FLRQ: Faster LLM Quantization with Flexible Low-Rank Matrix Sketching

Hongyaoxing Gu[1,2], Lijuan Hu[1], Shuzi Niu[1], Fangfang Liu[1,3*]

[1]Institute of Software Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]Key Laboratory of System Software (Chinese Academy of Sciences)

Abstract

Traditional post-training quantization (PTQ) is considered an effective approach to reduce model size and accelerate inference of large-scale language models (LLMs). However, existing low-rank PTQ methods require costly fine-tuning to determine a compromise rank for diverse data and layers in large models, failing to exploit their full potential. Additionally, the current SVD-based low-rank approximation compounds the computational overhead. In this work, we thoroughly analyze the varying effectiveness of low-rank approximation across different layers in representative models. Accordingly, we introduce Flexible Low-Rank Quantization (FLRQ), a novel solution designed to quickly identify the accuracy-optimal ranks and aggregate them to achieve minimal storage combinations. FLRQ comprises two powerful components, Rank1-Sketch-based Flexible Rank Selection (R1-FLR) and Best Low-rank Approximation under Clipping (BLC). R1-FLR applies the R1-Sketch with Gaussian projection for the fast low-rank approximation, enabling outlier-aware rank extraction for each layer. Meanwhile, BLC aims at minimizing the low-rank quantization error under the scaling and clipping strategy through an iterative method. FLRQ demonstrates strong effectiveness and robustness in comprehensive experiments, achieving state-of-the-art performance in both quantization quality and algorithm efficiency.

Introduction

Large language models (LLMs) have demonstrated outstanding capabilities across a wide range of tasks that significantly impact our daily lives. However, the scale of LLMs makes their deployment extremely challenging, necessitating quantization techniques for reducing model size while improving inference efficiency (Kuzmin et al. 2023).

In quantization methods, Quantization-Aware Training (QAT) incorporates pseudo-quantization layers to simulate the quantization process (Liu et al. 2024), requiring retraining with different quantization parameters. This approach inevitably demands higher computational costs and can be affected by modifications to the original model architecture. In contrast, Post-Training Quantization (PTQ) has recently gained more attention for a faster quantization process and state-of-the-art PTQ methods have achieved accuracy comparable to QAT (Ding et al. 2022; Hubara et al. 2021; Frantar et al. 2023). Among post-training quantization (PTQ) methods, low-rank approaches have garnered significant attention in recent years due to their effectiveness in capturing essential weight information, thereby enhancing quantization accuracy.

Method	Rank	Lora func
LoRC(Yao et al. 2024)	1,4,8,16,32	SVD
LoftQ(Li et al. 2024b)	16,32	SVD
LQER(Zhang et al. 2024)	600	SVD
L^2QER(Zhang et al. 2024)	32,64,256	SVD
SVD-Quant(Li et al. 2024a)	16,32,64	SVD
FLRQ(ours)	Flexible	R1-Sketch

Table 1: Previous low-rank quantization methods are constrained to restricted low-rank components, while FLRQ under R1-Sketch provides the flexibility to select effective low-rank approximations, thereby minimizing computational resource wastage and achieving high efficiency.

Current low-rank quantization approaches typically adopt a fixed rank and apply it uniformly across all layers of a model. However, recent studies reveal that different layers exhibit varying degrees of importance. This uniform-rank strategy has two primary drawbacks: 1. For models of different sizes, the memory usage ratio under the same rank varies, necessitating separate tuning of the fixed rank for each model; 2. A fixed rank leads to inconsistent effectiveness in capturing weight feature across layers, resulting in potential memory waste and insufficient performance.

Given these issues, we have proposed an iterative FLRQ method for flexible rank selection in different layers of the model, which is also shown in Figure 1. Furthermore, we enhance the matrix sketching algorithm to accelerate our proposed method. Our key contributions are as follows.

- We analyzed the effectiveness of reducing error in different layers of large model weights under low-rank approximation, and proposed FLRQ for selecting optimal low-rank components across various layers, which does not require expensive knowledge distillation, hyperparameter search, and is easily integrated with other approaches.

*Corresponding author

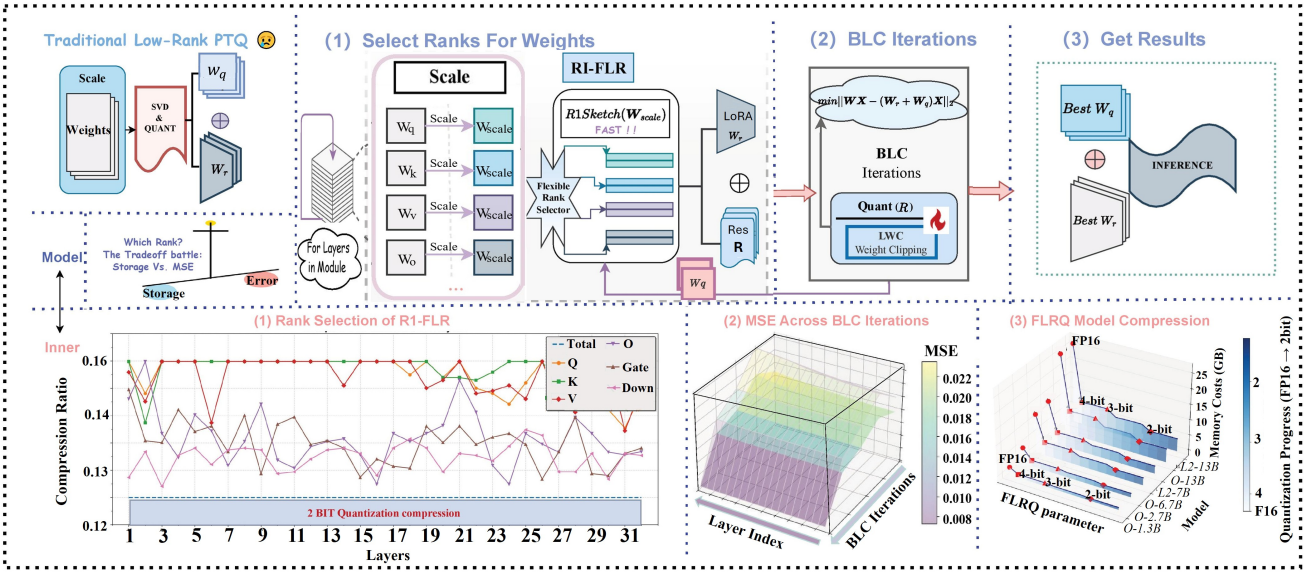

Figure 1: The presentation at the model layer and internals of FLRQ, (1) to (3) represent the three steps of the FLRQ algorithm respectively. In detail, FLRQ utilizes flexible rank selections (1), activation-based scaling, and iterative Best Low-rank Approximation under Clipping (BLC) algorithm (2), leading to higher quantization accuracy and smaller model size (3).

- We proposed rank-1 sketch (R1-Sketch) under Gaussian projection by reformulating the randomized SVD algorithm and carrying efficient implementation on GPU platform. This technique is suited for low-rank component extraction and solely utilizes BLAS Level-2 routines, ensuring highly efficient computations.
- Experiments demonstrate that FLRQ achieves comparable accuracy with fewer memory costs and outperforms conventional schemes by a significant margin in accuracy, especially for low-bit precision (INT2). Moreover, the R1-Sketch method effectively reduces computation time and has low inference latency.

Background and Related Work

Post-Training Quantization

PTQ (Post-Training Quantization), as a technique for model compression, has received extensive attention in recent years: Weight-only quantization in PTQ can be expressed as

$$\boldsymbol{O} = Quant(\boldsymbol{W}) \cdot \boldsymbol{X}. \tag{1}$$

AWQ (Lin et al. 2024) enhances the quantization accuracy by preserving the top 1% of significant activation channels by applying a scale matrix. GPTQ (Frantar et al. 2023) quantizes each parameter by OBS (Hassibi, Stork, and Wolff 1993) per block to mitigate the accuracy loss from quantization.

Recent studies like Omniquant (Shao et al. 2023) introduced learnable weight clipping, Quip# (Tseng et al. 2024) employes a randomized matrix transformation, while Affinequant (Ma et al. 2024) uses equivalent affine transformations, CALDERA (Saha et al. 2024) applies low-rank decomposition in quantization. These methods consider the challenging regime of sub-4 bit post-training LLM quantization but there are still challenges for 2-bit quantization.

Low-Rank methods in LLMs quantization

Low-rank methods in quantization have attracted considerable attention in recent years. Low-rank quantization takes advantage of the low-rank decomposition of the weight quantization matrix (Yao et al. 2024), followed by residual quantization represented as:

$$\boldsymbol{W}\boldsymbol{X} = (Quant(\boldsymbol{W} - \boldsymbol{W}_r) + \boldsymbol{W}_r)\boldsymbol{X}, \tag{2}$$

where thr rank r matrix $\boldsymbol{W}_r$ is the low-rank approximation of W with minimal error, which can be calculated by SVD (Singular value decomposition):

$$\boldsymbol{W}_r = SVD(\boldsymbol{W}) = (U_r\Sigma_r)V_r^T = \boldsymbol{W}_L\boldsymbol{W}_R. \tag{3}$$

A small subset of high singular value components can be isolated, and extracting these low-rank matrices effectively reduces the impact of outliers. This approach is broadly categorized into two types:

- **low-rank within quantization:** The former belongs to post-training quantization (PTQ) methods, where low-rank approximation is applied during quantization to reduce weight outliers and thereby minimize quantization error. For instance, ViTALiTy (Dass et al. 2023) employs a combination of sparsification and low-rank approximation, while LQER (Zhang et al. 2024) integrates Block Floating Point techniques to further enhance accuracy. Similarly, SVD-Quant (Li et al. 2024a) introduces low-rank methods into diffusion models, yielding significant performance improvements.
- **low-rank fine-tuning after quantization:** This involves refining the PTQ quantized model on a dataset to improve accuracy, as exemplified by methods such as LoftQ (Li et al. 2024b) achieves nearly lossless inference quantization by designing tailored singular value distributions.

Recent work like CALDERA (Saha et al. 2024) improves L2-loss by iteration and applies mix-precision in $\boldsymbol{W}_r$, and RILQ (Lee et al. 2025) considers the construct of $\boldsymbol{W}_r$ by model loss instead of linear loss, both methods perform low-rank fine-tuning based on state-of-the-art PTQ like Omniquant (Shao et al. 2023) and achieve high-precision 2-bit models on QUIP# (Tseng et al. 2024).

Sketching matrix and RSVD algorithm

Matrix sketching is a technique designed to efficiently handle large-scale matrices. Its core idea is to construct a "sketch" through randomization. This approach reduces storage and computational costs while preserving most of the essential information. Randomized Singular Value Decomposition (RSVD) (Halko, Martinsson, and Tropp 2011) represents one class of matrix sketching techniques utilized for low-rank approximation and has been widely applied in computer vision (Ji and Li 2014; Osawa et al. 2017) and machine learning (Guan, Li, and Guan 2017; Kumar 2016). This algorithm is generally divided into the following two steps:

1. Compute an approximate basis for the column space of $A \in \mathbb{R}^{m\times n}$, assume $m \leq n$. Attempt it in it times to obtain a matrix Q with r orthogonal columns that approximates matrix A. Formally, $A_r \approx QQ^*A$, where Q^* denotes the conjugate transpose of Q.
2. Utilize the orthogonal matrix Q to calculate a much smaller rank-k matrix Q^*A, and employ it to compute the desired low-rank matrix by Singular Value Decomposition.

It is evident that the computational requirements for RSVD are significantly reduced compared to SVD and the error satisfies [1]:

$$\mathbb{E}\|A - A_r\| \leq \sigma_{r+1} + \left[1 + 4\sqrt{\frac{2n}{r-1}}\right]^{1/(it+1)} \sigma_{r+1}. \quad (4)$$

where σ_i represents the i-th largest singular value of A.

Motivation: Why do we need flexible and fast rank selection?

In Low-rank methods, rank selection often relies on empirical insights in specific layers, and the selected rank is a fixed value for all weights. For example, LoRC in ZeroQuantv2 (Yao et al. 2024) fixes the rank as 2^n, RILQ and LoftQ use fixed ranks of 16 or 32, and experiments in LQER suggest that a rank of 64 is sufficient to maintain the accuracy of the OPT-1.3b model, SVD-Quant adopts a fixed rank of 32 while CALDERA applies ranks from 64 to 256.

A fixed rank introduces non-negligible memory and latency costs. Under 2-bit quantization, LQER with rank 256 incurs roughly **50%** additional memory, whereas CALDERA with rank 256, despite achieving higher accuracy, it brings 20% extra memory and increasing inference latency by nearly **30%** on 7B model. The overhead of SVD or fine-tuning–based quantization is also substantial: RILQ demonstrates competitive accuracy at rank 16 yet must be combined with other quantization methods—e.g., RILQ+OmniQuant requires 3.1 h for quantization and about 1 h for fine-tuning on LLaMA2-7B and SVD-based methods severely slow quantization, diminishing usability.

[1]The proof of this theorem is complex; for specifics, one may refer to (Halko, Martinsson, and Tropp 2011).

Method

In quantization, low-rank methods frequently employ a fixed rank approach, which is not ideally suited for all layers. We introduce **FLRQ** (Flexible Low-Rank Quantization), structured into two primary components and presented in Algorithm 2:

- **R1-FLR** (R1-Sketch-based Flexible Rank Selection): This component focuses on dynamically determining the rank for each layer under rank-1 sketch (R1-Sketch), allowing for adaptability according to the specific characteristics and requirements of individual layers.
- **BLC** (Best Low-rank Approximation under Clipping): This segment aims at achieving efficient low-rank approximations when utilizing clipping strategies to minimize quantization errors while preserving critical data features.

In this section, we will first present the R1-Sketch algorithm, followed by the introduction of two optimization methods: **R1-FLR** and **BLC**.

Low rank under R1-Sketch

To address the inefficiency of SVD, we proposed a simplification to the randomized SVD (RSVD) algorithm under the rank-1 condition. This simplification introduces a rank-1 matrix approximation technique, as described in below:

Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$. For a standard RSVD (Randomized Singular Value Decomposition) algorithm prototype, it typically consists of the following two steps:

Stage A:

1. Generate an $\mathbb{R}^{n\times r}$ Gaussian test matrix $\boldsymbol{S}$.
2. Form $\boldsymbol{Y} = (\boldsymbol{A}\boldsymbol{A}^*)^{it}\boldsymbol{A}\boldsymbol{S}$.
3. Construct a matrix $\boldsymbol{Q} = QR(\boldsymbol{Y})$ by QR decomposition whose columns form an orthonormal basis of $\boldsymbol{Y}$.

Stage B:

1. Form $\boldsymbol{B} = \boldsymbol{Q}^*\boldsymbol{A}$.
2. Compute an SVD of the small matrix: $\boldsymbol{B} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$.
3. Set $\boldsymbol{U} = \boldsymbol{Q}\boldsymbol{U}$.

If a rank-1 matrix $\boldsymbol{S} \in \mathbb{R}^{n\times 1}$ is utilized for low-rank approximation of a matrix, substitute the rank-1 matrix in the two stages, we also have $Y = (\boldsymbol{A}\boldsymbol{A}^*)^{it}\boldsymbol{A}S$.

then for the matrix $\boldsymbol{Y} \in \mathbb{R}^{m\times 1}$, the QR decomposition can be directly represented as follows.

$$\boldsymbol{Q} = \frac{\boldsymbol{Y}}{\|\boldsymbol{Y}\|} \in \mathbb{R}^{m\times 1}, \boldsymbol{R} = \|\boldsymbol{Y}\| \in \mathbb{R}^{1\times 1}. \quad (5)$$

Similarly, the SVD decomposition for rank-1 matrix $\boldsymbol{B} = \boldsymbol{Q}^*\boldsymbol{A}$ can be represented as follows:

$$\boldsymbol{U} = \{1\}, \boldsymbol{\Sigma} = \|\boldsymbol{B}\|, \boldsymbol{V} = \frac{\boldsymbol{B}}{\|\boldsymbol{B}\|}. \quad (6)$$

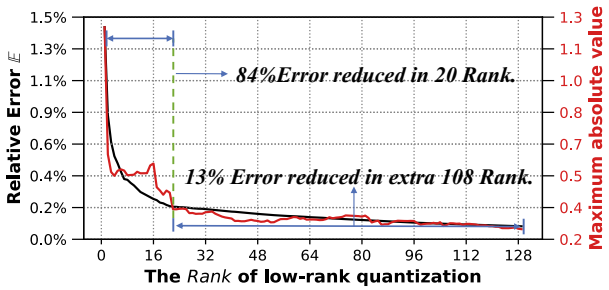

Figure 2: As the extraction rank increases, the curves of the relative error $\mathbb{E}$ under L_2 norm and the $amax$ value decrease.

Denote $A_L = \boldsymbol{QU\Sigma}$ and $A_R = \boldsymbol{V}$. Apply the results of SVD decomposition, we have:

$$A_L = \frac{(\boldsymbol{AA}^*)^{it}\boldsymbol{AS}}{\|(\boldsymbol{AA}^*)^{it}\boldsymbol{AS}\|} \cdot \frac{\|\boldsymbol{S}^*\boldsymbol{A}^*(\boldsymbol{AA}^*)^{it}\boldsymbol{A}\|}{\|(\boldsymbol{AA}^*)^{it}\boldsymbol{AS}\|},$$
$$A_R = \frac{\boldsymbol{B}}{\|\boldsymbol{B}\|} = \frac{\boldsymbol{S}^*\boldsymbol{A}^*(\boldsymbol{AA}^*)^{it}\boldsymbol{A}}{\|\boldsymbol{S}^*\boldsymbol{A}^*(\boldsymbol{AA}^*)^{it}\boldsymbol{A}\|}. \quad (7)$$

R1-Sketch-based Flexible Low-Rank Selection

Given a layer $\mathbf{W} \in \mathbb{R}^{m*n}$ in d_{fp} and perform a pseudo-quantization with r rank $\mathbf{W_r}$ under d bit.

$$\widehat{\boldsymbol{W}_q} = clamp(\lfloor \boldsymbol{R}/s_r \rceil) * s_r;\ s_r = \frac{2^{d-1}\text{-}1}{amax(\boldsymbol{R})}, \quad (8)$$

where $\boldsymbol{R} = \boldsymbol{W}\text{-}\boldsymbol{W}_r$. Then $\widehat{\boldsymbol{W}}$ can be decomposed into $\widehat{\boldsymbol{W}} = \widehat{\boldsymbol{W}_q} + \boldsymbol{W}_r$. The low-rank component is stored in original precision which is nearly lossless.

As a example illustrated in Figure 2, the quantization error $\mathbb{E} = \|\boldsymbol{WX} - \widehat{\boldsymbol{W}}\boldsymbol{X}\|/\|\boldsymbol{WX}\|$ gradually decreases as the rank increases. However, repeated recalculation of $\mathbb{E}$ after layer re-quantization following rank adjustments incurs significant computational costs. In order to improve quantization efficiency, we propose a flexible rank selection approach based on $amax$ aimed at minimizing quantization errors while optimizing computational efficiency.

Assume that the corresponding *amax* for this rank is w_r. Then the maximum quantization error under rank r quantization is $E_r = \frac{1}{2s_r}$. Thus, the error reduced by $p = \frac{E_0}{E_r} = \frac{w_0}{w_r}$, the precision improvement corresponding to an additional $d' = \log_2(p)$ bits. Consequently, the model bit-width increases to $d + d'$, and the effective quantization bits increase by q in Eq.9; For memory costs, after incorporating rank-r matrix, the model size increased by k in Eq.9.

$$q = \frac{d + d'}{d}, k = 1 + \frac{d_{fp} \times r \times (m + n)}{d \times m \times n}. \quad (9)$$

Then, **R1-FLR** workflow is as follows, and shown in Algorithm 1:

Starting the traversal of r from 1, applying the R1-Sketch method for computing the low-rank matrix corresponding to the current rank-1 matrix, then utilizing Equation 9 to calculate the parameters q and k during each iteration.

Algorithm 1: R1-Sketch Flexible Low-Rank Selection (R1-FLR)

Data: $W \in \mathbb{R}^{m*n}$(weight), X(value from calibration),)
Result: W_r, W_q
Begin **R1-FLR** processes: ;
for $i = 1$ ***to*** n **do**
  Obtain the sketch rank-1 matrix by Eq.7:
  $\{\boldsymbol{U}_1, \boldsymbol{V}_1\} \leftarrow calR1matrix(\boldsymbol{R})$;
  $\boldsymbol{R} \leftarrow \boldsymbol{R} - \boldsymbol{U}_1 \cdot \boldsymbol{V}_1$;
  get amax now: $maxAbs \leftarrow amax(\boldsymbol{R})$;
  Calculate Q, K by Eq.9 and the slope:
  $sNow \leftarrow getSlope(maxAbs)$;
  if $K > Q$ *or* $K > 1 + x$ *or* $sNow < t$ **then**
    Endloop;
  end
  $\boldsymbol{W}_L$.append(U_1); $\boldsymbol{W}_R$.append(V_1);
end
return $\boldsymbol{W}_L, \boldsymbol{W}_R$;

1. If $q > k$, it demonstrates that the precision improvement achieved exceeds the increase in model size. Thus, the r-rank quantization is effective.
2. If $q \leq k$, it indicates that the precision improvement achieved is not larger than the increase in model size due to the extraction of low-rank components.

Best Low-rank Approximation under Clipping

Low-rank quantization with calibration: Recent studies have shown that the data distribution of activation can influence the effectiveness of quantization Lin et al. (2024); Zhang et al. (2024). Scaling the weights according to the activation values can significantly reduce quantization errors. Specifically, this involves using a calibration dataset to perform inference on each layer during quantization to obtain the current layer's activation value distribution, then calculating a scale vector that is applied before low-rank approximation, which can perform as:

$$\{\boldsymbol{U}', \boldsymbol{V}\} = \text{R1-FLR}(\alpha \boldsymbol{W}),\ \boldsymbol{U} = \alpha^{-1}\boldsymbol{U}'. \quad (10)$$

The calculation of α is:

$$\alpha = \overline{\boldsymbol{X}}^{2.5} / \sqrt{max(\overline{\boldsymbol{X}}) * min(\overline{\boldsymbol{X}})}, \quad (11)$$

where $\boldsymbol{X}$ is the corresponding activation value of $\boldsymbol{W}$ and $\overline{\boldsymbol{X}}$ is per-token normalized mean of $\boldsymbol{X}$. The calculation here is similar to that in AWQ (Lin et al. 2024). In addition, during quantization, setting a portion of the numbers with the largest absolute values to zero by clipping can improve quantization accuracy.

Get best low-rank quantization by iteration: With the aforementioned strategies, the algorithm flow is as follows:

1. Under calibration, use R1-FLR to compute the low-rank matrix with scaling to obtain $\boldsymbol{W}_r$.
2. Apply clipping to find a p_{clp} and cut off the elements whose absolute values exceed p_{clp}, where $\boldsymbol{W}_{clp} = Clipping(\boldsymbol{W} - \boldsymbol{W}_r, p_{clp})$.

Precision	Method	OPT-1.3b		OPT-6.7b		OPT-13b		LLaMA2-7b		LLaMA2-13b	
		Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4
FP16	Baseline	14.62	14.72	10.86	11.74	10.13	11.19	5.47	6.97	4.88	6.47
W4A16	RTN	31.96	21.45	12.05	13.37	11.41	12.41	5.88	7.30	5.12	6.60
	AWQ	15.22	15.04	10.93	11.87	10.21	11.28	5.61	7.13	4.97	6.56
	OmniQuant	14.88	15.03	10.96	11.85	10.20	11.29	5.58	7.12	4.95	6.56
	AffineQuant	14.79	14.98	10.92	11.84	10.19	11.27	5.58	7.12	4.95	6.56
	FLRQ	**14.65**↓	**14.97**↓	**10.84**↓	**11.84**↓	**10.13**↓	**11.28**↓	**5.55**↓	**7.06**↓	**4.94**↓	**6.52**↓
W3A16	RTN	119.1	126.47	23.54	32.56	46.03	44.12	6.66	8.4	5.51	7.18
	AWQ	16.32	16.27	11.41	12.30	10.68	11.61	6.24	7.84	5.32	6.94
	OmniQuant	15.72	16.11	11.27	12.31	10.47	11.63	6.03	7.75	5.28	6.98
	AffineQuant	15.61	**16.02**	11.18	**12.21**	10.51	11.63	6.08	7.83	5.28	6.99
	FLRQ	**15.53**↓	16.07	**11.18**↓	12.37	**10.52**↓	**11.68**↓	**5.88**↓	**7.45**↓	**5.16**↓	**6.77** ↓
W2A16	RTN	1.3e4	7.7e3	7.8e3	5.2e3	7.6e4	2.8e4	4.2e3	4.9e3	122.2	139.6
	AWQ	47.97	38.40	16.2	16.48	14.32	14.73	2.2e5	1.7e5	1.2e5	9.4e4
	OmniQuant	23.95	27.33	14.43	16.67	12.94	14.92	11.06	15.02	8.26	11.05
	AffineQuant	23.32	23.28	14.18	15.62	12.88	14.60	10.87	13.13	7.64	10.32
	FLRQ	**22.99**↓	**22.52**↓	**14.05**↓	**15.23**↓	**12.60**↓	**13.81**↓	**9.14**↓	**12.10**↓	**6.77**↓	**8.87**↓

Table 2: WikiText2 and C4 perplexity (PPL ↓) results on OPT and LLaMA-2 models, context length is 2048.

Algorithm 2: Flexible Low-Rank Matrix Sketching Quantization

Data: $W \in \mathbb{R}^{m*n}$(weight), X(value from calibration)
Result: W_r, W_q
Init quantization: $\boldsymbol{W}_r = SVD(\boldsymbol{W})$,
$\boldsymbol{W}_q = Quant(\boldsymbol{W} - \boldsymbol{W}_r), bestE \leftarrow inf$;
Start **BLC** processes;
for $i = 1$ ***to*** $epochs$ **do**
  $\mathbb{E} \leftarrow ||\boldsymbol{W}\boldsymbol{X} - (\boldsymbol{W}_r + \boldsymbol{W}_q)\boldsymbol{X}||_2$;
  if $\mathbb{E} < minE$ **then**
    $minE \leftarrow \mathbb{E}$;
    $\{bestW_r, bestW_q\} \leftarrow \{\boldsymbol{W}_r, \boldsymbol{W}_q\}$;
  end
  Calculate the resident matrix: $\boldsymbol{R} \leftarrow \boldsymbol{W} - \boldsymbol{W}_q$;
  $\boldsymbol{W}_r = \{\boldsymbol{W}_L, \boldsymbol{W}_R\} \leftarrow R1\text{-}FLR(\boldsymbol{R})$;
  Apply weight clipping to rest of $\boldsymbol{W}$:
  $\boldsymbol{W}_{clp} \leftarrow Clipping(\boldsymbol{W} - \boldsymbol{W}_r, p_{clp})$;
  Quantize the W_{clip}: $\boldsymbol{W}_q = Quant(\boldsymbol{W}_{clp})$;
end
return $best\boldsymbol{W}_r, best\boldsymbol{W}_q$;

3. Quantize the clipped matrix $\boldsymbol{W}_q = Quant(\boldsymbol{W}_{clp})$.

In this context, is such a decomposition optimal? Considering this from error under L_2 norm, we aim to find a d bit quantized matrix $\boldsymbol{W}_q$ and a rank r matrix $\boldsymbol{W}_r$ that minimize

$$\min_{r,p_{clp}} \left[\mathbb{E}||\boldsymbol{W}\boldsymbol{X} - (\boldsymbol{W}_r + \boldsymbol{W}_q)\boldsymbol{X}||_2\right]. \tag{12}$$

Since the original problem is non-trivial and difficult to solve directly, we propose an iterative method **BLC** to progressively compress the residual space between the low-rank quantization and the original weights, thereby obtaining a minimized error. The core strategy of **BLC** involves the alternating update of $\boldsymbol{W}_r$ and p_{clp}. Firstly, we apply **R1-FLR** and clipping to obtain $\boldsymbol{W}_r$ and $\boldsymbol{W}_q$. Then loop through the following three operations below to find the $\boldsymbol{W}_q$ and $\boldsymbol{W}_r$ corresponding to the minimum error $\mathbb{E}$:

1. Calculate $\mathbb{E} = ||\boldsymbol{W}\boldsymbol{X} - (\boldsymbol{W}_r + \boldsymbol{W}_q)\boldsymbol{X}||_2$.
2. Calculate the quantized residual matrix $\mathbf{R} = \boldsymbol{W} - \boldsymbol{W}_q$ and obtain $\boldsymbol{W}_r = \textbf{R1-FLR}(\mathbf{R})$.
3. Apply **clipping** to find p'_{clp} for clipping and $\boldsymbol{W}_q = Quant(Clipping(\boldsymbol{W} - \boldsymbol{W}_r, p'_{clp}))$.
4. Update the $\boldsymbol{W}_q$, $\boldsymbol{W}_r$ corresponding to the minimum $\mathbb{E}$.

Now we present FLRQ algorithm, which incorporates the BLC iteration and R1-FLR flexible rank selection and the pseudo-code is shown in Algorithm 2.

Experiments

SetUp: We performed a series of evaluations on FLRQ. During quantization, the hyperparameter it at **2** in main evaluations and the group size was fixed at **128**, aligning with the settings in AWQ quantization. We employ a calibration dataset consisting of 128 randomly selected 2048 token segments from WikiText2 (Merity et al. 2016), which proved to be a good sampling strategy in OminiQuant (Shao et al. 2023). All experiments were conducted on an Nvidia A100 40G GPU. Our evaluations included perplexity and zero-shot tests in the OPT (Zhang et al. 2022), LLaMA-2 and LLaMA-3 (Touvron et al. 2023) model families for language generation tasks.

Baselines: We used AWQ (Lin et al. 2024), LQER (Zhang et al. 2024), OmniQuant (Shao et al. 2023) and Affinequant (Ma et al. 2024) as the primary baselines for comparison.

Evaluation Tasks: For the evaluation tasks, we conducted perplexity experiments on the WikiText2 (Merity et al. 2016)

Bit	OPT			LLaMA2	
	1.3B	6.7B	13B	7B	13B
4	30.5/0.34	27.1/0.16	27.0/0.12	36.1/0.21	38.6/0.18
3	28.8/0.33	27.7/0.16	26.4/0.12	35.8/0.21	38.4/0.18
2	27.6/0.33	32.7/0.19	33.6/0.15	39.2/0.24	41.9/0.20

Table 3: The extracted rank and extra average bit width of FLRQ at different x values as (rank/extra-avg.bit).

Bit	Method	extra.bit↓	avg.rank↓	Wiki2↓	C4↓
3	LQER	0.21	32	6.23	8.82
	FLRQ	0.23	36	**5.88**	**7.45**
2	LQER	1.60	256	10.33	12.12
	FLRQ	0.24	39	**9.14**	**12.10**

Table 4: Compare with LQER on LLaMA2-7b.

and C4 (Raffel et al. 2020) datasets. We performed zero-shot experiments using test sets including ARC(challenge,easy) (Boratko et al. 2018), BOOLQ (Clark et al. 2019), OpenBookQA (Mihaylov et al. 2018), PIQA (Bisk et al. 2020), and Winogrande (Sakaguchi et al. 2021), with the lm-evaluation-harness (Gao et al. 2024) framework testing.

Language Generation

We tested FLRQ on model perplexity at 4,3,2-bit. The results are presented in Table 2 and the memory costs are shown in Table 3. According to the results, FLRQ outperforms four PTQ methods on most tasks. Then we compared FLRQ with LQER and the results are presented in Table 4. In LQER's 2-bit quantization, a significantly high fixed rank of 256 is required to maintain accuracy, whereas in FLRQ, the average rank at 2-bit is only around 40, achieving better perplexity. This is attributed to our flexible rank selection and BLC method, which reduces the residual space thereby making our approach superior to non-iterative methods with fixed ranks as shown in Figure 1.(2).

Recently, low-rank quantized fine-tuning algorithms have received widespread attention. We compare our approach with two fine-tuning methods, CALDERA and RILQ, both of which achieve their optimal implementations based on Quip#. However, CALDERA, despite achieving the highest accuracy, incurs significant latency during inference due to its selection of a larger rank (256-rank in int4). Furthermore, low-rank methods generally underperform compared to rotation-based approaches such as Quip#; nevertheless, with the incorporation of fine-tuning, they can achieve comparable accuracy, demonstrating the robustness of the FLRQ algorithm.

Downstream task accuracy

We reused the quantization configuration in language generation and performed a zero-shot evaluation on six downstream tasks, including ARC (easy), ARC (challenge), PIQA, OpenBookQA, BOOLQ, and Winogrande, and the results are presented in Table 6. FLRQ's average accuracy across the

Method	avg. rank	extra. bit	GEN Tasks ↓ Wiki2	C4	low-rank latency↓
Quip#	-	-	12.74	16.84	-
FLRQ	40	0.24	14.12	17.81	4.8%
Quip&CALD	256	0.4	8.87	12.02	26.7%
Quip&RILQ	64	0.4	9.64	12.96	7.1%
FLRQ&RILQ	56	0.36	9.78	13.03	6.5%

Table 5: 2-bit PPL and inference latency on LLaMa3-8B, where CALD is short of CALDERA. QUIP# and FLRQ are PTQ quantization methods, while RILQ and CALDERA are low-rank fine-tuning methods.

Method	OPT			Llama2	
	1.3b	6.7b	13b	7b	13b
FP16	48.8%	55.2%	55.6%	62.7%	63.4%
AWQ(4)	47.4%	54.7%	55.7%	62.2%	64.1%
Omni(4)	47.8%	54.9%	55.6%	62.5%	65.0%
FLRQ(4)	**48.4%**↑	**55.0%**↑	**55.1%**↑	**62.7%**↑	**65.4%**↑
AWQ(3)	46.6%	53.2%	53.5%	60.9%	62.7%
Omni(3)	47.4%	53.7%	54.8%	61.1%	63.4%
FLRQ(3)	**47.6%**↑	**54.4%**↑	**55.3%**↑	**61.4%**↑	**64.2%**↑
Omni(2)	41.9%	46.9%	47.3%	50.1%	53.9%
FLRQ(2)	**45.5%**↑	**51.5%**↑	**52.3%**↑	**56.4%**↑	**60.4%**↑

Table 6: Zero-Shot results (↑) in an average across six zero-shot tasks. Omni is in short of Omniquant and the quantization bit is given in parentheses.

six downstream tasks is consistent with the FP16 baseline, showing near-lossless accuracy in 4bit and 3bit quantization, and maintains considerable accuracy at 2 bits.

The hyperparameters in FLRQ

The choice of iterations it: The accuracy of r1-sketch is determined by the number of iterations it. We evaluated the convergence and execution efficiency under varying values of it and the effectiveness of parameter selection it in the FLRQ method. The evaluation concludes with PPL and execution time in Table 7. The findings indicate that while a higher it improves the accuracy of R1-Sketch, setting $it = 2$ in FLRQ is enough. In this case, R1-Sketch only takes **6** $GEMV$ of $O(N^2)$ and some $O(N)$ routines, providing a balance between accuracy and computational efficiency.

Time efficiency

We evaluated the efficiency of FLRQ from two perspectives: **Quantization Efficiency:** We tested the quantization speed of FLRQ against other PTQ methods, with the results presented in Table 8. In the 4-bit and 3-bit quantization, where accuracy drop is less, we selected GPTQ and LQER algorithms for comparison due to faster quantization speed. However, in the 2-bit scenario, AWQ and LQER cannot maintain accuracy; therefore, we chose the higher-precision PTQ methods OmniQuant and AffineQuant for evaluation. The result

it	OPT-1.3B		OPT-6.7B	
	PPL	Time	PPL	Time
0	16.84	6.1m/1.2s	11.65	26.5m/3.1s
1	15.56	6.1m/1.8s	11.42	26.5m/5.2s
2	**15.53**	**6.1m/2.1s**	**11.18**	**26.6m/9.3s**
4	15.53	6.1m/3.5s	11.18	26.7m/16.9s
8	15.53	6.2m/6.8s	11.18	27.0m/29.5s
SVD	15.53	8.7m/2.6m	11.18	56.4m/20.3m

Table 7: The PPL and time costs (**FLRQ** total time)/(R1-FLR partial time) of FLRQ with different it parameters on 3-bit quantized OPT models under wikitext2 dataset.

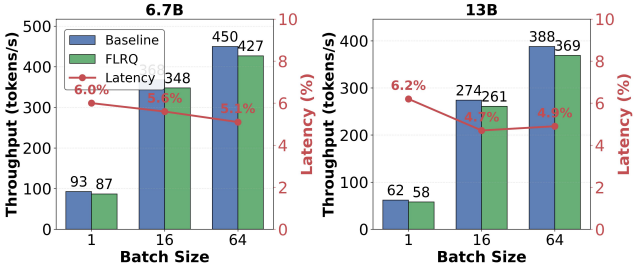

Figure 3: Comparison of Throughput and Latency between Baseline (W4A16) and FLRQ (W4A16+Lora).

shows that, in 3 and 4-bit, FLRQ demonstrated comparable quantization times to methods like AWQ, while being over **30%** faster than LQER, which employs a SVD. Particularly noteworthy is the performance of FLRQ in the challenging 2-bit quantization scenario, where FLRQ exhibited at least a **30%** faster quantization speed compared to OmniQuant, and more than **5** times faster than AffineQuant.

Inference Efficiency: We implemented an efficient fusion kernel for low-rank quantization to the AutoGPTQ quantization framework, with the inference results presented in Figure 3. Due to the adoption of the flexible rank selection strategy, FLRQ introduces only a **4%** to **6%** in inference latency.

Ablation studies

To validate the effectiveness of each strategy within FLRQ, we conducted ablation studies divided into two parts:

- Ablation on **R1-FLR**: To demonstrate the effectiveness of **R1-FLR**, we compared it with fixed-rank methods without applying calibration. As shown in Table 9, FLRQ achieves similar or better PPL to fixed-rank methods, yet with a reduced average bit width, indicating higher compression efficiency. Specifically, in the LLaMA2-13B model, FLRQ achieved the same PPL as a fixed rank of 64 while reducing **40%** extra memory costs.
- Ablation on **BLC**: We compare the performance with and without **BLC**. The results are presented in Table 10. And the findings indicate that the use of **BLC** further reduces the residual space in low-rank quantization, thereby enhancing the accuracy of low-rank quantization at every bit level. Specifically, in the 2-bit scenario, the **BLC** technique notably avoids quantization distortion, demonstrating superior performance.

Bit	Method	OPT			LLaMA2	
		1.3b	6.7b	13b	7b	13b
3.4	AWQ	6.7m	28.3m	52.8m	25.2m	40.5m
	LQER	8.2m	35.3m	1.1h	45.2m	1.2h
	FLRQ	6.2m	26.6m	51.3m	23.0m	40.3m
2	Omni	1.2h	3.9h	6.8h	3.1h	5.3h
	Affine	3.2h	14.4h	25.3h	12.2h	23.1h
	FLRQ	33.2m	2.5h	4.9h	2.0h	3.8h

Table 8: The quantization time costs on a single A100 GPU, where 'm' stands for minutes and 'h' for hours:

Models	RANK=32		RANK=64		FLRQ (NO BLC)	
	avg. bit	PPL	avg. bit	PPL	avg. rank(bit)	PPL
7B	4.32	5.83	4.52	5.73	36.03(4.31)↓	5.74
13B	4.28	4.99	4.44	4.98	21.9(4.24)↓	4.98

Table 9: 4-bit PPL under fixed rank and FLRQ on Wiki2.

The results indicate that **R1-FLR** is capable of significantly reducing the additional overhead associated with low-rank quantization while maintaining quantization accuracy and **BLC** not only improves accuracy across different bit depths but also significantly solves quantization distortion, especially under more challenging 2-bit conditions.

Bit	BLC	OPT			LLaMA2	
		1.3b	6.7b	13b	7b	13b
4	×	14.58	10.89	10.11	5.55	4.94
	✓	14.55↓	10.84↓	10.13	5.55	4.94
3	×	15.80	11.32	10.54	5.89	5.18
	✓	15.53↓	11.18↓	10.52↓	5.88↓	5.16↓
2	×	29.32	17.23	15.41	2.1e6	1.2e6
	✓	22.99↓	14.05↓	12.60↓	9.14↓	6.77↓

Table 10: PPL(↓) on WikiText-2 dataset. Note that "×" indicates without employing the iterative strategy from **BLC**. Other settings remain consistent with the main experiments.

Conclusion

In this work, we propose FLRQ, a low-rank quantization method that employs flexible rank selection based on R1-Sketch and iteratively minimizes quantization errors. Compared to other low-rank PTQ, FLRQ features lower additional memory consumption and faster quantization speeds. Extensive experiments demonstrate that FLRQ achieves better precision than several other PTQ methods in model quantization, particularly excelling in 2-bit quantization. Furthermore, we demonstrate the robustness of the FLRQ algorithm and achieve low inference latency through efficient kernel fusion.

Acknowledgements

This work is partially supported by the Strategic Priority Research Program of Chinese Academy of Sciences (XDB0500101), and the Basic Research Project of the Institute of Software, Chinese Academy of Sciences (ISCAS-JCMS-202304).

References

Bisk, Y.; Zellers, R.; Le bras, R.; Gao, J.; and Choi, Y. 2020. PIQA: Reasoning about Physical Commonsense in Natural Language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 7432–7439.

Boratko, M.; Padigela, H.; Mikkilineni, D.; Yuvraj, P.; Das, R.; McCallum, A.; Chang, M.; Fokoue-Nkoutche, A.; Kapanipathi, P.; Mattei, N.; et al. 2018. A systematic classification of knowledge, reasoning, and context within the ARC dataset. *arXiv preprint arXiv:1806.00358*.

Clark, C.; Lee, K.; Chang, M.-W.; Kwiatkowski, T.; Collins, M.; and Toutanova, K. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Dass, J.; Wu, S.; Shi, H.; Li, C.; Ye, Z.; Wang, Z.; and Lin, Y. 2023. Vitality: Unifying low-rank and sparse approximation for vision transformer acceleration with a linear taylor attention. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 415–428. IEEE.

Ding, Y.; Qin, H.; Yan, Q.; Chai, Z.; Liu, J.; Wei, X.; and Liu, X. 2022. Towards accurate post-training quantization for vision transformer. In *Proceedings of the 30th ACM international conference on multimedia*, 5380–5388.

Frantar, E.; Ashkboos, S.; Hoefler, T.; and Alistarh, D. 2023. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. In *The Eleventh International Conference on Learning Representations*.

Gao, L.; Tow, J.; Abbasi, B.; Biderman, S.; Black, S.; DiPofi, A.; Foster, C.; Golding, L.; Hsu, J.; Le Noac'h, A.; Li, H.; McDonell, K.; Muennighoff, N.; Ociepa, C.; Phang, J.; Reynolds, L.; Schoelkopf, H.; Skowron, A.; Sutawika, L.; Tang, E.; Thite, A.; Wang, B.; Wang, K.; and Zou, A. 2024. A framework for few-shot language model evaluation.

Guan, X.; Li, C.-T.; and Guan, Y. 2017. Matrix factorization with rating completion: An enhanced SVD model for collaborative filtering recommender systems. *IEEE access*, 5: 27668–27678.

Halko, N.; Martinsson, P.-G.; and Tropp, J. A. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2): 217–288.

Hassibi, B.; Stork, D.; and Wolff, G. 1993. Optimal Brain Surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, 293–299 vol.1.

Hubara, I.; Nahshan, Y.; Hanani, Y.; Banner, R.; and Soudry, D. 2021. Accurate post training quantization with small calibration sets. In *International Conference on Machine Learning*, 4466–4475. PMLR.

Ji, H.; and Li, Y. 2014. GPU accelerated randomized singular value decomposition and its application in image compression. *Proc. of MSVESCC*, 39–45.

Kumar, B. 2016. A novel latent factor model for recommender system. *JISTEM-Journal of Information Systems and Technology Management*, 13(3): 497–514.

Kuzmin, A.; Nagel, M.; Van Baalen, M.; Behboodi, A.; and Blankevoort, T. 2023. Pruning vs quantization: Which is better? *Advances in neural information processing systems*, 36: 62414–62427.

Lee, G.; Lee, J.; Hong, S.; Kim, M.; Ahn, E.; Chang, D.-S.; and Choi, J. 2025. RILQ: Rank-Insensitive LoRA-based Quantization Error Compensation for Boosting 2-bit Large Language Model Accuracy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 18091–18100.

Li, M.; Lin, Y.; Zhang, Z.; Cai, T.; Li, X.; Guo, J.; Xie, E.; Meng, C.; Zhu, J.-Y.; and Han, S. 2024a. Svdqunat: Absorbing outliers by low-rank components for 4-bit diffusion models. *arXiv preprint arXiv:2411.05007*.

Li, Y.; Yu, Y.; Liang, C.; Karampatziakis, N.; He, P.; Chen, W.; and Zhao, T. 2024b. LoftQ: LoRA-Fine-Tuning-aware Quantization for Large Language Models. In *The Twelfth International Conference on Learning Representations*.

Lin, J.; Tang, J.; Tang, H.; Yang, S.; Chen, W.-M.; Wang, W.-C.; Xiao, G.; Dang, X.; Gan, C.; and Han, S. 2024. AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. *Proceedings of Machine Learning and Systems*, 6: 87–100.

Liu, Z.; Oguz, B.; Zhao, C.; Chang, E.; Stock, P.; Mehdad, Y.; Shi, Y.; Krishnamoorthi, R.; and Chandra, V. 2024. LLM-QAT: Data-Free Quantization Aware Training for Large Language Models. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Findings of the Association for Computational Linguistics: ACL 2024*, 467–484. Bangkok, Thailand: Association for Computational Linguistics.

Ma, Y.; Li, H.; Zheng, X.; Ling, F.; Xiao, X.; Wang, R.; Wen, S.; Chao, F.; and Ji, R. 2024. AffineQuant: Affine Transformation Quantization for Large Language Models. In *The Twelfth International Conference on Learning Representations*.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Mihaylov, T.; Clark, P.; Khot, T.; and Sabharwal, A. 2018. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Osawa, K.; Sekiya, A.; Naganuma, H.; and Yokota, R. 2017. Accelerating matrix multiplication in deep learning by using low-rank approximation. In *2017 International Conference on High Performance Computing & Simulation (HPCS)*, 186–192. IEEE.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67.

Saha, R.; Sagan, N.; Srivastava, V.; Goldsmith, A.; and Pilanci, M. 2024. Compressing large language models using low rank and low precision decomposition. *Advances in Neural Information Processing Systems*, 37: 88981–89018.

Sakaguchi, K.; Bras, R. L.; Bhagavatula, C.; and Choi, Y. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9): 99–106.

Shao, W.; Chen, M.; Zhang, Z.; Xu, P.; Zhao, L.; Li, Z.; Zhang, K.; Gao, P.; Qiao, Y.; and Luo, P. 2023. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Tseng, A.; Chee, J.; Sun, Q.; Kuleshov, V.; and De Sa, C. 2024. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *arXiv preprint arXiv:2402.04396*.

Yao, Z.; Wu, X.; Li, C.; Youn, S.; and He, Y. 2024. Exploring post-training quantization in llms from comprehensive study to low rank compensation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 19377–19385.

Zhang, C.; Cheng, J.; Constantinides, G. A.; and Zhao, Y. 2024. LQER: Low-Rank Quantization Error Reconstruction for LLMs. *arXiv preprint arXiv:2402.02446*.

Zhang, S.; Roller, S.; Goyal, N.; Artetxe, M.; Chen, M.; Chen, S.; Dewan, C.; Diab, M.; Li, X.; Lin, X. V.; et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.