.3 Prefill and Decoding Pipeline

For layer ℓ , the compressed-cache state is $\mathcal{B}^{(\ell)} = (P_c, I, C_{\text{fp}}, Q_K, Q_V, P_r, R_K, R_V)$. Here, I stores one routing index per compressed token; C_{fp} stores the FP16 leading coefficients; Q_K and Q_V de-

Algorithm 1 MOB-KV prefill at transformer layer ℓ .

Require: Layer inputs $X_{1:T}$, positions $P_{1:T}$, basis pool $\mathcal{V}^{(\ell)}$, w_{fp} , g_k , g_v , r_{fp} , b_k , and b_v

Ensure: Prefill output $O_{1:T}$ and cache state $\mathcal{B}^{(\ell)}$

- 1: $(Q^{\text{pre}}, K^{\text{pre}}, V) \leftarrow (XW_Q, XW_K, XW_V)$
- 2: $Q \leftarrow \text{RoPE}(Q^{\text{pre}}, P_{1:T})$; $K \leftarrow \text{RoPE}(K^{\text{pre}}, P_{1:T})$
- 3: $O_{1:T} \leftarrow \text{CausalAttention}(Q, K, V)$
- 4: Initialize empty full-precision buffers (P_r, R_K, R_V)
- 5: **for** $t = 1, \dots, T$ **do**
- 6: Append $(P_t, K_t^{\text{pre}}, V_t)$ to (P_r, R_K, R_V) in FP16
- 7: **if** $|R_K| = w_{\text{fp}}$ **then**
- 8: **for** each buffered token i **do**
- 9: $I_i \leftarrow \arg \max_m \|(V_m^{(\ell)})^\top R_{K,i}\|_2^2$
- 10: $c_i \leftarrow (V_{I_i}^{(\ell)})^\top R_{K,i}$; split $c_i = [c_{\text{fp},i}; c_{q,i}]$
- 11: Append I_i and FP16 $c_{\text{fp},i}$ to (I, C_{fp})
- 12: **end for**
- 13: Partition the buffered tail coefficients into groups of g_k ; token-wise quantize and pack them into Q_K
- 14: Independently channel-wise quantize and pack all buffered values into Q_V
- 15: Append P_t to P_c and clear the full-precision buffers
- 16: **end if**
- 17: **end for**
- 18: Discard the temporary post-RoPE tensor K
- 19: **return** $O_{1:T}, \mathcal{B}^{(\ell)}$

note packed key-tail and value blocks, including their scale/zero-point metadata. The full-precision buffers (R_K, R_V) store pending pre-RoPE keys and exact values, respectively, while P_c and P_r record compressed and buffered token positions. The layer-specific basis pool $\mathcal{V}^{(\ell)}$ is stored separately. No post-RoPE key is persistent in $\mathcal{B}^{(\ell)}$. We use $\text{FuseKey}(m; \mathcal{B}^{(\ell)})$ to denote the operator that recovers the coefficient matrix of tokens routed to basis m by dequantizing the key tails and concatenating the FP16 leading coefficients.

Algorithm 2 One MOB-KV decoding step at transformer layer ℓ .

Require: Current input x_t , position p_t , basis pool $\mathcal{V}^{(\ell)}$, cache state $\mathcal{B}^{(\ell)}$, w_{fp} , and g_k

Ensure: Attention output o_t and updated cache state $\mathcal{B}^{(\ell)}$

- 1: $(q_t^{\text{pre}}, k_t^{\text{pre}}, v_t) \leftarrow (x_t W_Q, x_t W_K, x_t W_V)$
- 2: Append $(p_t, k_t^{\text{pre}}, v_t)$ to (P_r, R_K, R_V) in FP16
- 3: **if** $|R_K| = w_{fp}$ **then**
- 4: **for** each buffered token i **do**
- 5: $I_i \leftarrow \arg \max_m \|(V_m^{(\ell)})^\top R_{K,i}\|_2^2$
- 6: $c_i \leftarrow (V_{I_i}^{(\ell)})^\top R_{K,i}$; split $c_i = [c_{fp,i}; c_{q,i}]$
- 7: Append I_i and FP16 $c_{fp,i}$ to (I, C_{fp})
- 8: **end for**
- 9: Partition the buffered tail coefficients into groups of g_k ; token-wise quantize and pack them into $\mathcal{Q}_K$
- 10: Independently channel-wise quantize and pack the values being flushed into $\mathcal{Q}_V$
- 11: Append P_r to P_c and clear the full-precision buffers
- 12: **end if**
- 13: **for** $m = 1, \dots, M$ **do**
- 14: $C_m \leftarrow \text{FuseKey}(m; \mathcal{B}^{(\ell)})$
- 15: $\hat{K}_m^{\text{pre}} \leftarrow C_m (V_m^{(\ell)})^\top$
- 16: **end for**
- 17: $\hat{K}^{\text{pre}} \leftarrow \text{RestoreTokenOrder}(\{\hat{K}_m^{\text{pre}}\}_{m=1}^M)$
- 18: $K_c \leftarrow \text{RoPE}(\hat{K}^{\text{pre}}, P_c)$
- 19: $V_c \leftarrow \text{Dequantize}(\mathcal{Q}_V)$
- 20: $K_r \leftarrow \text{RoPE}(R_K, P_r)$
- 21: $q_t \leftarrow \text{RoPE}(q_t^{\text{pre}}, p_t)$
- 22: $o_t \leftarrow \text{Attention}(q_t, [K_c; K_r], [V_c; R_V])$
- 23: Discard C , $\hat{K}^{\text{pre}}$, K_c , K_r , and V_c after attention
- 24: **return** $o_t, \mathcal{B}^{(\ell)}$

RoPE-transformed keys and dequantized values in Algorithm 2 are temporary attention inputs rather than persistent cache entries.

D Comparison with TurboQuant

As shown in Table 5, we additionally compare MOB-KV with TurboQuant. Since TurboQuant does not provide an official implementation, we report results obtained using a community implementation. We evaluate TurboQuant under its 3-bit quantization setting and compare it with the full-precision baseline and the best-performing MOB-KV configuration.

Consistent with the trend reported in the TurboQuant paper, TurboQuant preserves LongBench performance reasonably well at a low KV cache memory ratio, indicating that its quantization scheme can retain useful long-context information for retrieval- and comprehension-oriented tasks. However, on reasoning benchmarks that were not reported in the original TurboQuant paper, including GSM8K-CoT, BBH-CoT, and MMLU-CoT, TurboQuant shows a substantial accuracy drop compared with both the full-precision baseline and MOB-KV. This suggests that CoT reasoning workloads are more sensitive to the quantization errors

introduced by TurboQuant under low-bit compression. Since these results are based on a community implementation rather than an official release, we present them only as a reference comparison and exclude TurboQuant from the main experimental tables.

E Calibration Data

Table 6 summarizes the datasets, configurations, and splits used for calibration. We stream examples from the training split of each dataset, except for MMLU, where we use the auxiliary_train split because no standard training split is provided. Each example is formatted into a user–assistant message pair using the model tokenizer’s chat template, and approximately 10,000 tokens are collected per task. During calibration, we register forward hooks on k_proj to collect pre-RoPE key activations from all layers; when value statistics are needed, we similarly collect v_proj outputs. The activations are flattened over tokens and KV heads, converted to float32, and stored on CPU for offline basis construction.

F Pre-RoPE Key Subspace Overlap Across Calibration Distributions

To analyze the stability observed in the calibration-distribution shift experiments, we compare the dominant pre-RoPE key subspaces extracted from different calibration distributions. Let X_l^D denote the activations collected at layer l from calibration distribution D , and let $U_{l,r}^D$ be the orthonormal basis formed by the top r eigenvectors of $(X_l^D)^\top X_l^D$. We define the rank- r subspace overlap between the default calibration mixture D_0 and an alternative calibration distribution D as

$$\text{Overlap}_{l,r}(D_0, D) = \frac{1}{r} \left\| (U_{l,r}^{D_0})^\top U_{l,r}^D \right\|_F^2. \quad (9)$$

This value equals 1 when the two subspaces coincide and approaches 0 as they become orthogonal. Table 7 reports the overlap between the default mixture and the WikiText, multilingual, and code calibration distributions. Averaged across all 32 layers, the overlap is already high at rank 256, ranging from 0.784 to 0.827, and exceeds 0.87 for every alternative distribution at rank 512. The representative layers exhibit the same consistent trend. These results indicate that the dominant low-dimensional structure of pre-RoPE keys is largely shared even

Table 5: Comparison with TurboQuant across three models on reasoning and long-context benchmarks. “KV” denotes KV cache size relative to the full-precision baseline. The KV size ratio includes quantization metadata, routing indices, and the recent full-precision window; therefore, it varies with the average sequence length of each benchmark.

Model	Method	GSM8K-CoT		BBH-CoT		MMLU-CoT		LongBench	
		KV	Acc.	KV	Acc.	KV	Acc.	KV	Score
LLaMA-3.1-8B-Instruct	Full (FP16)	100%	0.842	100%	0.693	100%	0.724	100%	0.431
	TurboQuant (3-bit)	29.57%	0.690	28.26%	0.369	44.56%	0.671	23.14%	0.424
	MOB-KV ($r=512, r_{fp}=128, r_q=384, K2V4$)	32.10%	0.841	30.82%	0.682	46.68%	0.723	25.86%	0.431
	MOB-KV ($r=512, r_{fp}=128, r_q=384, K2V2$)	26.41%	0.842	25.02%	0.669	42.20%	0.722	19.65%	0.427
	MOB-KV ($r=256, r_{fp}=32, r_q=224, K2V2$)	20.79%	0.827	19.31%	0.603	37.72%	0.715	13.53%	0.427
Mistral-7B-Instruct-v0.3	Full (FP16)	100%	0.523	100%	0.531	100%	0.596	100%	0.385
	TurboQuant (3-bit)	29.57%	0.434	28.26%	0.346	44.56%	0.553	23.14%	0.371
	MOB-KV ($r=512, r_{fp}=128, r_q=384, K2V4$)	32.10%	0.529	30.82%	0.514	46.68%	0.595	25.86%	0.385
	MOB-KV ($r=512, r_{fp}=128, r_q=384, K2V2$)	26.41%	0.513	25.02%	0.503	42.20%	0.594	19.65%	0.384
	MOB-KV ($r=256, r_{fp}=32, r_q=224, K2V2$)	20.79%	0.506	19.31%	0.473	37.72%	0.576	13.53%	0.384
LLaMA-3.2-3B-Instruct	Full (FP16)	100%	0.766	100%	0.548	100%	0.646	100%	0.331
	TurboQuant (3-bit)	29.57%	0.643	28.26%	0.352	44.56%	0.590	23.14%	0.319
	MOB-KV ($r=512, r_{fp}=128, r_q=384, K2V4$)	32.10%	0.766	30.82%	0.525	46.68%	0.640	25.86%	0.331
	MOB-KV ($r=512, r_{fp}=128, r_q=384, K2V2$)	26.41%	0.758	25.02%	0.502	42.20%	0.636	19.65%	0.329
	MOB-KV ($r=256, r_{fp}=32, r_q=224, K2V2$)	20.79%	0.754	19.31%	0.440	37.72%	0.633	13.53%	0.317

Table 6: Calibration datasets used for offline basis construction. We collect approximately 10,000 tokens from each task using streaming mode.

Task	HuggingFace Dataset	Config / Split
MBPP	mbpp	train
GSM8K	gsm8k	main / train
TriviaQA	trivia_qa	rc / train
SciQ	allenai/sciq	train
MMLU	cais/mmlu	all / auxiliary_train
ARC-Challenge	allenai/ai2_arc	ARC-Challenge / train
OpenBookQA	allenai/openbookqa	main / train
WinoGrande	allenai/winogrande	winogrande_debiased / train
HellaSwag	Rowan/hellaswag	train
NarrativeQA	deepmind/narrativeqa	train
HotpotQA	hotpotqa/hotpot_qa	distractor / train
GovReport	ccdv/govreport-summarization	train
LCC	microsoft/LCC_python	train

across substantially different calibration domains, including general-domain text, multilingual data, and code. Thus, the offline bases of MOB-KV are not strongly tied to a particular calibration workload, directly supporting the robustness observed in the distribution-shift and leave-one-dataset-out experiments in the main text.

G Seed Variance and Uncertainty Analysis

To determine whether the performance of MOB-KV depends on a particular random seed, we repeat the evaluation with five seeds on GSM8K-CoT, MMLU-CoT, BBH-CoT, and LongBench. All experiments use the main configuration from the paper: $M=4$ clusters, total key coefficient rank $r=512$, FP16 leading rank $r_{fp}=128$, quantized tail rank $r_q=384$, K2V4, quantization group size 32,

and a recent full-precision window of 64 tokens. Table 8 summarizes the major benchmark results as mean $\pm$ standard deviation, while Tables 9 and 10 report the individual-seed reasoning results and task-level LongBench statistics, respectively.

Variation across the five seeds is small on every benchmark. In particular, MMLU-CoT produces nearly identical scores across seeds, and the standard deviation of the average LongBench score is only 0.0008. When the absolute differences from the FP16 baseline are interpreted alongside seed variability, the differences on GSM8K-CoT, MMLU-CoT, and LongBench are 0.0004, 0.0004, and 0.0002, respectively, all within one standard deviation of MOB-KV. The very small gaps from FP16 on these three benchmarks are therefore difficult to distinguish from seed-level variation.

Table 7: Pre-RoPE key subspace overlap between the default calibration mixture and alternative calibration distributions. (a) reports the mean and standard deviation across 32 layers. (b) reports representative layer-wise results.

(a) Mean across 32 layers				
Rank	WikiText	Multilingual	Code	
256	0.812 ± 0.046	0.784 ± 0.052	0.827 ± 0.038	
512	0.884 ± 0.027	0.876 ± 0.030	0.901 ± 0.020	

(b) Representative layers				
Rank	Layer	WikiText	Multilingual	Code
256	0	0.945	0.929	0.947
256	16	0.828	0.804	0.850
256	31	0.800	0.758	0.832
512	0	0.962	0.960	0.964
512	16	0.885	0.877	0.909
512	31	0.859	0.847	0.885

Table 10 also reports the five-seed mean and standard deviation for each of the 16 LongBench tasks, enabling direct examination of task-level stability across multi-document QA, summarization, retrieval, and code completion in addition to the aggregate average.

H Matched-Memory Comparison

Because compression methods differ in quantization granularity and auxiliary storage, matching only their theoretical compression ratios does not fairly reflect actual memory usage. We therefore measure the empirical average KV-cache retention ratio on GSM8K-CoT, including compressed key/value tensors, scales and zero points, routing indices, basis storage, and the recent full-precision window, and compare MOB-KV, KIVI, GEAR-KCVT, Palu, and xKV at an empirical memory budget of approximately 26%. Exact matching is infeasible because retention varies with sequence length and method-specific metadata; Table 11 therefore uses the closest configurations within 26.1–26.4%.

At 26.4% retention, MOB-KV scores 0.842 on GSM8K-CoT, retaining higher accuracy than KIVI (0.750), GEAR-KCVT (0.766), Palu (0.000), and xKV (0.676) at nearly the same memory budget.

I Long-Context Robustness Across Context Lengths

We further evaluate RULER separately at 16K, 32K, and 64K to examine whether performance degrades as context length increases. Table 12 compares FP16 with three MOB-KV memory–accuracy operating points at each context length.

The $r=512$, $r_{fp}=128$, $r_q=384$, K2V4 configuration scores 0.933 and 0.875 at 16K and 32K, slightly exceeding the FP16 scores of 0.932 and 0.871, respectively, and matches FP16 at 64K with a score of 0.853. The K2V2 configuration at the same rank also scores 0.850 at 64K, only 0.003 below FP16. The smaller $r=256$, $r_{fp}=32$, $r_q=224$, K2V2 configuration exhibits a consistent modest reduction at all lengths but no new sharp performance collapse from 16K to 64K. Thus, the higher-memory configurations retain full-precision-level performance through 64K, while the loss under more aggressive compression is reported explicitly at each context length.

J SVDq and the Online–Offline Basis Trade-off

SVDq was omitted from the original comparison because the compression scopes differed: SVDq compresses only the KV cache of prefill tokens, whereas MOB-KV in the main experiments compresses both prefill and decoding tokens. To remove this discrepancy, we configure MOB-KV for prefill-only compression and compare the two methods at nearly identical empirical memory budgets. In Table 13, the retention ratios of MOB-KV and SVDq are 26.40% and 26.12%, respectively. SVDq scores 0.006, 0.002, and 0.012 higher on GSM8K-CoT, MMLU-CoT, and BBH-CoT, respectively, while both methods score 0.429 on LongBench. Thus, under matched scope and memory, the two methods provide broadly comparable accuracy–memory trade-offs, with SVDq retaining an advantage on the reasoning benchmarks.

Unlike the prefill-only comparison in Table 13,

Table 8: Five-seed uncertainty of the main results. The absolute gap is measured against the FP16 baseline. “Within 1 std.” indicates whether the FP16 gap is no larger than one standard deviation of MOB-KV.

Benchmark	FP16	MOB-KV (mean $\pm$ std.)	Absolute gap	Within 1 std.
GSM8K-CoT	0.842	0.8416 $\pm$ 0.0033	0.0004	Yes
MMLU-CoT	0.724	0.7236 $\pm$ 0.0005	0.0004	Yes
BBH-CoT	0.693	0.6837 $\pm$ 0.0039	0.0093	No
LongBench	0.431	0.4308 $\pm$ 0.0008	0.0002	Yes

Table 9: Reasoning performance across five random seeds.

Benchmark	Seed 0	Seed 1	Seed 2	Seed 3	Seed 4	Mean $\pm$ std.
GSM8K-CoT	0.8410	0.8450	0.8450	0.8380	0.8390	0.8416 $\pm$ 0.0033
MMLU-CoT	0.7230	0.7240	0.7240	0.7240	0.7230	0.7236 $\pm$ 0.0005
BBH-CoT	0.6820	0.6815	0.6810	0.6904	0.6835	0.6837 $\pm$ 0.0039

Table 10: LongBench task-level mean and standard deviation over five random seeds.

Task	Mean $\pm$ std.	Task	Mean $\pm$ std.
2WikiMQA	0.5195 $\pm$ 0.0053	PassageCount	0.0890 $\pm$ 0.0014
GovReport	0.3437 $\pm$ 0.0011	PassageRetrieval-en	1.0000 $\pm$ 0.0000
HotpotQA	0.5803 $\pm$ 0.0013	Qasper	0.4541 $\pm$ 0.0021
LCC	0.1871 $\pm$ 0.0008	QMSum	0.2548 $\pm$ 0.0020
MultiNews	0.2663 $\pm$ 0.0009	RepoBench-P	0.1745 $\pm$ 0.0012
MultiFieldQA-en	0.5717 $\pm$ 0.0019	SAMSum	0.2187 $\pm$ 0.0019
MuSiQue	0.3166 $\pm$ 0.0025	TREC	0.6880 $\pm$ 0.0057
NarrativeQA	0.3086 $\pm$ 0.0028	TriviaQA	0.9194 $\pm$ 0.0020
Average			0.4308 $\pm$ 0.0008

Table 11: Matched-memory comparison using the empirically measured average KV-cache retention ratio on GSM8K-CoT. The reported ratio includes method-specific auxiliary storage.

Method	Configuration	KV-cache retention	GSM8K-CoT
MOB-KV	$M=4$, $r=512$, $r_{fp}=128$, $r_q=384$, K2V4, $g=32$, $w=64$	26.4%	0.842
KIVI (K2V2)	K2V2, $g=32$, $w=64$	26.1%	0.750
GEAR-KCVT (K2V2)	K2V2, K4/V4 residual ranks, 3 power iterations, 2% sparse residual, $w=47$	26.1%	0.766
Palu	Matched-memory configuration	26.1%	0.000
xKV	$R_K=128$, $R_V=150$, layer-group size 2	26.1%	0.676

Table 14 compresses both prefill and decoding KV caches. The online-SVD baseline performs SVD once on the layer-wise pre-RoPE keys collected from each request’s prefill prompt to construct an input-specific basis. This basis is used to compress the prefill cache and then remains fixed for the duration of the request, being reused to compress every subsequently generated decoding key. We do not rerun SVD at each decoding step because repeated decomposition during autoregressive decoding incurs prohibitive runtime overhead. This prefill-once variant is used only for experimental tractability, since the fully adaptive online-SVD procedure would repeatedly update the basis during decoding and was too time-consuming to evaluate. Both the offline-basis and online-SVD settings use K2V4, group size 32, and a recent full-precision window of 64 tokens; the offline-basis setting uses $M=4$ clusters.

This comparison highlights the trade-off between input-specific adaptivity and runtime cost. Online SVD can fit its basis directly to the dominant subspace of the current prompt, but incurs one decomposition during prefill for every request. In contrast, MOB-KV eliminates this cost by reusing a basis pool constructed during calibration, deliberately sacrificing some input-specific adaptivity. At $r=512$, $r_{fp}=128$, and $r_q=384$ in Table 14, online SVD achieves the highest score of 0.848, confirming that a per-input basis can be more accurate when sufficient rank is available. This is a clear limitation of the offline basis pool.

Conversely, at $r=128$, $r_{fp}=32$, and $r_q=96$, online SVD drops sharply to 0.594, while the offline bases calibrated on different domains retain scores of 0.747–0.787. This result suggests that under aggressive low-rank compression, a reusable subspace learned from broad calibration data can be

Table 12: RULER results at different context lengths. r , r_{fp} , and r_q denote the total key coefficient rank, FP16 leading rank, and quantized tail rank, respectively.

Method	16K	32K	64K
FP16	0.932	0.871	0.853
MOB-KV ($r=256$, $r_{fp}=32$, $r_q=224$, K2V2)	0.918	0.859	0.838
MOB-KV ($r=512$, $r_{fp}=128$, $r_q=384$, K2V2)	0.928	0.865	0.850
MOB-KV ($r=512$, $r_{fp}=128$, $r_q=384$, K2V4)	0.933	0.875	0.853

Table 13: Matched-scope comparison between MOB-KV and SVDq. Both methods compress only the prefill KV cache and use closely matched memory budgets.

Method	KV-cache retention	GSM8K-CoT	MMLU-CoT	BBH-CoT	LongBench
MOB-KV (prefill-only)	26.40%	0.842	0.724	0.682	0.429
SVDq	26.12%	0.848	0.726	0.694	0.429

Table 14: Offline MOB-KV bases and per-input online SVD across three rank configurations on GSM8K-CoT. Unlike the prefill-only comparison in Table 13, this experiment compresses both prefill and decoding KV caches. The online basis is computed once from the prefill prompt and reused throughout decoding. The headers report total key coefficient rank r , FP16 leading rank r_{fp} , and quantized tail rank r_q ; all configurations use K2V4.

	$r=512$ $r_{fp}=128$ $r_q=384$	$r=256$ $r_{fp}=64$ $r_q=192$	$r=128$ $r_{fp}=32$ $r_q=96$
Basis construction			
WikiText offline basis	0.835	0.818	0.747
Multilingual offline basis	0.837	0.821	0.787
Code offline basis	0.831	0.831	0.782
Online SVD	0.848	0.808	0.594

more stable than a basis fitted to a single input. Overall, online SVD can provide a better input-specific fit at high rank, whereas offline MOB-KV is more robust at low rank and under calibration-domain shift while avoiding decomposition latency. The contribution of MOB-KV is therefore not to outperform online SVD in accuracy across every regime, but to trade some adaptivity for more stable low-rank behavior and practical serving efficiency. This choice may degrade performance when the offline basis pool does not sufficiently cover the deployment workload; lightweight online adaptation is an important direction for future work.

K MOB-KV Compression Configurations

Table 15 reports the MOB-KV configurations used in the Pareto figures. The table lists the memory ratio, number of basis clusters, total key coefficient rank, FP16 leading rank, quantized tail rank, and key/value quantization bit-widths. These settings define the operating points used to evaluate the memory-accuracy and latency trade-offs of MOB-KV.

L Extended Related Work

KV Cache Eviction. In addition to quantization and low-rank approximation, token-level KV cache eviction reduces memory by selectively preserving important tokens rather than retaining the KV states of all context tokens. Existing methods identify such tokens using attention-based importance signals, including heavy hitters, persistent token importance, and attention sinks (Zhang et al., 2023; Liu et al., 2023; Xiao et al., 2024). Recent work further improves eviction through prompt-aware selection, layer-wise budget allocation, or adaptive KV cache policies (Li et al., 2024; Cai et al., 2024; Feng et al., 2026). These methods are orthogonal to MOB-KV: eviction reduces the sequence dimension by removing tokens, whereas MOB-KV retains all tokens and compresses their hidden-dimensional representations via offline basis projection and quantization.

KV Cache Quantization. KV cache quantization reduces memory usage by storing key/value states in low-bit representations without evicting tokens. KIVI analyzes the distributional differences between keys and values, and applies 2-bit quantization using per-channel quantization for keys and per-token quantization for values (Liu et al., 2024b).

Table 15: MOB-KV configurations used in the pareto figures ($M = 4$ clusters).

KV-cache retention	M	Total key coefficient rank r	FP16 leading rank r_{fp}	Quantized tail rank r_q	KV bits
19.7%	4	128	32	96	K2V2
22.9%	4	512	32	480	K2V2
24.2%	4	256	128	128	K2V2
26.4%	4	512	128	384	K2V2
28.6%	4	512	32	480	K2V4
29.9%	4	256	128	128	K2V4
32.1%	4	512	128	384	K2V4

GEAR mitigates the quality degradation caused by quantization by compensating for the error introduced after low-bit quantization with low-rank and sparse components (Kang et al., 2024). In addition, various KV cache quantization methods have been proposed based on outlier-aware quantization, mixed-precision quantization, and progressive quantization (Hooper et al., 2024; Yang et al., 2024; Su et al., 2025; Liu et al., 2025). These approaches can reduce cache size while retaining all tokens, but under aggressive low-bit settings, activation outliers and quantization error can degrade attention quality.

KV Cache Low-Rank Approximation. Low-rank approximation-based methods exploit hidden-dimensional and cross-layer redundancy in the KV cache or KV projections to store the cache in a more compact representation. Palu compresses KV projections into low-rank forms and caches compact intermediate states (Chang et al., 2025b), while LoRC applies low-rank approximation to KV weight matrices with a progressive compression strategy (Zhang et al., 2024). xKV further exploits cross-layer redundancy through online SVD across layer groups (Chang et al., 2025a). Other methods have also explored calibration-based low-rank attention and layer-wise KV merging (Saxena et al., 2024; Liu et al., 2024a). These approaches can reduce the representation dimension while preserving all tokens; however, relying solely on low-rank approximation may discard residual information, and the reconstruction process can introduce non-negligible runtime overhead.

Existing KV cache compression methods mainly focus on either quantization or low-rank approximation: the former reduces numerical precision, while the latter exploits hidden-dimensional or cross-layer redundancy. Although recent work combines the two (Yankun et al., 2025), it does not separately and jointly analyze how their errors affect the final attention output. Moreover, such methods often require input-dependent online SVD or decompo-

sition, incurring impractical runtime overhead for LLM serving. We address these limitations by analyzing the interaction between quantization and low-rank approximation and deriving a practical KV cache compression design that avoids online SVD.