HqeKV: Towards Hybrid Quantization and Eviction for KV Cache in Long-Context LLM Inference

Anonymous ACL submission

Abstract

The autoregressive inference in large language models requires repeated computation across transformer layers. While caching intermediate key-value (KV) pairs eliminates redundancy, it introduces severe memory overhead, particularly in long-context settings. Existing cache compression methods operate solely on either quantization or eviction, based on importance estimation of cached data. However, they are limited by coarse compression choices and inaccurate importance assessment, leading to suboptimal inference quality. To address this, we propose HqeKV, a hybrid compression framework built on both quantization and eviction, offering finer-grained compression options that adapt smoothly to the varying importance of cached KV pairs. An integrated optimizer automatically selects the best compression action for each cached element, maximizing quality while insulating end-users from tedious low-level tuning details. We further design a joint K–V importance metric to provide more accurate importance assessment results so that the optimizer can make smarter decisions. Additionally, HqeKV supports flexible conversion policies across multiple quantization precision levels, to further reduce quality degradation. Extensive experiments show that HqeKV improves output quality from 40.53 to 49.98 under the same memory constraints, outperforming state-of-the-art alternatives. Code is available at https://github.com/skywclouds/HqeKV.

1 Introduction

Large Language Models (LLMs) are gaining increasing popularity due to their remarkable performance across a wide range of tasks. LLMs work through Inference-as-a-Service (IaaS) where the prompt from an end-user is processed autoregressively through stacked transformer layers to generate output tokens. Each newly generated output is appended to the input sequence and then processed together, resulting in substantial redundant computations for the repeated input. Caching intermediate key-value (KV) pairs of previously processed tokens can effectively mitigate this inefficiency, but introduces a significant memory bottleneck. The size of KV Cache grows linearly with sequence length and can exceed the model's own memory footprint, especially for long-context IaaS.

Many recent efforts have been devoted to compressing KV Cache, which primarily fall into two categories, i.e., quantization and eviction. Quantization methods reduce numerical precision, typically converting FP16 values to a single low-precision format (Su et al., 2025) or, at most, two mixed low-precision formats (Li et al., 2025). However, their coarse quantization granularity and the uniform converting strategy cannot adapt to the varying information density present in different cached pairs, forcing a difficult trade-off between inference quality and memory savings. Eviction methods, on the other hand, discard KV pairs deemed less important (Qin et al., 2025), avoiding (de)quantization runtime penalty and enabling faster inference, but often incur noticeable quality degradation. Crucially, both lines of compression works rely on distinguishing the importance of cached data. Current metrics, however, consider only the key (K) vectors and ignore the influence of value (V) vectors, leading to imprecise assessments that further exacerbate the quality loss.

Motivated by these limitations, we propose HqeKV, a hybrid framework that combines quantization and eviction for efficient KV Cache compression. Cached KV pairs are sorted by importance in ascending order. The top-tier pairs are quantized to high precision or even not compressed, the middle tier to medium or low precision, and the bottom tier is evicted. To better accommodate varying importance levels, we particularly expand the candidate precision levels to three, after balancing bit utilization and runtime latency. Furthermore, a

built-in optimizer automatically determines the optimal boundaries among FP16, three quantization precisions and eviction, based on available memory capacity, thereby maximizing inference quality.

The increased number of compression options amplifies the challenge of accurate importance assessment, which is fundamental to optimizing action boundaries. This problem is exacerbated by the standard cumulative attention weight metric, which relies solely on K vectors and excludes V vectors since the information of the latter is not readily available in advance. To address this gap, we first observe a positive correlation between inference quality degradation and the range of V vectors. Leveraging this observation and adhering to the principles of attention computation, we propose a new joint K-V metric to accurately assess the importance of cached pairs.

We further find that within our fine-grained quantization scheme, the optimal conversion strategies differ by bit-width. The uniform conversion works best for high bit-widths/precision, while the normalized conversion is superior for low bit-widths/precision. HqeKV accordingly applies the tailored strategy for each precision level, thereby enhancing inference quality.

Our contributions are summarized as below.

- Proposing a hybrid compression framework HqeKV that combines quantization and eviction where cached pairs are processed using one of five compression actions——FP16 (full precision), three fine-grained quantization precisions or eviction. By adaptively optimizing the action boundaries and selecting tailored conversion strategies for different precisions, HqeKV maximizes inference quality within given memory limits.

- Proposing a novel importance metric that jointly incorporates both key (K) and value (V) tensors, based on the observed positive correlation between quality degradation and the range of V vectors. This refined metric improves the distinction among cached pairs, leading to superior boundary decisions and enhanced inference quality.

- Extensive experiments on the widely used LongBench suite and multiple LLMs show that HqeKV significantly outperforms the state-of-the-art, improving quality from 40.53 to 49.98 under the same memory limit.

2 Preliminary

2.1 Transformer Architecture

Most of the state-of-art LLMs currently are built by stacking multiple decoder-only Transformer layers. Each Transformer layer maps its input vector x to an output vector of the same dimension.

The core of a single Transformer layer is the Multi-Head Self-Attention (MHSA) mechanism. Given input vector x, MHSA firstly generate Q, K, V as query, key, and value for every head. Then, the self-attention for every head is computed by:

$$O = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_h}}\right)V, \tag{1}$$

where the $\text{Softmax}(QK^\top/\sqrt{d_h})$ is called attention weights.

2.2 LLM Inference and KV Cache

LLM inference consists of two phases, including prefilling and decoding. In prefilling, text is encoded into tokens and then processed through transformer layers to produce the first output token x_0. In decoding, the model iteratively appends new tokens to the prefix and re-inputs it, causing redundant calculations of K, V and self-attention. To optimize this, KV Cache stores previous Keys and Values matrices. When generating x_{t+2}, only the newly produced vector x_{t+1} is projected into key and value spaces k_{t+1}, v_{t+1}. These projections are then concatenated with the cached matrices $K = [K|k_{t+1}], V = [V|v_{t+1}]$. This incremental reuse technique is called KV Cache mechanism.

2.3 Quantization Strategies

Quantization is a kind of methods mapping numbers from high precision to low precision. This paper employs uniform and normalized quantization. The following first introduces the uniform quantization. Let the vector to be quantized be $\Xi \in \mathbb{R}^{1\times D}$, then the quantization and de-quantization output is shown as follows:

$$\Xi_q = \frac{\Xi - z}{s}, \quad \Xi_{dq} = \Xi_q \cdot s + z,$$

where $z = \min(\Xi)$ is the offset, and $s = \frac{\max(\Xi)-\min(\Xi)}{2^b-1}$ is the scaling factor, and b is the quantization precision. This process can be regarded as placing each number at the center of a set of equal-length intervals.

Second, we introduce the normalized quantization. The first step in normalized quantization is shown as follows:

$$\Xi' = \frac{\Xi - m}{s},$$

where m and s is the mean value and standard deviation of ξ. Then, let $C = [c_0, \cdots, c_{2^b-1}]$ to be the $\frac{1}{2^b}, \cdots, \frac{2^b-1}{2^b}$ quantile of the standard normal distribution. The quantization output of numbers in vector Ξ is shown as follows:

$$\Xi_{iq} = \min_{0<j<2^b-1} \int_{\min(c_j, \Xi'_i)}^{\max(c_j, \Xi'_i)} \varphi(u)\mathrm{d}u,$$

where $\varphi(t)$ is the standard normal density function. And the de-quantization output is shown as follows:

$$\Xi_{dq} = C[\Xi_q] \cdot s + m.$$

This process can be regarded as placing each number at its nearest quantile of the standard normal distribution.

3 Method

3.1 Overview of HqeKV

We present HqeKV, a novel KV Cache compression method and the general framework is illustrated in figure 1. In the prefilling phase, the Q, K and V are generated and then in compression phase, a novel importance metric jointly incorporates attention weights and value token ranges guides the allocation of K and V into five precision based on the optimal ratio with specific compression strategy. In the decoding phase, the KV Cache is periodically re-quantized after every T generated tokens.

The remainder of this section is organized as follows. In Section 3.2, we introduce the proposed joint importance metric for KV Cache compression, which incorporates both attention scores and the range of value tokens. Section 3.3 presents a search algorithm designed to determine the optimal allocation ratio across different precision levels. In Section 3.4, we detail the precision-specific compression strategies employed for KV Cache. Finally, Section 3.5 describes the operational mechanism of HqeKV in the decoding phase.

3.2 Joint Importance Metric with Range

Most existing KV Cache compression methods rely on cumulative attention weights to evaluate importance, often overlooking the intrinsic properties of the KV Cache. In this section, we derive the relationship between conversion loss and data range, providing support for designing a more reasonable KV Cache importance metric.

The quantization process can be regarded as mapping each value to its nearest quantization center. Assuming the data to be quantized is independently and identically distributed as $p(\xi)$, the quantization bit-width b, and denoting the interval that is closest to quantization center c_i in possibility as I_i, the expected conversion loss for any value can be expressed as follows:

$$\begin{aligned} &\underset{\xi \sim p(\xi)}{E}(\mathcal{L}_\mathrm{n}(\xi) | \xi \in [\min(\Xi), \max(\Xi)]) \\ &= \frac{\sum_{i=0}^{2^b-1} \int_{\xi \in I_i \bigcap [\min(\Xi), \max(\Xi)]} (\xi - c_i)^2 p(\xi)\mathrm{d}\xi}{\int_{\min(\Xi)}^{\max(\Xi)} p(\xi)\mathrm{d}\xi}. \end{aligned} \tag{2}$$

Equation (2) can be interpreted as drawing a sample ξ from the probability distribution $p(\xi)$, and ξ may fall into any quantization interval. We compute the expected squared distance between ξ and the quantization center within each interval and then sum them up. Owing to the generic formulation, the expression holds for both uniform and normalized quantization—the sole distinction resides in the partitioning policy and the placement of the representative codewords.

From Equation (2), it follows that the quantization error of a vector is determined by its extremal values. Consequently, defining the range of a vectot Ξ as $R_\Xi = \max(\Xi) - \min(\Xi)$, the range emerges as a direct indicator of the incurred conversion loss.

To validate the viewpoint, We visualize the Equation (2) and by means of numerical integration. We assume that $p(\xi)$ is the standard normal density function, b is 4 and quantization strategy is uniform quantization in Equation (2). We also randomly select 1,000 tokens from the V Cache of Llama-3.1-8B-Instruct, corresponding to the first question in the Qasper dataset of LongBench (Bai et al., 2024) and quantize them to 4, 2 bit-widths under uniform quantization, 1 bit-width under normalized quantization and eviction, evaluating the range and conversion losses as shown in Figure 2.

From Figure 2 we can obtain that there is a positive correlation between conversion losses and the range of data. We also calculate the Pearson correlation coefficients (PCCs) of conversion losses from the randomly select tokens and range as shown in Table 1. And a positive correlation be-

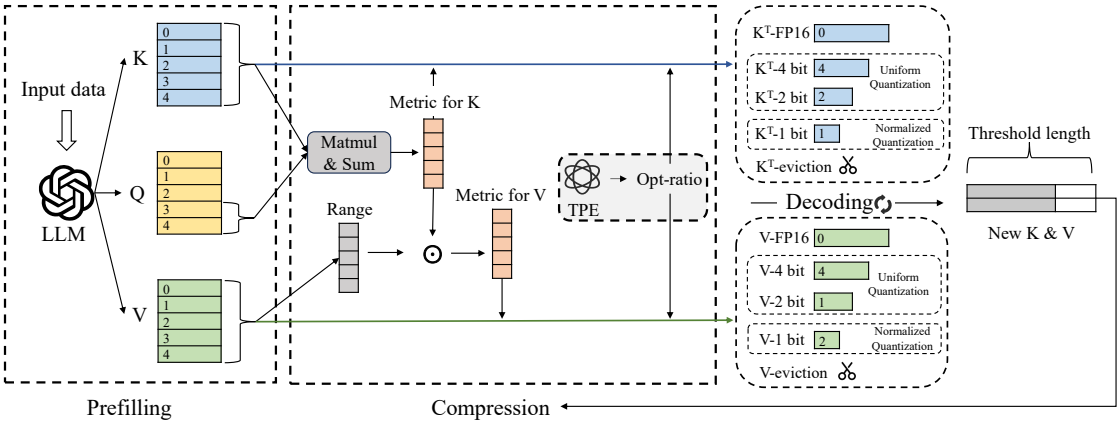
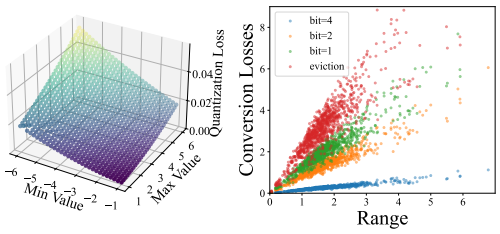


Figure 1: The inference framework of HqeKV in a single Transformer layer.



(a) Numerical integration of Equation (2)

(b) Conversion losses of tokens with range

Figure 2: Numerical integration of Equation (2) under 4 bit-width uniform quantization and the conversion losses of tokens from KV Cache in LongBench with range.

tween conversion losses and the range is indicated by the results.

In summary, the range serves as a viable metric for quantifying the conversion losses. Accordingly, we quantify K's importance through cumulative attention weights. Meanwhile, the importance of V is assessed through the product of the cumulative attention weights and the range of each token within V since in Equation (1) there is a multiplication between attention weights and V.

Bit-width	4-bit	2-bit	1-bit	eviction
PCCs (V Cache)	0.96	0.96	0.95	0.91

Table 1: Pearson correlation coefficients (PCCs) of conversion losses and range at 4, 2 and 1 bit-widths and eviction.

3.3 Optimizer for Precision-Ratio Allocation

The allocation of precision ratio in the scenario of mixed-precision compression is a highly critical issue. The existing approaches typically involve either manual allocation of the ratio of each precision or the establishment of certain thresholds to determine the precision to which a token should be quantized. However, these strategies are clearly not the optimal approach. In contrast, our work adopt an offline search method to determine the ratio of each precision.

Let the average bit-width of compressed KV Cache to be B_{avg} and the ratios of 4, 2, 1 bit-widths and eviction to be $\delta_1, \delta_2, \delta_3, \delta_4$. Then the allocation of ratios for different precision can be expressed by Equation (3).

$$\begin{cases} 4\delta_1 + 2\delta_2 + \delta_3 = B_{\text{avg}} \\ \delta_1 + \delta_2 + \delta_3 + \delta_4 = 1 \end{cases} \quad (3)$$
$$\text{s.t. } B_{\text{avg}} > 0,\ 0 \leq \delta_1, \cdots \delta_4 \leq 1.$$

To solve the infinite solutions of Equation (3), we set δ_3, δ_4 as free variables and represent δ_1, δ_2 in terms of them. We use the Tree-structured Parzen Estimator (TPE) (Bergstra et al., 2011) to efficiently search for the values of δ_3 and δ_4. Specifically, we set the number of iterations to 200. Due to the quantization dimensions of K and V are different, we partition the KV Cache into chunks with size G. Within each chunk, we apply the per-channel quantization to K and the per-token quantization to V. To determine the importance of each chunk, we utilize the average importance of the tokens within the same chunk as a representative measure. We denote the cross-entropy of the model's prefilling output in certain quantization ratio and the full-precision prefilling output on the calibration dataset as $\mathcal{L}_{\text{c}}$.

Additionally, we design a regularization term

$\mathcal{L}_{\text{reg}} = \alpha x_3 + \beta x_4$. We aim to reduce the ratios of low precision through the incorporation of this regularization term. So the final loss function is defined as $\mathcal{L} = \mathcal{L}_{\text{c}} + \mathcal{L}_{\text{reg}}$.

The ratio that minimizes $\mathcal{L}$ is considered as the optimal ratio. And we retain the most import 2 chunks in full precision to further improve the model performance under quantization. The process of the search is shown in Appendix A.1.

3.4 Precision-Specific Quantization Strategies

Existing KV Cache mixed-precision quantization methods uniformly apply a single quantization strategy across all precision levels, overlooking the fact that distinct strategies are better suited to different bit-widths. To substantiate this claim, we extract the KV Cache of Llama-3.1-8B-Instruct from the first question on Qasper dataset in LongBench (Bai et al., 2024), randomly selecting 128 channels for K and 128 tokens for V and quantizing them to 4, 2 and 1 bit-widths under uniform and normalized quantization. We avoid 8 bit-width since its 256 states exceed the 128-dimensional attention heads in current LLMs, causing unnecessary overhead. The conversion losses are shown in Table 2.

Strategies	K			V		
	4-bit	2-bit	1-bit	4-bit	2-bit	1-bit
Uniform	**5.28**	25.59	92.06	**0.29**	1.44	4.67
Normalized	9.28	23.29	**37.44**	0.62	1.47	**2.05**

Table 2: Conversion losses for the K and V cache across different bit-widths and quantization strategies. The quantization group size is set to 32.

Table 2 reveals that, for both K and V, normalized quantization produces larger conversion losses than uniform quantization at 4-bit. A reversal occurs only in the 1-bit regime, where normalized quantization exhibits a decisive advantage over its uniform counterpart.

Since NQKV (Cai et al., 2025) confirm that the data in KV Cache follows a normal distribution, we randomly sampled 128 vectors $\in \mathbb{R}^{128}$ from the standard normal distribution and quantize them to 4, 2 and 1 bit-widths. The conversion losses are shown in Table 3.

We can obtain the same phenomenon from Table 2 and Table 3. In summary, we decide to perform uniform quantization for 4 and 2 bit-width and normalized quantization for 1 bit-width. For quantization dimension, we adopt per-channel and per-token quantization to K and V.

Strategies	4-bit	2-bit	1-bit
Uniform	**0.87**	4.38	15.46
Normalized	1.47	4.07	**6.77**

Table 3: Conversion losses for random vectors from standard normal distribution across different bit-widths and quantization strategies. The quantization group size is set to 32.

To reduce the computational overhead, we only select m most recent tokens to calculate the cumulative attention weights. The detailed algorithm is shown in Appendix A.2.

3.5 Strategy in Decoding Phase

In the decoding phase, we re-quantize all KV Cache after every T generated tokens. If the precision of a token is lower than that in the previous quantization, we further quantize it. Otherwise, it is retained unchanged. This strategy is motivated by the observation that only a minimal fraction of the KV Cache transitions to higher precision. Further details are provided in Appendix A.3.

4 Experiments

4.1 Experimental Setup

We conduct our experiments on Llama-3.1-8B-Instruct (Grattafiori et al., 2024) and Qwen3-8B (Yang et al., 2025), which are widely used open-source models with a maximum context length of 128k.

We evaluate the model on LongBench (Bai et al., 2024), a pioneering bilingual, multi-task benchmark for long context understanding which comprises 21 datasets across 6 task categories in both English and Chinese, including single-doc QA, multi-doc QA, summarization, few-shot learning, synthetic tasks, and code completion.

We benchmark baselines against five state-of-the-art methods, including single-precision quantization methods KIVI (Liu et al., 2024) and OTT (Su et al., 2025), mixed-precision quantization methods ZipCache (He et al., 2024), CAKE (Qin et al., 2025) and eviction method KVTuner (Li et al., 2025). We set the average compression bit-width for HqeKV to 2, 3.2 and 3.25 corresponding to the compression bit-width of quantization baselines. And we set the eviction ratio of CAKE to the same level as 3.2 bit-width quantization. The detailed hyper-parameters setup is shown in appendix A.4.

Finally, all experiments are conducted on a NVIDIA RTX A6000 GPU (48GB) device.

4.2 Evaluation on Inference Quality

The brief evaluation results of HqeKV with all baselines on the LongBench dataset are shown in Table 4 and the detailed results with quantization baselines are shown in Appendix A.5.1. Moreover, the comparison of HqeKV and eviction method CAKE is shown in Appendix A.5.2.

As shown in Table 4, HqeKV demonstrates superior performance compared to all baselines. Notably, HqeKV achieves 123.69% improvement over CAKE on Multi-Document QA and 120.99% improvement on Single-Document QA. HqeKV not only achieves the highest average score, but also sets a new state-of-the-art in 81.25% of the individual tasks. Even in some tasks that do not exceed baselines, there is only a slight gap. Remarkably, in several scenarios HqeKV delivers superior accuracy while operating under a 14.29% stricter KV Cache memory budget. Specifically, ZipCache and KVTuner may encounter out-of-memory (OOM) issues and CUDA ERROR. It should be particularly noted that the "Avg" scores for ZipCache-2.8, ZipCache-3.2 and HqeKV-3.2 near ZipCache in Table 4 are calculated from non-OOM columns. And the "Avg" scores for KVTuner-3.25 and HqeKV-3.25 near KVTuner in Table 4 are calculated from the non-ERROR columns.

4.3 Evaluation on Inference Scalability

We evaluate the inference Scalability by examining memory usage and throughput. Specifically, we first investigate the maximum batch size achievable by different methods across varying context lengths. Additionally, we analyze the memory usage of different methods with varying batch sizes, focusing on a context length of 4096 tokens. The number of generated new tokens in both evaluations above is set to 32, and the results are illustrated in Figure 3.

As depicted in Figure 3, HqeKV enables the largest batch size in most cases and reduces memory usage by 29.6%, 8.3% and 8.6% on average compared to OTT, ZipCache and KVTuner. The reason is that their compression strategies introduce substantial storage overhead, degrading overall memory efficiency. OTT computes the 1-norm for every token, incurring substantial memory overhead. ZipCache samples 10% token for cumulative attention weights, leading to higher memory usage. KVTuner sacrifices efficiency by storing quantized values without bit-packing, leaving significant unused memory. In contrast, HqeKV adopts a memory-efficient design and incorporates a customized Triton kernel that densely packs quantized values, thereby minimizing the memory footprint.

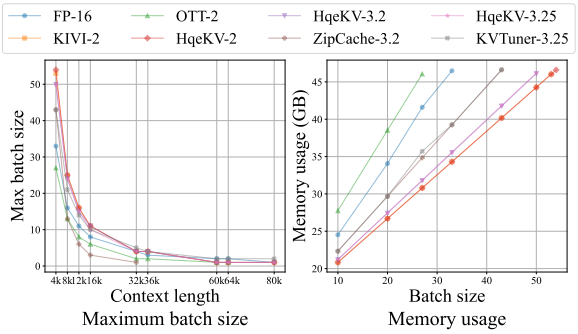


Figure 3: Maximum batch size and memory usage of different methods.

Second, we analyze the throughput efficiency by comparing the time per output token (TPOT) of HqeKV with that of quantization baselines on the Wikitext-2 dataset. Specifically, the comparison is conducted under two different settings, including one with a context length of 161 tokens and 338 newly generated tokens, and the other with a context length of 4096 tokens and 128 newly generated tokens. The results are illustrated in Figure 4.

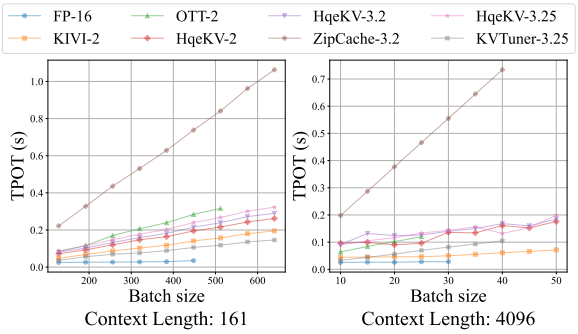


Figure 4: Time per output token (TPOT) of HqeKV with that of quantization baselines on Wikitext-2 dataset across varying context lengths.

The superior efficiency of HqeKV over OTT and ZipCache stems from that HqeKV introduces a specialized CUDA kernel that integrates the dequantization and vector-matrix multiplication processes. Although HqeKV exhibits higher TPOT than KIVI and KVTuner, this overhead remains within an acceptable range.

Methods	Single-Document QA	Multi-Document QA	Summarization	Few-shot Learning	Synthetic Task	Code Completion	Avg
			Llama-3.1-8B-Instruct				
FP16	47.7	40.45	26.19	63.26	66.81	56.59	49.76
KIVI-2	46.84	39.79	25.9	63.04	65.38	55.22	48.96
OTT-2	47.02	40.04	25.95	62.92	64	57.99	49.43
HqeKV-2	46.53	**40.12**	25.82	**63.07**	**66.4**	**58.17**	**49.71**
ZipCache-3.2	52.13	39.82	22.73	68.66	66.73	61.32	50.98
HqeKV-2.8	54.03	40.17	22.86	68.15	67.14	60.77	51.26
HqeKV-3.2	**54.2**	**40.93**	**22.88**	67.76	**67.3**	61.25	**51.49**
CAKE-3.2	21.44	17.28	22.96	61.19	58.89	57.6	40.3
HqeKV-3.2	**47.38**	**40.93**	**25.81**	**62.45**	**67.3**	**57.83**	**49.91**
KVTuner-3.25	46.78	43.06	25.78	62.58	66.98	60.91	49.94
HqeKV-3.25	**47.06**	**44.55**	**26.08**	**62.86**	66.17	**61.91**	**50.33**
CAKE-3.25	24.64	18.57	22.8	60.41	60.5	58.59	40.53
HqeKV-3.25	**47.06**	**41.54**	**26.08**	**62.86**	**66.17**	58.02	**49.98**
			Qwen3-8B				
FP16	47.14	40.52	23.9	63.12	66.5	66.11	51.24
KIVI-2	46.05	40.85	23.82	63.71	63.89	63.62	50.36
OTT-2	46.25	41.35	23.89	63.09	65	64.7	50.75
HqeKV-2	**46.44**	**41.68**	**24**	63.55	**66.64**	**65.83**	**51.37**
ZipCache-3.2	53.55	40.15	20.38	62.53	66.5	66.93	51.22
HqeKV-2.8	**54.94**	**41.03**	**20.95**	62.12	**67.42**	**68.72**	**51.98**
HqeKV-3.2	54.77	40.62	20.82	**63.05**	67.17	67.88	51.89
CAKE-3.2	47.06	40.71	22.01	62.13	66.33	64.17	50.45
HqeKV-3.2	46.86	40.62	**23.94**	**63.05**	**67.17**	**66.15**	**51.29**

Table 4: Inference scores of Llama-3.1-8B-Instruct and Qwen3-8B on LongBench.

4.4 Overall Performance Analysis

To intuitively compare the methods across multiple dimensions and thereby judge their overall performance, we plot a radar chart to provide a multi-dimensional comparison of the HqeKV, KIVI, OTT and ZipCache based on the obtained experimental results, as shown in Figure 5.

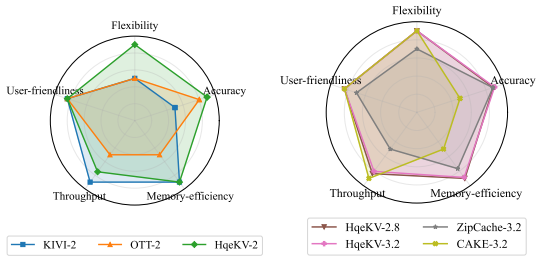


Figure 5: Multi-dimensional performance comparison of HqeKV and other methods.

The accuracy score in Figure 5 corresponds to the performance on the Multi-Document QA task in LongBench with Llama-3.1-8B-Instruct. The memory-efficiency score is evaluated under context length of 4096 where batch-size is 20, and the throughput score is evaluated under context length of 161 where batch-size is 448. All three metrics are linearly rescaled so that higher values always indicate superior performance. For flexibility score, KIVI and OTT are restricted to a single precision, exhibiting the lowest flexibility. ZipCache allows the compression bit-width between 2 and 4, but the narrow span confines its flexibility to a moderate level. In contrast, both HqeKV and CAKE support a more wide compression ratios spectrum, endowing them with the highest flexibility. For user-friendliness score, ZipCache obliges the user to manually derive and balance precision ratios, rendering it the least user-friendly. Other methods accept a single global compression ratio as direct input. Consequently, their user-friendliness is regarded as uniformly high.

Figure 5 reveals that that HqeKV surpasses other methods since its largest area.

4.5 Ablation Study

To systematically evaluate how each proposed innovations contributes, we conducted an ablation study of Llama-3.1-8B-Instruct on LongBench for evaluation. The brief results are shown in Table 5 and, the descriptions of "Method" column in Table 5 are provided in Appendix A.7.1, and the detailed results are given in Appendix A.7.2.

Table 5 shows that removing any one of the four innovations degrades performance, while adding any one improves it, confirming that all HqeKV innovations are effective.

We also explored the performance of the model by the hyper-parameters α and β. We conduct an evaluation on HotpotQA from LongBench with the Llama-3.1-8B-Instruct model. Under the configuration ($\alpha = 2$, $\beta = 4$), the model achieves the highest score of 56.75, outperforming the alternatives: 54.76 ($\alpha = 2$, $\beta = 2$), 55.15 ($\alpha = 4$, $\beta = 2$), and 55.49 ($\alpha = 4$, $\beta = 4$). These results indicate that $\alpha = 2$ and $\beta = 4$ are the optimal hyper-parameters among the considered configurations.

Method	Brief Description	Avg
HqeKV-3.2	-	**49.91**
w/o I	w/o metric with range	49.78
w/o II	uniform quantization	41
w/o III	randomly non-opt ratio	49.8
w/o eviction	w/o eviction	49.47
III on w/o II	opt ratio on uni-quant	49.87
only I	metric with range	30.38
only II	specific quant strategies	49.81
only III	opt ratio allocation	46.34
w/o I, II & III	only quant-eviction	29.91

Table 5: Scores on LongBench in ablation study.

5 Related work

5.1 KV Cache Eviction

Researchers have proposed multiple methods to reduce the memory usage of KV Cache by eviction. H2O (Zhang et al., 2023) utilizes cumulative attention weights to determine which tokens should be retained or evicted. Scissorhands (Liu et al., 2023) retains tokens with a relatively high cumulative attention score over a period of time in the past. SnapKV (Li et al., 2024) retains tokens with high cumulative attention weights in each head and several recently tokens. StreamingLLM (Xiao et al., 2024) retains only the initial few tokens and the most recent few tokens. PyramidKV (Cai et al., 2024) adopts a pyramid-style allocation to different layers, allocating a larger budget to the early layers and a smaller budget to the final layers. CAKE (Qin et al., 2025) quantifies the saliency of each layer by jointly considering both spatial and temporal characteristics and allocates memory to each layer. MorphKV (Ghadia et al., 2025) evicts the token with the lowest importance. ThinK (Xu et al., 2025) retains the top-ranked channels selected by the query-dependent Frobenius norm.

5.2 KV Cache Quantization

KV Cache Quantization is an effective strategy to reduce memory use of KV Cache. Atom (Zhao et al., 2024) adopts uniform group quantization on KV Cache. KIVI (Liu et al., 2024) applies per-channel and per-token uniform group quantization on Key Cache and Value Cache. KVQuant (Hooper et al., 2024) uses the K-means clustering weighted by the fisher information matrix and quantizes Key Cache before RoPE, storing outliers separately in full precision. SKVQ (Duanmu et al., 2024) clusters channels of similar orders of magnitude together and quantizes KV Cache per-token. QJL (Zandieh et al., 2025) performs the JL Transform to the Key Cache and quantizes it to 1 bit. OTT (Su et al., 2025) stores outliers token separately and quantizes others like (Liu et al., 2024). SpinQuant (Liu et al., 2025) employs learnable Hadamard transformations to reduce outliers in the KV Cache. MiKV (Yang et al., 2024) quantizes tokens with less importance in low precision and proposes a per-channel balancer to reduce outliers. ZipCache (He et al., 2024) uses channel-separable tokenwise quantization to reduce the impact of outliers. QAQ (Cheng et al., 2025) determines quantization precision based on token-level standard deviation. Oaken (Kim et al., 2025) quantizes numbers near and far from zero to low precision and others to high precision. PQCache (Zhang et al., 2025) uses product quantization to compress KV Cache. KVTuner (Li et al., 2025) leverages Pareto frontier theory to assign layer-specific quantization levels. Cocktail (Tao et al., 2025) adopts the methods in RAG to evaluate the importance of KV Cache and quantizes them to different precision.

6 Conclusion

In this paper, we propose HqeKV, a novel KV Cache compression method that integrates mixed-precision quantization and eviction. We provide both theoretical and empirical evidence demonstrating a positive correlation between conversion losses and range. We propose a joint KV Cache importance metric that combines cumulative attention weights and token range of V. HqeKV features fine-grained quantization precision levels with specific quantization strategies. Moreover, HqeKV employs a search algorithm to determine the optimal precision ratio allocation. Extensive experiments show that HqeKV outperforms other state-of-the-art KV Cache compression methods.

7 Limitations

Despite its strong empirical performance, HqeKV is not without limitations. The offline search for optimal precision ratios, while optimized using TPE, remains agnostic to user-level or task-level heterogeneity, precluding true personalization. While normalized quantization mitigates accuracy loss at 1-bit precision, significant conversion loss still occurs at such extreme compression levels, highlighting the trade-off between bit-width and accuracy. The fixed re-quantization interval during decoding, though empirically set, introduces a sensitivity that could affect both latency and quality, especially in longer-context scenarios. Furthermore, its behavior under multi-modal architectures requires further research.

References

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. 2024. Pyramidkv: Dynamic KV cache compression based on pyramidal information funneling. *CoRR*, abs/2406.02069.

Zhihang Cai, Xingjun Zhang, Zhendong Tan, and Zheng Wei. 2025. Nqkv: A kv cache quantization scheme based on normal distribution characteristics. *arXiv preprint arXiv:2505.16210*.

Wen Cheng, Shichen Dong, Jiayu Qin, and Wei Wang. 2025. Qaq: Quality adaptive quantization for llm kv cache. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2542–2550.

Haojie Duanmu, Zhihang Yuan, Xiuhong Li, Jiangfei Duan, Xingcheng Zhang, and Dahua Lin. 2024. Skvq: Sliding-window key and value cache quantization for large language models. *arXiv preprint arXiv:2405.06219*.

Ravi Ghadia, Avinash Kumar, Gaurav Jain, Prashant J. Nair, and Poulami Das. 2025. Dialogue without limits: Constant-sized KV caches for extended response in LLMs. In *Forty-second International Conference on Machine Learning*.

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*.

Yefei He, Luoming Zhang, Weijia Wu, Jing Liu, Hong Zhou, and Bohan Zhuang. 2024. Zipcache: Accurate and efficient kv cache quantization with salient token identification. In *Advances in Neural Information Processing Systems*, volume 37, pages 68287–68307. Curran Associates, Inc.

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

Minsu Kim, Seongmin Hong, RyeoWook Ko, Soongyu Choi, Hunjong Lee, Junsoo Kim, Joo-Young Kim, and Jongse Park. 2025. Oaken: Fast and efficient llm serving with online-offline hybrid kv cache quantization. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture*, ISCA '25, page 482–497, New York, NY, USA. Association for Computing Machinery.

Xing Li, Zeyu XING, Yiming Li, Linping Qu, Hui-Ling Zhen, Yiwu Yao, Wulong Liu, Sinno Jialin Pan, and Mingxuan Yuan. 2025. KVTuner: Sensitivity-aware layer-wise mixed-precision KV cache quantization for efficient and nearly lossless LLM inference. In *Forty-second International Conference on Machine Learning*.

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. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. 2025. Spinquant: LLM quantization with learned rotations. In *The Thirteenth International Conference on Learning Representations*.

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 (Henry) Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi

Chen, and Xia Hu. 2024. Kivi: a tuning-free asymmetric 2bit quantization for kv cache. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Ziran Qin, Yuchen Cao, Mingbao Lin, Wen Hu, Shixuan Fan, Ke Cheng, Weiyao Lin, and Jianguo Li. 2025. CAKE: Cascading and adaptive KV cache eviction with layer preferences. In *The Thirteenth International Conference on Learning Representations*.

Yi Su, Yuechi Zhou, Quantong Qiu, Juntao Li, Qingrong Xia, Ping Li, Xinyu Duan, Zhefeng Wang, and Min Zhang. 2025. Accurate KV cache quantization with outlier tokens tracing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12895–12915, Vienna, Austria. Association for Computational Linguistics.

Wei Tao, Bin Zhang, Xiaoyang Qu, Jiguang Wan, and Jianzong Wang. 2025. Cocktail: Chunk-adaptive mixed-precision quantization for long-context llm inference. In *2025 Design, Automation & Test in Europe Conference (DATE)*, pages 1–7. IEEE.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

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. *CoRR*, abs/2402.18096.

Amir Zandieh, Majid Daliri, and Insu Han. 2025. Qjl: 1-bit quantized jl transform for kv cache quantization with zero overhead. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25805–25813.

Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. 2025. Pqcache: Product quantization-based kvcache for long context llm inference. *Proceedings of the ACM on Management of Data*, 3(3):1–30.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong 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.

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

A Appendix

A.1 Algorithm of Optimal Precision Ratio Allocation Search

The search process is shown in Algorithm 1.

Algorithm 1 Optimal Precision Ratio Search

Input: The average compression bit-width $B_{\text{avg}} \in (0, 4)$

Output: Optimal ratio list R_{opt}

initialize minimum loss $L \leftarrow$ Inf
initialize Optimal ratio list $R_{\text{opt}} \leftarrow \{\}$
initialize input S from calibration dataset
$F \leftarrow \text{LLM}(S)$ ▷ Output in full precision
for $iteration = 0$ **to** 200 **step** 1 **do**
  $x_3, x_4 \leftarrow \text{TPE_Sample}(0, 1)$
  $x_1 \leftarrow B_{\text{avg}} - 2 + x_3 + 2x_4$
  $x_2 \leftarrow 4 - B_{\text{avg}} - 3x_3 - 4x_4$
  $R \leftarrow \{x_1, x_2, x_3, x_4\}$
  if $x_1 < 0$ or $x_2 < 0$ **then**
    $L_{\text{temp}} \leftarrow$ Inf
  else
    $L_{\text{temp}} \leftarrow \mathcal{L}(F, \text{LLM}(S, R)) + \mathcal{L}_{\text{reg}}$
  end if
  if $L_{\text{temp}} < L$ **then**
    $L \leftarrow L_{\text{temp}}, R_{\text{opt}} \leftarrow R$
  end if
  $\text{TPE_Update}(L_{temp}, x_3, x_4)$
end for
return R_{opt}

A.2 Algorithms of KV Cache Compression

The compression algorithms of HqeKV in prefilling and decoding phase are shown in Algorithm 2 and 3. In the decoding phase, Algorithm 3 will be continuously executed until the inference is completed.

A.3 Transitions from Lower to Higher Precision

We conduct a detailed statistical analysis of the transitions in the KV Cache from lower precision to higher precision during the second compression phase. This analysis is performed while running the first question from the DuReader dataset in LongBench, using a mixed-quantization ratio of 0.05 for 4 bit-width, 0.9 for 2 bit-width, and 0.05 for 1 bit-width, on the Llama-3.1-8B-Instruct model. The results are shown in Table 6.

Algorithm 2 KV Cache Compression in Prefilling Phase

Input: $Q \in \mathbb{R}^{b \times h_Q \times s \times d}$, $K, V \in \mathbb{R}^{b \times h_{KV} \times s \times d}$, quantization group size G, number of recent tokens calculating cumulative attention weights m, Optimal ratio list R_{opt}

Output: Compressed KV Cache K', V', residual full precision KV Cache $K_{\text{full}}, V_{\text{full}}$,recent attention weights window $A \in \mathbb{R}^{m \times d}$, range of V $Range$

initialize $Q_{\text{r}} \in \mathbb{R}^{b \times h_Q \times m \times d}$ from Q.
initialize compression length $C \leftarrow \left\lfloor \frac{s}{G} \right\rfloor \times G$
initialize $K_{\text{c}} \in \mathbb{R}^{b \times h_{KV} \times C \times d}$ from K.
initialize $V_{\text{c}} \in \mathbb{R}^{b \times h_{KV} \times C \times d}$ from V.
initialize $K_{\text{full}} \in \mathbb{R}^{b \times h_{KV} \times (s-C) \times d}$ from K.
initialize $V_{\text{full}} \in \mathbb{R}^{b \times h_{KV} \times (s-C) \times d}$ from V.
initialize $Range \leftarrow \text{range}(V_{\text{c}})$.
$A \leftarrow \text{softmax}\left(\frac{Q_{\text{r}} K_{\text{c}}^T}{\sqrt{d}}\right)$
$A \leftarrow \text{sum}(A, \text{dim} = (0, 1))$
$Range \leftarrow \text{sum}(Range, \text{dim} = (0, 1))$
$\text{metric}_K \leftarrow \text{reshape}(A, (m, \left\lfloor \frac{s}{G} \right\rfloor, G))$
$Range \leftarrow \text{reshape}(Range, (\left\lfloor \frac{s}{G} \right\rfloor, G))$
$\text{metric}_K \leftarrow \text{average}(\text{metric}_K, \text{dim} = 2)$
$Range \leftarrow \text{average}(Range, \text{dim} = 2)$
$\text{metric}_K \leftarrow \text{cumulative}(\text{metric}_K, \text{dim} = 0)$
$\text{metric}_V \leftarrow \text{metric}_K \odot Range$
$K' \leftarrow \text{compress}(K_{\text{c}}, \text{metric}_K, R_{\text{opt}})$
$V' \leftarrow \text{compress}(V_{\text{c}}, \text{metric}_V, R_{\text{opt}})$
return $K', V', K_{\text{full}}, V_{\text{full}}, A, Range$

K or V	$1 \rightarrow$ 2-bit	$2 \rightarrow$ 4-bit	$4 \rightarrow$ FP16	Sum
K Cache	1.18%	2.16%	0	3.33%
V Cache	1.3%	2.09%	0	3.39%

Table 6: Transitions of KV Cache from lower precision to higher precision in decoding phase.

As shown in Table 6, only a minimal fraction of the KV Cache transitions from lower precision to higher precision. These low transition rates suggest that the occurrence of higher precision transitions can be effectively disregarded.

A.4 Detailed Hyper-Parameters Setup

For KIVI (Liu et al., 2024), we configure quantization group-size to 32 and residual length to 128. For OTT (Su et al., 2025), we set number of outlier tokens to 5, size of outlier pool to 32 and residual length to 96. For ZipCache (He et al., 2024), we set the saliency ratio to 0.6 and streaming gap to 128. For CAKE (Qin et al., 2025), we set window

size to 32 and $\tau_1 = 1.6$, $\tau_2 = 0.4$. For KVTuner (Li et al., 2025), we employ per-channel and per-token quantization for K and V, with a quantization group-size of 32, residual length of 128, and an average bit-width of 3.25. For our method, we set the re-quantization gap T and quantization group-size G to 32. In addition, we set the number of tokens calculating cumulative attention weights m to 20 and hyperparameter α to 2 and β to 4.

Algorithm 3 KV Cache Compression in Decoding Phase

Input: $Q \in \mathbb{R}^{b\times h_Q\times 1\times d}$, $K, V \in \mathbb{R}^{b\times h_{KV}\times 1\times d}$, Compressed KV Cache K', V', residual full precision KV Cache $K_{\text{full}}, V_{\text{full}}$,recent attention weights window $A \in \mathbb{R}^{m\times d}$, range of V $Range$, quantization group size G, re-quanztization gap T, Optimal ratio list R_{opt}

Output: Compressed KV Cache K', V', residual full precision KV Cache $K_{\text{full}}, V_{\text{full}}$,recent attention weights window $A \in \mathbb{R}^{m\times d}$

initialize $Range_{\text{new}} \in \mathbb{R}^{b\times h_{KV}\times 1} \leftarrow \text{range}(V)$.
$A_{\text{new}} \leftarrow \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)$
$A_{\text{new}} \leftarrow \text{sum}(A, \text{dim} = (0, 1))$
$A \leftarrow \text{update}(A, A_{\text{new}})$
$Range_{\text{new}} \leftarrow \text{sum}(Range_{\text{new}}, \text{dim} = (0, 1))$
$Range \leftarrow \text{update}(Range, Range_{\text{new}})$
$K_{\text{full}} \leftarrow \text{Concat}(K_{\text{full}}, K)$
$V_{\text{full}} \leftarrow \text{Concat}(V_{\text{full}}, V)$
$FullLength = \text{shape}(K_{\text{full}}, \text{dim} = 2)$
if $FullLength == T$ **then**
    $\text{metric}_K \leftarrow \text{reshape}(A, (m, \lfloor \frac{s}{G} \rfloor, G))$
    $Range \leftarrow \text{reshape}(Range, (\lfloor \frac{s}{G} \rfloor, G))$
    $\text{metric}_K \leftarrow \text{average}(\text{metric}_K, \text{dim} = 2)$
    $Range \leftarrow \text{average}(Range, \text{dim} = 2)$
    $\text{metric}_K \leftarrow \text{cumulative}(\text{metric}_K, \text{dim} = 0)$
    $\text{metric}_V \leftarrow \text{metric}_K \odot Range$
    $K' \leftarrow \text{compress}(K', K_{\text{full}}, \text{metric}_K, R_{\text{opt}})$
    $V' \leftarrow \text{compress}(V', V_{\text{full}}, \text{metric}_V, R_{\text{opt}})$
    $K_{\text{full}} \leftarrow \text{Null}$
    $V_{\text{full}} \leftarrow \text{Null}$
end if
return $K', V', K_{\text{full}}, V_{\text{full}}, A$

A.5 Additional Experiment Results

A.5.1 Detailed Evaluation Results on LongBench

The detailed evaluation results on LongBench (Bai et al., 2024) with quantization baselines are shown in Table 8. It should be particularly noted that the "Avg" scores for ZipCache-2.8, ZipCache-3.2 and HqeKV-3.2 in Table 8 are calculated from non-OOM columns. In addition, the "Avg" scores for KVTuner-3.25 and HqeKV-3.25 in Table 8 are calculated from the columns that do not experience CUDA error in KVTuner.

A.5.2 Comparison with Eviction Methods

The evaluation results of HqeKV with CAKE (Qin et al., 2025) are shown in Table 9.

A.6 Detailed Compression Ratios

Methods	4-bit	2-bit	1-bit	eviction
Llama-3.1-8B-Instruct				
HqeKV-2	0.0315	0.922	0.0299	0.0165
HqeKV-2.8	0.416	0.568	0.0012	0.0153
HqeKV-3.2	0.637	0.307	0.037	0.0186
HqeKV-3.25	0.679	0.225	0.0847	0.0116
Qwen3-8B				
HqeKV-2	0.164	0.672	0.0006	0.164
HqeKV-2.8	0.659	0.081	0.001	0.259
HqeKV-3.2	0.778	0.0431	0.0009	0.178

Table 7: Detailed compression ratios.

A.7 Detailed Ablation Study

A.7.1 Detailed Ablation Study Methods Descriptions

Detailed descriptions of ablation study are shown in Table 10.

A.7.2 Detailed Ablation Study Results

Detailed results of ablation study are shown in Table 11.

Methods	Single-Document QA				Multi-Document QA				Summarization			
	NQA	Qasper	MFQA-en	MFQA-zh	HPQA	2WMHQA	MSQ	DR	GR	QMSum	MN	VCSUM
Llama-3.1-8B-Instruct												
FP16	29.12	45.26	55.18	63.11	53.83	46.51	29.62	31.84	34.81	25.44	27.16	17.35
KIVI-2	28.19	45.2	52.86	62.61	53.78	43.82	29.51	32.03	34.95	24.87	26.79	16.98
OTT-2	28.77	44.41	53.47	63.04	52.51	44.44	31.63	31.57	35.1	24.41	27.14	17.16
HqeKV-2	**28.86**	43.1	**53.98**	62.05	53.77	**45.4**	29.77	31.53	35	24.67	**26.81**	16.8
ZipCache-3.2	OOM	42.59	53.92	60.34	54.4	45.19	30.61	29.1	OOM	24.19	26.68	17.31
HqeKV-2.8	28.51	**46.54**	54.13	61.44	53.02	**44.04**	29.46	**34.17**	34.84	24.55	26.75	17.27
HqeKV-3.2	**28.62**	45.43	**55.13**	**62.27**	**56.75**	42.59	**30.9**	33.49	**34.59**	**25.11**	**26.85**	16.68
KVTuner-3.25	29.61	43.75	53.41	61.96	56.87	44.19	28.11	ERROR	34.51	24.72	26.64	17.25
HqeKV-3.25	**30.73**	**44.07**	**54.39**	60.89	55.97	**45.95**	**31.73**	**32.51**	**34.91**	**25.02**	**26.73**	**17.66**
Qwen3-8B												
FP16	26.16	47.17	54.37	62.67	59.37	42.43	33.11	27.16	33.17	23.91	24.68	13.84
KIVI-2	22.79	46.51	54.06	62.86	60.05	42.04	33.48	27.23	32.82	23.99	24.56	13.91
OTT-2	24.58	46.8	53.96	61.57	60.07	44.47	33.83	27.03	32.99	23.78	24.8	13.97
HqeKV-2	24.54	46.75	53.5	62.75	**61.76**	44.36	33.12	**27.5**	**33.28**	23.8	24.78	**14.13**
ZipCache-3.2	OOM	46.71	51.21	62.14	58.83	43.37	33.53	24.86	OOM	22.93	24.19	14.02
HqeKV-2.8	24.77	47.69	54.17	**62.78**	61.53	41.67	33.07	**27.88**	33.15	24.19	**24.66**	14
HqeKV-3.2	**24.94**	**47.79**	**54.2**	62.18	**59.19**	41.94	**34.37**	26.98	**33.29**	**24.23**	24.4	13.84

Methods	Few-shot Learning				Synthetic Task			Code Completion		Avg
	TREC	TriviaQA	SAMSum	LSHT	PC	PR-en	PR-zh	LCC	RB-P	
Llama-3.1-8B-Instruct										
FP16	73	90.9	43.62	45.5	7.59	99.5	93.35	59.8	53.38	49.76
KIVI-2	73.5	90.92	43.22	44.5	7.09	99	90.06	58.96	51.47	48.96
OTT-2	73.5	89.09	44.1	45	4.43	99.5	88.09	60.95	55.03	49.43
HqeKV-2	72.5	**91.17**	44.1	44.5	7.67	**100**	**91.53**	**61.94**	54.39	**49.71**
ZipCache-3.2	71.5	91.45	43.02	OOM	5.94	99.5	94.74	61.32	OOM	50.98
HqeKV-2.8	70	90.42	44.04	45.5	7.34	99.5	94.58	60.77	**54.83**	51.26
HqeKV-3.2	70	89.12	**44.16**	**46.5**	**8.67**	99.5	93.72	61.25	54.4	**51.49**
KVTuner-3.25	73	88.27	42.31	46.75	4.77	99.5	96.67	60.91	ERROR	49.94
HqeKV-3.25	71.5	**89.18**	**44.01**	46.75	5.55	99.5	93.47	**61.91**	**54.12**	**50.33**
Qwen3-8B										
FP16	73	89.32	44.4	45.75	1	100	98.5	67.52	64.69	51.24
KIVI-2	73.5	89.64	44.71	47	2.5	98	91.17	65.75	61.48	50.36
OTT-2	72	89.66	44.68	46	4.33	99.5	94.17	67.23	62.17	50.75
HqeKV-2	72	**90.36**	**44.85**	47	2.42	**100**	**97.5**	**68.53**	**63.12**	**51.37**
ZipCache-3.2	70	89.85	44.25	46	1	100	98.5	66.93	OOM	51.22
HqeKV-2.8	69	88.72	**44.77**	46	**3.75**	100	98.5	**68.72**	63.73	**51.98**
HqeKV-3.2	**72.5**	89.07	44.63	46	3	100	98.5	67.88	**64.41**	51.89

Table 8: Inference scores of Llama-3.1-8B-Instruct and Qwen3-8B on LongBench.

Methods	NQA	Qasper	MFQA-en	MFQA-zh	HPQA	2WMHQA	MSQ	DR	GR	QMSum	MN	VCSUM	TREC	TriviaQA	SAMSum	LSHT	PC	PR-en	PR-zh	LCC	RB-P	Avg
Llama-3.1-8B-Instruct																						
CAKE-3.2	28.65	12.97	26.04	19.24	19.13	14.44	11.01	24.54	29.91	22.26	24.41	15.24	68.5	89.78	40.98	45.5	5.17	96.16	75.34	61.09	54.11	40.03
HqeKV-3.2	28.62	45.43	55.13	62.27	56.75	42.59	30.9	33.49	34.59	25.11	26.85	16.68	70	89.12	44.16	46.5	8.67	99.5	93.72	61.25	54.4	**49.91**
CAKE-3.25	28.65	12.94	26.69	19.53	18.7	17.09	12.68	25.81	29.52	22.3	24.06	15.31	64.5	91.72	40.9	44.5	7.51	95.04	78.96	61.67	55.5	40.53
HqeKV-3.25	30.73	44.07	54.39	60.89	55.97	45.95	31.73	32.51	34.91	25.02	26.73	17.66	71.5	89.18	44.01	46.75	5.55	99.5	93.47	61.91	54.12	**49.98**
Qwen3-8B																						
CAKE-3.2	26.97	46.69	53.87	62.42	60.22	42.53	35.07	25.01	30.69	23.8	20.71	12.83	71	89.23	42.27	46	1	100	98	66.4	63.02	50.45
HqeKV-3.2	24.94	47.79	54.2	62.18	59.19	41.94	34.37	26.98	33.29	24.23	24.4	13.84	72.5	89.07	44.63	46	3	100	98.5	67.88	64.41	**51.29**

Table 9: Comparison with CAKE on LongBench based on Llama-3.1-8B-Instruct and Qwen3-8B.

Method	Description
w/o I	opt ratio and specific quantization strategy and metric with range, w/o metric with range
w/o II	uniform quantization to 4, 2 and 1 bit-widths base on the opt-ratio from specific quantization strategy and metric with range
w/o III	randomly selected non-opt ratio allocation on specific quantization strategy and metric with range
w/o eviction	opt ratio on 4, 2 and 1 bit-width quantization, specific quantization strategy and metric with range
III on w/o II	opt ratio on uniform quantization to 4, 2 and 1 bit-widths base on the opt-ratio from specific quantization strategy and metric with range
only I	randomly selected non-opt ratio allocation on uniform quantization to 4, 2 and 1 bit-widths and metric with range
only II	randomly selected non-opt ratio allocation on specific quantization strategies, w/o metric with range
only III	opt ratio allocation on 4, 2 and 1 bit-width uniform quantization and eviction, w/o metric with range
w/o I, II & III	randomly selected non-opt ratio allocation, w/o specific quantization strategies and w/o metric with range

Table 10: Detailed description of methods in ablation study.

Methods	NQA	Qasper	MFQA-en	MFQA-zh	HPQA	2WMHQA	MSQ	DR	GR	QMSum	MN	VCSUM	TREC	TriviaQA	SAMSum	LSHT	PC	PR-en	PR-zh	LCC	RB-P	Avg
HqeKV-3.2 bit	28.62	45.43	55.13	62.27	56.75	42.59	30.9	33.49	34.59	25.11	26.85	16.68	70	89.12	44.16	46.5	8.67	99.5	93.72	61.25	54.4	49.91
w/o innovation I	28.81	44.2	53.97	61.76	51.57	43.15	31.19	33.23	34.81	24.29	26.82	17.18	71	86.04	44.23	47.5	4.97	99.5	94.49	62.5	56.31	49.78
w/o innovation II	24.37	27.56	42.35	48.73	48.75	36	24.92	12.83	18.74	17.73	18.04	10.99	53.5	90.53	39.06	30	4.72	99.5	94.86	51.07	45.37	41
w/o innovation III	29.11	43.83	53.25	60.41	55.71	44.64	29.8	31.42	34.8	25.29	26.5	16.62	72.5	90.39	43.94	46	6.78	99.5	93.81	61.69	55.05	49.8
w/o eviction	29.62	44.63	55.17	59.68	54.39	44.42	28.24	31.75	34.33	24.93	26.3	16.66	68	90.85	43.99	45.5	9.08	99.5	91.87	60.07	55.87	49.47
innovation III on w/o II	27.98	46.26	52.91	62.22	52.82	45.1	29.8	31.87	34.5	24.73	26.53	16.91	73.5	89.79	43.94	45.5	6.22	99.5	93.46	62.12	55.5	49.87
innovation I	16.99	17.47	24.37	27.41	35.97	18.38	13.42	9.02	18.49	14.37	15.52	10.39	45.5	86.33	26.2	23.5	7.56	98.29	37.08	37.02	35.5	30.38
innovation II	29.26	43.97	53.37	60.08	56.62	44.02	28.52	32.27	34.73	25.23	27.09	17.29	73	91.49	43.87	45	5.67	99.5	93.14	62.21	54.72	49.81
innovation III	27.91	40.25	49.96	55.67	53.11	44.51	29.21	21.03	22.06	20.97	21.41	11.6	64	87.45	43.37	41.5	6.44	99.5	95.62	58.33	52.69	46.34
w/o innovation I, II & III	16.33	16.38	23.37	27.02	32.73	20.88	14.79	8.59	18.64	14.42	14.95	10.22	41.75	84.29	25.88	26.75	5.67	99	34.3	37.34	34.73	29.91

Table 11: Ablation study results on LongBench of Llama-3.1-8B-Instruct.