Highlights

Enabling Efficient Low-bit Quantization based on Matrix Product Operators for KV Cache Compression

Jia-Qi Wang,Xiao-Qi Han,Peng-Jie Guo,Rong-Qiang He,Ze-Feng Gao,Zhong-Yi Lu

- This paper proposes a KV cache compression method, MPOQ, based on MPO decomposition for accelerating inference in large language models.
- As a post-training quantization method, MPOQ addresses quantization challenges by transferring them from the original matrix to local tensors, achieving a high KV cache compression rate with minimal accuracy loss.
- MPOQ offers several advantages, including the absence of data, training, or calibration requirements, as well as flexible quantization options that support separate quantization of weights, separate quantization of activations, or simultaneous quantization of both.
- MPOQ can achieve 4-bit KV cache quantization while maintaining comparable generation quality.

Enabling Efficient Low-bit Quantization based on Matrix Product Operators for KV Cache Compression[⋆,⋆⋆]

Jia-Qi Wang, Xiao-Qi Han, Peng-Jie Guo, Rong-Qiang He, Ze-Feng Gao[*] and Zhong-Yi Lu[*]

School of Physics, Renmin University of China, Beijing, China

ARTICLE INFO

Keywords:
Large Language Models
Quantization
Matrix Decomposition
KV Cache

ABSTRACT

Large language models (LLMs), despite their remarkable successes, remain significantly expensive to implement. Among various strategies, key-value cache (KV cache) stands out as a crucial technique for expediting the inference of LLMs, yet it comes with substantial memory costs. To reduce the KV cache size, conventional methods often sacrifice accuracy or require additional data for calibration, which restricts their feasibility in real-world LLMs applications. Here, we introduce MPOQ, a novel data-free quantization technique based on matrix product operators (MPO) to effectively compress the KV cache. The MPO can decompose the original matrix into a series of local tensors, effectively transferring the quantization challenges from the original matrix to these local tensors, thereby allowing us to adjust the distribution of outliers within the original matrix. Specifically, we have discovered that outliers are predominantly concentrated in smaller local tensors, whereas larger tensors exhibit a more constrained value range. Leveraging this insight, we propose a strategy that employs low-bit quantization for the large tensor while preserving a high-precision representation for the smaller tensor. Extensive experiments based on OPT, LLaMA and Mistral demonstrate the effectiveness of our method in improving both the performance and efficiency of LLMs (∼75% reduction in memory footprint while maintaining comparable generation quality).

1. Introduction

Large Language Models (LLMs) (such as GPT (Brown et al., 2020), OPT (Zhang et al., 2022) and LLaMA (Touvron et al., 2023)) have exhibited remarkable potential in driving advancements in the field of language intelligence. These sophisticated models can comprehend and produce intricate linguistic structures, delivering impressive results across a multitude of language-related tasks. However, the substantial size of these models often results in increased inference latency, presenting significant obstacles in real-world implementation. This extended latency implies that models take longer to process requests, potentially degrading user experience, particularly in scenarios that demand swift responses. Therefore, it is urgent to reduce the running cost of LLMs.

To enhance the performance of LLMs in the inference phase, a widely utilized strategy involves the implementation of key-value cache (KV cache), as demonstrated by Pope et al. (2023). Inference in LLMs follows an auto-regressive approach, where each step generates a token based on the previous steps. During each step, the key-value embeddings from the attention mechanism are stored in memory to avoid redundant key-value projection calculations in future steps. Unfortunately, the memory required for the KV cache, which includes prompts and previously generated tokens, can be surprisingly large [1]. Moreover, the substantial computational requirements of LLMs necessitate more substantial hardware resources, such as GPUs and TPUs. This not only escalates the costs associated with deployment but may also contribute to a rise in energy consumption. Furthermore, the administration of extensive cache systems typically involves a high volume of I/O read and write operations, which can result in significant delays. The challenge is exacerbated when these operations are required to stretch across several machines (Patel et al., 2024). Consequently, compressing the KV cache is crucial for enhancing inference efficiency of LLMs.

[⋆]This work was financially supported by the National Natural Science Foundation of China (Grant No.62476278, No.11934020, and No.12204533) and the National Key R&D Program of China (Grants No. 2024YFA1408601).

[⋆⋆]The code and data has been released in https://github.com/Chihiro1099/MPOQ

[*]Corresponding author

zfgao@ruc.edu.cn (Z. Gao); zlu@ruc.edu.cn (Z. Lu)

ORCID(s):

[1]OPT-175B consume approximately 325 GB of memory, with a batch size of 128 and a sequence length of 2048, the KV cache requires around 950 GB of memory. Considering that eight Nvidia A100-80GB GPUs provide a total of 640 GB of GPU memory, the memory usage for the KV cache is indeed a cause for concern.

Considering the issues previously discussed, several researches have been conducted on quantization and sparsity techniques for the KV cache in LLMs to minimize the computational cost during inference (Yao et al., 2022a; Park et al., 2022; Frantar and Alistarh, 2023; Bansal et al., 2022). With a fixed inference sequence length, the reduction in KV cache memory usage can directly correspond to an expanded batch size. An increase in batch size is particularly beneficial for systems designed for high-volume inference (Pope et al., 2023; Sheng et al., 2023; Chevalier et al., 2023). Zhang et al. (2024); Liu et al. (2024f); Mu et al. (2023) suggest using pruning to keep the KV cache compact. This method, while reducing memory consumption, might cause the loss of information when generating lengthy texts. Dettmers et al. (2022b) discovered during the quantization of the OPT model that when the parameter count exceeds 6.7 billion, a significant number of outliers begin to appear in the activation values. These outliers are approximately 100 times larger than most of the activation values. Additionally, post-quantization techniques, despite preserving the entire text history, often suffer from a decline in performance due to the quantization of activation values at low bit levels (Dettmers et al., 2022a). Moreover, many existing quantization techniques still rely on calibration or retraining (Frantar et al., 2022; Xiao et al., 2023) to maintain model performance, which limits their applicability in deployment scenarios requiring data-free KV cache compression.

To address this issue, we developed a novel approach that integrated matrix product operator (MPO) and quantization, called **MPOQ**, which aims to mitigate the impact of outliers on low-bit quantization. The MPO is an algorithm that factorized a matrix into a sequential product of local tensors, which have potential to adjust the distribution of outliers in the original matrix, making the resulting local tensors which are more suitable for quantization (Liu et al., 2024b). By applying MPO directly to the activation values, we can distribute the quantization difficulty across multiple tensors, while separately quantize the weights. This MPO decomposition shifts the complexity of activation quantization without affecting the weights. Compared to SmoothQuant (Xiao et al., 2023), which transfers quantization difficulty from activation to weights, this method avoids any impact on weight precision by directly operating on the activation themselves.

Note that MPOQ does not require retraining the original model or introducing significant computational overhead. It operates without relying on back propagation or reconstruction, thereby preserving the generalization ability of LLMs across different domains and modalities without over fitting to a calibration set. By applying MPO decomposition to activation and weights separately, MPOQ effectively distributes the quantization difficulty across multiple matrices, transferring the complexity of quantize activation into more manageable components.

Additionally, different datasets and models may influence the testing results. To thoroughly evaluate the quantization performance of MPOQ, we conducted separate quantization and testing on models before and after fine-tuning. To demonstrate the efficacy of our proposed MPOQ, we conduct extensive experiments on OPT (Zhang et al., 2022), LLaMA (Touvron et al., 2023) and Mistral (Zhao et al., 2023a). The results indicate that MPOQ can achieve 4-bit quantization for the KV cache and 8-bit quantization for weights and activation with almost no loss in performance. This suggests that the MPOQ serves as an effective post-training quantization approach, ensuring both accuracy and efficient inference. Consequently, MPOQ provides a viable quantization solution for LLMs, contributing to the compression of KV caches and accelerating the inference speed of LLMs.

2. Related Work

Quantization for LLMs. The quantization method not only effectively reduces the memory usage during the inference process of large models but also decreases inference latency, thereby accelerating inference speed. However, the presence of outlier activations and massive activations makes the quantization of activation values particularly challenging. To address this issue, Dettmers et al. (2022b) mitigated the difficulties of quantizing outliers by transferring them to the weights. While this approach is effective, the optimal smoothing factor for different models and tasks require careful tuning. Atom (Zhao et al., 2024) combined various quantization techniques, including mixed-precision quantization, dynamic activation quantization, and KV cache quantization. However, achieving optimal performance and accuracy may necessitate meticulous adjustments of parameters within Atom, such as quantization bit width and group size. Many quantization methods involve parameters that must be handled with care, making practical application difficult, especially in data-free calibration scenarios where pre-tuning is not feasible. In this regard, our proposed MPOQ method directly addresses activation quantization without calibration.

KV cache compression. Compressing the KV cache during LLMs inference can not only reduce cache pressure but also decrease the frequency of I/O operations, thereby lowering deployment requirements. However, the presence of outlier activations and massive activations significantly complicates the quantization process, making it challenging to simultaneously maintain model accuracy and achieve higher compression rates. One prominent research direction to address this challenge involves retaining key high-precision information while applying low-bit quantization to less critical data. RazorAttention (Tang et al., 2024) retained the full cache for retrieval heads, discarded distant tokens in non-retrieval heads, and introduced a "compensation token" mechanism to recover information from discarded tokens. MiniCache (Liu et al., 2024a) observed that KV cache states between adjacent layers in mid-to-deep layers were highly similar. It selectively preserved highly similar states that carried unique semantic information, though fine-tuned interpolation hyperparameters and retention thresholds was necessary to balance performance and efficiency. GEAR (Kang et al., 2024) first quantified most entries with similar magnitudes to ultra-low precision, then used a low-rank matrix to approximate quantization errors and a sparse matrix to correct individual errors for outlier entries. MiKV (Yang et al., 2024) adopt an importance-aware mixed-precision quantization strategy, retained evicted KV pairs in low precision while kept important KV pairs in high precision. However, simply removing tokens deemed unimportant for future context based on current information might not be ideal. Selecting the appropriate tokens for different models and tasks often requires manual tuning, and these operations introduce additional computational overhead. In contrast to other methods, our approach retains all tokens and ensures full context integrity, while the added computational overhead remains simple and manageable.

KV cache and communication management. The bottlenecks limiting LLMs inference speed come from two main factors: the large size of the KV cache and the frequent I/O operations required for reading and writing the cache. Metho (Kwon et al., 2023) used paged management of the KV cache, which improved memory efficiency and allowed for cache sharing between requests. InfiniGen (Lee et al., 2024), tailored for long-text generation, predicted key tokens to minimize CPU-to-GPU cache transfers, accelerated inference while dynamically selecting critical cache entries to reduce memory usage. Similarly, CacheGen (Liu et al., 2024e) lowered CPU-to-GPU cache transfer costs. These methods address KV cache management from the perspectives of memory and data transfer, and in principle, they can be used alongside KV cache compression techniques to further enhance inference efficiency. Our proposed MPOQ method directly handles activation quantization without requiring calibration, making it convenient for integration with these existing approaches. This compatibility can comprehensively improve the inference performance of LLMs, especially when deployed on resource-constrained terminals.

3. Preliminary

In this section, we introduce the background on LLMs quantization, the inference process, KV cache quantization, and the challenges in quantization caused by outliers.

3.1. LLMs quantization

Quantization is the process of mapping high-precision data in models, such as FP32 and FP16, to lower-precision data formats like INT8, INT4, or even INT2. This quantization operation significantly reduces the storage requirements for data, thereby lowering the inference costs of the model (Lin et al., 2023; Frantar et al., 2022; Dettmers et al., 2022b). The quantization process can be expressed as follows:

$$\bar{\mathrm{M}}_{\mathrm{INTx}} = \lceil \frac{\mathrm{M}_{\mathrm{FP16}}}{\Delta} \rfloor, \quad \Delta = \frac{\max(|\mathrm{M}|)}{2^{\mathrm{x}-1} - 1}, \tag{1}$$

In which, $\mathrm{M}_{\mathrm{FP16}}$ represents the floating-point tensor to be quantized, while $\bar{\mathrm{M}}_{\mathrm{INTx}}$ denotes its corresponding quantized version. Δ is the quantization step size, $\lceil\cdot\rfloor$ is the rounding function, and x indicates the number of bits. Here, we employ symmetric quantization. During the quantization process, the maximum absolute value is used to compute Δ, establishing a mapping from high precision to low precision for the activations. There are two methods for calculating Δ: static quantization, which computes it offline using activations from calibration samples, and dynamic quantization, which derives Δ using runtime statistics of the activations. The dynamic approach, which is used in this work, avoids the need for a calibration set, making it particularly advantageous in data-sensitive scenarios, such as those involving security and privacy concerns.

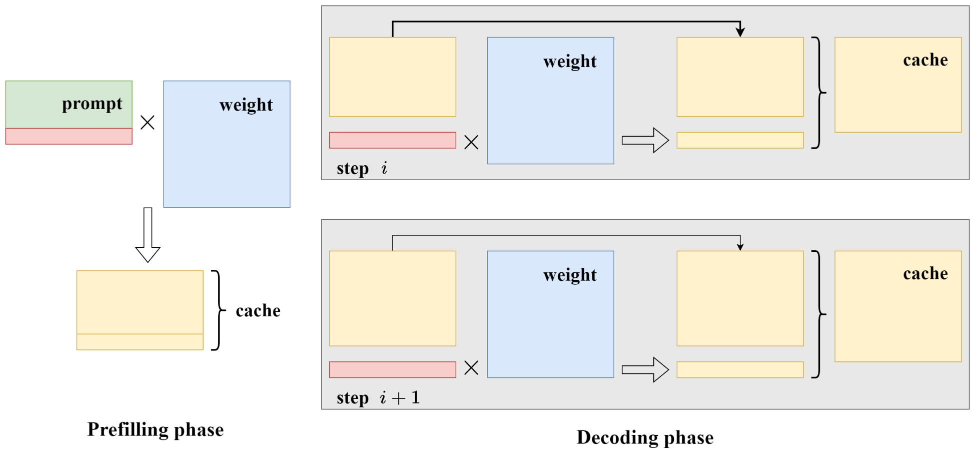


Figure 1: The KV cache technology of LLMs. The inference process of LLMs is divided into two parts: the prefilling phase and the decoding phase. In the prefilling phase, the model reads the prompt and generates the first token, storing the associated keys and values in the cache. During the decoding phase, the model generates new tokens one by one through autoregression. By storing the attention keys and value tensors related to each layer's historical tokens, the KV cache technology eliminates redundant computations of historical text, thereby accelerating the model's inference speed.

3.2. LLMS inference and KV cache

During the inference of LLMs, the task prompt is first input into the model to generate the first token, after which the remaining tokens are generated in an autoregressive manner (Zhao et al., 2023b; Zhong et al., 2024). The token-by-token text generation characteristic of LLMs results in significant computational redundancy during the inference process. To address this issue, researchers have introduced the KV cache technique (Pope et al., 2022). This method effectively avoids redundant calculations of historical text by storing previously generated historical tokens. Figure 1 illustrates how the KV cache technology prevents redundant calculations. The yellow blocks represent historical tokens stored in the cache, while each new token generated through autoregressive computation will also be added to the cache. This visualization effectively demonstrates the operational mechanism of KV cache and its crucial role in enhancing inference efficiency. This optimization not only significantly speeds up the model's inference but also allows for more efficient utilization of computational resources during the generation process, thereby improving overall performance.

Although the KV cache technique has made significant strides in reducing redundant computations, the parameter size of LLMs remains enormous. As the batch size and input text length increase, the memory overhead of the KV cache expands rapidly, often surpassing the memory requirements of the model itself. This substantial memory demand significantly complicates the practical deployment of LLMs in resource-constrained environments, such as mobile devices or edge computing scenarios. To address this challenge, researchers are focusing on developing more efficient cache management and quantization techniques. These advancements aim to optimize memory usage and improve the overall efficiency of LLMs inference, thereby enabling effective deployment in diverse settings while maintaining performance.

3.3. Challenges of activation quantization

The quantization of LLMs primarily involves weights and activation. While the distribution of weights tends to be relatively uniform and flat, making them easier to quantize, the quantization of activations is more challenging due to the presence of outliers and massive activations.

For example, as for the OPT model, Dettmers et al. (2022b) discovered that activations can contain outlier anomalies that are approximately 100 times larger than the majority of activations. Recently, research (Sun et al., 2024) in the LLaMA2 model revealed that some activations exhibit an extraordinary magnitude, being over four orders of magnitude larger than the median. These massive activations typically occur in a small number of feature dimensions that are independent of the input and often appear suddenly; they emerge after the computation of a single layer and tend to disappear in the final layers. This suggests that massive activations are distinct from typical outlier features and are relatively difficult to capture.

In practice, setting an appropriate value for the quantization step size Δ for activations is challenging. When such outlier anomalies exist, INT4 quantization can map numerous benign activations to the same value. At this point, the quantized activations cannot be effectively distinguished, making it challenging to accurately recover them during the dequantization inference process, which consequently leads to a rapid decline in model performance. Consequently, the presence of outliers complicates the quantization of activations, resulting in degraded accuracy for conventional quantization methods such as ZeroQuant (Yao et al., 2022b).

4. Methods

In this section, we introduce an effective quantization method based on MPO decomposition, dubbed MPOQ, designed to separate outliers in activations, reduce quantization errors, and utilize this method for quantizing the KV cache. During LLM inference, this method helps reduce storage requirements while minimizing the loss of inference accuracy after dequantization.

4.1. MPO decomposition and outlier distribution

We utilize the MPO (Gao et al., 2020a) to decompose matrices, aiming to adjust the distribution of outliers within the original matrix to alleviate quantization challenges. MPO can be viewed as a form of tensor decomposition, allowing the matrix to be broken down into a sequence of local tensor products (Rabanser et al., 2017; Kolda and Bader, 2009). The MPO decomposition of a matrix $\mathrm{M} \in \mathbb{R}^{I\times J}$ into a product of n local tensors can be represented as

$$\mathrm{MPO(M)} = \prod_{k=1}^{n} \mathcal{T}_{(k)}[d_{k-1}, i_k, j_k, d_k], \tag{2}$$

where $\mathcal{T}$ denotes the local tensor with size $d_{k-1} \times i_k \times j_k \times d_k$ in which $\prod_{k=1}^{n} i_k = I, \prod_{k=1}^{n} j_k = J$. Here, we set $n = 2$ and use biased decomposition to break the matrix into two tensors. The tensor with more parameters is designated as the central tensor $\mathcal{T}_C$, while the other tensor with fewer parameters is referred to as tensor $\mathcal{T}_O$. This can be expressed as MPO (M) $= \mathcal{T}_C \times \mathcal{T}_O$. According to Liu et al. (2021); Gao et al. (2020b), matrix decomposition can potentially adjust the outlier distribution of the original matrix. MPO shares this property, making the decomposed local tensors easier to quantize. Given that outliers in activations are relatively sparse, we adjust the decomposition ratio of $\mathcal{T}_C$ and $\mathcal{T}_O$, increasing the central tensor's proportion to about 99.4%, thereby ensuring it retains most of the original matrix's information.

We examine the outlier distributions in the original matrix M and the two tensors $\mathcal{T}_C$ and $\mathcal{T}_O$ post-MPO decomposition. As shown in Figure 2, the value distributions of the decomposed tensors are narrower than that of the original tensor, with $\mathcal{T}_C$ showing a significantly narrower distribution compared to both the original matrix and the other tensor $\mathcal{T}_O$. This indicates a reduction in outlier values within $\mathcal{T}_C$, effectively adjusting the outliers of the original matrix, which is beneficial for low-bit quantization. For the tensor $\mathcal{T}_O$, due to its relatively low parameter count, we can opt to maintain its original precision without quantization.

4.2. MPOQ method

Based on the above observations, we propose the MPOQ method and apply it to the inference of LLms. As illustrated in Figure 3, MPOQ performs MPO decomposition on the matrix to be quantized, M, yielding $\mathcal{T}_C$. The central tensor $\mathcal{T}_C$ undergoes low-bit quantization to produce $\mathcal{T}_C'$, while $\mathcal{T}_O$ remains unchanged. During the dequantization phase, $\mathcal{T}_C'$ is dequantized to obtain $\mathcal{T}_C''$, which is then combined with $\mathcal{T}_O$ to reconstruct the matrix M'.

The model inference process consists of two main stages: prefilling phase and decoding phase. When applying MPOQ to the KV cache in LLMs inference, specific adjustments are necessary. In the prefilling phase, the model reads the prompt and performs a forward pass to generate the first token. At this point, MPOQ processes and caches the intermediate layer keys and values, reducing initial cache pressure and enhancing inference efficiency. During the decoding phase, the model reads the historical KV cache and generates tokens one by one while producing new keys and values. As the sequence length increases, the KV cache grows linearly. To avoid the additional workload from frequent matrix decomposition and quantization, MPOQ executes MPO decomposition and quantization when the cache reaches a certain length (*e.g.,* 500).

In summary, based on the adjustment of outliers through MPO decomposition operations, we introduce a KV cache quantization method that leverages MPO decomposition. During the quantization phase, low-precision quantization is

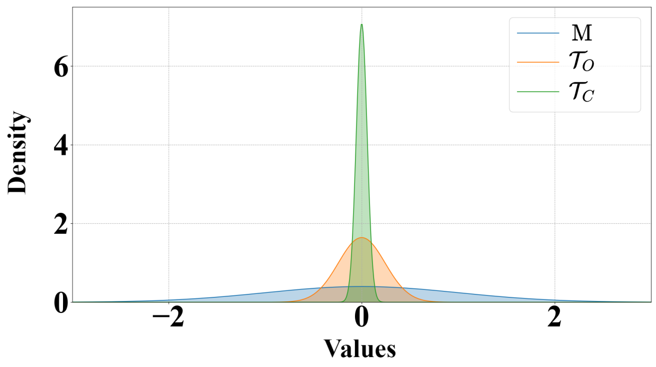


Figure 2: Outlier distributions of local matrices. After performing MPO decomposition, the distribution of values in the central tensor $\mathcal{T}_C$ is significantly narrowed compared to the original matrix M.

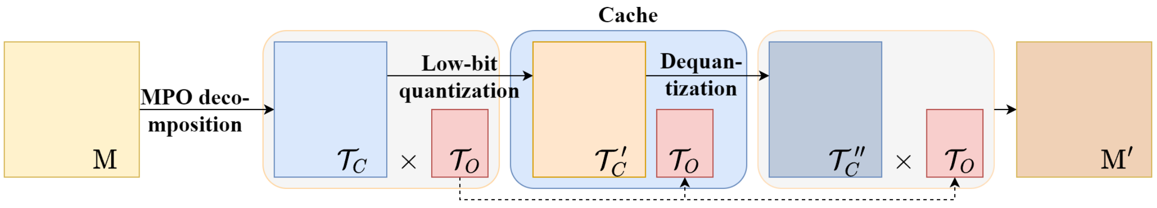


Figure 3: The quantization and dequantization process of MPOQ. During the quantization phase, the matrix M is decomposed using MPO to obtain the central tensor $\mathcal{T}_C$ and the other tensor $\mathcal{T}_O$. The tensor $\mathcal{T}_O$ remains unchanged, while the central tensor $\mathcal{T}_C$ undergoes low-bit quantization to produce the quantized tensor $\mathcal{T}_C'$, which is then stored in the cache along with $\mathcal{T}_O$. In the dequantization phase, $\mathcal{T}_C'$ is read from the cache and dequantized to obtain tensor $\mathcal{T}_C''$. By combining $\mathcal{T}_C''$ with $\mathcal{T}_O$, the matrix M′ is reconstructed. In the MPOQ setup, the central tensor $\mathcal{T}_C$ constitutes a significant portion of the overall representation (approximately 99.4%), allowing MPOQ to achieve a high compression rate while maintaining the accuracy of the dequantized results.

applied to the central tensor, while the other tensor retains its high-precision representation. In the dequantization phase, the original matrix is reconstructed, and the quality of this reconstruction is directly related to the quantization error. This quantization method is designed for the inference process and can be applied to the model's weights or activation values. Importantly, it imposes no requirements on the model or the training dataset, achieving a data-free post-training quantization method.

5. Experiments

Here, we evaluate the performance of MPOQ on language modeling tasks before and after model fine-tuning. Following this, we test its learning and adaptation capabilities in the Open-Ended Document Generation task under different sample sizes. Similar to the setup by Xiao et al. (2023), we quantize the weights and activations to various bit precisions (2/4/8/16 bits) and use accuracy as the evaluation metric.

5.1. Experimental setup

Baseline. We compare MPOQ with mainstream quantization methods (Round-to-nearest (RTN, (Lin et al., 2023)) and SmoothQuant (Xiao et al., 2023)) for KV cache compression. The RTN method maps real values to integer values through a straightforward rounding operation, which offers simplicity in implementation and computational efficiency. SmoothQuant achieves quantization by smoothing out activation outliers into weights, demonstrating excellent performance when dealing with unevenly distributed activation values.

Models. We considered several popular LLMs, including LLaMA (Touvron et al., 2023), OPT (Zhang et al., 2022), and Mistral (Zhao et al., 2023a). Based on observations by Dettmers et al. (2022b) regarding the conditions under which activation outliers occur in the OPT model, we selected a diverse set of models with parameter counts exceeding 6.7 billion. The benchmark models include medium-sized ones such as OPT-6.7B, LLaMA2-7B, LLaMA2-7B-Chat, LLaMA3-8B, and Mistral-7B, as well as larger models like OPT-13B and OPT-30B. For comparison purposes, we also included a smaller model, OPT-1.3B. Among them, several models from the LLaMA and OPT families are used to evaluate the scalability of the MPOQ method across different tasks and model sizes. The Mistral model serves as a supplement to assess the performance of MPOQ.

Datasets. We used the LAMBADA dataset (Paperno et al., 2016) to test the language modeling task. Language modeling is a fundamental task in natural language processing, representing the basic capability of language models to analyze input text data and learn the structure and patterns of language, predicting and generating the next word or sequence of words in the text. Commonly used datasets (Liu et al., 2024d) for language modeling tasks include Penn Treebank, WikiText-103, The Pile, and LAMBADA. Here, we chose to use the LAMBADA dataset, which is specifically designed to evaluate a model's ability to understand language based on context. This dataset contains 10,022 paragraphs and requires the model to comprehend entire passages, thereby increasing the difficulty of predictions and providing a better assessment of the model's reasoning capabilities. It is particularly beneficial for testing the performance of model compression in generating text with long-range dependencies.

During fine-tuning process, we also used the LAMBADA training set to train the model and tested the performance of different quantization methods on the fine-tuned model. We employed LoRA (Hu et al., 2021) for fine-tuning, maintaining the proportion of fine-tuning parameters at approximately 0.1% of the total parameters, and added the fine-tuned parameters to the original model for quantization testing.

We evaluated the effectiveness of MPOQ on downstream tasks across four datasets with varying context lengths: Boolq (long context), Subj (short context), Ag_news (medium context), and RTE (medium context). The experiments followed the setup by Chevalier et al. (2023).

We evaluate the performance of MPOQ-quantized models on a variety of downstream tasks using the LongBench (Bai et al., 2024). LongBench consists of 21 long-context understanding tasks, covering a wide range of task types including single-document question answering, multi-document question answering, summarization, few-shot learning, code completion, and synthetic tasks. The average input length in LongBench is 3,642 tokens. Our experimental setup follows that of Bai et al. (2024), adopting their evaluation protocols and metrics.

Data requirements and quantization settings. From the perspective of data requirements and quantization settings, our approach differs significantly from existing methods. Among the current methods (including RTN, LLM.int8, SmoothQuant, GPTQ, and AWQ), only RTN and LLM.int8 support quantization without data and training. Other methods, such as SmoothQuant, require a calibration set for tuning the smoothing factor. In terms of quantization settings, only RTN supports full quantization configurations, including weight-only (WxA16), activation-only (W16Ax), and weight & activation (WxAx). In contrast, GPTQ and AWQ only support WxA16, while LLM.int8 and SmoothQuant only support WxAx. MPOQ, however, implements all quantization settings (WxA16, W16Ax, and WxAx) and supports quantization without data and training, similar to RTN. Therefore, we consider RTN as the primary comparison object, while other methods will only be included when applicable conditions are met.

5.2. Main results

Task accuracy assessment of quantization methods. Table 1 shows the accuracy scores of different quantization methods on the LAMBADA dataset across six models (OPT-1.3B, OPT-6.7B, Mistral-7B, LLaMA3-8B, OPT-13B, and OPT-30B). It's evident that as bit precision decreases, all quantization methods significantly reduce the size of the KV cache. At 8-bit quantization, different methods perform similarly, with scores close to FP16. Here, the impact of outliers on the effective mapping process for model compression is minimal, allowing all methods in the table to maintain accuracy during the dequantization process. 4-bit quantization typically exhibits performance close to 16-bit, but results are clearly inferior to 8-bit, indicating that outliers begin to affect the quantization process. When using 2-bit quantization, the accuracy loss for other methods is substantial. However, the MPOQ method shows a clear advantage, effectively extracting outliers from activations and accurately restoring them during dequantization. Additionally, the rate of accuracy decline varies among models as quantization bit precision decreases, highlighting

Table 1
Comparison of model quantization results. The table compares the accuracy evaluation results of three methods—MPOQ, RTN, and SmoothQuant—on the LAMBADA dataset. In this table, "W-A-" indicates the quantization of the key and value modules at different levels. The results for the SmoothQuant method utilize the calibration dataset files and smoothing factor settings generated by the official code. The data for OPT-1.3B and OPT-6.7B are from Liu et al. (2024c). The best performance in each group is highlighted in bold, respectively. For all the results, we report the mean values of five runs using different random seeds.

Setting	Exp	#Bits	Size$_{\text{(MB)}}$	OPT-1.3B	OPT-6.7B	Mistral-7B	LLaMA3-8B	OPT-13B	OPT-30B	Average
	FP16	16-16	46.7	75.4	81.2	87.3	83.0	80.7	82.0	81.6
	RTN	16-8	23.3	75.3	81.2	87.3	82.9	**80.8**	82.0	81.6
	MPOQ	16-8	23.3	**75.4**	81.2	**87.4**	82.9	80.7	82.0	81.6
Activations	RTN	16-4	11.7	71.7	80.6	**85.9**	79.8	79.9	**81.8**	80.0
Only	MPOQ	16-4	11.7	**73.6**	**80.9**	85.8	**80.3**	**80.6**	81.6	**80.5**
	RTN	16-2	5.8	3.5	4.7	7.6	2.5	3.9	0.0	3.7
	MPOQ	16-2	5.8	**8.6**	**28.8**	**11.7**	**38.7**	**32.5**	**16.2**	**22.75**
	RTN	8-8	23.3	75.4	81.3	87.4	83.0	80.8	81.9	81.6
	SmoothQuant	8-8	23.3	75.3	81.3	87.3	83.0	80.8	81.2	81.5
	MPOQ	8-8	23.3	75.4	81.3	87.4	83.0	80.8	**82.0**	**81.7**
Weights	RTN	4-4	11.7	69.4	78.5	79.9	78.3	79.2	80.6	77.7
&	SmoothQuant	4-4	11.7	69.0	77.7	**84.9**	77.9	**79.6**	**81.4**	78.4
Activations	MPOQ	4-4	11.7	**70.8**	**79.1**	84.6	**78.5**	79.5	80.4	**78.8**
	RTN	2-2	5.8	**3.6**	**3.2**	**3.8**	0.6	0.3	0.1	1.9
	SmoothQuant	2-2	5.8	3.8	3.0	3.2	0.0	1.8	0.0	2.0
	MPOQ	2-2	5.8	1.8	2.9	3.4	**0.9**	**2.6**	0.1	2.0

distinct outlier distributions in different LLMs even for the same task. Overall, MPOQ demonstrates superior average accuracy compared to other methods.

Impact of fine-tuning on quantization. The quantization results of the model after fine-tuning are shown in Table 2, where the values represent the difference in accuracy between the fine-tuned model and the original model. The proportion of fine-tuning parameters to total parameters is maintained at approximately 0.1% (Mistral-7B: 0.09%, LLaMA3-8B: 0.08%, and OPT-13B: 0.10%), with a consistent fine-tuning step count of 200. Compared to Table 1, it can be observed that when the model is uncompressed (FP16), the task accuracy improves, indicating a positive enhancement in the model's task adaptability on the LAMBADA dataset. However, when using low-bit quantization for inference, the accuracy differences exhibit significant fluctuations and do not follow a consistent trend with varying compression rates. For some fine-tuned models (*e.g.,* LLaMA3-8B), the accuracy after compression is even lower than that of the original model, which may be related to the fine-tuning hyperparameter settings. To ensure consistency in testing, all models in Table 2 share the same fine-tuning parameter settings without individual optimization. The smoothing coefficient used in SmoothQuant relies on the dataset rather than parameter fine-tuning, so its smoothing factor is consistent with Table 1.

It is noteworthy that when comparing the accuracy of 8-bit and 4-bit quantization, the differences before and after fine-tuning for the same model are significant, and there are also considerable differences between different models. This variability may arise from changes in network parameters due to fine-tuning, leading to altered outlier distributions during inference on the LAMBADA dataset. The distribution range of activation values may narrow or widen, which could be attributed to changes caused by parameter fine-tuning. In this context, both the MPOQ method and other quantization techniques in the table demonstrate stability, with quantized accuracy not directly affected by fine-tuning. The MPOQ method maintains its advantage over other methods in Table 1, unaffected by specific task fine-tuning. This result further highlights the MPOQ method's strengths, which do not require data, training, or calibration.

Evaluation on long-text tasks. Table 3 shows the quantization performance of different methods under varying context lengths. We evaluated the context learning capability of MPOQ using the OPT-13B model across four datasets with varying context lengths. We selected different numbers of demonstration experiments (0-shot, 2-shot, and 10-shot) to test task accuracy. Results indicate that increasing the number of demonstrations generally

Table 2
Comparison of model quantization results after fine-tuning. The proportion of fine-tuning parameters to total parameters is maintained at approximately 0.1%. To ensure consistency in testing, all models share the same fine-tuning parameter settings, without individual optimization for each model. The smoothing factor for SmoothQuant is set as indicated in Table 1. The best performance in each group is highlighted in bold, respectively. For all the results, we report the mean values of five runs using different random seeds.

Setting	Exp	#Bits	Mistral-7B	LLaMA3-8B	OPT-13B	Average
Activations Only	FP16	16-16	87.3	83.0	80.9	83.7
	RTN	16-8	87.20	83.0	80.9	83.7
	MPOQ	16-8	**87.5**	83.0	80.9	**83.8**
	RTN	16-4	85.5	79.3	**81.2**	82.0
	MPOQ	16-4	**85.6**	**79.9**	80.9	**82.1**
	RTN	16-2	7.7	2.3	3.6	4.6
	MPOQ	16-2	**14.4**	**36.8**	**28.9**	**26.7**
Weights & Activations	RTN	8-8	87.4	82.9	**80.9**	83.7
	SmoothQuant	8-8	87.3	**83.1**	**80.9**	**83.8**
	MPOQ	8-8	**87.5**	**83.1**	80.4	83.7
	RTN	4-4	80.1	78.0	80.4	79.5
	SmoothQuant	4-4	**85.1**	78.0	**80.6**	**81.2**
	MPOQ	4-4	84.5	**78.6**	80.0	81.0
	RTN	2-2	3.8	0.6	**3.1**	2.5
	SmoothQuant	2-2	3.3	0.0	2.4	1.9
	MPOQ	2-2	**4.2**	**0.7**	2.9	**2.6**

Table 3
Results of in-context learning. The test results in the table are based on four datasets with different context lengths: the longer context length dataset Boolq, the medium context length datasets Ag_news and RTE, and the shorter context length dataset Subj. The best performance in each group is highlighted in bold, respectively. For all the results, we report the mean values of five runs using different random seeds.

Exp	#Bits	ICL	Boolq	Subj	Ag_news	RTE	Average
FP16	16	0-shot	57.2	52.0	71.8	54.5	58.9
	16	2-shot	64.6	64.5	74.2	49.9	63.3
	16	10-shot	62.7	93.4	70.8	57.8	71.2
RTN	4	2-shot	48.2	59.4	73.7	**47.0**	57.1
MPOQ	4	2-shot	**56.0**	**62.0**	**74.6**	42.8	**58.4**
RTN	4	10-shot	61.0	72.8	76.5	46.2	64.1
MPOQ	4	10-shot	**62.1**	**92.6**	**77.2**	**53.8**	**71.4**
RTN	2	2-shot	43.7	56.5	28.7	58.6	46.9
MPOQ	2	2-shot	**49.6**	**59.9**	**42.3**	**59.6**	**52.8**
RTN	2	10-shot	38.8	51.9	43.7	48.9	45.8
MPOQ	2	10-shot	**42.5**	**67.4**	**45.3**	**53.9**	**52.3**

enhances performance, although there are instances where accuracy decreases despite more demonstrations, potentially related to the emergence of outlier activations or large-scale activations. In the case of longer contexts (10-shot), RTN's performance declines significantly, while MPOQ performs relatively well. This further validates the superior performance of MPOQ.

Evaluation on diverse downstream tasks. Table 4 shows the quantization performance of different methods on LongBench. We present results for representative tasks, including TREC (few-shot learning), SAMSum (few-shot learning), 2wikimqa (multi-document QA), GovReport (summarization), QMSum (summarization), LCC (code completion), PassageRetrieval-zh (synthetic task), along with the average score across all 21 tasks.

Table 4
Results on Longbench. For all the other results, we report the mean values of five runs using different random seeds.

Model	Exp	#Bits	TREC	SAMSum	2wikimqa	GovReport	QMSum	LCC	Passage Retrieval-zh	Average
LLaMA2-7B	FP16	16-16	67.50	41.12	10.84	26.57	20.70	66.77	8.00	25.60
	RTN	16-4	63.50	38.12	**11.19**	21.98	19.10	**58.59**	4.35	**22.94**
	SmoothQuant	16-4	**65.00**	**38.80**	10.50	21.14	19.48	57.59	**6.42**	22.72
	MPOQ	16-4	59.50	38.64	8.78	**23.21**	**19.44**	57.83	4.63	22.86
	RTN	16-2	0.00	1.02	0.11	0.06	0.69	12.18	0.00	1.50
	SmoothQuant	16-2	0.00	0.81	0.31	0.03	1.15	12.88	0.25	1.57
	MPOQ	16-2	0.00	**1.27**	**0.33**	**0.88**	**2.83**	**16.69**	**0.95**	**2.71**
LLaMA2-7B -Chat	FP16	16-16	64.00	41.26	27.53	26.42	21.20	58.53	12.00	30.13
	RTN	16-4	59.50	39.60	27.69	25.20	**20.67**	**49.34**	4.22	**26.79**
	SmoothQuant	16-4	**63.00**	38.56	**30.30**	24.46	20.02	47.60	**4.93**	26.59
	MPOQ	16-4	58.50	**39.96**	27.52	**25.89**	19.29	48.52	3.80	26.62
	RTN	16-2	0.00	1.49	0.26	0.04	1.39	10.66	0.50	1.34
	SmoothQuant	16-2	0.00	1.40	0.18	0.04	0.99	10.78	0.00	1.39
	MPOQ	16-2	**2.00**	**3.14**	**0.74**	**1.65**	**5.00**	**20.33**	**1.29**	**4.26**

The results show that under W16A4 quantization, MPOQ outperforms both RTN and SmoothQuant on the GovReport task across two models. On the SAMSum, QMSum, and LCC tasks, MPOQ achieves scores comparable to those of RTN and SmoothQuant. Overall, the three methods—RTN, SmoothQuant, and MPOQ—exhibit similar average performance under W16A4 quantization. However, under W16A2 quantization, MPOQ consistently outperforms RTN and SmoothQuant across almost all tasks, demonstrating a clear advantage in lower-bit quantization scenarios.

The compression rate and latency of MPOQ. Finally, we evaluate the compression rate and time complexity of MPOQ on the OPT-13B model. Theoretically, the quantization compression rate λ for the matrix W can be expressed as:

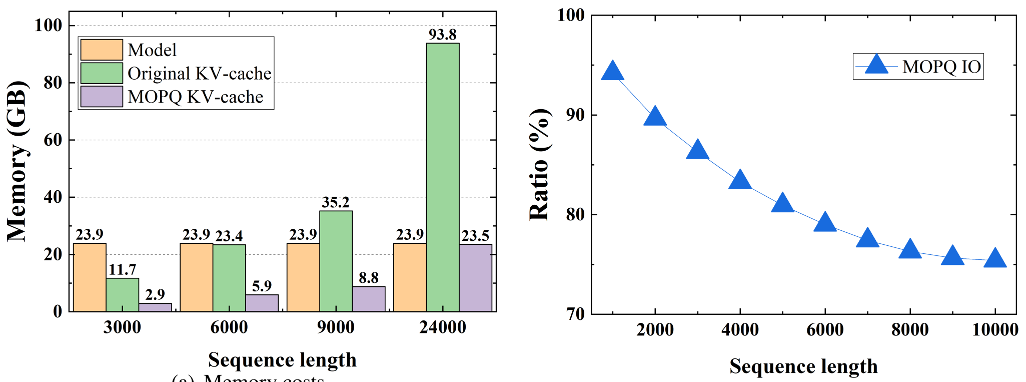
$$\lambda = \frac{\mathrm{num}(\mathcal{T}_C) \times \mathrm{x} + \left[\mathrm{num}(\mathcal{T}_O) + \mathrm{num}(\Delta)\right] \times 16}{\mathrm{num}(\mathrm{W}) \times 16}, \tag{3}$$

where num(·) denotes the number of elements in a tensor. MPOQ quantizes the central tensor $\mathcal{T}_C$ to a low-precision representation of x bits, while the other tensor $\mathcal{T}_O$ and the quantization step size Δ retain their original precision. Since the number of parameters in $\mathcal{T}_O$ andi Δ is significantly smaller than that in $\mathcal{T}_C$, we can approximate $\lambda \approx \frac{\mathrm{num}(\mathcal{T}_C)\times \mathrm{x}}{\mathrm{num}(\mathrm{W})\times 16}$. Additionally, the total number of tensor parameters obtained through MPO decomposition is slightly larger than that of the original matrix, but this increase is negligible compared to the quantization benefits of the central tensor. When the central tensor is quantized to a 4-bit integer, the compression rate of MPOQ approaches 1/4.

The cache evaluation results of the model are shown in Figure 4(a). The sequence length increases from 3k to 24k. When the sequence length reaches 6k, the size of the KV cache is comparable to that of the model, while the cache after MPOQ compression is only one-quarter the size of the model. It is not until the sequence length grows to 24k that the size of the compressed cache matches that of the model. Additionally, MPOQ reduces both the KV cache size and communication costs. As shown in Figure 4(b), as the text sequence length increases, the communication latency of MPOQ significantly decreases. These experiments demonstrate that, during the inference process, MPOQ is highly effective in reducing cache usage and in lowering inference latency.

5.3. Discussion

MPOQ continues the approach from DecoQuant (Liu et al., 2024c) and conducts more extensive experiments, addressing several issues that were previously overlooked, thus thoroughly demonstrating the effectiveness of the method. We tested a wider range of models, including the Mistral-7B, OPT-13B and OPT-30B, with a particular focus on the quantization effects of the MPOQ method on the latest version of the open-source model, LLaMA3. Furthermore, we conduct extensive evaluations on a wide range of downstream tasks from the LongBench, and compare our results



(a) Memory costs (b) I/O latency

Figure 4: Memory costs and I/O latency. Testing the efficiency of KV cache compression and inference latency of MPOQ on the OPT-13B model.

with those achieved by state-of-the-art methods. More importantly, this study considers the impact of fine-tuning on quantization, evaluating the quantization effects before and after fine-tuning and comparing these results with other methods.

6. Conclusion

This paper proposes a KV cache compression method, MPOQ, based on MPO decomposition for accelerating inference in LLMs. MPOQ decomposes the original matrix into local tensors (the central tensor and the other tensor), effectively transferring the quantization challenges from the original matrix to these local tensors. This allows us to adjust the distribution of outliers within the original matrix. The results indicate that MPOQ can simultaneously reduce the KV cache size while ensuring accuracy, thus enabling efficient inference for LLMs.

As a post-training quantization method, MPOQ has two main advantages: (1) it requires no data, training, or calibration; and (2) it is highly flexible, supporting separate quantization of weights, separate quantization of activations, or simultaneous quantization of both. Extensive experiments demonstrate that MPOQ can achieve 4-bit quantization of the KV cache with minimal performance loss, highlighting its effectiveness in enhancing the inference efficiency of LLMs while maintaining model performance. Furthermore, when 2-bit quantization is applied solely to the activations, the performance of MPOQ significantly surpasses that of the RTN method. In the future, we plan to further explore the potential of MPOQ in 2-bit KV cache compression.

7. Limitations

While our method demonstrates promising performance, it is not without limitations. The current experimental results are based on a limited set of tasks and datasets, and its generalizability across broader application scenarios has yet to be fully validated. Moreover, as a post-training quantization method, our approach may facilitate the deployment of LLMs on various end devices, such as personal phones and computers. However, the potential biases inherent in these models could raise social concerns when deploying LLMs in real-world settings.

Acknowledgments

This work was financially supported by the National Natural Science Foundation of China (Grant No.62476278, No.11934020, and No.12204533) and the National Key R&D Program of China (Grants No. 2024YFA1408601). Computational resources have been provided by the Physical Laboratory of High Performance Computing at Renmin University of China.

References

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., et al. (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.

Bansal, H., Gopalakrishnan, K., Dingliwal, S., Bodapati, S., Kirchhoff, K., and Roth, D. (2022). Rethinking the role of scale for in-context learning: An interpretability-based case study at 66 billion scale. *arXiv preprint arXiv:2212.09095*.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Chevalier, A., Wettig, A., Ajith, A., and Chen, D. (2023). Adapting language models to compress contexts. In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3829–3846. Association for Computational Linguistics.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. (2022a). Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. (2022b). Llm.int8(): 8-bit matrix multiplication for transformers at scale. *CoRR*, abs/2208.07339.

Frantar, E. and Alistarh, D. (2023). Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. (2022). GPTQ: accurate post-training quantization for generative pre-trained transformers. *CoRR*, abs/2210.17323.

Gao, Z.-F., Cheng, S., He, R.-Q., Xie, Z.-Y., Zhao, H.-H., Lu, Z.-Y., and Xiang, T. (2020a). Compressing deep neural networks by matrix product operators. *Physical Review Research*, 2(2):023300.

Gao, Z.-F., Cheng, S., He, R.-Q., Xie, Z.-Y., Zhao, H.-H., Lu, Z.-Y., and Xiang, T. (2020b). Compressing deep neural networks by matrix product operators. *Physical Review Research*, 2(2):023300.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Kang, H., Zhang, Q., Kundu, S., Jeong, G., Liu, Z., Krishna, T., and Zhao, T. (2024). Gear: An efficient kv cache compression recipe for near-lossless generative inference of llm. *arXiv preprint arXiv:2403.05527*.

Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., and Stoica, I. (2023). Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA. Association for Computing Machinery.

Lee, W., Lee, J., Seo, J., and Sim, J. (2024). InfiniGen: Efficient generative inference of large language models with dynamic KV cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 155–172, Santa Clara, CA. USENIX Association.

Lin, J., Tang, J., Tang, H., Yang, S., Dang, X., and Han, S. (2023). AWQ: activation-aware weight quantization for LLM compression and acceleration. *CoRR*, abs/2306.00978.

Liu, A., Liu, J., Pan, Z., He, Y., Haffari, G., and Zhuang, B. (2024a). Minicache: Kv cache compression in depth dimension for large language models. *arXiv preprint arXiv:2405.14366*.

Liu, P., Gao, Z., Zhao, W. X., Xie, Z., Lu, Z., and Wen, J. (2021). Enabling lightweight fine-tuning for pre-trained language model compression based on matrix product operators. In Zong, C., Xia, F., Li, W., and Navigli, R., editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 5388–5398. Association for Computational Linguistics.

Liu, P., Gao, Z.-F., Zhao, W. X., Ma, Y., Wang, T., and Wen, J.-R. (2024b). Unlocking data-free low-bit quantization with matrix decomposition for kv cache compression. *arXiv preprint arXiv:2405.12591*.

Liu, P., Gao, Z.-F., Zhao, X., Ma, Y., Wang, T., and Wen, J.-R. (2024c). Unlocking data-free low-bit quantization with matrix decomposition for KV cache compression. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2430–2440, Bangkok, Thailand. Association for Computational Linguistics.

Liu, Y., Cao, J., Liu, C., Ding, K., and Jin, L. (2024d). Datasets for large language models: A comprehensive survey. *arXiv preprint arXiv:2402.18041*.

Liu, Y., Li, H., Cheng, Y., Ray, S., Huang, Y., Zhang, Q., Du, K., Yao, J., Lu, S., Ananthanarayanan, G., Maire, M., Hoffmann, H., Holtzman, A., and Jiang, J. (2024e). Cachegen: Kv cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, page 38–56, New York, NY, USA. Association for Computing Machinery.

Liu, Z., Desai, A., Liao, F., Wang, W., Xie, V., Xu, Z., Kyrillidis, A., and Shrivastava, A. (2024f). Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36.

Liu, Z., Yuan, J., Jin, H., Zhong, S., Xu, Z., Braverman, V., Chen, B., and Hu, X. (2024g). Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*.

Mu, J., Li, X. L., and Goodman, N. D. (2023). Learning to compress prompts with gist tokens. *CoRR*, abs/2304.08467.

Paperno, D., Kruszewski, G., Lazaridou, A., Pham, Q. N., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., and Fernández, R. (2016). The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Park, G., Park, B., Kim, M., Lee, S., Kim, J., Kwon, B., Kwon, S. J., Kim, B., Lee, Y., and Lee, D. (2022). Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models. *arXiv preprint arXiv:2206.09557*.

Patel, P., Choukse, E., Zhang, C., Shah, A., Goiri, Í., Maleki, S., and Bianchini, R. (2024). Splitwise: Efficient generative llm inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 118–132. IEEE.

Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Heek, J., Xiao, K., Agrawal, S., and Dean, J. (2023). Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5:606–624.

Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Levskaya, A., Heek, J., Xiao, K., Agrawal, S., and Dean, J. (2022). Efficiently scaling transformer inference. *CoRR*, abs/2211.05102.

Rabanser, S., Shchur, O., and Günnemann, S. (2017). Introduction to tensor decompositions and their applications in machine learning. *CoRR*, abs/1711.10781.

Sheng, Y., Zheng, L., Yuan, B., Li, Z., Ryabinin, M., Chen, B., Liang, P., Ré, C., Stoica, I., and Zhang, C. (2023). Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pages 31094–31116. PMLR.

Sun, M., Chen, X., Kolter, J. Z., and Liu, Z. (2024). Massive activations in large language models. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*.

Tang, H., Lin, Y., Lin, J., Han, Q., Hong, S., Yao, Y., and Wang, G. (2024). Razorattention: Efficient kv cache compression through retrieval heads. *arXiv preprint arXiv:2407.15891*.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023). Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. (2023). Smoothquant: Accurate and efficient post-training quantization for large language models. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 38087–38099. PMLR.

Yang, J. Y., Kim, B., Bae, J., Kwon, B., Park, G., Yang, E., Kwon, S. J., and Lee, D. (2024). No token left behind: Reliable kv cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*.

Yao, Z., Aminabadi, R., Zhang, M., Wu, X., Li, C., and He, Y. Z. (2022a). Efficient and affordable post-training quantization for large-scale transformers, 2022. *URL https://arxiv. org/abs/2206.01861*.

Yao, Z., Yazdani Aminabadi, R., Zhang, M., Wu, X., Li, C., and He, Y. (2022b). Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M. T., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T., and Zettlemoyer, L. (2022). OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068.

Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., et al. (2024). H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36.

Zhao, L., Yu, E., Ge, Z., Yang, J., Wei, H., Zhou, H., Sun, J., Peng, Y., Dong, R., Han, C., and Zhang, X. (2023a). Chatspot: Bootstrapping multimodal llms via precise referring instruction tuning. *arXiv preprint arXiv:2307.09474*.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J., and Wen, J. (2023b). A survey of large language models. *CoRR*, abs/2303.18223.

Zhao, Y., Lin, C.-Y., Zhu, K., Ye, Z., Chen, L., Zheng, S., Ceze, L., Krishnamurthy, A., Chen, T., and Kasikci, B. (2024). Atom: Low-bit quantization for efficient and accurate llm serving. In Gibbons, P., Pekhimenko, G., and Sa, C. D., editors, *Proceedings of Machine Learning and Systems*, volume 6, pages 196–209.

Zhong, Y., Liu, S., Chen, J., Hu, J., Zhu, Y., Liu, X., Jin, X., and Zhang, H. (2024). Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving.

Table 5
Comparisons with GEAR, MiniCache, and RazorAttention on LongBench. "*" indicates the data are from GEAR (Kang et al., 2024), MiniCache (Liu et al., 2024a), and RazorAttention (Tang et al., 2024) respectively. MPOQ are evaluated under a unified quantization setting of W16A4. For all the other results, we report the mean values of five runs using different random seeds.

Model	Exp	TREC	SAMSum	2wikimqa	GovReport	QMSum	LCC	PassageRetrieval-zh	Average
LLaMA2-7B	FP16	67.50	41.12	10.84	26.57	20.70	66.77	8.00	25.60
	GEAR	58.43	**41.32***	**9.23**	**26.97***	**21.28***	54.68	**5.72**	**27.80***
	MPOQ	**59.50**	38.64	8.78	23.21	19.44	**57.83**	4.63	22.86
LLaMA2-7B -Chat	FP16	64.00	41.26	27.53	26.42	21.20	58.53	12.00	30.13
	MiniCache	**64.00***	38.89	**30.58***	25.32*	19.21	**58.03***	**4.26**	**35.44***
	MPOQ	58.50	**39.96**	27.52	**25.89**	**19.29**	48.52	3.80	26.62
LLaMA3-8B -Instruct	FP16	54.00	20.82	30.54	17.46	8.79	55.37	65.65	21.70
	RazorAttention	0.00*	**19.22**	**26.89***	**26.56***	**23.86***	30.25*	**32.31**	**34.86***
	MPOQ	**51.27**	19.06	21.02	16.79	10.34	**36.53**	30.46	17.68

A. Appendix

Compare with GEAR, MiniCache, and RazorAttention

The comparison results with GEAR, MiniCache, and RazorAttention are shown in Table 5, although there remains a performance gap compared to RazorAttention, MiniCache, and GEAR in average scores, MPOQ achieves higher scores than GEAR and RazorAttention on the TREC and LCC tasks, and outperforms MiniCache on the SAMSum, GovReport, and QMSum tasks. These results suggest that MPOQ may offer certain advantages over the compared methods on specific tasks.

It is worth noting that GEAR, similar to RTN, SmoothQuant, and MPOQ, also incorporates a low-bit compression process for activations. Among these comparable approaches, MPOQ exhibits significant competitiveness. Unlike low-bit quantization methods, RazorAttention employs eviction and compensation mechanisms to reduce KV cache size while achieving compression ratios comparable to 4-bit quantization, enabling direct comparison in Table 5. MiniCache leverages similarities in cross-layer KV caches to reduce memory usage by merging adjacent layer caches, with its reported results based on 4-bit KIVI (Liu et al., 2024g) implementation, allowing for fair comparison. This comparative analysis indicates that, although MPOQ may not achieve the state-of-the-art average scores under 4-bit quantization, it maintains competitive performance on specific tasks such as TREC, GovReport, and LCC.

The effects of varying the decomposition ratio Table 6 illustrates the impact of different decomposition ratios in the MPOQ method on generation quality under two quantization settings: W16A4 and W16A2. The experiments are conducted on three models (Mistral-7B, LLaMA3-8B, and OPT-13B) using the LAMBADA dataset. The results show that as the decomposition ratio decreases, the generation quality improves across all models; however, the activation compression ratio drops rapidly. This suggests that, in order to maintain generation quality without significant degradation, it is advisable to adopt a relatively higher decomposition ratio to achieve better compression during inference.

The effects of different quantization bit widths Table 7 presents the effects of different quantization bit widths for $\mathcal{T}_C$ and $\mathcal{T}_O$ on model performance. The experiments are conducted on three models (Mistral-7B, LLaMA3-8B, and OPT-13B) using the LAMBADA dataset. The results show that as the bit width of $\mathcal{T}_O$ decreases, the model scores drop sharply. Acceptable performance is observed only when $\mathcal{T}_O$ maintains a relatively high bit width.

The effects of sequence length on performance

The effects of sequence length on performance is shown in Figure 5. In the figure, the vertical axis represents the score retention efficiency after quantization, calculated as

$$S = 1 - \frac{|S_{\mathrm{FP16}} - S_{\mathrm{MPOQ}}|}{S_{\mathrm{FP16}}}, \tag{4}$$

Table 6
The effects of varying the decomposition ratio under two quantization settings: W16A4 and W16A2.

#Bits	Setting	Mistral-7B	LLaMA3-8B	OPT-13B	Average	Compression Ratio
16-4	99%	85.8	80.3	80.6	82.2	3.9x
	75%	85.9	80.9	80.7	82.5	2.3x
	50%	86.2	81.4	80.7	82.8	1.6x
	25%	86.5	81.7	80.6	82.9	1.2x
16-2	99%	11.7	38.7	32.5	27.6	7.7x
	75%	16.9	39.3	34.0	30.1	2.9x
	50%	25.6	43.9	37.4	35.6	1.8x
	25%	32.2	48.5	44.3	41.7	1.3x

Table 7
The effects of different quantization bit widths for $\mathcal{T}_C$ and $\mathcal{T}_O$ on model performance.

Setting	Mistral-7B	LLaMA3-8B	OPT-13B	Average
16-16	87.3	83.0	80.7	83.7
8-16	87.4	82.9	80.7	83.7
8-8	25.1	36.3	37.2	32.9
8-4	3.9	12.0	9.4	8.4
8-2	3.7	1.9	1.3	2.3
4-16	85.8	80.3	80.6	82.2
4-8	16.3	32.7	31.8	26.9
4-4	3.1	8.3	6.6	6.0
4-2	1.2	0.7	0.3	0.7
2-16	11.7	38.7	32.5	27.6
2-8	5.2	7.4	4.3	5.6
2-4	1.4	1.7	0.6	1.2
2-2	0.3	0.1	0.2	0.2

where S_{FP16} is the score from the FP16 experiment and S_{MPOQ} is the score obtained using MPOQ under the W16A4 quantization setting. The horizontal axis 'Avg len'(average length) denotes the number of words for the English (code) datasets and the number of characters for the Chinese datasets of LongBench (Bai et al., 2024). The figure presents the results of MPOQ quantization on two models: LLaMA2-7B and LLaMA2-7B-Chat. As shown, there is no clear linear correlation between score retention efficiency and context length. Instead, the results are more significantly influenced by task type and model architecture. Context length directly affects the memory cost and I/O latency during quantized inference, as illustrated in Figure 4(a) and Figure 4(b).

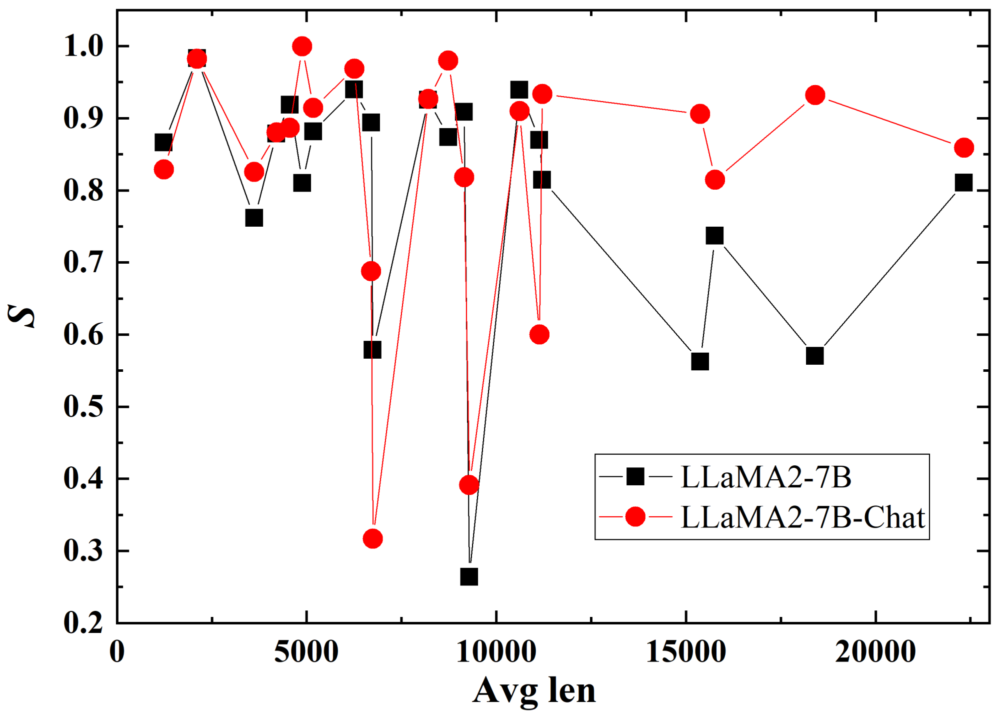


Figure 5: The effects of sequence length on performance. The results of MPOQ quantization are test on LongBench on two models: LLaMA2-7B and LLaMA2-7B-Chat. The horizontal axis denotes the number of tokens.