WKVQuant: Quantizing Weight and Key/Value Cache for Large Language Models Gains More

Anonymous ACL submission

Abstract

Large Language Models (LLMs) face significant deployment challenges due to their substantial memory requirements and the computational demands of auto-regressive text generation process. This paper addresses these challenges by focusing on the quantization of LLMs, a technique that reduces memory consumption by converting model parameters and activations into low-bit integers. We critically analyze the existing quantization approaches, identifying their limitations in balancing the accuracy and efficiency of the quantized LLMs. To advance beyond these limitations, we propose *WKVQuant*, a PTQ framework especially designed for quantizing weights and the key/value (KV) cache of LLMs. Specifically, we incorporates past-only quantization to improve the computation of attention. Additionally, we introduce two-dimensional quantization strategy to handle the distribution of KV cache, along with a cross-block reconstruction regularization for parameter optimization. Experiments show that *WKVQuant* achieves almost comparable memory savings to weight-activation quantization, while also approaching the performance of weight-only quantization.

1 Introduction

Large language models (LLMs) such as GPT (Brown et al., 2020; Ouyang et al., 2022), OPT (Zhang et al., 2022), and LLaMA (Touvron et al., 2023a,b) are essential in natural language processing, demonstrating unparalleled abilities to understand and generate text. However, their large size poses significant challenges for deployment. Firstly, the large number of weights in LLMs consumes a considerable amount of memory. For instance, the LLaMA-13b model requires approximately 26GB of memory when stored in FP16 format, which can only be accommodated by high-end GPUs. Secondly, due to the auto-regressive nature of text generation in LLMs, it is common practice to store certain intermediate results for reuse

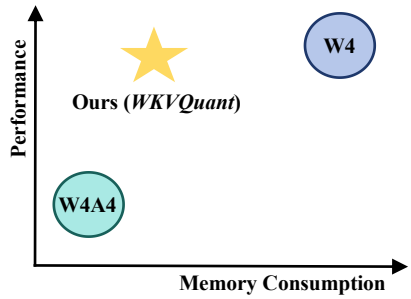

Figure 1: Memory-Performance curve.

during iteration to avoid redundant computations. These stored data, known as key/value cache (KV cache), occupy memory that increases as the batch size and sequence length grow (Miao et al., 2023). To address the issue of excessive memory usage in LLMs, quantization has emerged as a widely adopted solution (Zhao et al., 2023). By quantizing tensors into low-bit integers to represent numerical values, the memory footprint can be drastically reduced by 2-8 times. Existing quantization methods for LLMs can be categorized into two types, including weight-only quantization and weight-activation quantization (Zhao et al., 2023). As the KV cache is not quantized, weight-only quantization methods have a minimal impact on model accuracy, but is not effective enough at reducing memory consumption. In contrast, the weight-activation methods quantize both the weights and activations, including the KV cache. However, it suffers from a significant drop in accuracy. Both of these quantization paradigms have their merits and drawbacks, making it infeasible to simultaneously achieve the benefits of both approaches.

To overcome the dilemma between the accuracy and efficiency of the quantized LLMs, we consider the activation quantization on a finer-grained scale. The activations in LLMs include KV cache and other temporary activations. In this paper, we demonstrate that it is not cost-effective to quantize temporary activations. Specifically, ❶ our analysis

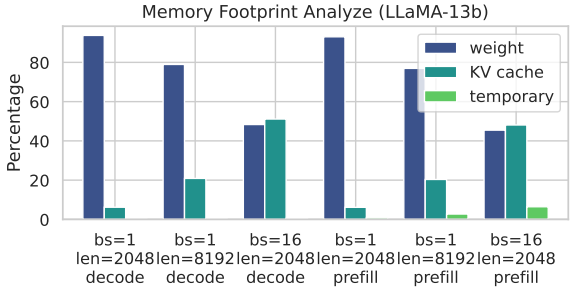

Figure 2: The memory usage proportions of weight, KV cache, and temporary storage on LLaMA-2-13b.

reveals that the temporary activation of memory is short-lived, as the memory content generated by one Transformer layer can be completely overwritten by the next layer. The memory footprint of temporary activations is significantly smaller compared to the memory usage of weights and the KV cache as shown in Figure 2. ❷ Temporary activations is highly sensitive to quantization, leading to high accuracy drop (Dettmers et al., 2022; Xiao et al., 2022). ❸ In decoding phase of LLM, as the computation is bound by memory access of weights and KV cache, quantizing temporary activations to utilize low-precision arithmetic units does not significantly reduce inference latency. Therefore, exclusive quantization of weights and KV cache is a beneficial trade-off between accuracy and memory savings. Existing methods are not designed for the exclusive quantization of weights and KV cache, which can lead to a decrease in accuracy. In this paper, we propose the *WKVQuant* framework, which is specifically designed for quantizing weights and KV cache. ❶ We introduce a new approach called Past Only Quantization (POQ) to enable higher precision in the Attention mechanism. Instead of discarding the full-precision Key and Value values (KV) after quantization, we temporarily retain them. This allows us to use unquantized KV during matrix multiplication in the Attention mechanism, improving the accuracy of Attention computation. In the prefill phase, POQ allows the network to achieve the same level of accuracy as weight-only quantization. In the decode phase, POQ combines the current unquantized KV with the previously quantized KV cache for prediction, improving the prediction accuracy. ❷ We observe significant variations in numerical values between channels and tokens in KV cache. Towards this issue, we propose Two-dimensional Quantization (2D-Quantization). Static channel smoothing aligns large values between channels, while dynamic token-wise quantization addresses variations between tokens. ❸ We discover a bias in the reconstruction loss function used for parameter optimization in the existing PTQ method (Shao et al., 2023). So we further introduce cross-block reconstruction regularization to reduce such bias and obtain better quantized parameters and smoothing parameters, thereby reducing quantization errors.

We evaluate the performance of *WKVQuant* on the LLaMA (Touvron et al., 2023) and LLaMA-2 (Touvron et al., 2023) models. Our experiments demonstrate that the W4KV4 quantized network with *WKVQuant* achieves significantly higher accuracy compared to the W4A4 quantized network. Specifically, when generating short sequences, W4KV4 performs equally well as the W4 quantization. For long sequence generation, W4KV4 exhibits performance close to that of W4 quantization. Meanwhile, the memory consumption of W4KV4 is nearly identical to that of W4A4.

In conclusion, our proposed *WKVQuant* provides a promising trade-off between accuracy and efficiency as shown in Figure 1.

2 Related Work

Existing quantization methods for LLMs can be classified into two types: weight-only quantization and weight-activation quantization (Zhao et al., 2023).

Weight-only Quantization for LLMs. For weight-only quantization, some works make efforts in the realm of Quantization-Aware Training (QAT), LLM-QAT(Liu et al., 2023b) innovatively tackles the challenges in acquiring training data for LLMs by leveraging pre-trained models for data-free distillation. Moreover, works such as QLORA (Dettmers et al., 2023a), PEQA (Kim et al., 2023a), QA-lora (Xu et al., 2023) and LoftQ (Li et al., 2023b) leverage Parameter Efficient Fine-Tuning (PEFT) techniques on performing fine-tuning tasks while achieving model compression. For Post-Training Quantization (PTQ) on LLMs(Kim et al., 2023b; Dettmers et al., 2023b; Shang et al., 2023), GPTQ (Frantar et al., 2022) and QuIP (Chee et al., 2023) achieve high compression rates by optimizing matrix multiplications operation and propose a novel layer-wise quantization solution. Besides, AWQ (Lin et al., 2023) and OWQ (Lee et al., 2023) take into account the impact of activation outliers on weight quantiza-

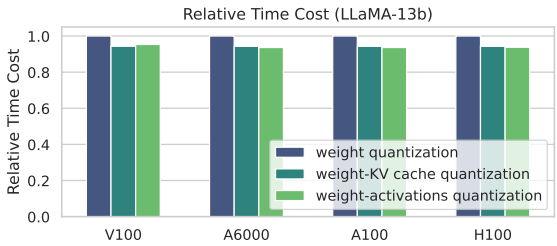

Figure 3: Relative time cost to generate 2048 tokens for a 2048 length input prompt using different quantization settings on LLaMA-2-13b (batchsize=1).

tion, showing quantization performance improvements. PB-LLM (Shang et al., 2023) exploits the salient-weight property of LLM and achieves extreme quantization to the lowest possible bit. While weight-only quantization can alleviate the computational burdens, its impact on memory usage and acceleration is still limited compared to weight-activation quantization.

Weight-activation Quantization for LLMs. Different with weight-only quantization methods, the weight-activation quantization methods (Wei et al., 2022, 2023; Li et al., 2023a; Yuan et al., 2023b) quantize both the weights and activations, including the KV cache. The most challenge for quantizing activations lies in outliers, which often lead to great quantization error. To tackle this issue, ZeroQuant (Yao et al., 2022) proposes a fine-grained hardware-friendly quantization scheme for both weight and activations. SmoothQuant (Xiao et al., 2022) migrates the quantization difficulty from activations to weights with a mathematically equivalent transformation (i.e., per-channel scaling). OmniQuant (Shao et al., 2023) further enhances performance by training the quantization parameters. RPTQ (Yuan et al., 2023a) proposes grouped quantization after clustering similar channels to tackle the outliers. While these methods have mitigated the quantization error, they commonly focus on addressing outliers in intermediate computational results. In this study, our principal emphasis is on quantizing weights and KV cache with PTQ techniques. Particularly, to our knowledge, *WKVQuant* is the first method for tackling weight and KV cache quantization predicaments.

3 Method

In this section, we first analysis the memory consumption when deploying LLMs in Section 3.1 and present the proposed *WKVQuant* in Section 3.2.

	Weight	KV Cache	Temporary Act
INT8	4.89	4.88	4.92
INT4	4.99	5.27	785.56

Table 1: Perplexity on WikiText2 under quantizing each part of LLaMA-2-13b (perplexity of FP16 model is 4.88). Weight is quantized by GPTQ, and activations are dynamically per-token quantized. Act denotes activations.

We then describe the details of *WKVQuant* in the following sections.

3.1 Memory Consumption Analysis

Deploying LLMs faces a significant challenge due to their enormous memory consumption. The memory usage of LLMs is mainly contributed by three resources: ❶ the weights of LLMs need to be stored in memory. For example, LLaMA-13b with 13 billion weights occupies around 26GB of memory in FP16 format. ❷ Temporary activations are generated during inference. For instance, Transformer inputs are kept in memory until the residual connection is executed. ❸ For auto-regressive LLMs, caching key and value activations (KV cache) into memory is necessary for subsequent token generation.

We use LLMViewer (Yuan, 2024) to analyze memory consumption and show the results in Figure 2. The analysis reveals that the memory footprint of temporary activations is relatively small, especially in decode phase. This is due to their short lifespan, as their memory can be released after usage. In contrast, the memory allocated for the KV cache cannot be released until the completion of a full answer generation process, which involves multiple inference passes through the network. Moreover, the memory consumption of the KV cache increases with larger batch sizes and longer input sequences as the model needs to store more KV pairs.

3.2 Weight-KV Cache Quantization

As mentioned in Section 2, there are two primary quantization methods for LLMs: weight-only quantization and weight-activation quantization. Weight-only quantization can compress the model weights to 4 bits or lower while minimizing the impact on accuracy. However, these method still result in relatively higher memory usage due to lack of activation quantization. Differently, weight-activation quantization aims to minimize the memory usage of LLMs to the greatest extent possible.

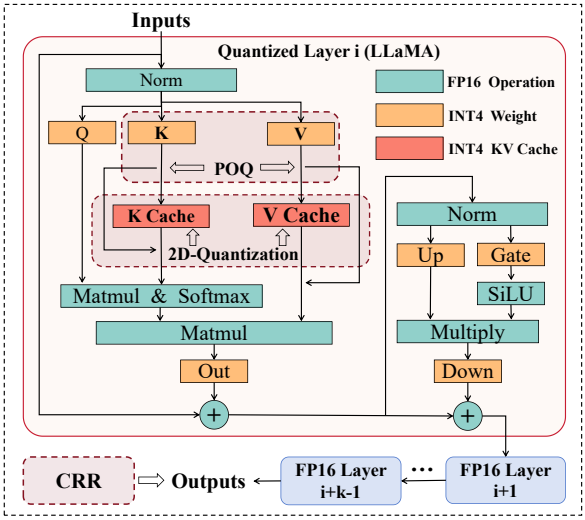

Figure 4: The framework of *WKVQuant*. Past Only Quantization (POQ) utilizes the original key and value state of the current token in the forward process for maximizing the preservation of original features. 2D-Quantization is applied for KV cache, which suppresses the outliers in both channel and token dimension. Cross-block Reconstruction Regularization (CRR) benefits training parameters by computing a more proper loss from a more global perspective.

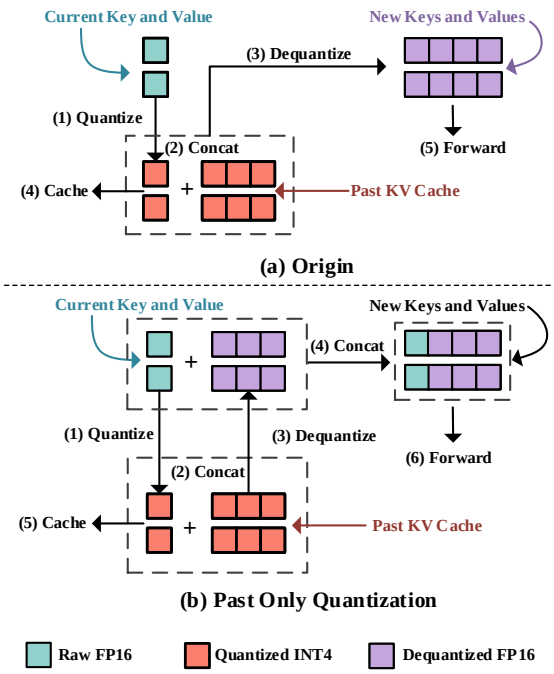

Figure 5: Comparison between original quantization method and Past Only Quantization (POQ) for KV cache. POQ utilizes the current key and value in raw full-precision and only quantizes the past KV cache.

However, quantizing activations has a larger impact on model accuracy compared to weight-only quantization. Furthermore, we opt to quantize the representative LLaMA-2-13b model and evaluate perplexity on the WikiText2 (Merity et al., 2016) dataset to investigate the impact of each part. As shown in Table 1, we observe that both 4-bit quantization of KV cache and temporary activations lead to decreased network performance, with temporary activations having a particularly significant drop.

Based on above observations, we believe that exclusively quantizing the weights and KV cache is a favorable choice. However, not quantizing temporary activations has two drawbacks: increased memory usage and the inability to utilize faster computation units, such as Nvidia's INT8 acceleration units. Nevertheless, these drawbacks do not significantly impact network inference. Firstly, as discussed earlier, temporary activations occupy minimal memory. Secondly, the unavailability of faster computation units does not greatly affect the inference speed. Figure 3 shows that the time cost of weight-KV cache quantization is nearly the same as weight-activation quantization. This is because the inference of LLMs is primarily constrained by memory access. Therefore, exclusive quantization of weights and KV cache is a beneficial trade-off between retaining model accuracy and achieving memory savings when compared to quantizing both weights and activations.

To achieve this trade-off, we propose a Post-Training Quantization (PTQ) framework, *WKVQuant*, which is specifically tailored for quantizing weights and KV cache of LLMs. Notably, to our knowledge, WKVQuant is the first method developed to address the exclusive quantization of weights and KV cache. The overview of *WKVQuant* is shown in Figure 4. Our *WKVQuant* comprises three main features: Past-Only Quantization (POQ) to enhance attention computation, 2D-Quantization strategy to handle the distribution of KV cache, and cross-block reconstruction regularization for parameter optimization. We will describe the details in the following section.

3.3 Past Only Quantization

In auto-regressive token generation, it is a common practice to store the keys and values of each layer into cache. This KV cache serves as input for the next token generation process, reducing redundant computations between steps. While this computation optimization is beneficial for LLMs, it inevitably results in a higher memory footprint, especially when dealing with large batch size or long input sequence (see Figure 2). Furthermore, as shown in Figure 5. (a), we can observe that in the

existing quantization method, the current key and value undergoes quantization and de-quantization before being passed into inference, which is a sub-optimal approach.

To this end, we propose a novel inference process called Past Only Quantization (POQ) to preserve the accuracy of the current key and value. As shown in Figure 5.(b), POQ does not quantize the representations generated from the current token. Instead, it uses the original full-precision representations when merging them with the de-quantized past KV cache during the forward process. Only the past KV loaded from KV cache are quantized in current decoding inference phase. By using unquantized KV during matrix multiplication in the Attention mechanism, POQ enhances the accuracy of Attention computation. During the prefill phase, POQ enables the network to attain the same level of accuracy as weight-only quantization since all the KV used in Attention are full-precision. In the decode phase, POQ combines the current unquantized KV with the previously quantized KV cache for prediction, thereby improving the prediction accuracy.

3.4 Two-dimensional Quantization

Previous studies have highlighted significant variations in numerical values among activation channels (Xiao et al., 2022; Wei et al., 2023). Similarly, we have observed substantial variations between channels and tokens in the KV cache. Quantizing them directly using the same parameters leads to substantial quantization errors. To tackle this issue, we introduce Two-dimensional Quantization. This approach employs static channel smoothing to align large values across channels and dynamic token-wise quantization to address variations between tokens.

The projection process of attention operation takes an input token sequence $\mathbf{X} \in \mathbb{R}^{T \times C_{\text{in}}}$, a weight matrix $\mathbf{W} \in \mathbb{R}^{C_{\text{in}} \times C_{\text{out}}}$, and a bias vector $\mathbf{B} \in \mathbb{R}^{1 \times C_{\text{out}}}$, where T, C_{in}, and C_{out} represent token length, input channels, and output channels, respectively. The mathematically projection process of attention operation can be expressed as: $\mathbf{Y} = \mathbf{X}\mathbf{W} + \mathbf{B}$, where $\mathbf{Y}$ represents KV cache in our settings.

Inspired by previous methods (Xiao et al., 2022; Wei et al., 2023), we introduce a learnable shifting parameter $\delta \in \mathbb{R}^{1 \times C_{\text{out}}}$ to align the centers of each channel, and also a learnable smoothing parameter $s \in \mathbb{R}^{1 \times C_{\text{out}}}$ to adjust each channel to appropriate range. By introducing those parameters, the projection process of attention operation can be transformed into:

$$\mathbf{Y} = \underbrace{(\mathbf{Y} - \delta) \oslash s}_{\tilde{\mathbf{Y}}} \odot s + \delta, \tag{1}$$

where'$\oslash$'and '$\odot$'represent element-wise division and multiplication, respectively. $\tilde{\mathbf{Y}}$ is the smoothed key or value after channel-wise shifting and scaling. The value difference among channels are significantly decreased by smoothing. Note that part of the parameters can be absorbed into the original linear weight and bias, which can be formulated as:

$$\tilde{\mathbf{Y}} = (\mathbf{X}\mathbf{W} + \mathbf{B} - \delta) \oslash s = \mathbf{X}\underbrace{\mathbf{W} \oslash s}_{\tilde{\mathbf{W}}} + \underbrace{(\mathbf{B} - \delta) \oslash s}_{\tilde{\mathbf{B}}}, \tag{2}$$

where $\tilde{\mathbf{W}}$ and $\tilde{\mathbf{B}}$ denote transformed weight and bias, respectively. After that, the quantization operation is applied to the smoothed weight and bias.

To further suppress the outliers in the token dimension, we utilize dynamic token-wise fine-grained quantization as formulated by:

$$\begin{aligned} Q_t(\tilde{\mathbf{Y}}) &= \text{clamp}(\lfloor \frac{\tilde{\mathbf{Y}} - m}{n} \rceil, -2^{N-1}, 2^{N-1} - 1), \\ \text{where } n &= \frac{\max(\text{abs}(\tilde{\mathbf{Y}} - m))}{2^{N-1}}, \; m = \text{mean}(\tilde{\mathbf{Y}}), \end{aligned} \tag{3}$$

where Q_t represents quantization process. $\lfloor \cdot \rceil$ denotes round operation and N is the target bit number. Note that the max, abs, and mean operations are dynamically calculated in the token dimension. By shifting each value of every token close to their average, the outliers have less effect on per-token quantization. In addition, the fine-grained quantization method can be applied to this step, which calculates the m and n in a finer-grained (such as 128 values as a group). This approach can further enhance the quantization performance.

For the weight quantization, we adopt the method in Omniquant (Shao et al., 2023), which is formulated as:

$$\mathbf{W}_q = \text{clamp}\left(\lfloor \frac{\mathbf{W}}{h} \rceil + z, 0, 2^{N-1}\right), \tag{4}$$

where $h = (\gamma \max(\mathbf{W}) - \beta \min(\mathbf{W}))/2^{N-1}$, $z = -\lfloor \beta \min(\mathbf{W})/h \rceil$. γ and β are trainable clipping parameters.

3.5 Cross-block Reconstruction Regularization

Based on above description, there are some parameters need to be determined, including the clipping parameters γ and β, as well as the smoothing parameters s and δ. Previous methods (Shao

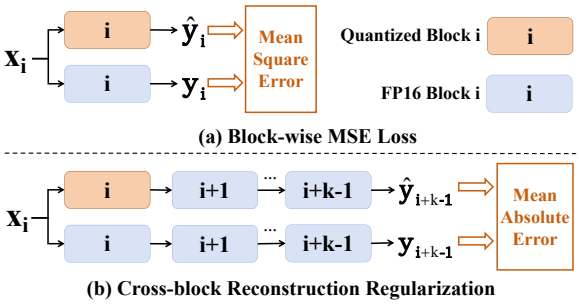

Figure 6: Comparison between Block-wise Mean Square Error (MSE) loss and Cross-block Reconstruction Regularization (CRR): CRR computes Mean Absolute Error (MAE) in the block dimension of $i+k-1$ while Block-wise MSE loss computes MSE in the block dimension of i.

et al., 2023) optimized these parameters using gradient descent. The approach involves calculating the local reconstruction loss between the output of the quantized Transformer layer and the full-precision Transformer layer, computing the gradients of these parameters with respect to the loss, and updating the parameters accordingly. However, it is worth noting that the local reconstruction loss introduces a bias and does not align with the final task loss (Yuan et al., 2022; Liu et al., 2023a; Zhou et al., 2024). This is due to the varying impact of each activation in the Transformer layer on the final network output. Additionally, it has been observed that outliers within the activations significantly affect on the Mean Squared Error (MSE) loss, which is square-based.

We propose a method called Cross-block Reconstruction Regularization (CRR) to mitigate this issue without significantly increasing computational and memory overhead. As shown in Figure 6, we pass the output of the Transformer layer to the subsequent layers, and then compute the loss by calculating the difference between the output results of the subsequent layers. Compared to the block-wise MSE loss, CRR provides a closer approximation to the error of network's final output, resulting in a smaller bias. Besides, to reduce the impact of outliers, we use the Mean Absolute Error (MAE) as a substitute for MSE, which helps to minimize the amplification of errors caused by quantizing outliers. The loss of quantizing the i-th block (Qblock_i) can be obtained by computing Mean Absolute Error (MAE) between quantized outputs $\hat{y}_{i+k-1}$ and full-precision outputs y_{i+k-1}, where k denotes the number of used blocks, x_i denotes the raw inputs for the original i-th decoder block_i. The parameters are optimized using the following equation:

$$\underset{\gamma,\beta,s,\delta}{\text{argmin}}\, \text{MAE}(\hat{y}_{i+k-1}, y_{i+k-1}). \tag{5}$$

4 Experiments

In this section, we introduce the detailed experimental settings as well as demonstrate the effectiveness of the proposed *WKVQuant*.

4.1 Experimental Settings

Models. We evaluate our *WKVQuant* on LLaMA (Touvron et al., 2023a) and LLaMA-2 (Touvron et al., 2023b) models (i.e., LLaMA-2-7B, LLaMA-2-13B, LLaMA-7B, and LLaMA-13B).

Baselines. We perform our *WKVQuant* in W4KV4 (quantizing weights to 4 bit and keys/values to 4 bit) setting. We also display results on OmniQuant (Shao et al., 2023) in W4A4 (quantizing weights to 4 bit and activations to 4 bit) setting and on GPTQ (Frantar et al., 2022) in W4 setting. Since there is no work for weight-KV/cache quantization before, we combine the scaling parameter for weights in W4 OmniQuant and the scaling parameter between query and key states in W4A4 OmniQuant to formalize so-called OmniQuant† in W4KV4 setting for comparison. The weight quantization in all methods are performed with group size of 128.

Calibration. The calibration dataset contains 128 randomly selected 2048-token segments from WikiText2 (Merity et al., 2016). The scaling factor s and shifting factor δ are initialized by computing the maximum absolute value and the mean value of corresponding feature representation across channels in the calibration dataset, respectively. The group size for KV cache quantization is set as 128. The learnable clipping factors γ and β are initialized as in OmniQuant. To optimize the learnable parameters, we utilize AdamW optimizer with zero weight decay. The learning rate for two-dimensional quantization (2D-Quantization) and learnable weight clipping is set as 5e−4 and 1e−2, respectively. We utilize Cross-block Reconstruction Regularization (CRR) for optimizing and the hyper-parameter k is fixed to 5 (see A.1). The optimization process is facilitated on a single Nvidia A100 GPU, utilizing a batch size of 1 over 5 epochs. Since the KV cache is not used in our calibration dataset, we deactivate past only quantization (POQ) while training and activate POQ after training.

Model	Method	Setting	Longtext avg↑	Zero-shot avg↑	WikiText2 ppl↓	PTB ppl↓	C4 ppl↓
LLaMA-2-7B	-	FP16	34.69	58.30%	5.47	37.91	7.26
	GPTQ	W4	34.71	57.74%	5.62	241.52	7.46
	OmniQuant	W4A4	11.16	45.44%	14.34	10304.68	20.18
	OmniQuant†	W4KV4	31.34	56.75%	6.09	64.39	8.98
	WKVQuant	W4KV4	34.48	58.38%	5.64	38.85	7.49
LLaMA-2-13B	-	FP16	34.12	61.32%	4.88	50.93	6.72
	GPTQ	W4	34.06	60.55%	4.99	50.25	6.84
	OmniQuant	W4A4	16.35	47.46%	12.39	263.60	17.51
	OmniQuant†	W4KV4	31.72	59.82%	5.18	55.95	7.30
	WKVQuant	W4KV4	32.52	60.34%	5.00	52.36	6.89
LLaMA-7B	-	FP16	34.80	57.68%	5.67	41.15	7.34
	GPTQ	W4	33.54	57.30%	5.83	43.70	7.51
	OmniQuant	W4A4	7.83	46.53%	11.58	231.46	16.19
	OmniQuant†	W4KV4	28.80	56.84%	6.25	51.96	8.17
	WKVQuant	W4KV4	33.86	57.91%	5.80	44.02	7.54
LLaMA-13B	-	FP16	36.02	60.60%	5.09	28.09	6.79
	GPTQ	W4	35.14	60.10%	5.19	29.25	6.90
	OmniQuant	W4A4	11.55	43.56%	11.18	115.38	16.37
	OmniQuant†	W4KV4	28.79	58.92%	5.64	32.78	7.86
	WKVQuant	W4KV4	35.50	60.44%	5.21	27.74	6.93

Table 2: The overall experimental results. Our results are shown in gray line.

Model	Method	Setting	Qasper↑	2WikiMultihopQA↑	HotpotQA↑	TriviaQA↑	LCC↑	Longtext avg↑
LLaMA-2-7B	-	FP16	7.81	7.53	8.44	84.21	65.49	34.69
	GPTQ	W4	7.68	9.35	8.55	84.88	63.12	34.71
	OmniQuant	W4A4	4.47	5.51	3.94	18.55	23.35	11.16
	OmniQuant†	W4KV4	6.18	8.69	7.07	77.81	56.96	31.34
	WKVQuant	W4KV4	7.57	9.64	8.31	84.11	62.79	34.48
LLaMA-7B	-	FP16	7.19	9.60	9.70	83.49	64.04	34.80
	GPTQ	W4	6.31	9.91	9.36	79.51	62.61	33.54
	OmniQuant	W4A4	2.77	4.57	3.54	13.19	15.11	7.83
	OmniQuant†	W4KV4	4.53	8.01	7.51	66.23	57.74	28.80
	WKVQuant	W4KV4	6.75	10.23	9.18	81.21	61.93	33.86

Table 3: Longtext scores. Results of LLaMA-2-13B and LLaMA-13B can be found in A.5.

Evaluated tasks and datasets. To eliminate the effectiveness of quantized LLMs under long input sequence circumstances, we perform experiments by using LongBench (Bai et al., 2023) framework on various datasets, i.e., Qasper (Dasigi et al., 2021), 2WikiMultihopQA (Ho et al., 2020), HotpotQA (Yang et al., 2018), TriviaQA (Joshi et al., 2017), LCC (Guo et al., 2023), and MultiFieldQA-en (Bai et al., 2023). The corresponding results are shown as Longtext score. We display the detailed information of the used datasets, which can be found in A.2. Following previous works, we also evaluate them on five zero-shot common sense reasoning tasks, i.e., PIQA (Tata and Patel, 2003), ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), and WinoGrande (Sakaguchi et al., 2021). Moreover, we basically evaluate the quantized models' perplexity (ppl) scores on WikiText2 (Merity et al., 2016), Pen Treebank (PTB) (Marcus et al., 1994) and C4 (Raffel et al., 2020).

4.2 Overall Results

As shown in Table 2, we conduct various experiments to evaluate our proposed *WKVQuant*. In general, we can find that methods in W4KV4 setting have obvious improvement compared to the W4A4 setting, highlighting the value of quantizing only the KV cache within the range of activations. As shown in Table 3, *WKVQuant* outperforms OmniQuant† in terms of average performance on the Longtext evaluation datasets, indicating the superior accuracy of our method for weigh-KV cache quantization. Notably, the evaluation processes of zero-shot accuracy and perplexity do not utilize KV cache, thus the results of *WKVQuant* on these tasks are turned to be under the W4 setting because of POQ. In this case, *WKVQuant* exhibits comparable performance with GPTQ, demonstrating the feasibility of co-optimizing the parameters for weights and KV cache quantization. It is worth mentioning that *WKVQuant* provides comparable performance with GPTQ in Longtext datasets, suggesting that

Quantization Method	Evaluation Score↑
FP16	24.42
RTN	2.3
+ LWC	3.88 (1.58↑)
+ 2D-Quant-channel	19.03 (16.73↑)
+ 2D-Quant-token	19.37 (17.07↑)
+ POQ	16.11 (13.81↑)

Table 4: The accuracy gain by utilizing each technique based on quantized LLaMA-2-7B model with RTN method.

Quantization Method	Evaluation Score↑
FP16	24.42
WKVQuant	25.29
- LWC	24.08 (1.21↓)
- 2D-Quant-channel	22.74 (2.55↓)
- 2D-Quant-token	23.14 (2.15↓)
- POQ	19.95 (5.34↓)

Table 5: The accuracy drop by deactivating each technique based on quantized LLaMA-2-7B model with *WKVQuant* method.

it can effectively quantize huge memory consumption caused by KV cache based on weight-only quantization, without introducing significant accuracy degradation. For detailed zero-shot results, refer to A.4.

4.3 Ablation Study

Ablation study for POQ and 2D-Quantization. To evaluate the effectiveness of the proposed POQ and 2D-Quantization, we first perform ablation study on LLaMA-2-7B and MultiFieldQA-en dataset with LongBench framework in W4KV4 setting considering that the methodology of POQ changes W4KV4 setting to W4 setting if KV cache is not used. We first apply different techniques used in *WKVQuant*,i.e., learnable weight clipping (LWC), static channel-wise smoothing (2D-Quant-channel), dynamic token-wise fine-grained quantization (2D-Quant-token), and POQ, to the RTN model separately to evaluate their contributions. Here, the RTN method refers to the vanilla round-to-nearest quantization. As shown in Table 4, each part of *WKVQuant* has positive effects on model performance especially POQ and 2D-Quantization. We then deactivate those used techniques from complete *WKVQuant* method to observe the accuracy change with their absence. Results in Table 5 shows that POQ holds paramount significance for performance, followed by 2D-Quantization.

Longtext Dataset	Block-wise MSE	CRR
Qasper↑	7.13	7.57
2WikiMultihopQA↑	9.41	9.64
HotpotQA↑	7.99	8.31
TriviaQA↑	84.60	84.11
LCC↑	61.24	62.79
Loingtext avg↑	34.07	34.48

Table 6: Ablation study of CRR.

	Model	FP	W4	W4KV4	W4A4
bs=1 len=2048	LLaMA-2-7b	14.0	4.3	3.5	3.5
	LLaMA-2-13b	27.1	8.0	6.8	6.8
bs=1 len=9012	LLaMA-2-7b	17.2	7.5	4.3	4.3
	LLaMA-2-13b	32.1	13.1	8.0	8.0
bs=16 len=2048	LLaMA-2-7b	30.1	20.4	7.5	7.5
	LLaMA-2-13b	52.2	33.2	13.1	13.1

Table 7: The memory consumption (GB) of decoding using different quantization settings.

Ablation study for CRR. Here, we ablate the effect of proposed Cross-block Reconstruction Regularization (CRR). As shown in Table 6, the proposed CRR is more effective than the commonly utilized block-wise MSE loss, which demonstrate the benefits training parameters by computing a more proper loss from a more global perspective.

4.4 Memory Consumption

We use LLMViewer (Yuan, 2024) to analyze memory consumption and show the results in Table 7. It can be observed that the weight-KV cache quantization has comparable memory savings with the weight-activation quantization, which means that *WKVQuant* improves the accuracy with almost negligibly increased memory.

5 Conclusion

In this paper, we have identified the limitations of existing quantization approaches in balancing accuracy and efficiency for LLMs. To address these limitations, we propose *WKVQuant*, a novel quantization methodology specifically designed for quantizing weights and the KV cache of LLMs. WKVQuant incorporates past-only quantization to optimize the improve the computation of attention. Additionally, we introduce two-dimensional quantization to handle the distribution of KV cache, along with a cross-block reconstruction loss for parameter optimization. Our proposed *WKVQuant* provides a promising trade-off between accuracy and efficiency, making LLMs more practical for deployment in resource-constrained environments.

Limitations

Large Language Models (LLMs) face significant problem when deploying because of the vast size. In this work, we analyze the memory consumption of different parts of LLMs (i.e., weights, Key/Value cache, and other temporary activations) and their corresponding quantization difficulties. From the analysis we know that the temporary activations are of low value to be quantized. We thus propose *WKVQuant*, which optimizes the exclusive quantization of weight and KV cache. Although this approach can both maintain model accuracy and reduce memory consumption, it is limited by not quantizing temporary activations.

Specifically, (1) this limitation means increased memory consumption since the temporary activations are remained in full-precision, especially when in prefill stage and when the batch-size/sequence length is large. We don't choose to quantize this part of activations because it consumes few memory space in most cases. (2) This limitation means that our approach can not utilize faster computation units, such as Nvidia's INT8 acceleration units. Weight-activation methods quantized all operation variables into the same bit-width, thus having a good implementation on the mentioned units. Although there should be a time cost difference, we claim that the gap is minimal because the inference of LLMs is primarily constrained by memory access.

In addition, we note that the cross-block reconstruction regularization will lead to an increase in the time required for optimizing quantization parameters. Specifically, for a 7b model, the process takes approximately 3 hours, while for a 13b model, it takes around 4 hours. However, it is worth mentioning that the process is typically performed only once before deployment. Therefore, we believe that the additional time is generally acceptable in the majority of cases.

References

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. 2023. Quip: 2-bit quantization of large language models with guarantees. *arXiv preprint arXiv:2307.13304*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610.

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

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023a. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2023b. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*.

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

Daya Guo, Canwen Xu, Nan Duan, Jian Yin, and Julian McAuley. 2023. Longcoder: A long-range pre-trained language model for code completion. *arXiv preprint arXiv:2306.14893*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. 2023a. Memory-efficient fine-tuning of

compressed large language models via sub-4-bit integer quantization. *arXiv preprint arXiv:2305.14152*.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. 2023b. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*.

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

Liang Li, Qingyuan Li, Bo Zhang, and Xiangxiang Chu. 2023a. Norm tweaking: High-performance low-bit quantization of large language models. *arXiv preprint arXiv:2309.02784*.

Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. 2023b. Loftq: Lora-fine-tuning-aware quantization for large language models. *arXiv preprint arXiv:2310.08659*.

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

Jiawei Liu, Lin Niu, Zhihang Yuan, Dawei Yang, Xinggang Wang, and Wenyu Liu. 2023a. Pd-quant: Post-training quantization based on prediction difference metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24427–24437.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023b. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*.

Mitch Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: Annotating predicate argument structure. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. 2023. Towards efficient generative large language model serving: A survey from algorithms to systems. *arXiv preprint arXiv:2312.15234*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*.

Yuzhang Shang, Zhihang Yuan, Qiang Wu, and Zhen Dong. 2023. Pb-llm: Partially binarized large language models. *ICLR*.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2023. Omniquant: Omnidirectionally calibrated quantization for large language models. *CoRR*, abs/2308.13137.

Sandeep Tata and Jignesh M Patel. 2003. Piqa: An algebra for querying protein data sets. In *15th International Conference on Scientific and Statistical Database Management, 2003.*, pages 141–150. IEEE.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. 2023. Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *arXiv preprint arXiv:2304.09145*.

Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. 2022. Outlier suppression: Pushing the limit of low-bit transformer language models. *NeurIPS*.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Julien Demouth, and Song Han. 2022. Smoothquant: Accurate and efficient post-training quantization for large language models. *arXiv preprint arXiv:2211.10438*.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhensu Chen, Xiaopeng Zhang, and Qi Tian. 2023. Qa-lora: Quantization-aware low-rank adaptation of large language models. *arXiv preprint arXiv:2309.14717*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *arXiv preprint arXiv:2206.01861*.

Zhihang Yuan. 2024. LLMViewer. https://github.com/hahnyuan/LLMViewer. Accessed: February 14, 2024.

Zhihang Yuan, Lin Niu, Jiawei Liu, Wenyu Liu, Xinggang Wang, Yuzhang Shang, Guangyu Sun, Qiang Wu, Jiaxiang Wu, and Bingzhe Wu. 2023a. Rptq: Reorder-based post-training quantization for large language models. *arXiv preprint arXiv:2304.01089*.

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. 2023b. Asvd: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*.

Zhihang Yuan, Chenhao Xue, Yiqi Chen, Qiang Wu, and Guangyu Sun. 2022. Ptq4vit: Post-training quantization for vision transformers with twin uniform quantization. In *Computer Vision – ECCV 2022*, pages 191–207, Cham. Springer Nature Switzerland.

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

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

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Sifan Zhou, Liang Li, Xinyu Zhang, Bo Zhang, Shipeng Bai, Miao Sun, Ziyu Zhao, Xiaobo Lu, and Xiangxiang Chu. 2024. Lidar-ptq: Post-training quantization for point cloud 3d object detection. *ICLR*.

A Appendix

A.1 Find the Best Number of Blocks for Cross-block Reconstruction Regularization

We propose the Cross-block Reconstruction Regularization for our *WKVQuant*, which computes Mean Absolute Error loss between the original and quantized outputs after several decoder blocks. To find out the best number of blocks, we perform ablation study designed for this hyper-parameter. Results are displayed in Table 8. From the table we can observe that the best number for k should be 5. So we fix it as 5 during the whole experiments.

k	WikiText2 ppl↓	PTB ppl↓	C4 ppl↓
1	5.7094	42.9600	7.5700
2	5.6758	43.7015	7.5517
3	5.6677	46.0000	7.5498
4	5.6668	41.9152	7.5417
5	5.6622	39.7035	7.5413
6	5.6642	40.7976	7.5437

Table 8: Results of perplexity on WikiText2, PTB, and C4 dataset by applying different number of blocks used in Cross-block Reconstruction Regularization. It can be observed from the table that setting k to 5 lead to the best results.

A.2 Detailed Information of the Used LongBench Datasets

To perform experiments on long texts (Longtexts), we utilize LongBench (Bai et al., 2023) framework to perfrom predictions on various datasets. Detailed information of the used datasets are shown in Table 9. The first four datasets each have 200 samples. LCC dataset has 500 samples and MultiFieldQA-en dataset has 150 samples.

Dataset	Task Type	Metric	Length
Qasper	Single-doc QA	F1	3619
2WikiMultihopQA	Multi-doc QA	F1	4887
HotpotQA	Multi-doc QA	F1	9151
TriviaQA	Few-Shot QA	F1	8209
LCC	Code	Edit Sim	1235
MultiFieldQA-en	Single-doc	F1	4559

Table 9: Description of the used LongBench datasets.

A.3 Implementation Details

To optimize the learnable parameters, we utilize AdamW optimizer with zero weight decay. The learning rate for two-dimensional quantization (2D-Quantization) and learnable weight clipping is set as 5e−4 and 1e−2, respectively. The optimization process is facilitated on a single Nvidia A100 GPU, utilizing a batch size of 1 over 5 epochs. Since the KV cache is not used in our calibration dataset, we deactivate past only quantization (POQ) while training and activate POQ after training.

A.4 Zero-shot Accuracy

See Table 10. This table can be seen as an extension to Table 2 as we display the zero-shot accuracy results on all the used datasets here.

A.5 Longtext Scores of LLaMA-2-13B and LLaMA-13B models

See Tabel 11. This table is a supplement to Table 3 as we display Longtext scores of LLaMA-2-13B and LLaMA-13B models here.

Model	Method	Setting	PIQA↑	ARC-Challenge↑	HellaSwag↑	WinoGrande↑	Zero-shot avg↑
LLaMA-2-7B	-	FP16	78.40%	39.84%	56.67%	67.24%	58.30%
	GPTQ	W4	78.34%	38.82%	56.06%	66.92%	57.74%
	OmniQuant	W4A4	66.15%	28.15%	42.04%	53.67%	45.45%
	OmniQuant†	W4KV4	77.09%	38.99%	54.17%	65.43%	56.75%
	WKVQuant	W4KV4	78.23%	40.78%	56.14%	67.48%	58.38%
LLaMA-2-13B	-	FP16	78.72%	45.56%	59.69%	69.69%	61.32%
	GPTQ	W4	78.83%	43.60%	59.24%	68.58%	60.55%
	OmniQuant	W4A4	67.02%	30.54%	44.83%	53.90%	47.46%
	OmniQuant†	W4KV4	78.07%	43.17%	58.22%	68.74%	59.82%
	WKVQuant	W4KV4	78.12%	43.94%	58.98%	68.75%	60.34%
LLaMA-7B	-	FP16	78.40%	38.22%	56.42%	66.92%	57.68%
	GPTQ	W4	78.18%	37.62%	56.11%	65.98%	57.30%
	OmniQuant	W4A4	66.97%	29.01%	43.63%	53.35%	46.53%
	OmniQuant†	W4KV4	77.63%	37.88%	55.02%	65.35%	56.84%
	WKVQuant	W4KV4	78.23%	39.76%	55.75%	65.58%	57.91%
LLaMA-13B	-	FP16	78.78%	43.94%	59.10%	70.08%	60.61%
	GPTQ	W4	78.29%	43.51%	58.50%	69.13%	60.10%
	OmniQuant	W4A4	64.25%	24.57%	41.88%	52.09%	43.56%
	OmniQuant†	W4KV4	78.45%	40.95%	57.37%	70.32%	58.92%
	WKVQuant	W4KV4	79.71%	43.00%	58.62%	71.19%	60.44%

Table 10: Zero-shot accuracy of all methods on PIQA, ARC-Challenge, HellaSwag, and WinoGrande dataset.

Model	Method	Setting	Qasper↑	2WikiMultihopQA↑	HotpotQA↑	TriviaQA↑	LCC↑	Longtext avg↑
LLaMA-2-13B	-	FP16	6.55	8.32	8.86	84.97	61.92	34.12
	GPTQ	W4	7.02	8.72	8.66	85.64	60.28	34.06
	OmniQuant	W4A4	4.71	5.41	4.70	28.96	37.98	16.35
	OmniQuant†	W4KV4	6.45	8.79	8.09	81.58	53.69	31.72
	WKVQuant	W4KV4	6.00	7.73	8.26	82.15	58.5	32.52
LLaMA-13B	-	FP16	8.16	9.88	9.69	85.50	66.87	36.02
	GPTQ	W4	6.97	10.00	9.54	84.05	65.16	35.14
	OmniQuant	W4A4	3.75	4.84	4.44	21.09	23.66	11.55
	OmniQuant†	W4KV4	5.35	8.31	8.33	66.14	55.84	28.79
	WKVQuant	W4KV4	6.71	9.47	9.04	86.60	65.69	35.50

Table 11: Longtext scores. Our results are shown in gray line.