QAQ: Quality Adaptive Quantization for LLM KV Cache

Anonymous Author(s)

Abstract

The emergence of LLMs has ignited a fresh surge of breakthroughs in NLP applications, particularly in domains such as question-answering systems and text generation. As the need for longer context grows, a significant bottleneck in model deployment emerges due to the linear expansion of the Key-Value (KV) cache with the context length. Existing methods primarily rely on various hypotheses, such as sorting the KV cache based on attention scores for replacement or eviction, to compress the KV cache and improve model throughput. However, heuristics used by these strategies may wrongly evict essential KV cache, which can significantly degrade model performance. In this paper, we propose QAQ, a Quality Adaptive Quantization scheme for the KV cache. We theoretically demonstrate that key cache and value cache exhibit distinct sensitivities to quantization, leading to the formulation of separate quantization strategies for their non-uniform quantization. Through the integration of dedicated outlier handling, as well as an improved attention-aware approach, QAQ achieves up to $10\times$ the compression ratio of the KV cache size with a neglectable impact on model performance. QAQ significantly reduces the practical hurdles of deploying LLMs, opening up new possibilities for longer-context applications. The code is available in supplementary materials for reproducibility.

1 Introduction

Large Language Models (LLMs) demonstrate state-of-the-art performance on various Natural Language Processing (NLP) benchmarks [Beeching *et al.*, 2023]. These LLMs showcased exceptional potential across a multitude of practical applications, including but not limited to text generation, conversation processing, and knowledge question answering [Chang *et al.*, 2023]. However, deploying these models efficiently poses a challenge due to the sequential nature of the generative inference process. That is, sequentially processing one token at a time requires accessing all previously generated tokens for computation. In practical computations, such as GPT series [Brown *et al.*, 2020], LLaMA series [Touvron *et al.*, 2023], and OPT series [Zhang *et al.*, 2022], the generative inference of these LLMs typically incorporates the *KV cache* mechanism to improve the efficiency of computation resource utilization. KV cache stores the computed values of the Key-Value vector from previous attention calculations and reuses them when computing the current token to save extra costs associated with redundant calculations. While being a widely used optimization method, as the model size and context length continue to increase, the storage overhead of the KV cache itself also grows dramatically, imposing significant pressure on the on-device, especially the high-cost GPU memory. *Reducing the memory footprint of the KV cache has become a highly active research topic* [Zhu *et al.*, 2023].

Currently, there is a substantial body of research addressing the efficient utilization of GPU memory in memory-constrained scenarios. Offloading is an intuitive solution for handling insufficient GPU memory during model inference. Although offloading can alleviate the pressure on GPU memory, implementing offloading specifically for the KV cache is a non-trivial problem, as it is constrained by various factors such as data transmission bandwidth limitations. Additionally, approaches like sparse transformers [Child *et al.*, 2019] and multi-query attention [Pope *et al.*, 2023] are designed to reduce cache sizes directly; however, applying them directly to optimize the KV cache may result in significant performance degradation [Ge *et al.*, 2023]. Recently, pioneering studies have emerged, focusing on the direct optimization of the KV cache to minimize its footprint in GPU memory. However, these methods often rely on attention values to eliminate redundant portions from the KV cache, retaining what is considered essential. Nevertheless, these approaches may lead to erroneously removing crucial KV cache and significantly degrading the performance of the model [Zhang *et al.*, 2023]. Naturally, it leads to the question: *is there a direct quantization method that can avoid the drawbacks mentioned above, while achieving leading performance?*

In this paper, we propose QAQ, a quality adaptive quantization scheme for KV cache in LLMs. Quantization is a commonly employed method for compressing model sizes and has been widely utilized for the compression of weights and activations [Zhu *et al.*, 2023]. However, compressing the KV cache remains a challenging task. There are three key insights that inspire QAQ.

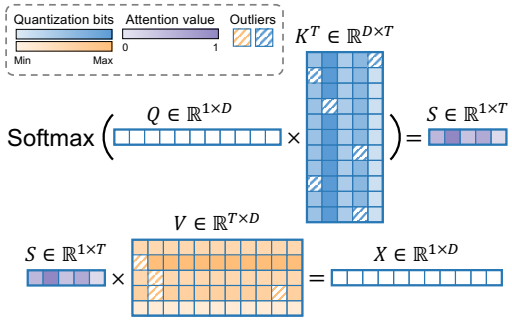


Figure 1: Calculation process of quantized key and value vectors, $\frac{1}{\sqrt{D}}$ is neglected.

- First, key cache and value cache exhibit distinct sensitivities to quantization, a proposition validated through theoretical analyses and empirical experiments. This necessitates the development of separate quantization strategies for key cache and value cache.
- Second, the hypothesis of persistence of importance has exceptions. Previous works proposed the hypothesis of the persistence of importance, advocating compression based on importance (attention-aware). We discovered that, despite its validity in the majority of cases, there exist a number of exceptions. This implies the need for careful handling of exceptions when quantizing based on attention.
- Third, outliers play a crucial role. While this has been acknowledged in weight and activation quantization, we verified that outliers also exert a significant impact on the KV cache quantization. Consequently, a dedicated treatment for quantizing outliers is required.

With these considerations, QAQ achieves nearly a $10\times$ compression of the KV cache size with minimal impact on model inference performance. In comparison to existing attention-based KV cache eviction or replacement approaches, QAQ achieves a further nearly $2\times$ compression. We make our code publicly available for replication.

2 Problem Description and Related Work

In this section, we first introduce problem profiling, followed by related work.

2.1 Problem Description

The inference process of an auto-regressive generative LLM typically includes two procedures, prompt encoding and token generation. In the prompt encoding procedure, for each token generation, the LLM requires contextual information from previous tokens, *i.e.*, key and value vectors (KV vectors). KV vectors are stored in the KV cache once they are generated to eliminate redundant computations. When a new token is generated in the token generation procedure, its associated KV vectors are appended to the KV cache, which implies that the size of the KV cache linearly increases with the length of the token sequence. However, the KV cache demonstrates a linear growth relationship with sequence length. As the model requires longer contexts, the KV cache becomes a substantial performance bottleneck. Taking OPT-175B as an example, with a total of 96 layers and a hidden size of 12288, its weights occupy 325GB memory, while the KV cache is $3.54\times$ larger, reaching 1152GB under its maximum sequence length.

We formally define the problem of quantization of the KV cache, we focus on the memory footprint of the KV cache in the attention calculation. Denote $\mathbf{Q} \in \mathbb{R}^{1\times D}$, $\mathbf{K} \in \mathbb{R}^{T\times D}$, and $\mathbf{V} \in \mathbb{R}^{T\times D}$ as the query tensor, key tensor, and value tensor within each head in every layer, respectively. The softmax operation output is denoted by $\mathbf{S} \in \mathbb{R}^{1\times T}$, calculated as $\text{Softmax}(\frac{1}{\sqrt{D}}\mathbf{QK}^T)$. Additionally, $\mathbf{X} \in \mathbb{R}^{1\times D}$ represents the product of $\mathbf{S}$ and $\mathbf{V}$. All the notions and their shape are illustrated in Figure 1.

For a specific quantization method C, we mark the quantized KV cache as $\hat{\mathbf{K}}, \hat{\mathbf{V}} = f(\mathbf{K}, \mathbf{V}, C)$, we choose the method that minimizes the loss of accuracy and at the same time using the least memory, as follows.

$$C^* = \text{argmin}_{C\in\mathcal{C}} \text{KVCacheMemory}(C) \quad \text{s.t.}$$
$$\left\|\text{Softmax}\left(\frac{1}{\sqrt{D}}\mathbf{QK}^T\right) - \text{Softmax}\left(\frac{1}{\sqrt{D}}\mathbf{Q}\hat{\mathbf{K}}^T\right)\right\|_2^2 \le \Delta_S$$
$$\left\|\text{Softmax}\left(\frac{1}{\sqrt{D}}\mathbf{QK}^T\right)\mathbf{V} - \text{Softmax}\left(\frac{1}{\sqrt{D}}\mathbf{Q}\hat{\mathbf{K}}^T\right)\hat{\mathbf{V}}\right\|_2^2 \le \Delta_X,$$

where $\mathcal{C}$ represents the set of all quantization methods, $\text{KVCacheMemory}(C)$ is the KV cache memory cost, which in our design is calculated by the quantized bit, of method C, Δ_S, Δ_X is the constrained quantization loss for $\mathbf{S}$ and $\mathbf{X}$, respectively, and are hyper-parameters to control the aim of accuracy.

2.2 Related Work

To alleviate the practical deployment challenges associated with the increasing scale of models, numerous methods have been developed in recent years for model compression. The memory footprint of a LLM consists of three components: model weights, activation, and KV cache. Early compression techniques primarily targeted the first two components [Zhu *et al.*, 2023]. Among these, the most representative categories include quantization, pruning, distillation, and low-rank approximation. In the field of model compression, quantization is a widely embraced method. Quantization involves converting the floating-point representations within the model into discrete forms, such as integers, which can significantly reduce the storage requirement. Carefully designed quantization methods aim to minimize the accuracy loss to an acceptable range. Quantization can be categorized into two main types: quantization-aware training (QAT) and post-training quantization (PTQ). QAT is less practical due to the substantial re-training costs, while PTQ without careful design may lead to severe accuracy degradation. In the early stage of PTQ, certain approaches focus on quantizing only the weight of LLMs. OPTQ [Frantar *et al.*, 2022b] introduces 3 or 4 bits quantization for weights with improved performance. LLM.int8() [Dettmers *et al.*, 2022] exploits the importance of the outliers and employs vector-wise quantization and mixed-precision decomposition for *outliers*. AWQ [Lin *et al.*, 2023]

finds only 1% of overall weights have a great impact on the performance of the model, it proposes attention-aware quantization based on this insight. ZeroQuant [Yao *et al.*, 2022] integrates a quantization scheme for both weight and activation in LLMs.

As the model demands higher capabilities for handling extremely long contexts, compressing the KV cache becomes pronounced. There is a limited body of recent art directly towards compressing the KV cache in LLM to mitigate the bottleneck. FastGen [Ge *et al.*, 2023] develops an adaptive compression method for the KV cache, leveraging the observation that abundant structures exist in attention modules. H2O [Zhang *et al.*, 2023] exploits the importance of a small set of tokens and proposes an efficient eviction strategy for the KV cache. Scissorhands [Liu *et al.*, 2023] validates the persistence of importance hypothesis for the KV cache and reduces the storage buffer. However, these methods rely on attention values to eliminate redundant parts in the KV cache, retaining the so-called important portions. Nevertheless, any misjudgment of importance leading to the loss of crucial cache can significantly degrade the performance of the model.

3 Insights

In this section, we will introduce the exposition of the three key insights that inspire our design.

3.1 Key Cache and Value Cache Exhibit Distinct Sensitivities to Quantization

Given the distinct roles of key vector and value vector in attention computations, the impact of quantization on the key cache and value cache yields disparate outcomes. The theoretical derivation supporting this assertion is as follows.

We first investigate the partial derivative of X_j concerning V_{ti} as follows:

$$\frac{\partial X_j}{\partial V_{ti}} = \begin{cases} 0 & i \neq j \\ S_t & i = j \end{cases}.$$

Similarly, we investigate the partial derivative of X_j with respect to K_{ti} as follows:

$$\frac{\partial X_j}{\partial K_{ti}} = S_t \cdot Q_i(V_{tj} - X_j).$$

From the above two equations, it is evident that changes in K have a more pronounced impact on X compared to V. In other words, the key cache is more sensitive to quantization, quantizing key cache results in more severe performance degradation. To validate our theoretical analysis, we utilize a uniform quantization approach to quantize the key cache and value cache individually in LLaMA 2-7B. Subsequently, we assess the model's accuracy in responding to 1,000 randomly sampled questions from HellaSwag [Zellers *et al.*, 2019]. The results are depicted in Figure 2(a). Notably, the key cache and value cache exhibit distinct sensitivities to the same uniform quantization granularity. Specifically, when uniformly quantizing the value cache to 2 bits only, the model's performance maintains a relatively high accuracy. However, in the case of only uniformly quantizing the key cache to 2 bits, the model's performance degrades significantly, aligning with our derived conclusions. This underscores the necessity of employing distinct quantization strategies for the key cache and value cache to acquire the best performance. To the best of our knowledge, we are the first ever to exploit this observation.

3.2 Persistence of Importance has Exceptions

The persistence of important tokens, which means the specific tokens that have larger attention value are the only ones significant for now and future steps in the LLM inference process, has been raised in the past works [Liu *et al.*, 2023]. Although this finding holds in most cases, there are still some exceptions where tokens that were initially less significant become suddenly crucial in the subsequent process of generation.

Figure 2(b) illustrates the attention computations matrix for the 1^{st} layer, 8^{th} head of LLaMA 2-7B [Touvron *et al.*, 2023], after the inference process on a randomly selected question from HellaSwag [Zellers *et al.*, 2019]. It is evident that the importance of the majority of tokens tends to persist, indicating a relatively stable importance level. However, there are exceptions where the importance of a few tokens deviates. Specifically, as illustrated by exception #1 and exception #2, certain tokens, initially considered less important, undergo a sudden shift in importance during a specific instance of inference. Without additional treatment for these exceptional cases, based on the assumptions of existing methodologies, these exceptional tokens might be evicted or discarded before demonstrating their importance. This could result in the loss of information when their importance changes and they are required for computation, subsequently impacting the model's performance.

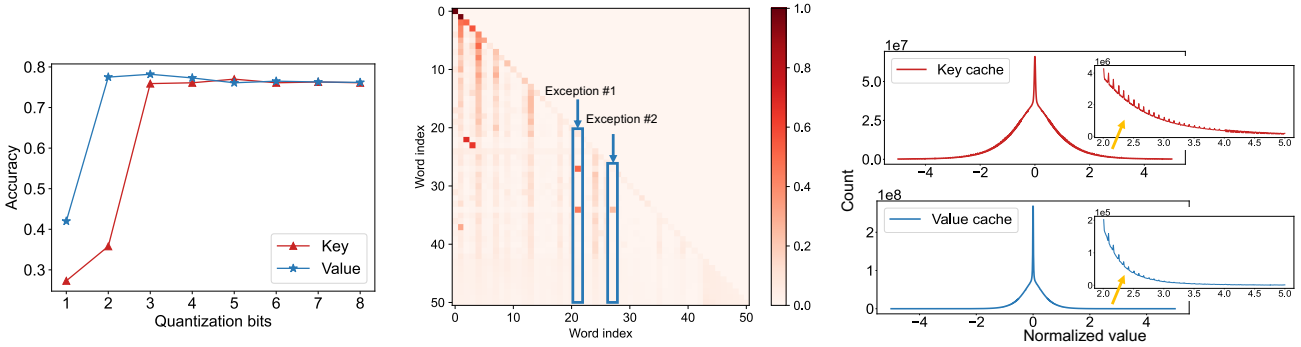
In our design, we propose the *attention window* as the solution, which stores the maximum value from the previous n attention scores for each token, addressing the exceptional cases mentioned above.

3.3 Outliers in KV Cache Matter

The handling of outliers is of paramount importance as they can significantly impact model performance when formulating quantization strategies. Existing works have demonstrated a profound understanding of outliers in model weights and activations. For instance, LLM.int8() [Dettmers *et al.*, 2022] employs vector-wise quantization and mixed-precision decomposition to address outliers in the model's weights. OWQ [Lee *et al.*, 2023] theoretically analyzes how activation outliers amplify errors in weight quantization.

The outliers in the KV cache also play an important role. After randomly testing 1,000 questions from HellaSwag using the LLaMA 2-7B model, the normalized numerical distributions of the key cache and value cache are depicted in Figure 2(c). We have magnified the tail for better visibility. Please note that the jagged distribution in the figure is a result of floating-point precision. The graph reveals a substantial presence of outliers in both the key cache and value cache. Further experiments indicate that neglecting to address these outliers significantly impacts the model's performance.

Our approach involves the implementation of mixed precision, specifically assigning a separate storage precision for



(a) KV cache exhibits different sensitivity to quantization.

(b) Exceptions in attention matrix.

(c) Distribution of KV cache value and outliers.

Figure 2: The experiment demonstration of our key insights.

outliers during quantization. This strategy aims to minimize performance degradation attributable to precision loss associated with outlier values.

4 Quality Adaptive Quantization Method for Key Cache and Value Cache

In this section, we first derive the formulas for KV cache quantization. Next, we show how this quantification approach is employed in the text generation procedure.

4.1 Quantitative Formula Derivation

The idea behind the derivation of our formula is as follows: We regard the quantized KV cache ($\hat{\mathbf{K}}$ or $\hat{\mathbf{V}}$) as a set of independent random variables, with means equivalent to the unquantized KV cache ($\mathbf{K}$ or $\mathbf{V}$) and customizable standard deviations ($\sigma_t^{(\mathbf{K})}$ or $\sigma_t^{(\mathbf{V})}$) for each token t, controlled by the number of quantization bits. Then, we treat the attention value $\hat{\mathbf{S}}$ as a function of $\hat{\mathbf{K}}$ and the output of the self-attention module $\hat{\mathbf{X}}$ as a function of $\hat{\mathbf{V}}$, both of which are also considered as random variables. To keep the accuracy optimal, it is desirable that $\hat{\mathbf{S}}$ and $\hat{\mathbf{X}}$ are close to their unquantized counterparts $\mathbf{S}$ and $\mathbf{X}$. We achieve this by restricting the standard deviation of $\hat{\mathbf{S}}$ or $\hat{\mathbf{X}}$ less than a given hyperparameter $\sigma_{\text{max}}^{(\mathbf{S})}$ or $\sigma_{\text{max}}^{(\mathbf{X})}$, which provides an upper bound for $\sigma_t^{(\mathbf{K})}$ or $\sigma_t^{(\mathbf{V})}$, respectively. Finally, we calculate the minimum number of quantization bits of KV cache for each token t that ensures the constraints of $\sigma_t^{(\mathbf{K})}$ or $\sigma_t^{(\mathbf{V})}$ are satisfied.

Value cache quantization. We start with the quantization of value cache, which is simpler in comparison. The value at index d in the output of the self-attention module is given by:

$$\hat{X}_d = \sum_{t=1}^{T} S_t \cdot \hat{V}_{td},$$

where the standard deviation of $\hat{V}_{td}$ is $\sigma_t^{(\mathbf{V})}$ for each $t = 1, \ldots, T$ and $d = 1, \ldots, D$. Therefore, the variance of $\hat{X}_d$ can be expressed as:

$$\sigma_d^{2(\mathbf{X})} = \sum_{t=1}^{T} S_t^2 \cdot \sigma_t^{2(\mathbf{V})},$$

which implies that the error of X_d accumulates from the error of each token. Ideally, the error contribution from each token, $S_t^2 \sigma_t^{2(\mathbf{V})}$, should be uniform across all tokens. Furthermore, to ensure that $\sigma_d^{(\mathbf{X})}$ remains below the given upper bound $\sigma_{\text{max}}^{(\mathbf{X})}$, the following constraint must be satisfied:

$$\sigma_t^{(\mathbf{V})} \leq \frac{1}{\sqrt{T}} \cdot \frac{\sigma_{\text{max}}^{(\mathbf{X})}}{|S_t|}.$$

This formula suggests that the error of value cache at each token should be inversely proportional to its corresponding attention value.

Key cache quantization. Due to the involvement of the Softmax function, the quantization of key cache is inherently more complex. The attention score of token t is given by:

$$\hat{S}_t = \frac{e^{\hat{A}_t}}{\sum_{i=1}^{T} e^{\hat{A}_i}}, \text{ where } \hat{A}_t = \sum_{d=1}^{D} Q_d \cdot \hat{K}_{td}.$$

Given that $\hat{A}_t$ is the sum of a collection of independent random variables, it follows the normal distribution $N(\mu_t^{(\mathbf{A})}, \sigma_t^{(\mathbf{A})})$ according to the central limit theorem, where:

$$\mu_t^{(\mathbf{A})} = \sum_{d=1}^{D} Q_d \cdot K_{td} \text{ and } \sigma_t^{2(\mathbf{A})} = \sum_{d=1}^{D} Q_d^2 \cdot \sigma_t^{2(\mathbf{K})}.$$

By definition, $e^{\hat{A}_t}$ follows the log-normal distribution $\text{LogNormal}(\mu_t^{(\mathbf{A})}, \sigma_t^{(\mathbf{A})})$ with the mean and variance being:

$$\mathbb{E}\left[e^{\hat{A}_t}\right] = e^{\mu_t^{(\mathbf{A})} + \sigma_t^{2(\mathbf{A})}/2},$$
$$\text{Var}\left[e^{\hat{A}_t}\right] = e^{2\mu_t^{(\mathbf{A})} + \sigma_t^{2(\mathbf{A})}} \left(e^{\sigma_t^{2(\mathbf{A})}} - 1\right).$$

Similarly, we expect the uniformity in error contribution from each token, implying that $e^{\hat{A}_t}$ should be identically distributed for all $t = 1, \ldots, T$. $\hat{S}_t$ is the ratio distribution between $e^{\hat{A}_t}$ at token t and the sum of $e^{\hat{A}_t}$ over all tokens. Applying a second-order Taylor expansion to the ratio [Seltman, 2012], the variance of $\hat{S}_t$ is approximately:

$$\begin{aligned}\sigma_t^{2(\mathbf{S})} \approx & \frac{1}{T^2} \cdot \left(1 - \frac{1}{T}\right) \cdot \frac{\left(\mathbb{E}\left[e^{\hat{A}_t}\right]\right)^2}{\text{Var}\left[e^{\hat{A}_t}\right]} \\ = & \frac{1}{T^2} \cdot \left(1 - \frac{1}{T}\right) \cdot \left(e^{\sum_{d=1}^{D} Q_d^2 \cdot \sigma_t^{2(\mathbf{K})}} - 1\right).\end{aligned}$$

To ensure that $\sigma_t^{2(\mathbf{S})}$ does not exceed the given upper bound $\sigma_{\text{max}}^{2(\mathbf{S})}$, we have:

$$\sigma_t^{2(\mathbf{K})} \leq \frac{1}{\sum_{d=1}^{D} Q_d^2} \cdot \log\left(\frac{T^3}{T-1} \cdot \sigma_{\text{max}}^{2(\mathbf{S})} + 1\right).$$

This inequality suggests that the upper bound of $\sigma_t^{(\mathbf{K})}$ is dependent on the squared norm of query tensors (*i.e.*, $\sum_{d=1}^{D} Q_d^2$), which varies with each inference. To address this variability, we precompute the distribution of the squared norm of query tensors and use the upper 10% quantile of this distribution in the formula. In this way, the inequality holds in 90% of cases.

Determining quantization bits. In the final step, we determine the quantization bits of KV cache for each token t ($B_t^{(\mathbf{K})}$ and $B_t^{(\mathbf{V})}$) based on the upper bound standard deviations of quantized KV cache ($\sigma_t^{(\mathbf{K})}$ and $\sigma_t^{(\mathbf{V})}$) calculated above. Owing to the symmetry between the key cache and value cache in this step, we only demonstrate the derivation using key cache $\mathbf{K}$.

To quantize key cache into $B_t^{(\mathbf{K})}$ bits, we uniformly split the range $\left[K_t^{\text{min}}, K_t^{\text{max}}\right]$ into $2^{B_t^{(\mathbf{K})}}$ segments, where K_t^{min} and K_t^{max} represents the min and max value of key cache at token t. We then round values in each segments to the corresponding midpoint. Using this quantization method, the quantization errors of key cache are uniformly distributed in $\left[-\Delta_t^{(\mathbf{K})}, \Delta_t^{(\mathbf{K})}\right]$, where $\Delta_t^{(\mathbf{K})}$ satisfies:

$$2 \cdot \Delta_t^{(\mathbf{K})} \cdot 2^{B_t^{(\mathbf{K})}} = K_t^{\text{max}} - K_t^{\text{min}}.$$

Given that the standard deviation of the above uniform distribution $\sigma_t^{(\mathbf{K})}$ is $\frac{1}{\sqrt{3}}\Delta_t^{(\mathbf{K})}$, we get:

$$B_t^{(\mathbf{K})} = \left\lceil \log_2\left(\frac{K_t^{\text{max}} - K_t^{\text{min}}}{2\sqrt{3} \cdot \sigma_t^{(\mathbf{K})}}\right)\right\rceil.$$

4.2 Methods

Attention score prediction. Our quantization method necessitates the attention scores that quantify the degree to which future tokens attend to preceding tokens. However, these attention scores are not available at the time of quantization. To address this limitation, we invoke the *persistence of importance* theory, which posits that the attention scores (*i.e.*, importance) of each token remain relatively constant (*i.e.*, persistence) throughout the process of token generation. Consequently, we can approximate future attention scores by using the current ones, and use it in the quantization formulas.

Nevertheless, as stated in Section 3.2, certain attention scores exhibit abrupt increments, which poses challenges to the quantization method, since quantization is irreversible, we can only quantize caches from higher to lower bits and not vice versa. To mitigate this issue, we propose *attention window*, a method that keeps track of the attention scores of each token and predicts the future attention scores as the maximum value within a window of the preceding n scores. This strategy ensures that aggressive quantization to lower bits is undertaken only after a sequence of consistently low attention scores has been observed, thereby being confident that future scores will remain low.

Outliers. Outliers of KV cache have a significant impact on the model's performance, as highlighted in Section 3.3. We define outliers as the values in the KV cache that exceed the $\alpha\%$ quantile at both the maximum and minimum ends, where α is a hyperparameter that controls the proportion of values deemed as outliers. To mitigate the impact of outliers, we introduce a mixed-precision quantization approach. Specifically, we keep outliers unquantized and store them in a sparse matrix at full precision.

Compared to quantizing KV cache uniformly, the benefits of this method are twofold: 1) the important outliers themselves are stored accurately without quantization error; 2) the quantization of the remaining values in KV cache can be more granular because the distribution range is significantly reduced without outliers. Our experiments also demonstrate that this method effectively avoids the performance degradation caused by quantization.

Integration in text generation process. Current autoregressive LLMs generate text token-by-token. Within each inference iteration with our quantization method integrated, the model takes in a new token, combined with the quantized KV cache of previous tokens, and outputs the KV cache of the new token. We copy the unquantized new KV cache into CPU memory for future use, and calculate the quantization bits for all existing tokens using the previously derived formula. For tokens whose newly-calculated quantization bits are lower than the current bits, we further quantize them to the lower bits. For those otherwise, we take the unquantized KV cache from CPU memory and re-quantize it to the required bits. Although this method introduces additional transfers between CPU and GPU memory, our attention window technique can effectively reduce this overhead though cautious quantization strategy.

5 Evaluation

In this section, we first introduce the experiment setting then we present the results that show QAQ archives up to near $10\times$ compression of KV cache memory footprint with no compromise in accuracy.

Methods	Backbone model	Task	Compression ratio with less than 1% acc. drop
Scissorhands[Liu *et al.*, 2023]	OPT-6B	HellaSwag-Five shot	3
		PIQA-Five shot	5
		MathQA-Five shot	5
	OPT-13B	HellaSwag-Five shot	5
		PIQA-Five shot	5
		MathQA-Five shot	5
H2O[Zhang *et al.*, 2023]	OPT-30B	PIQA	5
		MathQA	5
QAQ	LLaMA 2-7B	HellaSwag-Zero shot	7.477
		PIQA-Zero shot	9.022
		MathQA-Zero shot	7.477
	LLaMA 2-13B	HellaSwag-Zero shot	8.394
		PIQA-Zero shot	9.024
		MathQA-Zero shot	7.888

Table 1: The experiment performance of QAQ vs. SOTA methods.

Model	Task	w/o outliers acc.	w/o outliers compression ration	w 1% outliers acc.	w 1% outliers compression ration
LLaMA 2-7B	HellaSwag	0.572	7.981	0.722	7.629
	PIQA	0.707	7.695	0.763	7.333
	MathQA	0.250	7.969	0.279	7.497
LLaMA 2-13B	HellaSwag	0.660	7.938	0.766	7.583
	PIQA	0.737	7.614	0.789	7.223
	MathQA	0.257	7.928	0.287	7.453

Table 2: The ablation result of outliers in QAQ.

5.1 Experiment Settings

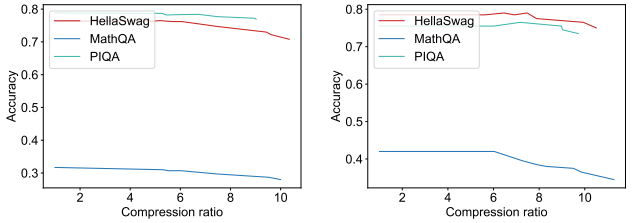
Our experiments are based on a representative LLM model family, LLaMA 2 with model sizes ranging from 7 billion and 13 billion. The outliers ratio is set as 1%, and the size of attention window is set as 5. We compare the accuracy of QAQ-LLaMA against the original LLaMA on a number of downstream tasks: HellaSwag [Zellers *et al.*, 2019], MathQA [Amini *et al.*, 2019], PIQA [Bisk *et al.*, 2020]. The evaluation uses a similar architecture as lm-eval-harness [Gao *et al.*, 2021] with zero shot in every task, all experiments are conducted on a server with 8 NVIDIA V100 32GB GPUs.

5.2 Compression Ratio vs. Accuracy

We present the experimental evaluation results depicting the variations in accuracy and compression ratio for QAQ in Figure 3. The compression ratio is defined as the average ratio of the quantized KV cache size to the original KV cache size, where $1\times$ denotes the original uncompressed LLaMA model. For both 7B and 13B model specifications, a relatively smooth curve is observed in the graph across four commonly employed downstream tasks. Even with compression ratios reaching up to $8\times$, there is minimal impact on the model's performance. Remarkably, at compression ratios as high as $10\times$, the overall performance of the model remains exceptionally high. Notably, in tasks such as PIQA, the $10\times$ compressed model's performance approaches that of the uncompressed model, indicating the robust capability of QAQ to significantly compress the KV cache size without compromising performance.

5.3 Comparison

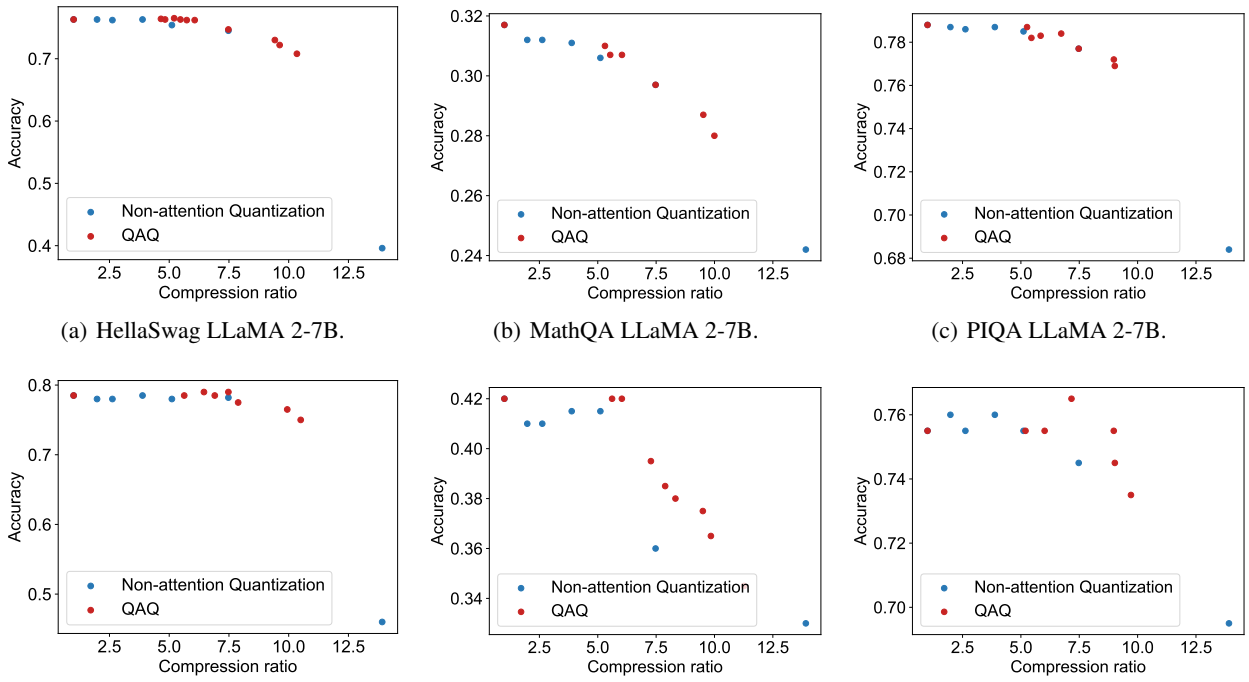
We conducted a comparative analysis with existing state-of-the-art compression methodologies. In the realm of model quantization, the widespread practice of uniformly quantizing parameters has exhibited notable compression efficacy



(a) LLaMA 2-7B Zero shot. (b) LLaMA 2-13B Zero shot.

Figure 3: Experiment performance of QAQ.

[Frantar *et al.*, 2022a]. Considering the pivotal role played by outliers during quantization, we retained outliers within the framework of uniform quantization. The performance evaluation of QAQ is juxtaposed with that of the uniform quantization approach, as illustrated in Figure 4. The compression ratio in the scatter plot is determined by the ratio of the compressed KV cache size to the original KV cache size. Closer proximity of data points to the upper-right corner of the figure indicates higher compression ratios and better performance guarantees, signifying superior overall performance. It is evident that the envelope formed by the QAQ, which is colored in red, data points encompasses the blue data points across all tasks in both models. This implies that the LLaMA models quantized using QAQ, whether in the 7B or 13B parameter specifications, exhibit significantly superior performance in the four downstream tasks compared to uniform quantization. This underscores the exceptional efficacy of the QAQ quantization strategy. For the existing SOTA techniques in compressing KV cache, we present a comparative analysis of QAQ in Table 1. It is evident that QAQ achieves a notable $1.6 - 1.8\times$ improvement in lossless compression ratio



(a) HellaSwag LLaMA 2-7B. (b) MathQA LLaMA 2-7B. (c) PIQA LLaMA 2-7B.

(d) HellaSwag LLaMA 2-13B. (e) MathQA LLaMA 2-13B. (f) PIQA LLaMA 2-13B.

Figure 4: Experiment result of QAQ v.s. non-attention quantization.

across multiple downstream tasks when compared to the current SOTA methods. The results substantiate the outstanding performance of QAQ.

5.4 Ablations

To verify the crucial role of outliers in quantization, we conducted ablation experiments while keeping the remaining experimental conditions unchanged.

Outliers. To evaluate the outliers' role in the quantization method, we conduct the ablation experiment on the outliers. The experiments were conducted on the same three downstream tasks, distinguishing between two groups: one where outliers were treated separately and another where no special treatment was applied. The presence of outliers was determined based on the numerical distribution. The experimental results were averaged over 10 trials and are presented in Table 2. The experimental results indicate that outliers have a substantial impact on KV cache quantization. In cases where outliers are not handled separately, the model's performance on downstream tasks experiences a significant decline of $12\% - 26\%$. Conversely, treating outliers individually incurs only a 4% additional overhead on the compression ratio. This demonstrates the efficient and accurate handling of outliers by QAQ.

Attention window size. To verify the importance of handling exceptional cases in quantization, we conducted ablation experiments to investigate the impact of treating such cases differently. The experiments are conducted on the same three downstream tasks, involving two groups: one with a specified window size and the other without, where the absence of a specific setting implies a default window size of 1. The results were averaged over 10 trials and are presented in Table 3. The experimental results are presented in Table 3. The findings indicate that, across the three downstream tasks, handling exceptional cases results in an improvement of approximately $2\% - 4\%$ in performance. This further advances the performance of QAQ.

Model	Task	Attention window size	Acc.	Compression ratio
LLaMA 2-7B	HellaSwag	1	0.722	7.628
		5	0.730	7.377
	PIQA	1	0.755	7.820
		5	0.778	6.797
	MathQA	1	0.276	7.633
		5	0.284	7.331
LLaMA 2-13B	HellaSwag	1	0.766	7.583
		5	0.772	7.340
	PIQA	1	0.788	7.692
		5	0.794	6.802
	MathQA	1	0.283	7.588
		5	0.295	7.321

Table 3: The ablation result of attention window in QAQ.

6 Conclusion

Inspired by three key insights, we propose a quality adaptive quantization scheme, QAQ, for the KV cache to reduce its memory footprint. Our method demonstrates memory reduction of $10\times$ in the KV cache with neglectable loss of accuracy. The superior performance achieved by QAQ allows the model to accommodate longer contextual inputs, creating new possibilities for a broader range of applications.

References

[Amini *et al.*, 2019] Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*, 2019.

[Beeching *et al.*, 2023] Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. Open LLM leaderboard. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard, 2023.

[Bisk *et al.*, 2020] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

[Brown *et al.*, 2020] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[Chang *et al.*, 2023] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Kaijie Zhu, Hao Chen, Linyi Yang, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *arXiv preprint arXiv:2307.03109*, 2023.

[Child *et al.*, 2019] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[Dettmers *et al.*, 2022] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

[Frantar *et al.*, 2022a] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

[Frantar *et al.*, 2022b] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Optq: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2022.

[Gao *et al.*, 2021] Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*, 2021.

[Ge *et al.*, 2023] Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for LLMs. *arXiv preprint arXiv:2310.01801*, 2023.

[Lee *et al.*, 2023] Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. OWQ: Lessons learned from activation outliers for weight quantization in large language models. *arXiv preprint arXiv:2306.02272*, 2023.

[Lin *et al.*, 2023] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

[Liu *et al.*, 2023] Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. *arXiv preprint arXiv:2305.17118*, 2023.

[Pope *et al.*, 2023] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5, 2023.

[Seltman, 2012] Howard Seltman. Approximations for mean and variance of a ratio. *unpublished note*, 2012.

[Touvron *et al.*, 2023] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[Yao *et al.*, 2022] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183, 2022.

[Zellers *et al.*, 2019] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[Zhang *et al.*, 2022] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[Zhang *et al.*, 2023] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. *arXiv preprint arXiv:2306.14048*, 2023.

[Zhu *et al.*, 2023] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*, 2023.