LRQ: Optimizing Post-Training Quantization for Large Language Models by Learning Low-Rank Weight-Scaling Matrices

Anonymous ACL submission

Abstract

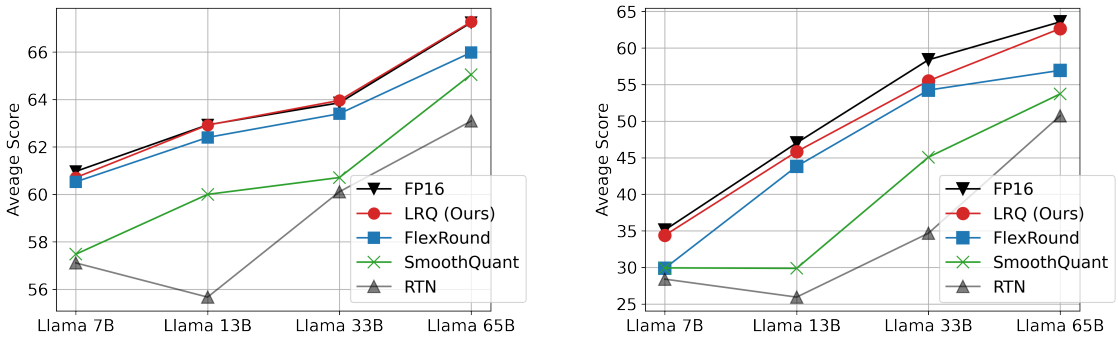
With the commercialization of large language models (LLMs), weight-activation quantization has emerged to compress and accelerate LLMs, achieving high throughput while reducing inference costs. However, existing post-training quantization (PTQ) techniques for quantizing weights and activations of LLMs still suffer from non-negligible accuracy drops, especially on massive multitask language understanding. To address this issue, we propose Low-Rank Quantization (LRQ) − a simple yet effective post-training weight quantization method for LLMs that reconstructs the outputs of an intermediate Transformer block by leveraging low-rank weight-scaling matrices, replacing the conventional full weight-scaling matrices that entail as many learnable scales as their associated weights. Thanks to parameter sharing via low-rank structure, LRQ only needs to learn significantly fewer parameters while enabling the individual scaling of weights, thus boosting the generalization capability of quantized LLMs. We show the superiority of LRQ over prior LLM PTQ works under (i) 8-bit weight and per-tensor activation quantization, (ii) 4-bit weight and 8-bit per-token activation quantization, and (iii) low-bit weight-only quantization schemes. Our code is available at Software.

1 Introduction

As ChatGPT and GPT-4 (OpenAI, 2023) have showcased unprecedented capabilities across various domains such as common sense reasoning, mathematical problem-solving, and coding proficiency, there has been an exponential surge in interest surrounding the development of Large Language Models (LLMs). This surge in interest has culminated in the recent release of cutting-edge LLMs like Llama (Touvron et al., 2023a), PaLM 2 (Google et al., 2023), and Llama 2 (Touvron et al., 2023b). Accordingly, serving LLMs has rapidly emerged as a significant concern in both academia and industry. This stems from the substantial memory footprint and considerable computational cost incurred when operating these language models with tens or hundreds of millions of parameters in FP16 format. Therefore, extensive efforts (Frantar et al., 2023; Liu et al., 2023b) such as quantization or pruning are underway to compress LLMs and provide efficient deployment. In particular, quantization has garnered considerable interest among LLM engineers and researchers because quantization aids in not just model compression but also inference acceleration.

LLM quantization techniques fall into two primary categories: weight-only quantization and weight-activation quantization. Weight-only quantization concentrates on enhancing memory-bound operations like matrix-vector multiplication by quantizing weights of LLMs into low-bit integers (e.g., 2-4 bits). With activations kept in FP16, weight-only quantization exhibits marginal accuracy degradation but is only effective in accelerating text generation inference for small batch sizes (e.g., a single batch). In contrast, weight-activation quantization aims to expedite computationally intensive operations, such as matrix-matrix multiplication, typically by quantizing both weights and activations of LLMs into 8-bit integers and employing INT8 GEMM kernels. This comprehensive quantization approach enables LLM serving for large batch sizes, thus enhancing LLM throughput and expediting LLM inference through integer matrix multiplication. Yet, it comes with the trade-off of potential non-negligible accuracy drop. While each approach boasts its own set of strengths and weaknesses, we first focus on weight-activation quantization on the grounds that achieving high-throughput LLM inference is important to handle a substantial volume of user requests in real time.

Recent studies (Dettmers et al., 2022; Yao et al., 2022; Xiao et al., 2022; Lee et al., 2023b; Liu et al., 2023a) have attempted to quantize both weights



(a) Common Sense Reasoning tasks

(b) Massive Multitask Language Understanding

Figure 1: (a) Zero-shot performance and (b) five-shot accuracy of Llama with 8-bit per-channel asymmetric weight quantization and 8-bit per-tensor asymmetric static activation quantization, while keeping the KV cache in FP16.

and activations of LLMs. Among these works, only SmoothQuant (Xiao et al., 2022) and FlexRound (Lee et al., 2023b) demonstrated the potential for a hardware-efficient per-tensor static activation quantization scheme that can reduce the inference latency and memory usage by up to two-thirds and half respectively compared to FP16 baselines as thoroughly elucidated in Xiao et al. (2022). Given the compelling advantages of this scheme, we also stick mainly to per-tensor static activation quantization, with a primary focus on preventing non-negligible performance degradation, one of its key drawbacks, from occurring.

Despite promising results that SmoothQuant and FlexRound yielded, they still possess inherent limitations on enhancing model accuracy when using per-tensor static activation quantization. Although SmoothQuant is a potent technique for alleviating the difficulty of quantizing activation outliers, it uniformly divides activations in each channel and multiplies the weights in the corresponding input channel by some constant. Since such an uniform per-channel smoothing transformation can only scale the weights collectively per channel, not individually, SmoothQuant may lead to non-negligible accuracy loss after quantization for certain models as in Figure 1. On the other hand, as FlexRound learns a separate scale for each weight and thus enables flexible weight quantization based on individual characteristics of each weight, FlexRound can show marginal zero-shot accuracy drop on common sense reasoning tasks in Figure 1(a). However, as depicted in Figure 1(b), FlexRound falls short in performing well on massive multitask language understanding (MMLU), which necessitates problem-solving skills, specialized knowledge, as well as basic knowledge across diverse subjects. We empirically confirm that this phenomenon is because FlexRound has to learn too many scales relative to limited calibration samples due to the assignment of an independent scale to every weight.

To improve generalization performance on such a challenging benchmark, we propose a new post-training weight quantization approach, "Low-Rank Quantization (LRQ)", as a middle ground between SmoothQuant and FlexRound. LRQ is designed to minimize the mean squared error between the outputs of an intermediate FP16 Transformer block and those of its quantized counterpart with respect to low-rank weight-scaling matrices instead of full weight-scaling matrices that involve as many scales as their associated weights. Using such low-rank matrices, we can reduce the number of learnable parameters effectively while maintaining the concept of scaling weights individually by sharing learnable parameters via low-rank structure. As a result, LRQ can attain comparable accuracy to FP16 baselines on both common sense reasoning tasks and MMLU for every Llama model as seen in Figure 1.

Our main contribution is threefold:

- We propose a new post-training weight quantization method coined LRQ that leverages low-rank weight-scaling matrices for intermediate Transformer block output reconstruction, which improves the generalization performance of quantized LLMs as in Figure 1.
- We provide empirical insights into the significance of reducing the number of learnable parameters and how the utilization of low-rank matrices to effectively decrease learnable parameters impacts the generalization ability of quantized LLMs.
- We validate the effectiveness of LRQ under a wide variety of quantization schemes (8-bit

weight and per-tensor activation quantization, 4-bit weight and 8-bit per-token activation quantization, and low-bit weight-only quantization) with marginal accuracy loss.

2 Method

In this section, we outline the post-training quantization (PTQ) background that our method, LRQ is based on, figure out the problem arising when quantizing LLMs, and formulate LRQ. Finally, we deepen an empirical understanding of how LRQ can improve generalization in quantized LLMs.

2.1 Background

Block-wise Reconstruction First of all, our method is based on block-wise reconstruction, which originates from BRECQ (Li et al., 2021) for the purpose of taking into account the intra-block dependency and has been widely used in QDrop (Wei et al., 2022), FlexRound (Lee et al., 2023b), and AQuant (Li et al., 2023) due to its efficacy to yield less generalization error than layer-wise reconstruction. As we concentrate on weight-activation quantization of LLMs that are generally Transformer-based models, the block-wise reconstruction process is applied to every Transformer block in the order of arrangement. To be more concrete, with a small set of calibration data, the objective of block-wise reconstruction is to find quantized weights $\widehat{\boldsymbol{W}}$ by minimizing the block reconstruction error $\|\boldsymbol{W}\boldsymbol{X} - \widehat{\boldsymbol{W}}\widetilde{\boldsymbol{X}}\|_2^2$ where $\boldsymbol{W}$ and $\boldsymbol{X}$ are the weights and inputs of a FP16 Transformer block while $\widetilde{\boldsymbol{X}}$ is the inputs of its quantized counterpart (i.e., the outputs of its immediately preceding Transformer block with all its previous Transformer blocks quantized).

FlexRound Among PTQ studies that take advantage of block-wise reconstruction, FlexRound shows the state-of-the-art performance for a wide variety of models ranging from computer vision models to large language models including Llama. In FlexRound, the formulation of $\widehat{\boldsymbol{W}}$ is written as

$$\widehat{\boldsymbol{W}} = \boldsymbol{s}_1 \left\lfloor \frac{\boldsymbol{W}}{\boldsymbol{s}_1 \odot \exp(\boldsymbol{S}_2)} \right\rceil, \qquad (1)$$

where $\boldsymbol{s}_1$ is a quantization step size, $\boldsymbol{S}_2$ is a weight-scaling matrix whose shape is exactly the same as that of $\boldsymbol{W}$, $\lfloor \cdot \rceil$ and $\exp(\cdot)$ indicate the rounding and exponential function, and $\odot$ and $/$ represent element-wise multiplication and division. Depending on the type of $\boldsymbol{W}$, some supplementary vectors

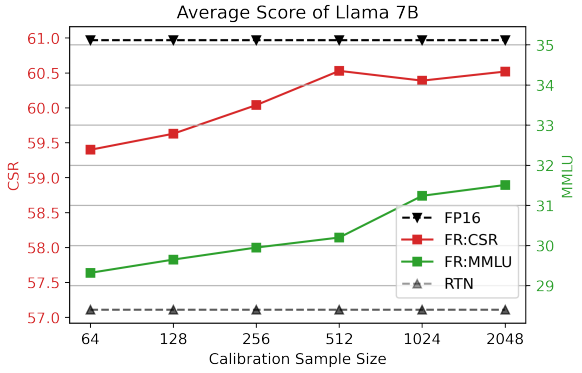


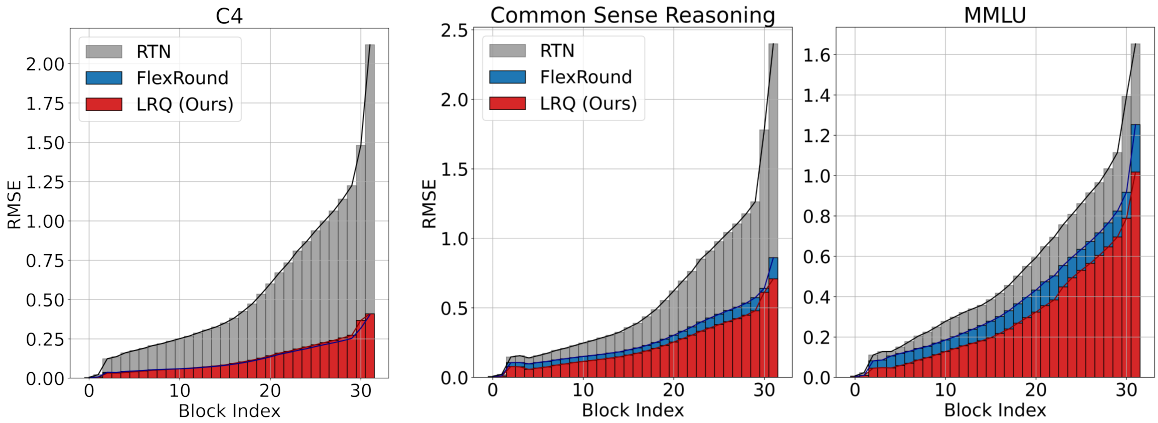
Figure 2: Zero-shot performance and five-shot accuracy of Llama 7B for FlexRound (FR) on common sense reasoning (CSR) tasks and MMLU according to the calibration sample size, with 8-bit per-channel asymmetric weight and 8-bit per-tensor asymmetric static activation quantization, while keeping the KV cache in FP16.

are added to $\boldsymbol{S}_2$, but we exclude these additional vectors to keep the expression uncluttered. At the beginning of learning, $\boldsymbol{S}_2$ is set to a zero matrix and $\boldsymbol{s}_1$ is initialized to $\arg\min_{\boldsymbol{s}_1} \|\boldsymbol{W} - \widehat{\boldsymbol{W}}\|_2^2$ to start learning from rounding-to-nearest (RTN). Then, both $\boldsymbol{s}_1$ and $\boldsymbol{S}_2$ are learned to minimize the block reconstruction error $\|\boldsymbol{W}\boldsymbol{X} - \widehat{\boldsymbol{W}}\widetilde{\boldsymbol{X}}\|_2^2$ with a small amount of calibration data as explained above. As FlexRound learns a separate scale for each weight, FlexRound can quantize each weight to one of not just the two nearest quantization grids but also more distant ones, based on individual characteristics of each weight. Nonetheless, as shown in Figure 1, quantized LLMs via FlexRound might exhibit reduced scores on challenging tasks like MMLU.

2.2 Motivation

We hypothesize that the failure to generalize well on challenging benchmarks like MMLU arises from the necessity of learning an individual scale for every weight with limited calibration samples. Now that $\boldsymbol{S}_2$ has as many learnable parameters as the size of $\boldsymbol{W}$ in Eq. 1, FlexRound's objective to achieve flexible weight quantization through the assignment of an independent scale to each weight may be deemed excessive when applied to LLM.

For instance, for Llama 7B, the smallest model in Llama, FlexRound has to learn more than 200 million scales with only just a few hundred or thousand calibration samples. FlexRound may be therefore prone to overfitting when quantizing LLMs. To resolve this issue, there might be two solutions: (i) increasing calibration samples, and (ii) decreasing learnable parameters. In the former case, as shown in Figure 2, the accuracy of FlexRound on



(a) Calibration sample (b) Unseen sample

Figure 3: Accumulated root mean square error (RMSE) between $\boldsymbol{W}\boldsymbol{X}$ and $\widehat{\boldsymbol{W}}\widetilde{\boldsymbol{X}}$ for RTN, FlexRound, and LRQ on (a) a calibration sample from the C4 dataset and (b) an unseen sample from common sense reasoning and MMLU benchmarks, ranging from the first Transformer block to the last Transformer block of Llama 7B. Here, weights and activations are quantized to 8-bit with per-channel asymmetric quantization and per-tensor asymmetric static quantization, while the KV cache remains in FP16. Note that RMSE tends to rise in line with the block index due to the presence of $\widetilde{\boldsymbol{X}}$ that accumulates quantization error resulting from previous quantized Transformer blocks.

MMLU increases as the calibration sample size grows larger. Yet, FlexRound still falls behind the FP16 baseline on MMLU by more than 3.5 percent, even when utilizing 2048 calibration samples, the maximum number we can use on a single NVIDIA A100-80GB GPU during the block-wise reconstruction process. Thus, we turn our focus toward reducing the number of learnable parameters.

2.3 Low-Rank Quantization

To reduce the number of learnable parameters, we decompose a weight-scaling matrix, $\boldsymbol{S}_2$, into a low-rank matrix before performing the reconstruction process. To be more specific, for $\boldsymbol{W} \in \mathbb{R}^{C_{out} \times C_{in}}$, $\boldsymbol{S}_2 \in \mathbb{R}^{C_{out} \times C_{in}}$ is factorized into $\boldsymbol{L}_2\boldsymbol{U}_2$ where $\boldsymbol{L}_2 \in \mathbb{R}^{C_{out} \times r}$ and $\boldsymbol{U}_2 \in \mathbb{R}^{r \times C_{in}}$ for $r < \min(C_{out}, C_{in})$. Additionally, we supplement $\boldsymbol{L}_2\boldsymbol{U}_2$ with a row vector, $\boldsymbol{r}_2 \in \mathbb{R}^{C_{out} \times 1}$ and a column vector, $\boldsymbol{c}_2 \in \mathbb{R}^{1 \times C_{in}}$, which is inspired by the addition of a row or column vector (or both) to a low-rank matrix in recommendation systems, one of the most popular applications of low-rank structure, for better prediction of ratings by considering a bias for each user or each item (Jahrer and Töscher, 2012; Goodfellow et al., 2016; Koren et al., 2021). As a result, we formulate $\widehat{\boldsymbol{W}}$ as

$$\widehat{\boldsymbol{W}} = \boldsymbol{s}_1 \left\lfloor \frac{\boldsymbol{W}}{\boldsymbol{s}_1 \odot \exp(\boldsymbol{L}_2\boldsymbol{U}_2 + \boldsymbol{r}_2 + \boldsymbol{c}_2)} \right\rceil, \quad (2)$$

which we refer to as 'Low-Rank Quantization (LRQ)'. At first, $\boldsymbol{L}_2$ and $\boldsymbol{U}_2$ are initialized to zeros and random values from a normal distribution respectively, and $\boldsymbol{r}_2$ and $\boldsymbol{c}_2$ are set to zero vectors so that $\boldsymbol{L}_2\boldsymbol{U}_2 + \boldsymbol{r}_2 + \boldsymbol{c}_2$ starts from a zero matrix like $\boldsymbol{S}_2$ in Eq. 1. Then, $\boldsymbol{s}_1$, $\boldsymbol{L}_2$, $\boldsymbol{U}_2$, $\boldsymbol{r}_2$, and $\boldsymbol{c}_2$ are learned to minimize $\|\boldsymbol{W}\boldsymbol{X} - \widehat{\boldsymbol{W}}\widetilde{\boldsymbol{X}}\|_2^2$ in a block-by-block manner. The ablation study on the effect of $\boldsymbol{r}_2$ and $\boldsymbol{c}_2$ in LRQ is presented in Appendix B.

2.4 Effect of Low-rank Matrices on Generalization of Quantized LLMs

Considering that a full weight-scaling matrix is substituted with a low-rank matrix as seen in Eq. 2 derived from Eq. 1, one might wonder (i) whether the minimization of block reconstruction error on calibration samples is feasible despite the use of low-rank matrices, and (ii) how the utilization of low-rank matrices can result in improved generalization performance on unseen benchmarks as Figure 1 demonstrates. To address these concerns, we conduct a comparative analysis of accumulated root mean square error (RMSE) between $\boldsymbol{W}\boldsymbol{X}$ and $\widehat{\boldsymbol{W}}\widetilde{\boldsymbol{X}}$ for RTN, FlexRound, and LRQ.

For a calibration sample that is selected from the C4 dataset, even if both FlexRound and LRQ initially start their learning process from the same RTN baseline, LRQ achieves an almost identical accumulated RMSE to FlexRound, as illustrated in Figure 3(a). This observation underscores that the use of low-rank weight-scaling matrices does not pose any noticeable obstacle to the minimization of block reconstruction error on calibration data. For common sense reasoning and MMLU benchmarks that are unseen during the reconstruction stage, however, accumulated RMSE for LRQ is much smaller than that for FlexRound as well as RTN as described in Figure 3(b). This compelling result implies that harnessing the parameter-efficiency of

Table 1: Zero-shot performance of Llama on common sense reasoning tasks (BoolQ, PIQA, HellaSwag, WinoGrande, ARC easy and challenge, and OpenBookQA) with per-channel asymmetric weight quantization, per-tensor asymmetric static activation quantization, and per-token asymmetric KV cache quantization. The accuracy (%) is reported for all tasks. The number of bits used for weights, activations, and KV cache is 8-bit.

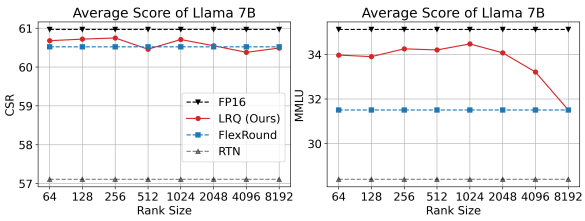
Method	# Bits (W/A/KV)	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average
Llama 7B	16/16/16	73.15	77.31	72.96	67.09	52.48	41.38	42.40	60.97
SmoothQuant	8/8/8	69.42	72.63	69.07	64.72	48.61	37.12	39.20	57.25
FlexRound	8/8/8	72.54	76.50	71.88	66.77	53.03	39.76	42.00	60.35
LRQ (Ours)	8/8/8	72.84	77.37	72.04	67.01	53.03	40.53	41.60	**60.63**
Llama 13B	16/16/16	68.53	79.11	76.23	70.01	59.89	44.54	42.20	62.93
SmoothQuant	8/8/8	67.34	75.19	71.78	69.06	54.92	40.44	38.80	59.65
FlexRound	8/8/8	68.78	78.51	75.23	70.56	58.46	44.03	41.00	62.37
LRQ (Ours)	8/8/8	68.84	78.78	75.56	70.80	59.13	44.62	41.60	**62.76**
Llama 33B	16/16/16	68.38	80.09	79.21	72.93	58.92	45.48	42.00	63.86
SmoothQuant	8/8/8	71.31	75.30	71.29	68.98	53.66	43.26	41.00	60.69
FlexRound	8/8/8	69.05	79.49	77.49	70.88	56.86	43.60	42.00	62.77
LRQ (Ours)	8/8/8	68.84	79.98	78.52	73.72	58.21	45.73	43.00	**64.00**
Llama 65B	16/16/16	82.32	80.85	80.71	77.19	58.71	46.33	44.60	67.24
SmoothQuant	8/8/8	78.78	79.54	79.11	73.32	56.23	45.90	43.80	65.24
FlexRound	8/8/8	80.46	79.38	79.23	74.98	57.20	46.42	45.00	66.10
LRQ (Ours)	8/8/8	82.35	81.12	79.96	75.61	58.96	46.59	45.40	**67.14**

low-rank matrices can facilitate superior generalization on unseen benchmarks. In light of these findings, the incorporation of low-rank matrices into block-wise reconstruction is indeed a pivotal step in enhancing the generalization capability of quantized LLMs. For visual representation across various samples, three figures are incorporated in Appendix C, each depicting the accumulated RMSE for three distinct samples. In addition, we illustrate the sensitivity of the accumulated RMSE to the number of calibration samples in Appendix D.

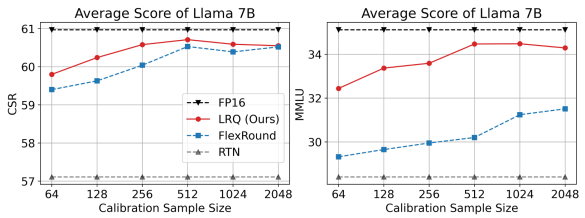
3 Experiments

In this section, we first explore the influence of the rank r in Eq. 2 and the quantity of calibration samples on the performance of LRQ. Next, to verify the effectiveness of LRQ, we compare LRQ with existing state-of-the-art post-training quantization (PTQ) methods for large language models (LLMs).

We use just a single NVIDIA A100-80GB GPU to quantize LLMs via LRQ. We randomly choose 512 calibration samples with a token length of 1024 from the training set of C4 (Raffel et al., 2020). Unless otherwise mentioned, LRQ is applied to all linear layers in both attention and feed-forward modules, and the rank r in Eq. 2 is set to 2048 for large language models beyond 30B parameters or to 1024 for smaller models so as to reduce the number of learnable parameters by approximately half compared to FlexRound. The exact ratio of the number of learnable parameters in LRQ to the number of pre-trained weights for an intermediate



(a) Rank Study



(b) Calibration Sample Size Study

Figure 4: Zero-shot and five-shot performances of Llama 7B on common sense reasoning (CSR) tasks and MMLU, where weights and activations are quantized to 8-bit while the KV cache is kept in FP16.

Transformer block of each Llama model is given in Appendix J. Quantized models are evaluated on MMLU (Hendrycks et al., 2021) in the five-shot setting or common sense reasoning benchmarks: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC easy and challenge (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018) in the zero-shot setting. More details are deferred to Appendix I.

3.1 Ablation Study

Rank Study To examine the impact of the rank r in Eq. 2 on the generalization on unseen bench-

Table 2: Zero-shot performance of Llama 2 on common sense reasoning tasks (BoolQ, PIQA, HellaSwag, WinoGrande, ARC easy and challenge, and OpenBookQA) with per-channel asymmetric weight quantization, per-tensor asymmetric static activation quantization, and per-token asymmetric KV cache quantization. The accuracy (%) is reported for all tasks. The number of bits used for weights, activations, and KV cache is 8-bit.

Method	# Bits (W/A/KV)	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average
Llama 2 7B	16/16/16	71.07	76.99	72.96	67.25	53.58	40.53	40.80	60.45
SmoothQuant	8/8/8	67.65	73.29	67.52	62.90	51.35	37.80	37.60	56.87
FlexRound	8/8/8	72.05	77.26	71.30	65.98	54.88	39.16	39.20	**59.98**
LRQ (Ours)	8/8/8	67.86	76.99	71.97	67.01	54.71	40.19	40.00	59.82
Llama 2 13B	16/16/16	69.02	79.05	76.62	69.61	57.95	44.28	42.00	62.65
SmoothQuant	8/8/8	63.55	75.95	70.99	66.30	53.96	40.10	40.60	58.78
FlexRound	8/8/8	66.94	79.00	75.32	69.38	58.54	42.92	40.40	61.79
LRQ (Ours)	8/8/8	68.59	78.67	75.83	70.64	58.16	43.34	39.80	**62.15**
Llama 2 70B	16/16/16	76.70	80.85	80.85	76.95	59.72	47.95	44.40	66.77
SmoothQuant	8/8/8	76.21	76.55	79.30	74.11	55.85	46.25	45.60	64.84
FlexRound	8/8/8	76.18	80.36	79.09	75.06	60.10	46.42	43.80	65.86
LRQ (Ours)	8/8/8	77.95	81.23	79.78	74.82	57.83	46.33	43.60	**65.93**

Table 3: Five-shot accuracy of Llama on Massive Multitask Language Understanding with per-channel asymmetric weight quantization, per-tensor asymmetric static activation quantization, and per-token asymmetric KV cache quantization. The accuracy (%) is reported for four disciplines. The number of bits used for weights, activations, and KV cache is 8-bit, the same as Table 1.

Method	STEM	Humanities	Social Science	Other	Average
Llama 7B	30.58	33.88	38.19	38.25	35.12
SmoothQuant	28.40	28.69	32.79	30.48	29.94
FlexRound	27.60	28.71	29.61	31.99	29.43
LRQ (Ours)	29.72	32.79	37.44	38.16	**34.39**
Llama 13B	36.35	44.97	54.14	53.15	47.02
SmoothQuant	27.24	30.12	30.58	31.31	29.87
FlexRound	33.63	42.81	48.65	49.26	43.60
LRQ (Ours)	35.16	44.55	51.74	52.04	**45.83**
Llama 33B	46.69	56.39	67.40	63.60	58.38
SmoothQuant	37.94	41.64	50.57	51.48	45.07
FlexRound	43.47	52.20	61.94	59.90	54.24
LRQ (Ours)	45.26	52.58	63.99	61.26	**55.51**
Llama 65B	51.95	61.87	73.32	67.58	63.57
SmoothQuant	44.83	50.82	63.34	57.09	53.72
FlexRound	46.32	54.60	65.06	62.49	56.94
LRQ (Ours)	50.96	61.28	71.99	66.66	**62.65**

Table 4: Five-shot accuracy of Llama 2 on Massive Multitask Language Understanding with per-channel asymmetric weight quantization, per-tensor asymmetric static activation quantization, and per-token asymmetric KV cache quantization. The accuracy (%) is reported for four disciplines. The number of bits used for weights, activations, and KV cache is 8-bit, the same as Table 2.

Method	STEM	Humanities	Social Science	Other	Average
Llama 2 7B	37.04	43.38	51.84	52.44	45.96
SmoothQuant	30.42	27.95	34.29	34.27	31.33
FlexRound	33.40	36.96	43.13	46.30	39.70
LRQ (Ours)	34.82	39.91	46.47	47.62	**42.04**
Llama 2 13B	44.27	54.43	63.41	60.76	55.68
SmoothQuant	30.98	29.29	35.36	35.29	32.37
FlexRound	41.09	51.58	61.39	59.41	53.28
LRQ (Ours)	42.88	51.97	62.14	59.93	**54.08**
Llama 2 70B	57.79	65.16	80.44	74.61	69.11
SmoothQuant	47.51	53.84	68.35	63.94	57.99
FlexRound	54.27	61.11	77.45	71.31	65.57
LRQ (Ours)	54.44	62.61	76.99	71.78	**66.12**

marks, we compare LRQ with different r (spanning from 64 to 8192) to FlexRound for Llama 7B as shown in Figure 4(a). The performance of LRQ (depicted by the red curve) either remains relatively stable (the left side of Figure 4(a)) or increases gradually from 33.97% to 34.47% (the right side of Figure 4(a)) with the rise in the rank r from 64 to 1024. As the rank r continuously increases from 2048 to 8192, however, the performance of LRQ eventually declines to match that of FlexRound (indicated by the blue curve) on both common sense reasoning tasks and MMLU, which leads us to set the rank r to 1024 for Llama 7B. Hence, selecting an appropriate low rank r becomes crucial to enable quantized LLMs via LRQ to achieve well-rounded performance across various tasks encompassing common sense reasoning and MMLU benchmarks.

Calibration Sample Size Study To identify whether the performance of LRQ improves with an increase in the number of calibration samples, we also conduct experiments on LRQ for Llama 7B with various calibration sample size while fixing the rank r to 1024. The accuracy of LRQ rises with a larger calibration sample size, but it reaches a saturation point when exceeding 1024 calibration samples as depicted in Figure 4(b). Nevertheless, LRQ can surpass FlexRound irrespective of the calibration sample size not only on common sense reasoning tasks but also on the MMLU benchmark, which sheds light on the effect of low-rank matrices on enhancing the generalization ability of quantized LLMs as we elaborate on in Section 2.4.

Table 5: Zero-shot performance of Llama 2 on common sense reasoning tasks (BoolQ, PIQA, HellaSwag, WinoGrande, ARC easy and challenge, and OpenBookQA) with per-channel asymmetric weight quantization, per-token asymmetric activation quantization, and per-token asymmetric KV cache quantization. The accuracy (%) is reported for all tasks. Here, weights are quantized to 4-bits, while activations and KV cache utilize 8-bit quantization.

Method	# Bits (W/A/KV)	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average
Llama 2 7B	16/16/16	71.07	76.99	72.96	67.25	53.58	40.53	40.80	60.45
SmoothQuant	4/8/8	43.03	63.71	41.08	54.30	35.69	27.99	32.60	42.63
FlexRound	4/8/8	71.71	76.77	72.24	66.14	53.49	40.02	40.40	60.11
LRQ (Ours)	4/8/8	73.00	76.99	71.90	65.98	54.38	39.68	41.20	**60.45**
Llama 2 13B	16/16/16	69.02	79.05	76.62	69.61	57.95	44.28	42.00	62.65
SmoothQuant	4/8/8	61.62	56.53	37.31	51.38	31.57	24.74	30.60	41.96
FlexRound	4/8/8	69.05	78.51	75.51	69.53	58.75	43.60	41.20	62.31
LRQ (Ours)	4/8/8	71.13	78.29	75.79	68.90	57.83	43.34	41.20	**62.35**
Llama 2 70B	16/16/16	76.70	80.85	80.85	76.95	59.72	47.95	44.40	66.77
SmoothQuant	4/8/8	50.46	71.60	48.35	55.09	44.87	32.17	37.40	48.56
FlexRound	4/8/8	77.31	80.96	79.89	75.30	60.19	48.21	43.40	66.47
LRQ (Ours)	4/8/8	77.92	81.28	80.42	75.06	60.94	48.04	42.60	**66.61**

Table 6: Five-shot accuracy of Llama 2 on Massive Multitask Language Understanding with per-channel asymmetric weight quantization, per-token asymmetric activation quantization, and per-token asymmetric KV cache quantization. The accuracy (%) is reported for four disciplines. Here, weights are quantized to 4-bits, while activations and KV cache utilize 8-bit quantization, which is the same as Table 5.

Method	STEM	Humanities	Social Science	Other	Average
Llama 2 7B	37.04	43.38	51.84	52.44	45.96
SmoothQuant	26.77	24.87	22.81	25.85	25.05
FlexRound	37.81	42.55	50.47	50.65	45.14
LRQ (Ours)	36.88	42.53	50.80	52.22	**45.36**
Llama 2 13B	44.27	54.43	63.41	60.76	55.68
SmoothQuant	27.07	24.25	25.22	26.43	25.57
FlexRound	42.88	50.71	61.94	59.93	53.77
LRQ (Ours)	43.90	52.56	62.07	59.96	**54.49**
Llama 2 70B	57.79	65.16	80.44	74.61	69.11
SmoothQuant	27.37	24.59	27.59	25.94	26.16
FlexRound	56.26	62.89	78.78	72.92	67.26
LRQ (Ours)	55.57	64.65	78.97	72.52	**67.65**

3.2 Per-tensor Static Activation Quantization

As meticulously studied in Xiao et al. (2022), per-tensor static activation quantization is hardware-efficient and can be implemented on off-the-shelf GPUs with FasterTransformer, the state-of-the-art Transformer inference framework provided from NVIDIA, to achieve up to 1.5× inference speed-up and almost halving the memory footprint compared to FP16 baselines. Accordingly, we employ per-tensor asymmetric static activation quantization as well as per-channel asymmetric weight quantization. Moreover, we also quantize the KV cache to 8-bit with a per-token asymmetric quantization scheme. It is worth noting that for large batch sizes, the KV cache can consume a much larger amount of memory than the model size, thus causing a bottleneck in high-throughput LLM inference. Fortunately, the performance discrepancy before and after per-token asymmetric KV cache quantization is almost insignificant no matter which quantization method is selected, as presented in Appendix H. For this reason, we also additionally utilize per-token asymmetric KV cache quantization. Further experimental details are provided in Appendix I.

Table 1, 2, 3, and 4 reveal the efficacy of LRQ on common sense reasoning tasks and MMLU. For common sense reasoning tasks, the zero-shot accuracy of LRQ is almost close to that of FP16 baselines, being superior to that of both SmoothQuant and FlexRound for most of the Llama and Llama 2 models. Not only that, LRQ also considerably outperforms SmoothQuant and FlexRound on MMLU.

3.3 Per-token Activation Quantization

Although LRQ shows better performance than SmoothQuant and FlexRound on both common sense reasoning tasks and MMLU when employing per-tensor asymmetric static activation quantization, there is still the five-shot accuracy gap on MMLU between LRQ and FP16 baselines for Llama 2 as in Table 4. Thus, we also conduct experiments on Llama 2 with a per-token asymmetric activation quantization scheme. More details about experimental settings are given in Appendix I.

In Table 5 and 6, when quantizing weights to 4-bit and both activations and KV cache to 8-bit, LRQ can attain similar zero-shot performance to FP16 baselines on common sense reasoning benchmarks and narrow the five-shot performance difference between FP16 baselines and quantized models to less than 1.5 percent on the MMLU benchmark.

Table 7: Zero-shot performance of Llama 2 on common sense reasoning tasks (BoolQ, PIQA, HellaSwag, WinoGrande, ARC easy and challenge, and OpenBookQA) and the causal language modeling task on WikiText2 with per-channel asymmetric weight-only quantization. The accuracy (%) and the perplexity (PPL) are reported for common sense reasoning tasks and the causal language modeling task, respectively. The lower PPL, the better.

Method	# Bits (W/A/KV)	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average	WikiText2
Llama 2 7B	16/16/16	71.07	76.99	72.96	67.25	53.58	40.53	40.80	60.45	5.47
OmniQuant	3/16/16	65.72	73.99	67.65	63.61	49.71	36.26	39.80	56.58	6.57
FlexRound	3/16/16	70.15	75.73	69.92	66.46	51.43	38.31	39.20	58.74	**6.34**
LRQ (Ours)	3/16/16	71.31	76.44	70.35	64.88	52.23	39.08	39.20	**59.07**	6.48
OmniQuant	4/16/16	68.99	77.48	71.26	67.01	53.66	39.08	40.00	59.64	**5.74**
FlexRound	4/16/16	73.24	76.55	72.09	67.09	52.82	39.51	40.80	60.30	5.83
LRQ (Ours)	4/16/16	71.80	76.88	72.40	67.88	53.24	40.27	40.20	**60.38**	5.75
Llama 2 13B	16/16/16	69.02	79.05	76.62	69.61	57.95	44.28	42.00	62.65	4.88
OmniQuant	3/16/16	69.02	77.69	72.77	65.90	54.00	42.75	38.80	60.13	5.58
FlexRound	3/16/16	66.02	78.29	74.41	67.32	57.15	42.15	40.60	60.85	5.59
LRQ (Ours)	3/16/16	67.49	78.45	74.25	69.30	56.23	42.58	41.60	**61.41**	**5.57**
OmniQuant	4/16/16	65.17	78.94	75.39	67.80	56.44	42.75	41.60	61.16	5.02
FlexRound	4/16/16	69.94	79.00	75.93	69.06	58.96	43.26	40.40	62.36	**5.01**
LRQ (Ours)	4/16/16	70.49	78.78	76.13	69.93	59.85	43.52	41.20	**62.84**	5.02
Llama 2 70B	16/16/16	76.70	80.85	80.85	76.95	59.72	47.95	44.40	66.77	3.31
OmniQuant	3/16/16	66.54	80.74	78.13	73.48	57.20	45.90	42.60	63.51	3.93
FlexRound	3/16/16	75.02	80.25	79.02	70.24	58.63	46.93	42.20	64.61	3.92
LRQ (Ours)	3/16/16	78.13	80.47	79.54	75.37	59.55	46.84	43.60	**66.21**	**3.89**
OmniQuant	4/16/16	77.13	80.96	80.53	75.85	59.76	46.76	42.40	66.20	3.47
FlexRound	4/16/16	78.38	80.58	80.49	75.85	59.51	47.87	43.60	66.61	**3.45**
LRQ (Ours)	4/16/16	79.72	80.36	80.58	77.03	59.72	47.44	43.80	**66.95**	3.47

Table 8: Average zero-shot accuracy and perplexity of Llama 3 8B on common sense reasoning tasks (PIQA, HellaSwag, WinoGrande, ARC easy and challenge) and WikiText2 respectively, using 4-bit per-channel asymmetric weight-only quantization. The lower PPL, the better. More details are given in Table 11. The results of GPTQ, AWQ, and QuIP come from Huang et al. (2024).

Method	# Bits (W/A/KV)	Average	WikiText2
Llama 3 8B	16/16/16	68.6	6.1
GPTQ	4/16/16	64.8	7.0
AWQ	4/16/16	67.0	7.1
QuIP	4/16/16	67.1	**6.5**
FlexRound	4/16/16	67.8	6.9
LRQ (Ours)	4/16/16	**68.0**	6.9

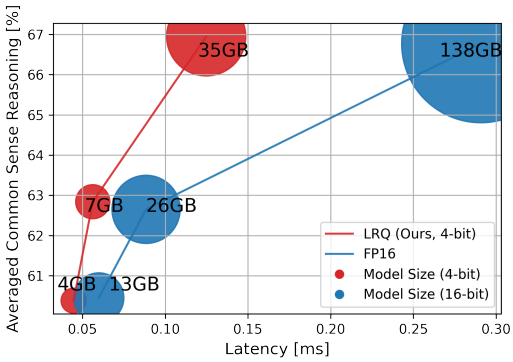


Figure 5: Average zero-shot accuracy over latency for Llama 2 7B, 13B, and 70B, respectively. The blue expresses FP16 baselines while the red represents 4-bit quantized models via LRQ. The size of a circle indicates the model size. More details are given in Appendix G.

3.4 Per-channel Weight-only Quantization

As LRQ is designed as a post-training weight quantization technique for LLMs, we also run experiments on weight-only quantization for Llama 2 and Llama 3 8B (Dubey et al., 2024) on common sense reasoning tasks and WikiText2 (Merity et al., 2016). In Table 7 and 8, we use per-channel weight-only quantization instead of group-wise weight-only quantization. Table 7 and 8 show that the average zero-shot accuracy of LRQ is consistently higher than that of all PTQ methods on common sense reasoning tasks. Specifically, in Table 7, LRQ can perform closely to FP16 baselines on common sense reasoning tasks even with 3-bit per-channel weight-only quantization. Despite the fact that Llama 3 is hard to quantize, Table 8 exhibits that LRQ attains the smallest accuracy drop (less than 0.7 percent compared to the FP16 baseline) on common sense reasoning tasks. In addition, Figure 5 displays compression and acceleration after quantizing Llama 2 to 4-bit via LRQ, showing that LRQ can perform comparably to FP16 baselines while reducing both latency and model size noticeably.

4 Conclusion

We propose a simple yet effective post-training weight quantization approach for LLMs, LRQ that learns low-rank weight-scaling matrices for block-by-block reconstructing the outputs of an intermediate Transformer block. Using such low-rank matrices, we can decrease the number of learnable parameters effectively while allowing for scaling weights individually due to the sharing of learnable parameters through a low-rank structure, thereby enhancing the generalization of quantized LLMs.

Limitations

To push the limit of post-training weight-activation quantization, two research directions emerge: (i) 4-bit weight and 8-bit activation quantization, and (ii) INT4 weight-activation quantization. As explained in Appendix A, Lee et al. (2023a) attempted to quantize LLMs with 4-bit weight and 8-bit activation quantization, whereas Wei et al. (2023) and Shao et al. (2024) strived to quantize LLMs with INT6 and even INT4 weight-activation quantization. In this paper, we only deal with the former quantization scheme, 4-bit weight and 8-bit activation quantization.

Like Wei et al. (2023) and Shao et al. (2024), we could also focus on INT4 weight-activation quantization rather than 4-bit weight and 8-bit activation quantization in Section 3.3. However, Liu et al. (2023a), an earlier LLM quantization work than Wei et al. (2023) and Shao et al. (2024), already exhibited the non-marginal accuracy degradation of 4-bit weight and 8-bit activation quantization despite the fact that Liu et al. (2023a) exploited quantization-aware training, not post-training quantization. Furthermore, in terms of serving throughput, Lin et al. (2024) shows the superiority of 4-bit weight and 8-bit activation quantization over INT4 weight-activation quantization as well as INT8 weight-activation quantization. For these reasons, we prioritize 4-bit weight and 8-bit activation quantization over INT4 weight-activation quantization in this paper.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *Preprint*, arXiv:2305.13245.

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

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. 2023. QuIP: 2-bit quantization of large language models with guarantees. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

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*.

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*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon

Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. 2020. Learned step size quantization. In *International Conference on Learning Representations*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. OPTQ: Accurate quantization for

generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021. A framework for few-shot language model evaluation.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Google, :, Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. Palm 2 technical report. *Preprint*, arXiv:2305.10403.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Wei Huang, Xingyu Zheng, Xudong Ma, Haotong Qin, Chengtao Lv, Hong Chen, Jie Luo, Xiaojuan Qi, Xianglong Liu, and Michele Magno. 2024. An empirical study of llama3 quantization: From llms to mllms. *Preprint*, arXiv:2404.14047.

Michael Jahrer and Andreas Töscher. 2012. Collaborative filtering ensemble for ranking. In *Proceedings of KDD Cup 2011*, volume 18 of *Proceedings of Machine Learning Research*, pages 153–167. PMLR.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. 2019. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4350–4359.

Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. 2023. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. In *Advances in Neural Information Processing Systems*, volume 36, pages 36187–36207. Curran Associates, Inc.

Yehuda Koren, Steffen Rendle, and Robert Bell. 2021. Advances in collaborative filtering. *Recommender systems handbook*, pages 91–142.

Janghwan Lee, Minsoo Kim, Seungcheol Baek, Seok Hwang, Wonyong Sung, and Jungwook Choi. 2023a. Enhancing computation efficiency in large language models through weight and activation quantization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14726–14739, Singapore. Association for Computational Linguistics.

Jung Hyun Lee, Jeonghoon Kim, Se Jung Kwon, and Dongsoo Lee. 2023b. Flexround: Learnable rounding based on element-wise division for post-training quantization. *Preprint*, arXiv:2306.00317.

Jung Hyun Lee, Jihun Yun, Sung Ju Hwang, and Eunho Yang. 2021. Cluster-promoting quantization with bit-drop for minimizing network quantization loss. *Preprint*, arXiv:2109.02100.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. 2021. BRECQ: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations*.

Zhengyi Li, Cong Guo, Zhanda Zhu, Yangjie Zhou, Yuxian Qiu, Xiaotian Gao, Jingwen Leng, and Minyi Guo. 2023. Efficient adaptive activation rounding for post-training quantization. *Preprint*, arXiv:2208.11945.

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*.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. 2024. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. *Preprint*, arXiv:2405.04532.

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

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. 2023b. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR.

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

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Gunho Park, Baeseong Park, Minsub Kim, Sungjae Lee, Jeonghoon Kim, Beomseok Kwon, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. 2024. Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models. *Preprint*, arXiv:2206.09557.

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. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2024. Omniquant: Omnidirectionally calibrated quantization for large language models. In *The Twelfth International Conference on Learning Representations*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Xiuying Wei, Ruihao Gong, Yuhang Li, Xianglong Liu, and Fengwei Yu. 2022. QDrop: Randomly dropping quantization for extremely low-bit post-training quantization. In *International Conference on Learning Representations*.

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 effective shifting and scaling. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1648–1665, Singapore. Association for Computational Linguistics.

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*.

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*.

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.

A Related Work

Quantization works can be generally categorized into quantization-aware training (QAT) and post-training quantization (PTQ). As QAT can maintain the performance of FP32/FP16 baselines, QAT has been applied to computer vision models (Jung et al., 2019; Esser et al., 2020; Lee et al., 2021). Notwithstanding, there exist many challenges associated with applying QAT to large language models (LLMs) due to the sheer scale of pre-training data and a huge amount of computational resources required for training on the whole pre-training dataset. Although Liu et al. (2023a) presented the possibility of applying QAT to LLMs, unfortunately, they did not perform experiments on Llama 65B, the largest and best performing model among the Llama models, in spite of using a single 8-GPU node. Kim et al. (2023) introduced Parameter-Efficient and Quantization-aware Adaptation (PEQA) that fine-tunes only quantization step sizes. While PEQA consumes less computational resources and can perform better than QAT, it still requires a single 8-GPU node to quantize Llama 65B.

As Frantar et al. (2023) demonstrated the application of PTQ to LLMs only with a single GPU, many researchers have recently paid attention to PTQ for LLMs. LLM PTQ can be classified into two categories: LLM weight-only quantization (Frantar et al., 2023; Lin et al., 2023; Chee et al., 2023) and LLM weight-activation quantization (Dettmers et al., 2022; Yao et al., 2022; Xiao et al., 2022; Lee et al., 2023b; Wei et al., 2023; Shao et al., 2024). For the former quantization, Frantar et al. (2023); Chee et al. (2023) quantized the weights of LLMs into low-bit integers based on layer-wise reconstruction, whereas Lin et al. (2023) did by not counting on reconstruction but per-channel scaling in consideration of both weight and activation magnitudes. Despite the fact that all these studies exhibited decent quantization performance, the main benefit of weight-only quantization does not align with serving LLMs with high throughput as delineated in Section 1. In this light, we concentrates on weight-activation quantization.

When it comes to weight-activation quantization, Yao et al. (2022) presented ZeroQuant with a 8-bit group-wise weight quantization scheme and a 8-bit per-token activation quantization scheme based on layer-wise knowledge distillation, and Dettmers et al. (2022) proposed LLM.int8() with a 8-bit per-channel weight quantization scheme and a 8-bit per-token activation quantization scheme while keeping activation outliers in FP16. As discussed in Xiao et al. (2022), however, ZeroQuant incurs severe accuracy degradation for an open-source LLM, and the inference latency of LLM.int8() can be higher than that of the FP16 baseline. To deal with both issues, Xiao et al. (2022) devised SmoothQuant that can preserve the accuracy of OPT (Zhang et al., 2022) by easing the difficulty of activation quantization and accelerate LLM inference by up to 1.5 times. Yet, SmoothQuant suffers from non-negligible performance degradation for other open-source models such as Llama and Llama 2 with a 8-bit per-tensor static activation quantization scheme as illustrated in Figure 1. FlexRound that Lee et al. (2023b) created showed the experimental results of Llama up to 33B with a 8-bit per-channel weight quantization scheme and a 8-bit per-tensor static activation quantization scheme, but FlexRound incurs considerable performance degradation on the massive multitask language understanding (MMLU) benchmark as described in Figure 1(b). Beyond INT8 weight-activation quantization, Lee et al. (2023a) attempted to quantize LLMs with 4-bit weight and 8-bit activation quantization, whereas Wei et al. (2023) and Shao et al. (2024) strived to quantize LLMs with INT6 and even INT4 weight-activation quantization. However, in terms of serving throughput, Lin et al. (2024) showed the superiority of 4-bit weight and 8-bit activation quantization over INT4 weight-activation quantization as well as INT8 weight-activation quantization.

B Effect of $\boldsymbol{r}_2$ and $\boldsymbol{c}_2$ in LRQ

To show the effect of $\boldsymbol{r}_2$ and $\boldsymbol{c}_2$ in Eq. 2, we compare FlexRound, FlexRound with $\boldsymbol{S}_2 = \boldsymbol{L}_2\boldsymbol{U}_2$, and LRQ for Llama 7B and 13B.

Table 9: Zero-shot performance of FlexRound, FlexRound with $\boldsymbol{S}_2 = \boldsymbol{L}_2\boldsymbol{U}_2$, and LRQ on common sense reasoning tasks (BoolQ, PIQA, HellaSwag, WinoGrande, ARC easy and challenge, and OpenBookQA) with per-channel asymmetric weight quantization, per-tensor asymmetric static activation quantization, and per-token asymmetric KV cache quantization (if applied). The number of bits used for weights, activations, and KV cache is expressed as W/A/KV.

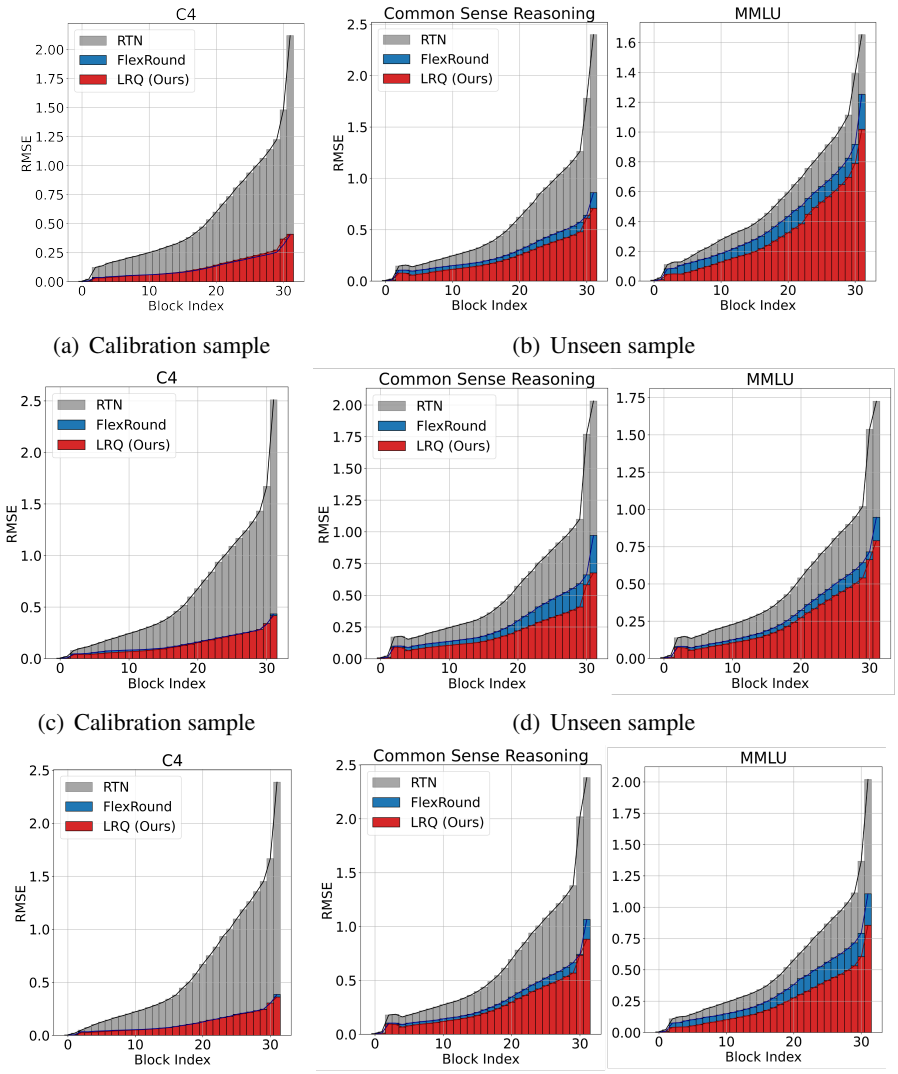
Method	# Bits (W/A/KV)	Llama 7B	Llama 13B
FlexRound	8/8/16	60.53	62.40
FlexRound with $\boldsymbol{S}_2 = \boldsymbol{L}_2\boldsymbol{U}_2$	8/8/16	60.69	62.62
LRQ (Ours)	8/8/16	**60.71**	**62.92**
FlexRound	8/8/8	60.35	62.37
FlexRound with $\boldsymbol{S}_2 = \boldsymbol{L}_2\boldsymbol{U}_2$	8/8/8	60.49	62.62
LRQ (Ours)	8/8/8	**60.63**	**62.76**

Table 10: Five-shot performance of FlexRound, FlexRound with $\boldsymbol{S}_2 = \boldsymbol{L}_2\boldsymbol{U}_2$, and LRQ on Massive Multitask Language Understanding with per-channel asymmetric weight quantization, per-tensor asymmetric static activation quantization, and per-token asymmetric KV cache quantization (if applied). The number of bits used for weights, activations, and KV cache is expressed as W/A/KV.

Method	# Bits (W/A/KV)	Llama 7B	Llama 13B
FlexRound	8/8/16	30.20	43.82
FlexRound with $\boldsymbol{S}_2 = \boldsymbol{L}_2\boldsymbol{U}_2$	8/8/16	33.86	45.48
LRQ (Ours)	8/8/16	**34.47**	**45.83**
FlexRound	8/8/8	29.43	43.60
FlexRound with $\boldsymbol{S}_2 = \boldsymbol{L}_2\boldsymbol{U}_2$	8/8/8	33.96	45.21
LRQ (Ours)	8/8/8	**34.39**	**45.83**

As evident from the tables above, FlexRound with $\boldsymbol{S}_2 = \boldsymbol{L}_2\boldsymbol{U}_2$ surpasses the performance of FlexRound but falls short of LRQ, which implies that the effect of $\boldsymbol{r}_2$ and $\boldsymbol{c}_2$ cannot be ignored. It is noteworthy that the five-shot accuracy on MMLU can witness an increase ranging from 1.5% to 4% by simply substituting $\boldsymbol{S}_2$ with $\boldsymbol{L}_2\boldsymbol{U}_2$, which corroborates the significance of leveraging the parameter-efficiency inherent in low-rank weight-scaling matrices.

C Figures of Accumulated RMSE on Assorted Samples



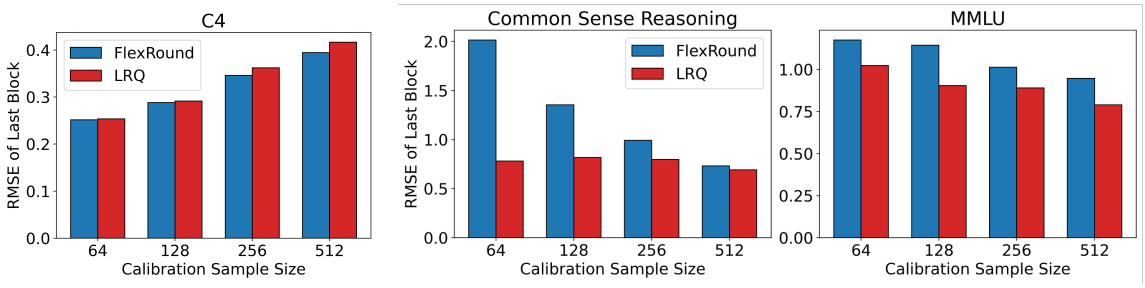
(a) Calibration sample (b) Unseen sample

(c) Calibration sample (d) Unseen sample

(e) Calibration sample (f) Unseen sample

Figure 6: Accumulated root mean square error (RMSE) between $\boldsymbol{W}\boldsymbol{X}$ and $\widehat{\boldsymbol{W}}\widetilde{\boldsymbol{X}}$ for RTN, FlexRound, and LRQ on (a), (c), (e) three different calibration samples from the C4 dataset and (b), (d), (f) three different unseen samples from common sense reasoning and MMLU benchmarks, ranging from the first Transformer block to the last Transformer block of Llama 7B. Here, weights and activations are quantized to 8-bit with per-channel asymmetric quantization and per-tensor asymmetric static quantization respectively, while the KV cache remains in FP16. Note that RMSE tends to rise in line with the block index due to the presence of $\widetilde{\boldsymbol{X}}$ that accumulates quantization error resulting from previous quantized Transformer blocks.

D Sensitivity of Accumulated RMSE to the Number of Calibration Samples



(a) Calibration sample

(b) Unseen sample

Figure 7: Accumulated root mean square error (RMSE) between $\boldsymbol{W}\boldsymbol{X}$ and $\widehat{\boldsymbol{W}}\widetilde{\boldsymbol{X}}$ for FlexRound and LRQ on (a) a calibration sample from the C4 dataset and (b) an unseen sample from common sense reasoning and MMLU benchmarks at the last Transformer block of Llama 7B. Here, weights and activations are quantized to 8-bit with per-channel asymmetric quantization and per-tensor asymmetric static quantization, respectively.

To figure out the sensitivity of accumulated root mean square error (RMSE) to the number of calibration samples used for the block-wise reconstruction, we compare accumulated RMSE between $\boldsymbol{W}\boldsymbol{X}$ and $\widehat{\boldsymbol{W}}\widetilde{\boldsymbol{X}}$ for FlexRound and LRQ at the last Transformer block of Llama 7B with the number of calibration samples varying from 64 to 512. As depicted in Figure 8(a), the accumulated RMSE of the last Transformer block on a calibration sample diminishes with a reduction in the number of calibration samples. This phenomenon is because FlexRound and LRQ are more likely to be fitted to calibration samples as the number of calibration samples becomes smaller. Conversely, Figure 8(b) reveals that the accumulated RMSE of the last Transformer block on each unseen sample from common sense reasoning and MMLU decreases with a larger number of calibration samples.

Notably, the pattern elucidated in Section 2.4 persists consistently across varying calibration sample sizes from 64 to 512. In other words, for every calibration sample size spanning from 64 to 512, LRQ consistently attains nearly identical accumulated RMSE to FlexRound for a calibration sample from the C4 dataset. Concurrently, the accumulated RMSE of LRQ remains markedly smaller than that of FlexRound for an unseen sample from common sense reasoning and MMLU. This observation provides additional support for the insight presented in Figure 3, as discussed in Section 2.4.

E Detailed Experimental Results for Llama 3 8B and Mistral 7B

Table 11: Zero-shot performance of Llama 3 8B on common sense reasoning tasks (PIQA, HellaSwag, WinoGrande, ARC easy and challenge) and the causal language modeling task on WikiText2 with 4-bit per-channel asymmetric weight-only quantization. The accuracy (%) and the perplexity (PPL) are reported for common sense reasoning tasks and the causal language modeling task, respectively. The lower PPL, the better. The experimental results of GPTQ, AWQ, and QuIP originate from Huang et al. (2024).

Method	# Bits (W/A/KV)	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	Average	WikiText2
Llama 3 8B	16/16/16	79.9	60.2	72.8	80.1	50.4	68.6	6.1
GPTQ	4/16/16	76.8	57.4	72.8	74.3	42.4	64.8	7.0
AWQ	4/16/16	78.3	58.6	72.5	77.6	48.3	67.0	7.1
QuIP	4/16/16	78.2	58.6	73.2	78.2	47.4	67.1	**6.5**
FlexRound	4/16/16	79.3	59.2	73.4	79.3	47.7	67.8	6.9
LRQ (Ours)	4/16/16	79.2	59.2	74.4	79.3	47.8	**68.0**	6.9

Table 12: Zero-shot performance of Mistral 7B on common sense reasoning tasks (BoolQ, PIQA, HellaSwag, WinoGrande, ARC easy and challenge, and OpenBookQA) and the causal language modeling task on WikiText2 with per-channel asymmetric weight-only quantization. The accuracy (%) and the perplexity (PPL) are reported for common sense reasoning tasks and the causal language modeling task, respectively. The lower PPL, the better.

Method	# Bits (W/A/KV)	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average	WikiText2
Mistral 7B	16/16/16	83.82	82.15	81.07	73.95	79.50	54.01	43.80	71.19	5.25
FlexRound	3/16/16	79.72	80.58	77.27	66.61	75.59	46.84	39.40	66.57	6.60
LRQ (Ours)	3/16/16	80.95	80.74	77.50	68.27	75.76	48.81	40.20	**67.46**	**6.13**
FlexRound	4/16/16	82.91	81.34	79.78	72.45	78.66	51.11	44.40	70.09	5.48
LRQ (Ours)	4/16/16	84.01	81.66	79.79	73.09	78.58	51.71	45.00	**70.55**	**5.45**

To further justify the effectiveness of LRQ for other model families than Llama, we conduct additional experiments for Mistral 7B v0.1 (Jiang et al., 2023) on common sense reasoning tasks and WikiText2 in a low-bit per-channel weight-only quantization scheme. In Table 12, LRQ outperforms FlexRound in both 3-bit and 4-bit per-channel asymmetric weight-only quantization schemes. In particular, in a 3-bit per-channel asymmetric weight-only quantization scheme, LRQ surpasses FlexRound by about 0.5 PPL on WikiText2 and by almost 0.9 percent on common sense reasoning tasks.

F Comparison of Computation Cost to Complete the Quantization Process

For a comparative analysis of SmoothQuant, FlexRound, and LRQ in terms of computational cost to complete the quantization process, as delineated in Table 13, we measure the execution time and peak GPU memory usage while quantizing Llama 7B with 8-bit per-channel asymmetric weight quantization and 8-bit per-tensor asymmetric static activation quantization using 512 calibration samples and a batch size of 2. Since both FlexRound and LRQ involve gradient-based optimization in a block-wise manner while SmoothQuant is a learning-free quantization method, FlexRound and LRQ naturally spend more time and GPU memory quantizing LLMs than SmoothQuant. In Table 13, LRQ's extended processing time compared to FlexRound is attributed to the multiplication involving $\boldsymbol{L}_2$ and $\boldsymbol{U}_2$ in Eq. 2. Despite the slightly longer runtime, LRQ demonstrates an advantage in peak GPU memory usage, utilizing 23.5 GB compared to FlexRound's 25.4 GB. This efficiency is attributed to LRQ's fewer learnable parameters in comparison to FlexRound.

Table 13: Execution time and peak GPU memory usage while quantizing Llama 7B with 8-bit per-channel asymmetric weight quantization and 8-bit per-tensor asymmetric static activation quantization using 512 calibration samples and a batch size of 2

Method	Execution time (A)	Peak GPU memory usage
SmoothQuant	10 minutes	13.5 GB
FlexRound	5 hours 7 minutes	25.4 GB
LRQ (Ours)	5 hours 22 minutes	23.5 GB

Additionally, we also measure the execution time and peak GPU memory usage while quantizing Llama 2 7B with 4-bit per-channel asymmetric weight-only quantization using 512 calibration samples and a batch size of 2. Similar to Table 13, Table 14 shows that LRQ consumes more processing time but less peak GPU memory usage than FlexRound.

Table 14: Execution time and peak GPU memory usage while quantizing Llama 2 7B with 4-bit per-channel asymmetric weight-only quantization using 512 calibration samples and a batch size of 2

Method	Execution time (A)	Peak GPU memory usage
FlexRound	2 hours 50 minutes	23.6 GB
LRQ (Ours)	3 hours 3 minutes	21.3 GB

G Compression and Acceleration Effects after LRQ at Inference Time

Since only a quantization step size ($\boldsymbol{s}_1$) and an integer matrix ($\widetilde{\boldsymbol{W}}$) are required during inference, once we obtain an integer matrix by setting $\widetilde{\boldsymbol{W}}$ to $\left\lfloor \frac{\boldsymbol{W}}{\boldsymbol{s}_1 \odot \exp(\boldsymbol{L}_2\boldsymbol{U}_2+\boldsymbol{r}_2+\boldsymbol{c}_2)} \right\rceil$ after $\boldsymbol{L}_2$, $\boldsymbol{U}_2$, $\boldsymbol{r}_2$, and $\boldsymbol{c}_2$ are learned, there is no need to recompute the multiplication involving $\boldsymbol{L}_2$ and $\boldsymbol{U}_2$ at test time. In other words, for inference, like other uniform quantization methods including GPTQ, SmoothQuant, and AWQ, LRQ also requires only a quantization step size $\boldsymbol{s}_1$ and an integer matrix $\widetilde{\boldsymbol{W}}$ without the presence of $\boldsymbol{L}_2$, $\boldsymbol{U}_2$, $\boldsymbol{r}_2$, and $\boldsymbol{c}_2$. Therefore, packing/unpacking techniques and acceleration kernels (Frantar et al., 2023; Lin et al., 2023; Park et al., 2024) can be applied to LRQ without additional effort.

Figure 5 and Table 15 exhibit compression and acceleration effects after LRQ with 3-bit and 4-bit per-channel weight-only uniform quantization. Regarding the compression effect, 3-bit uniform quantization shows x4.55 compression ratio, and 4-bit uniform quantization shows x3.58 compression ratio on the model size for Llama 2 7B. To verify the acceleration effect, we measure the matrix multiplication latency of Llama 2 FFN layers across 7B to 70B on a single token generation. We utilize cuBLAS for FP16 baselines, while employing the LUT-GEMM kernel (Park et al., 2024) for per-channel weight-only uniformly quantized models. As shown in Table 15, for Llama 2 70B, the 4-bit per-channel weight-only quantized model shows x2.33 faster latency, and the 3-bit per-channel weight-only quantized model shows x2.77 faster latency than the FP16 cuBLAS baseline.

Table 15: Average zero-shot performance on common sense reasoning (CSR) tasks, model size, and latency when quantizing Llama 2 7B, 13B, and 70B with 3-bit and 4-bit per-channel weight-only uniform quantization. Matrix multiplication latency of each model's FFN layers is measured using the LUT-GEMM kernel (Park et al., 2024).

Model	Method	CSR Avg.	Model Size [GB]	Latency [ms]
Llama 2 7B	FP16	60.45	13.48	0.05987
	LRQ (3-bit)	59.07	2.95	0.03750
	LRQ (4-bit)	60.38	3.76	0.04462
Llama 2 13B	FP16	62.65	26.03	0.08843
	LRQ (3-bit)	61.41	5.41	0.04789
	LRQ (4-bit)	62.84	7.00	0.05621
Llama 2 70B	FP16	66.77	137.95	0.29088
	LRQ (3-bit)	66.21	26.72	0.10482
	LRQ (4-bit)	66.95	35.28	0.12479

H Comparison of Experimental Results before and after Per-token Asymmetric KV Cache Quantization

Table 16, 17, 18, 20, 22, and 24 show the comparison of experimental results before and after per-token asymmetric KV cache quantization. It can be easily seen that the performance difference before and after per-token asymmetric KV cache quantization is nearly inconsiderable no matter which quantization technique is chosen, as mentioned in Section 3.2. Furthermore, even without per-token asymmetric KV cache quantization, LRQ still outperforms prior state-of-the-art LLM post-training weight-activation quantization methods in most cases.

Table 16: Zero-shot performance of Llama on common sense reasoning tasks (BoolQ, PIQA, HellaSwag, WinoGrande, ARC easy and challenge, and OpenBookQA) with per-channel asymmetric weight quantization, per-tensor asymmetric static activation quantization, and per-token asymmetric KV cache quantization (if applied). Please refer to Figure 8. The accuracy (%) is reported for common sense reasoning tasks. The number of bits used for weights, activations, and KV cache is expressed as W/A/KV.

Method	# Bits (W/A/KV)	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average
Llama 7B	16/16/16	73.15	77.31	72.96	67.09	52.48	41.38	42.40	60.97
RTN	8/8/16	71.56	73.72	65.86	63.93	49.49	36.43	38.80	57.11
SmoothQuant	8/8/16	69.63	73.12	68.88	65.43	48.70	38.57	38.00	57.48
FlexRound	8/8/16	73.76	76.66	71.75	67.01	52.31	40.02	42.20	60.53
LRQ (Ours)	8/8/16	73.03	77.64	72.10	66.77	52.95	40.87	41.60	**60.71**
RTN	8/8/8	69.76	73.72	65.95	62.75	48.91	37.12	37.60	56.54
SmoothQuant	8/8/8	69.42	72.63	69.07	64.72	48.61	37.12	39.20	57.25
FlexRound	8/8/8	72.54	76.50	71.88	66.77	53.03	39.76	42.00	60.35
LRQ (Ours)	8/8/8	72.84	77.37	72.04	67.01	53.03	40.53	41.60	**60.63**
Llama 13B	16/16/16	68.53	79.11	76.23	70.01	59.89	44.54	42.20	62.93
RTN	8/8/16	66.06	71.82	65.70	62.98	50.97	35.58	36.60	55.67
SmoothQuant	8/8/16	68.29	75.30	71.82	68.03	55.18	40.19	41.20	60.00
FlexRound	8/8/16	68.59	78.67	75.21	70.64	58.88	43.60	41.20	62.40
LRQ (Ours)	8/8/16	68.99	79.22	75.61	71.19	58.92	43.52	43.00	**62.92**
RTN	8/8/8	65.87	72.25	62.52	62.19	51.81	35.41	38.40	55.49
SmoothQuant	8/8/8	67.34	75.19	71.78	69.06	54.92	40.44	38.80	59.65
FlexRound	8/8/8	68.78	78.51	75.23	70.56	58.46	44.03	41.00	62.37
LRQ (Ours)	8/8/8	68.84	78.78	75.56	70.80	59.13	44.62	41.60	**62.76**
Llama 33B	16/16/16	68.38	80.09	79.21	72.93	58.92	45.48	42.00	63.86
RTN	8/8/16	69.02	76.01	69.11	66.54	57.07	41.64	41.40	60.11
SmoothQuant	8/8/16	71.04	75.24	71.01	69.38	54.38	43.34	40.60	60.71
FlexRound	8/8/16	69.08	79.16	77.43	72.53	56.61	44.97	44.00	63.40
LRQ (Ours)	8/8/16	68.44	80.03	78.37	74.19	58.16	46.33	42.20	**63.96**
RTN	8/8/8	68.81	76.55	68.76	66.06	56.48	42.49	42.40	60.22
SmoothQuant	8/8/8	71.31	75.30	71.29	68.98	53.66	43.26	41.00	60.69
FlexRound	8/8/8	69.05	79.49	77.49	70.88	56.86	43.60	42.00	62.77
LRQ (Ours)	8/8/8	68.84	79.98	78.52	73.72	58.21	45.73	43.00	**64.00**
Llama 65B	16/16/16	82.32	80.85	80.71	77.19	58.71	46.33	44.60	67.24
RTN	8/8/16	79.48	77.04	74.15	71.19	52.48	43.52	43.80	63.09
SmoothQuant	8/8/16	78.72	78.84	79.12	74.03	56.23	45.22	43.20	65.05
FlexRound	8/8/16	81.31	79.33	79.16	73.56	57.83	46.08	44.60	65.98
LRQ (Ours)	8/8/16	82.45	80.69	79.92	76.64	58.92	46.67	45.60	**67.27**
RTN	8/8/8	79.51	75.79	74.13	71.35	51.85	44.03	43.60	62.89
SmoothQuant	8/8/8	78.78	79.54	79.11	73.32	56.23	45.90	43.80	65.24
FlexRound	8/8/8	80.46	79.38	79.23	74.98	57.20	46.42	45.00	66.10
LRQ (Ours)	8/8/8	82.35	81.12	79.96	75.61	58.96	46.59	45.40	**67.14**

Table 17: Five-shot performance of Llama on Massive Multitask Language Understanding with per-channel asymmetric weight quantization, per-tensor asymmetric static activation quantization, and per-token asymmetric KV cache quantization (if applied). Please refer to Figure 8. The accuracy (%) is reported for four groups of disciplines (STEM, Humanities, Social Science, and Other). The number of bits used for weights, activations, and KV cache is expressed as W/A/KV.

Method	# Bits (W/A/KV)	STEM	Humanities	Social Science	Other	Average
Llama 7B	16/16/16	30.58	33.88	38.19	38.25	35.12
RTN	8/8/16	27.40	27.16	29.18	30.38	28.40
SmoothQuant	8/8/16	28.36	27.89	32.63	30.41	29.61
FlexRound	8/8/16	28.30	29.20	30.13	33.47	30.20
LRQ (Ours)	8/8/16	29.69	32.48	37.63	38.80	**34.47**
RTN	8/8/8	27.04	27.23	29.28	30.38	28.36
SmoothQuant	8/8/8	28.40	28.69	32.79	30.48	29.94
FlexRound	8/8/8	27.60	28.71	29.61	31.99	29.43
LRQ (Ours)	8/8/8	29.72	32.79	37.44	38.16	**34.39**
Llama 13B	16/16/16	36.35	44.97	54.14	53.15	47.02
RTN	8/8/16	26.61	25.53	27.40	24.52	25.94
SmoothQuant	8/8/16	27.80	29.31	31.04	30.88	29.73
FlexRound	8/8/16	35.06	41.68	49.37	49.81	43.82
LRQ (Ours)	8/8/16	34.72	44.65	51.71	52.28	**45.83**
RTN	8/8/8	26.38	25.33	27.95	24.83	26.01
SmoothQuant	8/8/8	27.24	30.12	30.58	31.31	29.87
FlexRound	8/8/8	33.63	42.81	48.65	49.26	43.60
LRQ (Ours)	8/8/8	35.16	44.55	51.74	52.04	**45.83**
Llama 33B	16/16/16	46.69	56.39	67.40	63.60	58.38
RTN	8/8/16	32.14	32.22	37.11	38.25	34.67
SmoothQuant	8/8/16	38.17	41.45	50.37	51.08	44.92
FlexRound	8/8/16	43.94	52.31	62.14	60.21	54.49
LRQ (Ours)	8/8/16	45.13	52.99	64.12	61.88	**55.79**
RTN	8/8/8	32.47	32.37	38.35	40.59	35.60
SmoothQuant	8/8/8	37.94	41.64	50.57	51.48	45.07
FlexRound	8/8/8	43.47	52.20	61.94	59.90	54.24
LRQ (Ours)	8/8/8	45.26	52.58	63.99	61.26	**55.51**
Llama 65B	16/16/16	51.95	61.87	73.32	67.58	63.57
RTN	8/8/16	42.25	46.74	61.13	54.57	50.73
SmoothQuant	8/8/16	44.70	50.54	63.99	57.28	53.79
FlexRound	8/8/16	46.52	54.30	66.36	60.83	56.78
LRQ (Ours)	8/8/16	50.89	61.15	72.64	66.04	**62.59**
RTN	8/8/8	41.22	47.23	61.39	54.69	50.76
SmoothQuant	8/8/8	44.83	50.82	63.34	57.09	53.72
FlexRound	8/8/8	46.32	54.60	65.06	62.49	56.94
LRQ (Ours)	8/8/8	50.96	61.28	71.99	66.66	**62.65**

Table 18: Zero-shot performance of Llama 2 on common sense reasoning tasks (BoolQ, PIQA, HellaSwag, WinoGrande, ARC easy and challenge, and OpenBookQA) with per-channel asymmetric weight quantization, per-tensor asymmetric static activation quantization, and per-token asymmetric KV cache quantization (if applied). Please refer to Figure 8. The accuracy (%) is reported for common sense reasoning tasks. The number of bits used for weights, activations, and KV cache is expressed as W/A/KV.

Method	# Bits (W/A/KV)	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average
Llama 2 7B	16/16/16	71.07	76.99	72.96	67.25	53.58	40.53	40.80	60.45
RTN	8/8/16	60.86	67.19	57.53	59.43	45.50	32.00	34.20	50.96
SmoothQuant	8/8/16	67.09	72.03	67.34	65.43	50.88	37.12	38.20	56.87
FlexRound	8/8/16	71.99	77.04	71.23	65.11	54.42	40.44	38.80	59.86
LRQ (Ours)	8/8/16	67.49	77.58	72.19	67.96	54.76	39.59	40.40	**60.00**
RTN	8/8/8	60.58	67.08	57.66	60.54	45.83	31.57	34.40	51.09
SmoothQuant	8/8/8	67.65	73.29	67.52	62.90	51.35	37.80	37.60	56.87
FlexRound	8/8/8	72.05	77.26	71.30	65.98	54.88	39.16	39.20	**59.98**
LRQ (Ours)	8/8/8	67.86	76.99	71.97	67.01	54.71	40.19	40.00	59.82
Llama 2 13B	16/16/16	69.02	79.05	76.62	69.61	57.95	44.28	42.00	62.65
RTN	8/8/16	63.12	73.99	62.60	58.80	52.15	36.26	36.40	54.76
SmoothQuant	8/8/16	64.19	76.28	70.75	66.06	54.42	40.78	39.60	58.87
FlexRound	8/8/16	66.70	78.56	75.63	69.06	58.33	43.26	40.00	61.65
LRQ (Ours)	8/8/16	68.65	78.45	75.79	71.74	59.34	43.94	41.40	**62.76**
RTN	8/8/8	62.97	73.72	62.60	57.77	52.86	36.77	37.00	54.81
SmoothQuant	8/8/8	63.55	75.95	70.99	66.30	53.96	40.10	40.60	58.78
FlexRound	8/8/8	66.94	79.00	75.32	69.38	58.54	42.92	40.40	61.79
LRQ (Ours)	8/8/8	68.59	78.67	75.83	70.64	58.16	43.34	39.80	**62.15**
Llama 2 70B	16/16/16	76.70	80.85	80.85	76.95	59.72	47.95	44.40	66.77
RTN	8/8/16	73.27	78.18	76.89	69.69	57.91	45.90	41.60	63.35
SmoothQuant	8/8/16	76.82	76.82	79.35	72.77	56.06	45.39	43.20	64.34
FlexRound	8/8/16	75.81	80.25	79.03	74.59	59.43	46.42	43.40	65.56
LRQ (Ours)	8/8/16	77.71	80.69	79.83	74.11	57.91	45.99	43.60	**65.69**
RTN	8/8/8	72.39	78.51	76.49	69.61	57.74	44.62	40.40	62.82
SmoothQuant	8/8/8	76.21	76.55	79.30	74.11	55.85	46.25	45.60	64.84
FlexRound	8/8/8	76.18	80.36	79.09	75.06	60.10	46.42	43.80	65.86
LRQ (Ours)	8/8/8	77.95	81.23	79.78	74.82	57.83	46.33	43.60	**65.93**

Table 19: Zero-shot performance of Llama 2 on common sense reasoning tasks (BoolQ, PIQA, HellaSwag, WinoGrande, ARC easy and challenge, and OpenBookQA) with per-channel asymmetric weight quantization, per-tensor asymmetric static activation quantization, and per-token asymmetric KV cache quantization (if applied). Please refer to Figure 8. The accuracy (%) is reported for common sense reasoning tasks. The number of bits used for weights, activations, and KV cache is expressed as W/A/KV.

Method	# Bits (W/A/KV)	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average
Llama 2 7B	16/16/16	71.07	76.99	72.96	67.25	53.58	40.53	40.80	60.45
RTN	8/8/8	60.58	67.08	57.66	60.54	45.83	31.57	34.40	51.09
SmoothQuant	8/8/8	67.65	73.29	67.52	62.90	51.35	37.80	37.60	56.87
FlexRound	8/8/8	72.05	77.26	71.30	65.98	54.88	39.16	39.20	**59.98**
LRQ (Ours)	8/8/8	67.86	76.99	71.97	67.01	54.71	40.19	40.00	59.82
Llama 2 13B	16/16/16	69.02	79.05	76.62	69.61	57.95	44.28	42.00	62.65
RTN	8/8/8	62.97	73.72	62.60	57.77	52.86	36.77	37.00	54.81
SmoothQuant	8/8/8	63.55	75.95	70.99	66.30	53.96	40.10	40.60	58.78
FlexRound	8/8/8	66.94	79.00	75.32	69.38	58.54	42.92	40.40	61.79
LRQ (Ours)	8/8/8	68.59	78.67	75.83	70.64	58.16	43.34	39.80	**62.15**
Llama 2 70B	16/16/16	76.70	80.85	80.85	76.95	59.72	47.95	44.40	66.77
RTN	8/8/8	72.39	78.51	76.49	69.61	57.74	44.62	40.40	62.82
SmoothQuant	8/8/8	76.21	76.55	79.30	74.11	55.85	46.25	45.60	64.84
FlexRound	8/8/8	76.18	80.36	79.09	75.06	60.10	46.42	43.80	65.86
LRQ (Ours)	8/8/8	77.95	81.23	79.78	74.82	57.83	46.33	43.60	**65.93**

Table 20: Five-shot performance of Llama 2 on Massive Multitask Language Understanding with per-channel asymmetric weight quantization, per-tensor asymmetric static activation quantization, and per-token asymmetric KV cache quantization (if applied). Please refer to Figure 8. The accuracy (%) is reported for four groups of disciplines (STEM, Humanities, Social Science, and Other). The number of bits used for weights, activations, and KV cache is expressed as W/A/KV.

Method	# Bits (W/A/KV)	STEM	Humanities	Social Science	Other	Average
Llama 2 7B	16/16/16	37.04	43.38	51.84	52.44	45.96
RTN	8/8/16	28.26	24.65	31.39	24.68	26.91
SmoothQuant	8/8/16	28.99	29.14	35.33	34.98	31.81
FlexRound	8/8/16	32.70	38.38	43.58	45.77	40.01
LRQ (Ours)	8/8/16	34.36	40.02	46.64	47.32	**41.94**
RTN	8/8/8	29.66	24.06	30.45	24.49	26.76
SmoothQuant	8/8/8	30.42	27.95	34.29	34.27	31.33
FlexRound	8/8/8	33.40	36.96	43.13	46.30	39.70
LRQ (Ours)	8/8/8	34.82	39.91	46.47	47.62	**42.04**
Llama 2 13B	16/16/16	44.27	54.43	63.41	60.76	55.68
RTN	8/8/16	29.16	24.38	30.52	29.49	27.93
SmoothQuant	8/8/16	28.76	29.46	34.68	35.44	31.83
FlexRound	8/8/16	41.95	51.20	60.90	59.65	53.29
LRQ (Ours)	8/8/16	42.78	52.65	61.85	59.25	**54.07**
RTN	8/8/8	29.06	24.23	29.93	29.03	27.62
SmoothQuant	8/8/8	30.98	29.29	35.36	35.29	32.37
FlexRound	8/8/8	41.09	51.58	61.39	59.41	53.28
LRQ (Ours)	8/8/8	42.88	51.97	62.14	59.93	**54.08**
Llama 2 70B	16/16/16	57.79	65.16	80.44	74.61	69.11
RTN	8/8/16	45.99	52.69	65.52	59.16	55.58
SmoothQuant	8/8/16	48.11	54.05	68.12	63.23	57.98
FlexRound	8/8/16	53.64	61.36	77.35	71.90	65.64
LRQ (Ours)	8/8/16	54.41	62.78	77.48	71.56	**66.23**
RTN	8/8/8	46.82	53.37	66.23	58.51	55.97
SmoothQuant	8/8/8	47.51	53.84	68.35	63.94	57.99
FlexRound	8/8/8	54.27	61.11	77.45	71.31	65.57
LRQ (Ours)	8/8/8	54.44	62.61	76.99	71.78	**66.12**

Table 21: Five-shot performance of Llama 2 on Massive Multitask Language Understanding with per-channel asymmetric weight quantization, per-tensor asymmetric static activation quantization, and per-token asymmetric KV cache quantization (if applied). Please refer to Figure 8. The accuracy (%) is reported for four groups of disciplines (STEM, Humanities, Social Science, and Other). The number of bits used for weights, activations, and KV cache is expressed as W/A/KV.

Method	# Bits (W/A/KV)	STEM	Humanities	Social Science	Other	Average
Llama 2 7B	16/16/16	37.04	43.38	51.84	52.44	45.96
RTN	8/8/8	29.66	24.06	30.45	24.49	26.76
SmoothQuant	8/8/8	30.42	27.95	34.29	34.27	31.33
FlexRound	8/8/8	33.40	36.96	43.13	46.30	39.70
LRQ (Ours)	8/8/8	34.82	39.91	46.47	47.62	**42.04**
Llama 2 13B	16/16/16	44.27	54.43	63.41	60.76	55.68
RTN	8/8/8	29.06	24.23	29.93	29.03	27.62
SmoothQuant	8/8/8	30.98	29.29	35.36	35.29	32.37
FlexRound	8/8/8	41.09	51.58	61.39	59.41	53.28
LRQ (Ours)	8/8/8	42.88	51.97	62.14	59.93	**54.08**
Llama 2 70B	16/16/16	57.79	65.16	80.44	74.61	69.11
RTN	8/8/8	46.82	53.37	66.23	58.51	55.97
SmoothQuant	8/8/8	47.51	53.84	68.35	63.94	57.99
FlexRound	8/8/8	54.27	61.11	77.45	71.31	65.57
LRQ (Ours)	8/8/8	54.44	62.61	76.99	71.78	**66.12**

Table 22: Zero-shot performance of Llama 2 on common sense reasoning tasks (BoolQ, PIQA, HellaSwag, WinoGrande, ARC easy and challenge, and OpenBookQA) with per-channel asymmetric weight quantization, per-token asymmetric activation quantization, and per-token asymmetric KV cache quantization (if applied). Please refer to Figure 9. The accuracy (%) is reported for common sense reasoning tasks. The number of bits used for weights, activations, and KV cache is expressed as W/A/KV.

Method	# Bits (W/A/KV)	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average
Llama 2 7B	16/16/16	71.07	76.99	72.96	67.25	53.58	40.53	40.80	60.45
RTN	8/8/16	69.54	76.93	72.21	67.17	53.24	41.04	40.60	60.10
SmoothQuant	8/8/16	70.73	77.04	72.67	66.77	53.37	40.78	41.60	60.42
FlexRound	8/8/16	72.26	76.88	72.57	66.93	53.70	40.36	40.40	60.44
LRQ (Ours)	8/8/16	72.54	77.15	72.58	67.09	53.70	41.04	40.40	**60.64**
RTN	8/8/8	69.60	77.20	72.26	67.09	53.62	39.85	41.00	60.09
SmoothQuant	8/8/8	70.61	77.42	72.62	66.54	53.37	40.27	40.80	60.23
FlexRound	8/8/8	72.02	77.09	72.50	67.40	54.17	40.19	40.80	60.60
LRQ (Ours)	8/8/8	72.45	77.04	72.70	67.09	53.66	40.61	41.60	**60.74**
RTN	4/8/16	67.95	74.32	65.84	62.12	46.68	37.20	35.80	55.70
SmoothQuant	4/8/16	42.54	64.15	41.15	54.06	35.61	27.99	32.00	42.50
FlexRound	4/8/16	71.96	77.04	72.17	65.59	53.58	39.85	40.20	60.06
LRQ (Ours)	4/8/16	72.94	76.88	71.85	65.27	53.96	39.85	40.80	**60.22**
RTN	4/8/8	68.13	75.14	65.89	62.67	46.42	36.52	36.20	55.85
SmoothQuant	4/8/8	43.03	63.71	41.08	54.30	35.69	27.99	32.60	42.63
FlexRound	4/8/8	71.71	76.77	72.24	66.14	53.49	40.02	40.40	60.11
LRQ (Ours)	4/8/8	73.00	76.99	71.90	65.98	54.38	39.68	41.20	**60.45**
Llama 2 13B	16/16/16	69.02	79.05	76.62	69.61	57.95	44.28	42.00	62.65
RTN	8/8/16	67.80	78.89	75.61	68.90	58.08	43.69	41.60	62.08
SmoothQuant	8/8/16	68.99	79.33	76.47	70.64	57.53	44.71	41.80	**62.78**
FlexRound	8/8/16	69.27	78.73	76.62	69.69	57.62	44.71	42.20	62.69
LRQ (Ours)	8/8/16	69.24	78.67	76.48	69.30	57.79	44.03	42.40	62.56
RTN	8/8/8	67.46	78.73	75.57	68.51	58.12	44.28	41.40	62.01
SmoothQuant	8/8/8	68.81	78.78	76.39	70.72	57.11	44.20	41.60	62.52
FlexRound	8/8/8	69.36	79.16	76.67	69.53	57.83	44.37	42.80	**62.82**
LRQ (Ours)	8/8/8	69.02	78.78	76.48	69.93	57.83	43.86	42.00	62.56
RTN	4/8/16	65.20	73.61	60.00	58.80	49.12	36.18	34.80	53.96
SmoothQuant	4/8/16	61.74	57.07	37.17	52.09	31.61	24.40	31.40	42.21
FlexRound	4/8/16	69.14	78.67	75.67	68.98	58.92	44.20	41.00	62.37
LRQ (Ours)	4/8/16	71.10	78.29	75.75	69.30	57.74	43.69	41.00	**62.41**
RTN	4/8/8	65.23	74.05	60.04	58.64	49.07	35.92	35.20	54.02
SmoothQuant	4/8/8	61.62	56.53	37.31	51.38	31.57	24.74	30.60	41.96
FlexRound	4/8/8	69.05	78.51	75.51	69.53	58.75	43.60	41.20	62.31
LRQ (Ours)	4/8/8	71.13	78.29	75.79	68.90	57.83	43.34	41.20	**62.35**
Llama 2 70B	16/16/16	76.70	80.85	80.85	76.95	59.72	47.95	44.40	66.77
RTN	8/8/16	76.02	81.07	80.37	76.01	60.14	48.04	44.40	66.58
SmoothQuant	8/8/16	75.75	80.96	80.60	77.90	59.34	47.70	45.20	66.78
FlexRound	8/8/16	75.72	81.56	80.60	75.77	60.19	48.89	44.80	**66.79**
LRQ (Ours)	8/8/16	75.84	81.66	80.64	75.93	60.40	48.38	44.00	66.69
RTN	8/8/8	76.02	81.07	80.45	75.61	60.31	47.87	43.80	66.45
SmoothQuant	8/8/8	76.06	81.07	80.63	76.32	59.51	47.61	45.20	66.63
FlexRound	8/8/8	75.93	81.45	80.48	75.85	60.06	48.55	44.80	66.73
LRQ (Ours)	8/8/8	75.99	81.50	80.61	75.77	59.97	49.49	45.20	**66.93**
RTN	4/8/16	75.63	78.73	71.28	69.61	53.24	43.34	40.20	61.72
SmoothQuant	4/8/16	49.79	70.95	48.43	54.70	44.74	32.51	37.80	48.42
FlexRound	4/8/16	77.80	80.90	80.06	74.66	60.31	47.61	43.60	66.42
LRQ (Ours)	4/8/16	77.92	80.74	80.38	75.14	60.35	47.95	42.80	**66.47**
RTN	4/8/8	75.90	79.22	71.39	70.56	53.11	43.60	40.40	62.03
SmoothQuant	4/8/8	50.46	71.60	48.35	55.09	44.87	32.17	37.40	48.56
FlexRound	4/8/8	77.31	80.96	79.89	75.30	60.19	48.21	43.40	66.47
LRQ (Ours)	4/8/8	77.92	81.28	80.42	75.06	60.94	48.04	42.60	**66.61**

Table 23: Zero-shot performance of Llama 2 on common sense reasoning tasks (BoolQ, PIQA, HellaSwag, WinoGrande, ARC easy and challenge, and OpenBookQA) with per-channel asymmetric weight quantization, per-token asymmetric activation quantization, and per-token asymmetric KV cache quantization (if applied). Please refer to Figure 9. The accuracy (%) is reported for common sense reasoning tasks. The number of bits used for weights, activations, and KV cache is expressed as W/A/KV.

Method	# Bits (W/A/KV)	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average
Llama 2 7B	16/16/16	71.07	76.99	72.96	67.25	53.58	40.53	40.80	60.45
RTN	4/8/8	68.13	75.14	65.89	62.67	46.42	36.52	36.20	55.85
SmoothQuant	4/8/8	43.03	63.71	41.08	54.30	35.69	27.99	32.60	42.63
FlexRound	4/8/8	71.71	76.77	72.24	66.14	53.49	40.02	40.40	60.11
LRQ (Ours)	4/8/8	73.00	76.99	71.90	65.98	54.38	39.68	41.20	**60.45**
Llama 2 13B	16/16/16	69.02	79.05	76.62	69.61	57.95	44.28	42.00	62.65
RTN	4/8/8	65.23	74.05	60.04	58.64	49.07	35.92	35.20	54.02
SmoothQuant	4/8/8	61.62	56.53	37.31	51.38	31.57	24.74	30.60	41.96
FlexRound	4/8/8	69.05	78.51	75.51	69.53	58.75	43.60	41.20	62.31
LRQ (Ours)	4/8/8	71.13	78.29	75.79	68.90	57.83	43.34	41.20	**62.35**
Llama 2 70B	16/16/16	76.70	80.85	80.85	76.95	59.72	47.95	44.40	66.77
RTN	4/8/8	75.90	79.22	71.39	70.56	53.11	43.60	40.40	62.03
SmoothQuant	4/8/8	50.46	71.60	48.35	55.09	44.87	32.17	37.40	48.56
FlexRound	4/8/8	77.31	80.96	79.89	75.30	60.19	48.21	43.40	66.47
LRQ (Ours)	4/8/8	77.92	81.28	80.42	75.06	60.94	48.04	42.60	**66.61**

Table 24: Five-shot performance of Llama 2 on Massive Multitask Language Understanding with per-channel asymmetric weight quantization, per-token asymmetric activation quantization, and per-token asymmetric KV cache quantization (if applied). Please refer to Figure 9. The accuracy (%) is reported for four groups of disciplines (STEM, Humanities, Social Science, and Other). The number of bits used for weights, activations, and KV cache is expressed as W/A/KV.

Method	# Bits (W/A/KV)	STEM	Humanities	Social Science	Other	Average
Llama 2 7B	16/16/16	37.04	43.38	51.84	52.44	45.96
RTN	8/8/16	36.41	42.49	50.31	52.47	45.20
SmoothQuant	8/8/16	37.01	43.04	51.64	52.13	45.73
FlexRound	8/8/16	36.38	42.91	51.80	52.87	45.76
LRQ (Ours)	8/8/16	36.91	43.27	52.19	52.78	**46.05**
RTN	8/8/8	36.15	42.85	50.34	52.31	45.24
SmoothQuant	8/8/8	36.22	43.19	51.25	52.22	45.54
FlexRound	8/8/8	36.98	42.91	51.87	52.28	45.76
LRQ (Ours)	8/8/8	36.88	43.12	51.67	52.53	**45.83**
RTN	4/8/16	27.63	25.87	27.82	28.32	27.24
SmoothQuant	4/8/16	27.07	24.72	22.49	25.88	25.00
FlexRound	4/8/16	37.01	42.40	50.80	50.34	44.92
LRQ (Ours)	4/8/16	36.78	42.66	51.19	51.73	**45.36**
RTN	4/8/8	28.00	25.80	27.53	28.01	27.16
SmoothQuant	4/8/8	26.77	24.87	22.81	25.85	25.05
FlexRound	4/8/8	37.81	42.55	50.47	50.65	45.14
LRQ (Ours)	4/8/8	36.88	42.53	50.80	52.22	**45.36**
Llama 2 13B	16/16/16	44.27	54.43	63.41	60.76	55.68
RTN	8/8/16	43.57	52.88	61.88	61.17	54.76
SmoothQuant	8/8/16	44.04	53.69	63.21	61.35	55.47
FlexRound	8/8/16	43.84	53.65	63.37	61.10	55.39
LRQ (Ours)	8/8/16	44.80	53.75	63.47	60.73	**55.57**
RTN	8/8/8	43.87	52.88	62.33	60.67	54.81
SmoothQuant	8/8/8	44.10	53.58	63.11	60.95	55.33
FlexRound	8/8/8	44.17	52.88	63.76	61.29	55.33
LRQ (Ours)	8/8/8	44.50	53.07	63.24	61.26	**55.35**
RTN	4/8/16	30.55	26.08	33.51	35.07	30.74
SmoothQuant	4/8/16	27.04	24.23	25.48	26.16	25.55
FlexRound	4/8/16	42.91	50.80	62.11	60.27	53.77
LRQ (Ours)	4/8/16	43.24	52.41	61.78	60.24	**54.30**
RTN	4/8/8	30.95	26.31	32.92	34.58	30.67
SmoothQuant	4/8/8	27.07	24.25	25.22	26.43	25.57
FlexRound	4/8/8	42.88	50.71	61.94	59.93	53.77
LRQ (Ours)	4/8/8	43.90	52.56	62.07	59.96	**54.49**
Llama 2 70B	16/16/16	57.79	65.16	80.44	74.61	69.11
RTN	8/8/16	56.06	63.00	78.32	73.10	67.20
SmoothQuant	8/8/16	58.02	64.53	80.21	74.21	**68.80**
FlexRound	8/8/16	57.69	63.80	79.98	73.63	68.30
LRQ (Ours)	8/8/16	57.95	64.48	80.21	73.90	68.70
RTN	8/8/8	56.23	63.55	78.39	73.01	67.41
SmoothQuant	8/8/8	57.49	64.68	80.37	74.43	**68.82**
FlexRound	8/8/8	57.22	63.97	79.62	73.81	68.22
LRQ (Ours)	8/8/8	57.95	63.85	80.34	73.94	68.52
RTN	4/8/16	41.12	45.72	56.78	53.49	48.95
SmoothQuant	4/8/16	26.84	24.08	26.55	25.88	25.63
FlexRound	4/8/16	59.96	62.98	79.04	73.23	67.56
LRQ (Ours)	4/8/16	56.46	64.59	79.07	72.83	**67.92**
RTN	4/8/8	41.19	45.74	57.52	53.61	49.16
SmoothQuant	4/8/8	27.37	24.59	27.59	25.94	26.16
FlexRound	4/8/8	56.26	62.89	78.78	72.92	67.26
LRQ (Ours)	4/8/8	55.57	64.65	78.97	72.52	**67.65**

Table 25: Five-shot performance of Llama 2 on Massive Multitask Language Understanding with per-channel asymmetric weight quantization, per-token asymmetric activation quantization, and per-token asymmetric KV cache quantization (if applied). Please refer to Figure 9. The accuracy (%) is reported for four groups of disciplines (STEM, Humanities, Social Science, and Other). The number of bits used for weights, activations, and KV cache is expressed as W/A/KV.

Method	# Bits (W/A/KV)	STEM	Humanities	Social Science	Other	Average
Llama 2 7B	16/16/16	37.04	43.38	51.84	52.44	45.96
RTN	4/8/8	28.00	25.80	27.53	28.01	27.16
SmoothQuant	4/8/8	26.77	24.87	22.81	25.85	25.05
FlexRound	4/8/8	37.81	42.55	50.47	50.65	45.14
LRQ (Ours)	4/8/8	36.88	42.53	50.80	52.22	**45.36**
Llama 2 13B	16/16/16	44.27	54.43	63.41	60.76	55.68
RTN	4/8/8	30.95	26.31	32.92	34.58	30.67
SmoothQuant	4/8/8	27.07	24.25	25.22	26.43	25.57
FlexRound	4/8/8	42.88	50.71	61.94	59.93	53.77
LRQ (Ours)	4/8/8	43.90	52.56	62.07	59.96	**54.49**
Llama 2 70B	16/16/16	57.79	65.16	80.44	74.61	69.11
RTN	4/8/8	41.19	45.74	57.52	53.61	49.16
SmoothQuant	4/8/8	27.37	24.59	27.59	25.94	26.16
FlexRound	4/8/8	56.26	62.89	78.78	72.92	67.26
LRQ (Ours)	4/8/8	55.57	64.65	78.97	72.52	**67.65**

In Table 16 and 18, LRQ exhibits a slightly superior zero-shot performance on common sense reasoning tasks compared to FlexRound, which we believe is noteworthy since FlexRound already achieves the zero-shot performance on common sense reasoning tasks comparable to FP16 baselines. The close proximity in zero-shot performance between FlexRound and FP16 baselines on common sense reasoning tasks limits the potential for a substantial performance disparity between FlexRound and LRQ. Despite LRQ approaching the zero-shot performance of FP16 baselines more closely than FlexRound, the difference in zero-shot performance between FlexRound and LRQ cannot be anticipated to be large after all.

Nevertheless, as expounded in Section 1, it is crucial to emphasize that LRQ demonstrates competitive performance relative to FP16 baselines on both common sense reasoning tasks and Massive Multitask Language Understanding (MMLU), a feat not accomplished by FlexRound that excels solely on common sense reasoning tasks. Given the comprehensive evaluation of large language models (LLMs) across diverse benchmarks, the proficiency of LRQ in excelling across both common sense reasoning tasks and MMLU holds significant implications in the field of LLM quantization.

Regarding the 8-bit weight quantization presented in Table 22 and 24, the adoption of a per-token asymmetric activation quantization scheme results in even naive rounding-to-nearest (RTN) performing closely to the levels of FP16 baselines on both common sense reasoning tasks and MMLU. As a result, while LRQ exhibits slightly higher accuracy compared to SmoothQuant and FlexRound for most Llama 2 models, it can be concluded that SmoothQuant, FlexRound, and LRQ are nearly evenly matched.

In the context of 4-bit weight quantization as presented in Table 22, FlexRound achieves zero-shot accuracy levels comparable to FP16 baselines on common sense reasoning tasks, resulting in a relatively small zero-shot performance gap between FlexRound and LRQ, like the scenario depicted in Table 16 and 18. However, in the case of 4-bit weight quantization in Table 24, LRQ surpasses FlexRound by a margin ranging from 0.2 to 0.7 percent. Although these increments in five-shot accuracy on MMLU in Table 24 may seem modest compared to those in Table 17 and 20, we believe that the rise in five-shot accuracy by 0.2 to 0.7 percent on MMLU is significant. This is particularly noteworthy as it brings the five-shot accuracy gap between LRQ and FP16 baselines to less than 1.5 percent on MMLU, while the corresponding gap between FlexRound and FP16 baselines remains more or less at two percent for Llama 2 13B and 70B.

I Implementation Details

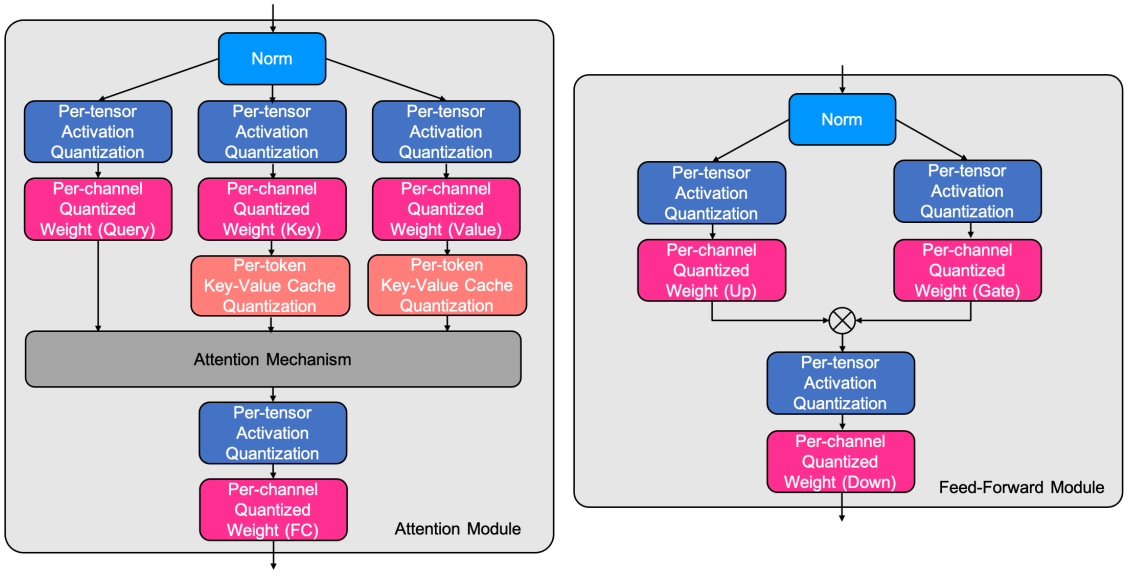


Figure 8: Illustration of a quantized Transformer block with per-channel asymmetric weight quantization, per-tensor asymmetric static activation quantization, and per-token asymmetric KV cache quantization. We remain the inputs of softmax and normalization layers in FP16.

Table 26: Learning rate and batch size for FlexRound and LRQ when employing a per-tensor asymmetric static activation quantization scheme (see Figure 8) in Table 1, 2, 3, 4, 16, 17, 18, and 20.

Method	Configuration	Llama 7B	Llama 13B	Llama 33B	Llama 65B	Llama 2 7B	Llama 2 13B	Llama 2 70B
FlexRound	Learning rate	3e-3	3e-3	1e-3	2e-3	3e-3	3e-3	1e-3
	Batch size	4	4	2	2	2	2	2
LRQ	Learning rate	3e-3	2e-3	1.5e-3	1e-3	1e-3	1.5e-3	1e-3
	Batch size	2	2	2	2	2	2	2

For the quantization scheme depicted in Figure 8, both FlexRound and LRQ are implemented in the experimental setting of QDrop (Wei et al., 2022) with the exception of the number of iterations for block-wise reconstruction, the batch size, and the learning rate. For all the Llama and Llama 2 models, the number of iterations for block-wise reconstruction is set to 5000 for both FlexRound and LRQ. The learning rate and the batch size for FlexRound and LRQ are described in 26. Notice that when applying LRQ to Llama 2 70B, the key and value projection weights are quantized via not LRQ but FlexRound due to the presence of GQA (Ainslie et al., 2023) in Llama 2 70B. To obtain the experimental results in Table 1 and 3, per-token asymmetric KV cache quantization is applied after completing block-wise reconstruction for all the Transformer blocks. For both activation quantization and KV cache quantization, we employ rounding-to-nearest.

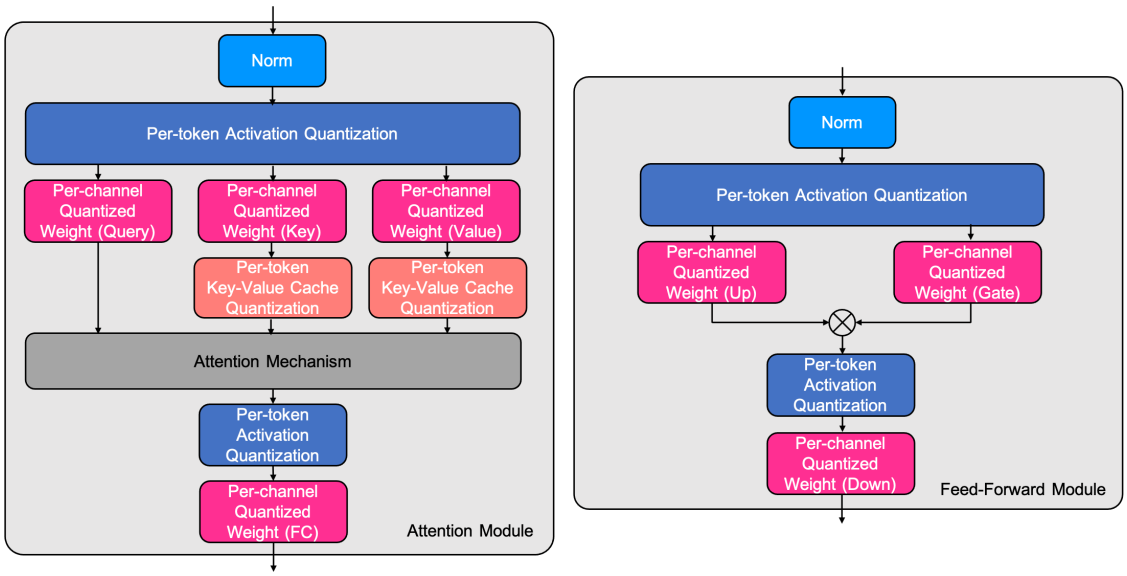


Figure 9: Illustration of a quantized Transformer block with per-channel asymmetric weight quantization, per-token asymmetric activation quantization, and per-token asymmetric KV cache quantization. We remain the inputs of softmax and normalization layers in FP16.

Table 27: Learning rate for FlexRound and LRQ when adopting a per-token asymmetric activation quantization scheme (see Figure 9) in Table 5, 6, 22, and 24.

Method	Weight	Llama 2 7B	Llama 2 13B	Llama 2 70B
FlexRound	8-bit	1e-4	4e-4	3e-4
	4-bit	5e-4	4e-4	5e-4
LRQ	8-bit	1e-4	2e-4	4e-4
	4-bit	5e-4	5e-4	4e-4

In the case of quantization scheme indicated in Figure 9, both FlexRound and LRQ are first implemented in the experimental setting of BRECQ (Li et al., 2021) with the exception of the number of iterations for block-wise reconstruction, the batch size, and the learning rate. The number of iterations for block-wise reconstruction and the batch size are set to 5000 and 2 respectively, for every Llama 2 model regardless of the number of bits used for weights. Table 27 exhibits the learning rate for FlexRound and LRQ in the case of 8-bit and 4-bit weight quantization, respectively. As explained in the above paragraph, when LRQ is applied to Llama 2 70B, weights in key and value projections are quantized via FlexRound. Here, when quantizing Llama 2 7B into 4-bit via LRQ, the attention module is quantized via LRQ, but the feed-forward module is quantized via FlexRound. In addition, when quantizing Llama 2 70B into 4-bit via LRQ, the feed-forward module is quantized via LRQ, but the attention module is quantized via FlexRound. To gain the experimental results in Table 5 and 6, per-token asymmetric activation quantization and per-token asymmetric KV cache quantization are sequentially applied after finishing block-wise reconstruction for all the Transformer blocks. For both activation quantization and KV cache quantization, we employ rounding-to-nearest.

All experiments about SmoothQuant are conducted based on the code provided in the SmoothQuant github repository[1]. Following Xiao et al. (2022), we select α, the hyperparameter to determine how much difficulty of activation quantization to shift to weight quantization, to be 0.8 for every Llama model, 0.85 for Llama 2 7B and 13B, and 0.9 for Llama 2 70B.

Table 28 displays the learning rate for LRQ employed in Table 7. For OmniQuant in Table 7, we

[1] https://github.com/mit-han-lab/smoothquant

Table 28: Learning rate for LRQ when adopting per-channel weight-only quantization in Table 7.

Method	Weight	Llama 2 7B	Llama 2 13B	Llama 2 70B
FlexRound	3-bit	2.5e-3	5e-4	5e-4
	4-bit	3e-3	7e-4	5e-4
LRQ	3-bit	6e-4	5e-4	5e-4
	4-bit	9e-4	6e-4	5e-4

utilized pre-trained OmniQuant models provided in the OmniQuant github repository[2].

For evaluation, we use Eleuther AI's *lm-evaluation-harness* (Gao et al., 2021) for common sense reasoning tasks and follow the evalution code in the MMLU github repository[3] for the MMLU benchmark.

[2] https://github.com/OpenGVLab/OmniQuant
[3] https://github.com/hendrycks/test

J Ratio of the number of learnable parameters in LRQ to the number of pre-trained weights

Table 29: Ratio of the number of learnable parameters in LRQ to the number of pre-trained weights for an intermediate Transformer block of each Llama model when setting the rank r to 2048 for large language models beyond 30B parameters or to 1024 for smaller models.

Model	Number of pre-trained weights (A)	Number of learnable parameters in LRQ (B)	Ratio (B/A)
Llama 7B	$4096 \times 4096 \times 4$ $+4096 \times 11008 \times 3$	$(4096 \times 1024 + 1024 \times 4096) \times 4$ $+(4096 \times 1024 + 1024 \times 11008) \times 3$	$\mathbf{39.51}\%$
Llama 13B	$5120 \times 5120 \times 4$ $+5120 \times 13824 \times 3$	$(5120 \times 1024 + 1024 \times 5120) \times 4$ $+(5120 \times 1024 + 1024 \times 13824) \times 3$	$\mathbf{31.57}\%$
Llama 33B	$6656 \times 6656 \times 4$ $+6656 \times 17920 \times 3$	$(6656 \times 2048 + 2048 \times 6656) \times 4$ $+(6656 \times 2048 + 2048 \times 17920) \times 3$	$\mathbf{48.60}\%$
Llama 65B	$8192 \times 8192 \times 4$ $+8192 \times 22016 \times 3$	$(8192 \times 2048 + 2048 \times 8192) \times 4$ $+(8192 \times 2048 + 2048 \times 22016) \times 3$	$\mathbf{39.51}\%$

K Average and Standard Deviation of FlexRound and LRQ

For common sense reasoning tasks in Table 1 and 2, LRQ slightly outperforms FlexRound in the case of Llama 2 70B and significantly surpasses FlexRound in the case of Llama 33B, but FlexRound is better than LRQ in the case of Llama 2 7B. To investigate how meaningful the improvement of LRQ over FlexRound is, we carry out three random trials for Llama 2 7B, Llama 33B, and Llama 2 70B, presenting the average and standard deviation of them.

Table 30: Average and standard deviation of zero-shot performance of FlexRound and LRQ over three random trials on common sense reasoning tasks (BoolQ, PIQA, HellaSwag, WinoGrande, ARC easy and challenge, and OpenBookQA) with per-channel asymmetric weight quantization, per-tensor asymmetric static activation quantization, and per-token asymmetric KV cache quantization. The number of bits used for weights, activations, and KV cache is 8-bit.

Method	# Bits (W/A/KV)	Llama 2 7B	Llama 33B	Llama 2 70B
FlexRound	8/8/8	59.72 ± 0.73	62.83 ± 0.36	65.65 ± 0.30
LRQ (Ours)	8/8/8	$\mathbf{59.90} \pm \mathbf{0.18}$	$\mathbf{63.81} \pm \mathbf{0.16}$	$\mathbf{65.89} \pm \mathbf{0.06}$

As seen in Table 30, not only does the average of LRQ surpass that of FlexRound, but the standard deviation of LRQ is also smaller than that of FlexRound, which strengthens our assertion that FlexRound might be prone to overfitting when applied to the quantization of LLMs.

L Combination of SmoothQuant with FlexRound and LRQ

Table 31: Zero-shot performance of Llama 7B on common sense reasoning tasks (BoolQ, PIQA, HellaSwag, WinoGrande, ARC easy and challenge, and OpenBookQA) with per-channel asymmetric weight quantization and per-tensor asymmetric static activation quantization, while keeping the KV cache in FP16. Here, 'SQ + FlexRound' and 'SQ + LRQ' denote FlexRound and LRQ that initially begin their own learning process from the SmoothQuant baseline in lieu of the rounding-to-nearest baseline, respectively. The accuracy (%) is reported for common sense reasoning tasks. The number of bits used for weights, activations, and KV cache is expressed as W/A/KV.

Method	# Bits (W/A/KV)	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average
Llama 7B	16/16/16	73.15	77.31	72.96	67.09	52.48	41.38	42.40	60.97
FlexRound	8/8/16	73.76	76.66	71.75	67.01	52.31	40.02	42.20	**60.53**
SQ+FlexRound	8/8/16	73.85	76.77	71.54	66.38	51.43	40.44	41.60	60.29
LRQ	8/8/16	73.03	77.64	72.10	66.77	52.95	40.87	41.60	**60.71**
SQ+LRQ	8/8/16	73.15	76.88	72.24	66.38	52.86	40.61	40.60	60.39

Table 32: Five-shot performance of Llama 7B on Massive Multitask Language Understanding with per-channel asymmetric weight quantization and per-tensor asymmetric static activation quantization, while keeping the KV cache in FP16. Here, 'SQ + FlexRound' and 'SQ + LRQ' denote FlexRound and LRQ that initially begin their own learning process from the SmoothQuant baseline in lieu of the rounding-to-nearest baseline, respectively. The accuracy (%) is reported for four groups of disciplines (STEM, Humanities, Social Science, and Other). The number of bits used for weights, activations, and KV cache is expressed as W/A/KV.

Method	# Bits (W/A/KV)	STEM	Humanities	Social Science	Other	Average
Llama 7B	16/16/16	30.58	33.88	38.19	38.25	35.12
FlexRound	8/8/16	28.30	29.20	30.13	33.47	30.20
SQ+FlexRound	8/8/16	30.98	29.71	33.80	35.26	**32.16**
LRQ	8/8/16	29.69	32.48	37.63	38.80	**34.47**
SQ+LRQ	8/8/16	30.35	31.84	37.44	37.32	34.01

As SmoothQuant is orthogonal to block-wise reconstruction, one might wonder how the performance of FlexRound and LRQ would change when FlexRound and LRQ start their own learning process from the SmoothQuant baseline in place of the RTN baseline. Table 31 and 32 reveal the performance of 'SmoothQuant (SQ) + FlexRound' and 'SmoothQuant (SQ) + LRQ' on common sense reasoning benchmarks and the MMLU benchmark, respectively. Unfortunately, in most cases, SmoothQuant does not display its efficacy when combined with FlexRound and LRQ. Although SmoothQuant enhances five-shot performance of FlexRound on MMLU by almost two percent, 'SQ + FlexRound' still underperforms LRQ as well as 'SQ + LRQ' on MMLU, which implies that employing low-rank weight-scaling matrices would be a better choice than using full weight-scaling matrices with additional pre-processing like an uniform per-channel scaling transformation in SmoothQuant.

M Process of $\boldsymbol{L}_2\boldsymbol{U}_2 + \boldsymbol{r}_2 + \boldsymbol{c}_2$ in Eq. 2

Similar to the broadcasting process in Python Numpy, we add $\boldsymbol{L}_2\boldsymbol{U}_2$, $\boldsymbol{r}_2$, and $\boldsymbol{c}_2$. To be more specific, let $\boldsymbol{L}_2\boldsymbol{U}_2$ be

$$\begin{bmatrix} LU_{(1,1)} & LU_{(1,2)} & \cdots & LU_{(1,C_{in})} \\ \vdots & \vdots & \vdots & \vdots \\ LU_{(C_{out},1)} & LU_{(C_{out},2)} & \cdots & LU_{(C_{out},C_{in})}, \end{bmatrix}$$

$\boldsymbol{r}_2$ be

$$\begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{C_{out}} \end{bmatrix},$$

and $\mathbf{c}_2$ be

$$\begin{bmatrix} c_1 & c_2 & \cdots & c_{C_{in}} \end{bmatrix}.$$

Then, by the broadcasting process, $\boldsymbol{L}_2\boldsymbol{U}_2 + \boldsymbol{r}_2 + \boldsymbol{c}_2$ can be expressed as

$$\begin{bmatrix} LU_{(1,1)} + r_1 + c_1 & LU_{(1,2)} + r_1 + c_2 & \cdots & LU_{(1,C_{in})} + r_1 + c_{C_{in}} \\ \vdots & \vdots & \vdots & \vdots \\ LU_{(C_{out},1)} + r_{C_{out}} + c_1 & LU_{(C_{out},2)} + r_{C_{out}} + c_2 & \cdots & LU_{(C_{out},C_{in})} + r_{C_{out}} + c_{C_{in}} \end{bmatrix}.$$