PTQ*1.61*: Push the Real Limit of Extremely Low-Bit Post-Training Quantization Methods for Large Language Models

Jiaqi Zhao[1], Miao Zhang[1*], Ming Wang[1], Yuzhang Shang[2], Kaihao Zhang[1], Weili Guan[1], Yaowei Wang[1], Min Zhang[1]

[1] Harbin Institute of Technology (Shenzhen)
[2] Illinois Institute of Technology

jiaqizhao0455@outlook.com, 190110509@stu.hit.edu.cn
{zhangmiao, guanweili, wangyaowei, zhangmin2021}@hit.edu.cn
yshang4@hawk.iit.edu, super.khzhang@gmail.com

Abstract

Large Language Models (LLMs) suffer severe performance degradation when facing extremely low-bit (sub 2-bit) quantization. Several existing sub 2-bit post-training quantization (PTQ) methods utilize a mix-precision scheme by leveraging an unstructured fine-grained mask to explicitly distinguish salient weights, while which introduces an extra 1-bit or more per weight. To explore the real limit of PTQ, we propose an extremely low-bit PTQ method called PTQ1.61, which enables weight quantization to 1.61-bit for the first time. Specifically, we first introduce a one-dimensional structured mask with negligibly additional 0.0002-bit per weight based on input activations from the perspective of reducing the upper bound of quantization error to allocate corresponding salient weight channels to 4-bit. For non-salient channels binarization, an efficient block-wise scaling factors optimization framework is then presented to take implicit row-wise correlations and angular biases into account. Different from prior works that concentrate on adjusting quantization methodologies, we further propose a novel paradigm called quantization preprocessing, where we argue that transforming the weight distribution of the pretrained model before quantization can alleviate the difficulty in per-channel extremely low-bit PTQ. Extensive experiments indicate our PTQ1.61 achieves state-of-the-art performance in extremely low-bit quantization. Codes are available at https://github.com/zjq0455/PTQ1.61.

1 Introduction

Large Language Models (LLMs) such as LLaMA (Touvron et al., 2023a,b) and GPT (Brown et al., 2020; Ouyang et al., 2022) have demonstrated remarkable success in various natural language processing tasks. However, their colossal numbers of parameters bring tremendous storage and inference
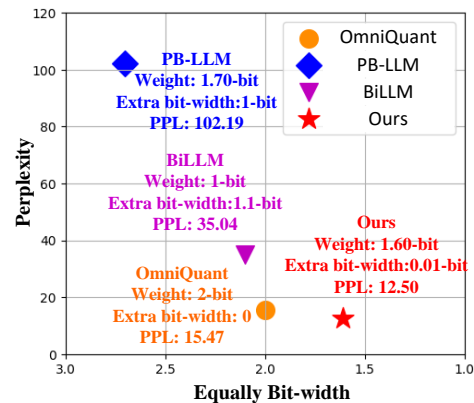

Figure 1: An overview of performance (Perplexity on WikiText2) and bit-width achieved by our **PTQ*1.61*** and other extremely low-bit PTQ methods on LLaMA-7B.

overheads. To alleviate the challenge, numerous model compression methods have been proposed such as quantization (Liu et al., 2022; Huang et al., 2019), pruning (Frantar and Alistarh, 2023; Ma et al., 2023) and knowledge-distillation (Gou et al., 2021; Tunstall et al., 2023). Among these works, post-training quantization (PTQ) methods (Yuan et al., 2023; Wei et al., 2022) have garnered particular attention for LLMs due to their computational efficiency compared to quantization-aware training (QAT) (Liu et al., 2023b; Esser et al., 2019) and other compression methods. Although maintaining nearly lossless performance at 4-bit or 8-bit, most existing state-of-the-art PTQ approaches fail when attempting to quantize weights to extremely low bit-width, *i.e.*, 1-bit or sub 2-bit.

PB-LLM (Shang et al., 2023) and BiLLM (Huang et al., 2024) are two most recent sub 2-bit PTQ methods for LLMs. They selectively preserve a portion of salient weights at 8-bit or with fine processing while quantizing the remaining weights to 1-bit. Although they demonstrate promising results, they are plagued by two critical issues. ***Firstly and the most importantly***, both methods introduce additional unstructured fine-grained masks to dis-

---

tinguish salient weights which requires additional 1-bit per weight to store the mask and leads the memory of the quantized model to exceeding 2-bit per weight, where PB-LLM with 2.7-bit and BiLLM with 2.1-bit respectively (see Appendix A). ***Secondly***, they independently and analytically derive the row-wise scaling factors used for mitigating binarization magnitude errors (Rastegari et al., 2016), violating the fact that weights exhibit implicit row-wise dependencies (Clark et al., 2019; Vig and Belinkov, 2019) and angular biases (Lin et al., 2020).

Motivated by issues above and to push the real limit of PTQ methods on extremely low-bit quantization, we propose an extremely low-bit (**1.61-bit**) PTQ method for LLMs called **PTQ*1.61***. Specifically, to eliminate the significant additional memory consumption caused by unstructured fine-grained masks, we dissect the quantization error through mathematical derivation to identify the structural influencing factors within it, and find that the upper bound of quantization error is significantly affected by input activation channels. Based on this discovery we propose a *one-dimensional structured mask* to preserve corresponding salient channels in the weight matrix at 4-bit, and successfully reduce the extra bit-width for each weight from over 1-bit to a negligible extent (0.0002-bit). Additionally, in order to capture the implicit row-wise correlations and directional biases jointly, we introduce a novel *efficient block-wise scaling factors optimization framework*.

In addition, unlike previous studies which always take the pretrained model with the best performance as the starting point for quantization, we find that the weights distribution also immensely affects the quantization performance. Specifically, existing per-channel PTQ methods usually consider a row-wise quantization pattern that assigns the same quantization parameter to all weights in a channel, while the distribution of salient weights in the pretrained model is scattered, which leads to significant quantization errors. Motivated by this row-wise nature, we propose a novel *preprocessing strategy* for LLMs quantization, which first transforms the weight distribution into a row-wise pattern through a lightweight restorative LoRA alignment, so that the preprocessed model is more suitable for per-channel PTQ than pretrained model. The proposed preprocessing strategy can be also applied to other extremely low-bit PTQ methods with notable performance enhancement, as shown

in Figure 5. We further discuss the differences and advantages of our preprocessing strategy from existing post-quantization parameter-efficient fine-tuning (PEFT) approaches (Dettmers et al., 2023) in Section 3.4 and Appendix D.

With these enhancements, **PTQ*1.61*** effectively quantizes the weights to extremely low-bit with outstanding performance, as illustrated in Figure 1. Our key contributions can be summarized as:

- To explore the real limitation of post-training quantization, we present an efficient extremely low-bit PTQ method for LLMs named **PTQ*1.61*** which is the ***first*** PTQ study to truly reduce the effective bit-width of LLMs weights to sub 2-bit (1.61-bit) with acceptable performance degradation.

- Different from leveraging memory-intolerable unstructured masks to preserve salient information, we propose a one-dimensional structured mask based on input activations to reduce the upper bound of quantization errors, which only introduces negligible 0.0002-bit for each weight.

- We further present a novel efficient block-wise optimization strategy to learn scaling factors to further consider the implicit row-wise dependencies and angular biases.

- We demonstrate that pretrained model is not amenable to per-channel PTQ and accordingly propose a quantization preprocessing paradigm based on restorative LoRA to transform salient weights as a row-wise pattern to further enhance the quantization performance.

2 Related Works

2.1 Post-Training Quantization

Post-training quantization is an efficient and expeditious quantization approach which merely necessitates a limited amount of calibration data to statistically determine the quantization parameters that help to scale float values to low-bit. AdaRound (Nagel et al., 2020) analyzes the quantization errors and employs a layer-wise optimization approach to learn the optimal rounding mechanism. BrecQ (Li et al., 2021) divides the model weights into multiple blocks and independently quantizes each block, allowing for finer control over quantization errors.
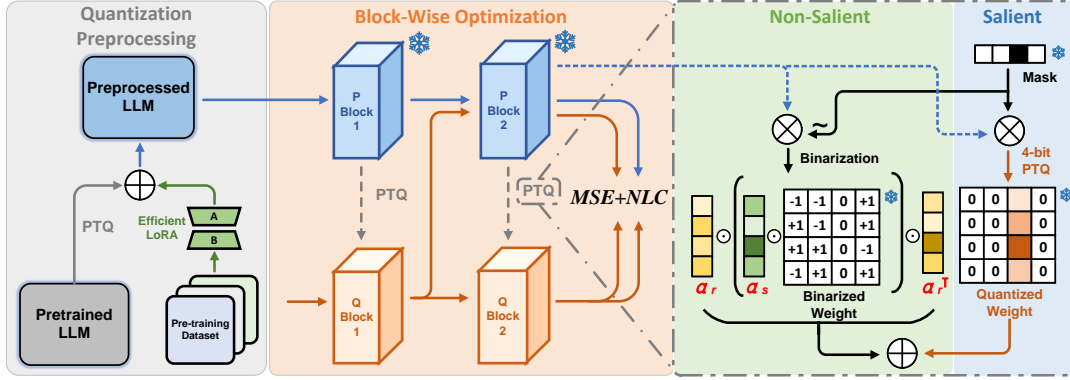
Figure 2: An overview of our **PTQ***1.61*. Utilizing quantization preprocessing, the pretrained model is transformed into a row-wise pattern which is amenable to channel-wise quantization. Initially, structured masks are obtained to distinguish salient weights channels based on channel-wise magnitude of input activation. Salient weight channels undergo 4-bit quantization to retain crucial information, while non-salient weights are binarized with the aid of learnable scaling factors updated by novel block-wise optimization framework.

In recent years, there has been a growing research interest in PTQ methods for LLMs. GPTQ (Frantar et al., 2022) quantizes weights column-wise based on the Hessian matrix and dynamically updates the remaining weights to compensate for quantization errors. AWQ (Lin et al., 2023) retains 1% of salient weights and calculates quantization parameters based on output activations. ZeroQuant (Yao et al., 2022) performs group-wise quantization for weights and finer-grained per-channel quantization for activations. SmoothQuant (Xiao et al., 2023) considers that the cause of quantization errors in activations lies in the presence of channel-wise outliers and proposes smoothing parameters to reduce their magnitude. OmniQuant (Shao et al., 2023) combines the advantages of previous works and learns the optimal smoothing and quantization parameters through back propagation, making it the current state-of-the-art PTQ method for LLMs.

Unfortunately, for extremely low-bit (sub 2-bit) quantization, which offers the highest compression ratio, the performance of such methods generally suffers significantly.

2.2 Extremely Low-Bit Quantization

Extremely low-bit quantization refers to approaches where the effective bit-width for weights is sub 2-bit. It has been widely welcomed due to significant compression ratio but suffers from severe performance degradation. BNN (Courbariaux et al., 2016) is the first model binarization method and XNOR-Net (Rastegari et al., 2016) presents scaling factors which reduce binarization errors with acceptable additional memory cost. RBNN (Lin et al., 2020) indicates that except for magni-

tude gaps, angular biases ought to be considered so that extra rotation matrices are introduced to overcome the drawback.

Several extremely low-bit QAT methods for LLMs (Xu et al., 2024; Wang et al., 2023; Ma et al., 2024) have been proposed recently. Regrettably, the immense computational resource consumption and the lack of open-source availability have hindered their widespread application so that there is a growing demand for more economical PTQ methods. PB-LLM (Shang et al., 2023) investigate the importance of salient weights and design extra 1-bit unstructured masks to retain them into 8-bit while binarizing the others. BiLLM (Huang et al., 2024) further presents finer-grained masks to divide into multi-groups for binarization using different scaling factors. However, the fine-grained masks that cannot be compressed in both methods results in the equivalent bitwidths exceeding 2 bits. To make contributions for truly extremely low-bit PTQ research, we propose **PTQ***1.61* which addresses the issues above and obtains promising performance.

3 PTQ*1.61*

In this section, we provide a detailed introduction to our **PTQ***1.61*, an extremely low-bit PTQ method for LLMs as demonstrated in Figure 2. We begin with briefly reviewing the basic concepts of model quantization and binarization in Section 3.1. Subsequently, to preserve salient information while avoiding insufferable memory overheads brought by unstructured masks in previous methods, we analyze the impact factors on quantization errors and then devise a one-dimensional mask based on

input activations with negligible bit-width in Section 3.2. In Section 3.3, a novel block-wise optimization strategy is introduced for binarization to obtain optimal scaling factors considering implicit dependencies and angular biases. In Section 3.4, we explain why the pretrained model is not suitable for per-channel PTQ and how our proposed quantization preprocessing strategy works.

3.1 Preliminaries

Model Quantization Model quantization aims to convert float weights into corresponding low-bit integer forms thereby reducing computational and memory overheads. The quantization function can be elaborated as:

$$\mathbf{W}_q = \text{clamp}(\lfloor \frac{\mathbf{W}}{S_q} \rceil + Z_q, 0, 2^b - 1), \quad (1)$$

where $\mathbf{W} \in \mathbb{R}^{n \times m}$ and $\mathbf{W}_q \in \mathbb{R}^{n \times m}$ indicate full-precision and quantized weights respectively. $\lfloor \cdot \rceil$ denotes round-to-nearest operator. S_q is the quantization scalar and Z_q represents zero-point.

Model Binarization Binarization represents the most extreme form of quantization (1-bit) where weights are assigned as ± 1 determined by the sign function. In more details:

$$\text{sign}(\mathbf{W}) = \begin{cases} +1, & \mathbf{W} \geq 0, \\ -1, & \mathbf{W} < 0. \end{cases}, \quad \mathbf{W}_b = \alpha \text{sign}(\mathbf{W}), \quad (2)$$

where α denotes scaling factors commonly used in previous methods (Bulat and Tzimiropoulos, 2019; Xu et al., 2021; Liu et al., 2018; Shang et al., 2023; Huang et al., 2024) to reduce binarization errors and $\mathbf{W}_b \in \mathbb{R}^{n \times m}$ is binarized weights with scaling factor α. Assuming that weights in each row of $\mathbf{W}$ are independent, we define w as a row of weights with n_w elements. The corresponding scaling factor α_w can be derived analytically by $\alpha_w = \frac{\|w\|_1}{n_w}$.

3.2 Structured Mask

Following (Shang et al., 2023) and (Huang et al., 2024), we recognize that partially preserving higher-bit salient information is crucial for reducing quantization errors. However, their unstructured masks cannot be compressed, resulting in additional memory overheads due to the scattered nature of salient weights within the weight matrix. Hence, our objective is to identify factors that significantly impact quantization errors while maintaining some degree of regularity.
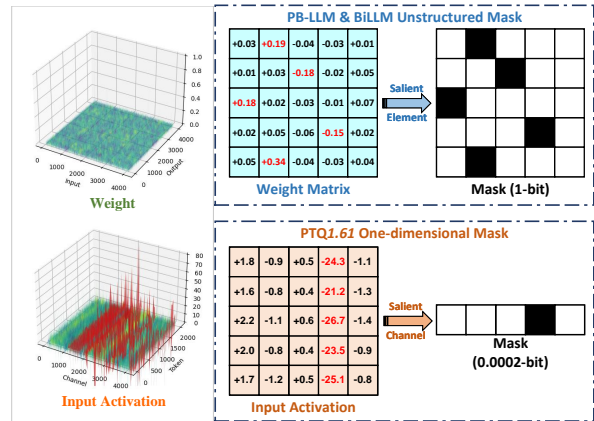

Figure 3: **(a)** The magnitude and distribution comparison between input activations and weights; **(b)** Previous masks are unstructured and uncompressible. In contrast our mask is one-dimensional with only 0.0002-bit.

For each quantized layer, its quantization error is expressed as the gap relative to full-precision output. This can be restated as:

$$\mathcal{E} = |\mathbf{X}\mathbf{W}_q^T - \mathbf{X}\mathbf{W}^T| = |\mathbf{X}(\mathbf{W}_q^T - \mathbf{W}^T)|, \quad (3)$$

where $\mathcal{E}$ is the layer-wise quantization error and $\mathbf{X} \in \mathbb{R}^{t \times m}$ indicates the input activation of the layer. To further explore hidden key factors, we give the visualization of X and W as Figure 3a from which we can observe that in contrast to the chaotic distribution of weights, input activations exhibit a clear channel-wise pattern, where the variance of each token within a channel is very small. Therefore, we re-define X as a column-wise set $\mathbf{X} = \{x_1, x_1, ..., x_m | x_1 \in \mathbb{R}^n\}$ so that Equation (3) will be modified as:

$$\mathcal{E} = |\sum_{i=1}^{m} x_i (w_{i,j}^q - w_{i,j})_{j=1,...,n}| \\ \leq \sum_{i=1}^{m} (|x_i| \sum_{j=1}^{n} |w_{i,j}^q - w_{i,j}|), \quad (4)$$

where $w_{i,j}$ denotes the element in position (i,j) of $\mathbf{W}^T$. As demonstrated in Equation (4), *the upper bound of quantization error is related to the magnitude of input activations for the i-th channel and weight matrix for the i-th row.* Figure 3a highlights that the magnitude of activations is about 1000 times larger than that of weights especially for top-20% channels. Therefore, we propose a one-dimensional mask to save the i-th row of W at 4-bit, maintaining salient information in channel i of input activations to reduce the upper bound

of quantization error as shown in Figure 3b. This improvement successfully reduces the additional bit-width introduced by masks to 0.0002-bit (see Appendix) and limit the equally weight bit-width at sub 2-bit (1.61-bit) for the first time among all existing PTQ methods. The light blue area of Figure 2 illustrates the role of our structured mask in the entire quantization process. Notably, OWQ (Lee et al., 2024) and AWQ (Lin et al., 2023) also take into account the relationship between input activation and weight, but our motivation and performance are entirely different. We have elaborated on this part in Appendix B.

3.3 Block-wise Scaling Factors Optimization

For non-salient weights, we perform binarization following Equation (2). However, previous analytically derived scaling factors ignore implicit correlations among rows and directional shifts which cannot be accurately captured through mathematical derivation. To address this issue, we set scaling factors as learnable and propose a novel efficient block-wise optimization pipeline to learn them while considering implicit row-wise dependencies and angular biases, as demonstrated in the orange and green area of Figure 2.

In order to conduct an effective distance metric for optimization, we first consider MSE loss to reduce magnitude gaps. Then for angular biases described above, we take cosine similarity, a metric considering directional gaps, into account. We formulate the joint metric as follows:

$$\mathbb{E}(f_1, f_2) = \|f_1 - f_2\|_2 + \mathcal{D}_{NLC}(f_1, f_2), \quad (5)$$

where $\mathbb{E}(\cdot)$ is the distance metric for optimization. f_1 and f_2 are different features. $D_{NLC}(\cdot)$ represents the negative logarithm of cosine similarity loss (Zhao et al., 2024), as given by:

$$\mathcal{D}_{NLC}(f_1, f_2) = -\log(\mathcal{C}(f_1, f_2)), \quad (6)$$

where $\mathcal{C}(\cdot)$ denotes cosine similarity. Followed by CBQ (Ding et al., 2023), our block-wise pipeline consists of two branches: the first branch aims at mitigating quantization error propagation and the second branch is tailored for quantifying the outputs distinction for the same inputs. Our final optimization objective is formulated as:

$$\underset{\alpha_s^*, \alpha_r^*}{\arg\min}(\mathbb{E}(\mathcal{F}(X, W), \mathcal{F}(X_q, W_q')) + \\ \mathbb{E}(\mathcal{F}(X_q, W), \mathcal{F}(X_q, W_q'))), \quad (7)$$
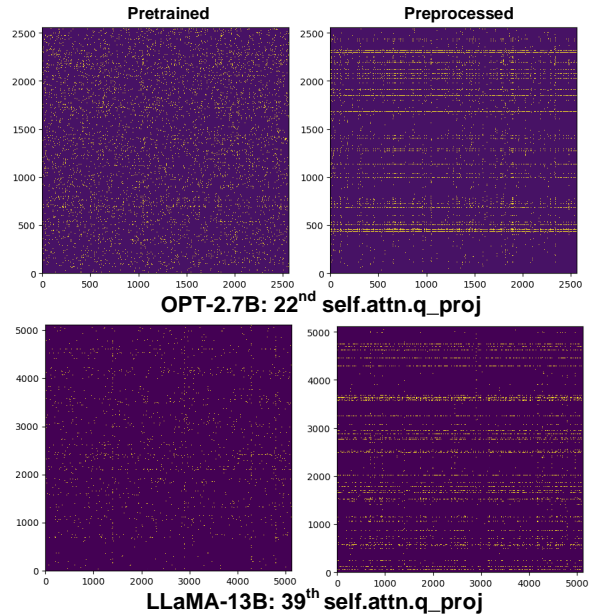

Figure 4: Salient weights distribution of the pretrained and preprocessed OPT-2.7B and LLaMA-13B. The salient weights of the pretrained model exhibit a scattered distribution while after quantization preprocessing a visible row-wise concentrated pattern appears. Each row in the weight matrices corresponds to an input channel. More visualization results please refer to Figure 10.

where α_s and α_r are scaling factors for magnitude and angular biases, respectively. W_q' denotes the quantized weights (see Appendix C.2) and $\mathcal{F}(\cdot)$ represents the embedding function of a block. X indicates the input activation of the full-precision block while X_q is that of the quantized block.

With novel optimization strategy, **PTQ*1.61*** outperforms previous low-bit PTQ methods significantly. The contribution is assessed in Table 3 and Appendix C.

3.4 Quantization Preprocessing

It is well-known that the pretrained models exhibit the best performance so prior PTQ studies intuitively believe that quantizing the pretrained models should also yield optimal results. However, we discover that this notion is somewhat biased. As illustrated in Figure 4, we highlight the salient weights of a linear layer in LLMs based on magnitude-metric (Shang et al., 2023) where a scattered distribution pattern can be observed. The scattered pattern results in significant quantization errors when calculating row-wise scaling factors in per-channel quantization scheme. Therefore, we infer that under the premise of minimizing the bad impact on the pretrained model, transforming

4487

Dataset	Methods	Bits	1-7	1-13	1-30	1-65	2-7	2-13	2-70	3-8
WikiText2	FP	16	5.68	5.09	4.10	3.53	5.47	4.88	3.31	6.14
	AWQ	2	2.5e5	2.8e5	2.4e5	7.4e4	2.2e5	1.2e5	-	8.2e5
	GPTQ	2	2.1e3	5.5e3	1.9e3	55.91	7.7e3	2.1e3	77.95	5.7e4
	QuIP	2	42.19	12.18	9.36	7.19	55.00	13.75	6.96	119.23
	OmniQuant	2	15.47	13.21	8.81	7.58	37.37	17.21	7.81	796.8 4
	PB-LLM	1.7(+1)	102.19	48.11	26.37	12.91	66.30	462.84	NAN	78.67
	BiLLM	1(+1.1)	35.04	15.14	10.52	8.51	32.48	21.77	12.80	50.59
	PTQ*1.61*	**1.61**	**12.50**	**9.67**	**7.95**	**7.02**	**12.70**	**9.74**	**6.94**	**22.90**
C4	FP	16	7.08	6.61	5.98	5.62	6.97	6.46	5.52	8.88
	AWQ	2	1.9e5	2.3e5	2.4e5	7.5e4	1.7e5	9.4e4	-	8.1e5
	GPTQ	2	689.13	2.5e3	234.95	40.58	NAN	323.12	48.82	1.0e5
	OmniQuant	2	24.89	18.31	13.67	10.77	90.64	26.76	12.28	2.4e3
	PB-LLM	1.7(+1)	67.92	34.20	22.45	13.70	66.23	333.54	NAN	78.98
	BiLLM	1(+1.1)	33.64	14.75	**10.97**	9.92	33.72	23.14	13.84	48.26
	PTQ*1.61*	**1.61**	**17.13**	**13.51**	10.98	**9.86**	**17.73**	**13.64**	**12.63**	**33.82**

Table 1: Perplexities comparison of PTQ methods on LLaMA families. For PB-LLM and BiLLM, *1.7(+1)* and *1(+1.1)* under **Bits** means *Weight bits(+Mask bits)*. OPT results can be found in Table 6.

weights with similar saliency in a row-wise distribution pattern can effectively enhance the quantization performance.

As claimed by LoRA (Hu et al., 2021), when fine-tuning a model on specific tasks, the weights compensation exhibits low-rank characteristics, indicating that fine-tuning may compensate for important information into specific dimensions of the weight matrices. Inspired by this proposition, we propose a novel quantization preprocessing paradigm. Specifically, we hope to perform a lightweight restorative LoRA on the pre-training dataset (*i.e.*, RedPajama (Computer, 2023), the pre-training dataset of LLaMA families) to partially restore the performance of an initial quantized model to its pretrained version while transforming the distribution of salient weights into a concentrated row-wise pattern. As shown in Figure 4, it is evident that the salient weights in the preprocessed model indeed exhibit a concentrated row-wise pattern and the results in Table 3 and 6 demonstrate its effectiveness.

We emphasize that **the goal of our preprocessing scheme is to produce a LLM that is more amenable to channel-wise quantization, rather than using the preprocessed model for inference directly**. In Appendix D.2, we further illustrate this point with example experiments. Additionally, it is crucial to clarify that our preprocessing scheme stands out significant advantages and differences from existing post-quantization PEFT methods, e.g., LoRA, QLoRA and QA-LoRA (Xu et al., 2023), please refer to Appendix D.

4 Experiments

In this section, we conduct extensive experiments to validate our novel extremely low-bit PTQ method **PTQ*1.61*** on various benchmarks and LLMs with existing methods to demonstrate that our approach achieves outstanding performance under extremely challenging quantization.

4.1 Experimental Setup

Baseline Since our **PTQ*1.61*** is an extremely low-bit PTQ method, we primarily choose PB-LLM (10% 8-bit) (Shang et al., 2023) and BiLLM (Huang et al., 2024), which claim to be extremely low-bit methods but actually have an equivalent bit-width larger than 2-bit, as baselines. Additionally, several state-of-the-art PTQ methods (2-bit) such as OmniQuant (Shao et al., 2023), AWQ (Lin et al., 2023), QuIP (Chee et al., 2024), and GPTQ (Frantar et al., 2022) are also be evaluated.

Models We evaluate our method mainly on LLaMA (Touvron et al., 2023a), LLaMA-2 (Touvron et al., 2023b) and LLaMA-3 (Dubey et al., 2024), for LLaMA-families are currently the most popular and widely applied among LLMs. Considering the comprehensiveness, experiments on OPT families (Zhang et al., 2022) are in Appendix D.

Training Details We initialize learnable scaling factors with $\alpha_w = \frac{\|w\|_1}{n_w}$ and AdamW optimizer (Loshchilov and Hutter, 2017) with zero weight decay is utilized to update them with learning rate 5e-4 and 1e-3. For PTQ, our calibration set sampled from WikiText2 (Merity et al., 2016) consists

LLaMA	Methods	Bits	PIQA	ARC-e	HellaS	Wing	Race	ARC-c	LAMB-o	LAMB-s	Avg.
	FP	16	78.67	75.29	56.99	70.01	40.29	41.81	73.57	67.82	63.06
1-8	GPTQ	2	53.64	26.09	25.87	47.75	22.68	22.44	0.0	0.0	24.81
	QuIP	2	60.23	38.26	34.83	51.22	28.90	22.10	15.10	8.52	32.40
	OmniQuant	2	58.22	39.94	32.45	52.49	32.25	22.10	23.66	12.77	34.24
	PB-LLM	1.7(+1)	55.71	29.12	28.31	48.86	26.41	19.80	10.42	10.09	28.59
	BiLLM	1(+1.1)	61.10	40.99	31.80	53.67	30.14	20.64	23.15	16.48	36.00
	PTQ*1.61*	**1.61**	**63.71**	**49.62**	**35.73**	**56.75**	**32.54**	**25.26**	**38.93**	**26.57**	**41.14**
	FP	16	79.16	77.36	59.92	72.77	39.62	46.42	76.15	71.08	65.31
1-13	GPTQ	2	51.90	25.84	26.08	49.72	24.11	22.18	0.0	0.0	24.98
	OmniQuant	2	67.14	51.43	**41.28**	56.20	32.73	29.52	23.40	17.85	39.94
	PB-LLM	1.7(+1)	60.45	37.46	30.79	51.07	30.24	18.69	27.71	21.33	34.72
	BiLLM	1(+1.1)	67.90	50.84	39.02	**62.19**	34.07	26.11	**49.93**	33.75	44.98
	PTQ*1.61*	**1.61**	**68.17**	**58.59**	40.02	58.33	**34.26**	**27.22**	45.95	**35.94**	**46.56**
	FP	16	80.96	80.39	63.34	75.69	40.57	52.82	77.59	73.34	68.09
1-30	GPTQ	2	52.50	20.39	25.88	51.38	23.25	21.33	0.06	0.0	24.35
	QuIP	2	**72.42**	58.80	45.64	63.30	35.69	30.20	52.45	36.64	49.39
	OmniQuant	2	70.35	58.03	44.82	58.17	34.93	31.74	41.88	31.75	46.46
	PB-LLM	1.7(+1)	63.76	40.11	33.32	61.17	30.91	21.33	43.78	33.09	40.93
	PTQ*1.61*	**1.61**	70.24	**63.64**	**46.82**	63.61	**37.13**	**32.17**	**55.95**	**44.61**	**51.77**
	FP	16	78.07	76.30	57.14	69.06	39.52	43.34	73.86	68.23	63.19
2-7	QuIP	2	56.53	28.70	27.52	48.78	24.40	18.94	3.18	2.06	26.26
	OmniQuant	2	57.34	38.80	30.11	51.78	27.37	20.73	3.98	1.47	30.20
	PB-LLM	1.7(+1)	54.46	28.20	27.03	49.09	26.70	19.11	7.08	5.71	27.17
	BiLLM	1(+1.1)	60.39	39.94	30.74	51.93	29.57	21.16	18.44	14.05	33.28
	PTQ*1.61*	**1.61**	**63.22**	**47.18**	**35.78**	**52.25**	**29.86**	**22.27**	**37.38**	**25.65**	**39.20**
	FP	16	79.11	79.46	60.04	72.14	40.57	48.46	76.77	70.33	65.86
2-13	QuIP	2	65.45	51.56	39.65	55.72	31.58	25.85	33.86	22.67	40.79
	OmniQuant	2	62.62	44.27	40.16	52.17	30.81	24.66	20.07	10.17	35.62
	PB-LLM	1.7(+1)	54.46	27.95	26.74	49.96	26.03	19.54	3.14	2.50	26.29
	BiLLM	1(+1.1)	63.55	49.83	34.36	**58.17**	32.34	23.81	40.81	25.15	41.00
	PTQ*1.61*	**1.61**	**66.54**	**56.86**	**40.32**	55.88	**33.30**	**26.45**	**47.23**	**31.21**	**44.72**
	FP	16	79.54	80.09	60.14	73.24	40.29	50.17	75.65	68.72	65.98
3-8	GPTQ	2	52.39	26.14	25.74	51.54	20.19	20.65	0.0	0.0	24.58
	QuIP	2	52.72	26.43	29.32	50.67	27.75	20.39	4.93	3.01	26.90
	OmniQuant	2	54.13	28.87	26.50	50.12	22.68	20.48	0.02	0.02	25.35
	PB-LLM	1.7(+1)	56.91	32.37	28.43	49.25	27.66	17.41	16.63	12.44	30.14
	BiLLM	1(+1.1)	60.01	38.26	31.48	**53.75**	**30.14**	19.45	26.30	15.51	34.36
	PTQ*1.61*	**1.61**	**63.22**	**46.17**	**34.71**	52.80	29.09	**23.04**	**30.27**	**18.26**	**37.20**

Table 2: Reasoning accuracies comparison on LLaMA family. More tasks are listed in Table 11 and 12.

of 128 random 2048 token-segments and the block-wise training process includes 20 epochs with a batch size of 1. For quantization preprocessing, the number of steps and ranks in lightweight restorative LoRA is 20K and 64 respectively. The entire process is deployed on 2 Nvidia A800 GPUs.

Datasets On language generation tasks which are the core objectives of LLMs, our test set comes from WikiText2 and C4 (Raffel et al., 2020). We also select several reasoning benchmarks, *i.e.*, PIQA (Bisk et al., 2020), ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2021), Race (Lai et al., 2017) and LAMBADA (Paperno et al., 2016), using the open-

sourced toolkit lm-evaluation-harness (Gao et al., 2023). We also assess the evaluation on MMLU (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021) and LongBench (Bai et al., 2024), please refer to Appendix E. In addition, RedPajama (Computer, 2023) is used for quantization preprocessing.

4.2 Experiments on Language Generation Tasks

The fundamental prowess of LLMs lies in their language generation capabilities. Consequently, evaluating such capabilities of a quantized model via perplexity serves as the core metric of a quantization method. As presented in Table 1, we compare
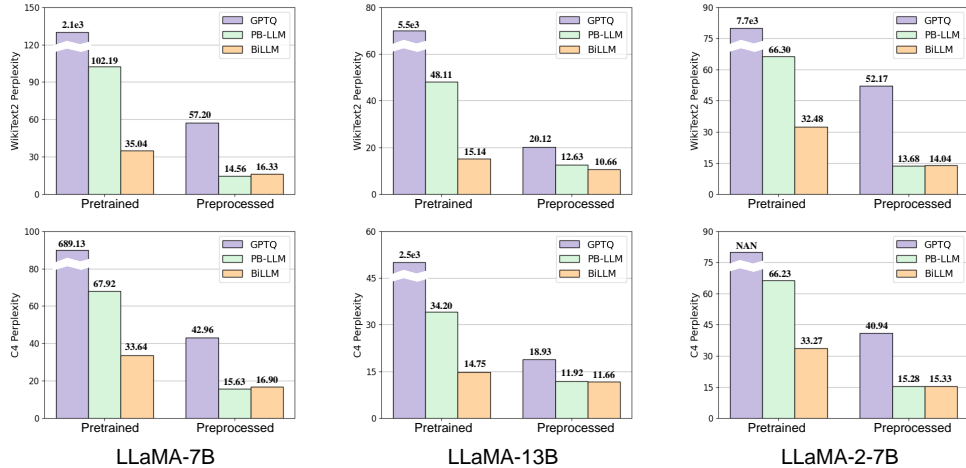
Figure 5: Our novel quantization preprocessing scheme on other existing PTQ methods. Results on more LLMs and common sense reasoning tasks can be found in Figure 8.

the perplexities between our **PTQ***1.61* and other baselines to valid the effectiveness on extremely low-bit PTQ tasks, from where we can see that our method achieves promising performance. In comparison to the two extremely low-bit methods PB-LLM and BiLLM, our performance significantly surpasses theirs while not introducing intolerable bit-width for each weight. In terms of OmniQuant, which performs best on LLaMA families among baselines, our method still surpasses it by a significant margin. For instance, on LLaMA-2-7B we achieves 12.70 while OmniQuant is 37.37 in Wiki-Text2. Significantly, our method performs better on LLaMA3, which is known to be harder to quantize.

4.3 Experiments on Reasoning Benchmarks

Reasoning capability is becoming a crucial metric for evaluating PTQ approaches. The comparison results are indicated in Table 2, where our method exhibits superiority in most benchmarks. For instance, compared with the second best baseline, BiLLM, our method showcases an average performance increase of $1.58\% \sim 5.92\%$. Particularly, on LAMBADA most baselines suffer from significant performance degradation while we still maintain an outstanding level. Considering the comparison results and the minimal 1.61-bit, we conclude that our method represents the state-of-the-art extremely low-bit PTQ scheme for LLMs.

4.4 Ablation Study

After demonstrating the advancement of our **PTQ***1.61*, we conduct ablation study on LLaMA-13B to further validate the effectiveness of our each innovation as indicated in Table 3. As the first

Structured Mask	Learnable Scalar	Preprocess	WikiText2	C4
-	-	-	14664	11377
✓	-	-	1370.4	772.83
-	-	✓	569.81	702.44
✓	✓	-	14.22	20.78
✓	✓	✓	**9.67**	**13.51**

Table 3: Ablation study (PPL) on LLaMA-13B.

row, without any additional improvements, directly using the derived analytically scaling factors for binarization would almost entirely compromise the text generation capability of LLMs. When utilizing our structured masks to retain salient weights as the second row, there is a significant improvement, indicating the importance of salient weights and the effectiveness of our masks, but there remains considerable room for enhancement. Furthermore, our novel block-wise strategy for non-salient weights binarization lifts the performance to a excellent level as demonstrated by the forth row. Ultimately, the last row illustrates that the row-wise pattern obtained by our quantization preprocessing brings a remarkable enhancement. More detailed ablation results are available in Appendix B.

4.5 Quantization Preprocessing on Baselines

In addition to our **PTQ***1.61*, we also employ the proposed quantization preprocessing scheme on other baselines to validate its scalability and the results can be found in Figure 5, which demonstrate that significant improvements appear in all baselines, especially for 2-bit GPTQ which completely collapses without preprocessing. With the effectiveness of our preprocessing scheme, future

research can take a fresh perspective to focus on finding a more appropriately pretrained model.

At present, despite its strong performance, our preprocessing scheme still suffers from drawbacks such as longer runtime (as indicated in Table 8). Therefore, we must clarify that our preprocessing scheme is viewed as an **optional component** of our **PTQ*1.61***. As indicated in Table 6, even without preprocessing, our **PTQ*1.61*** still achieves state-of-the-art performance under extremely low-bit setting. To avoid re-preprocessing from scratch, we have released several preprocessed LLMs on huggingface[1], where the followers can download and then perform quantization on them using our **PTQ*1.61***.

5 Conclusion

In this paper, we explore the real limit of post-training quantization and propose an extremely low-bit PTQ approach namely **PTQ*1.61***, which is truly the first PTQ method enables sub 2-bit quantization for LLMs. Firstly, one-dimensional structured mask with negligibly additional 0.0002-bit per weight is introduced to preserve salient weights. For non-salient weights binarization, we devise an efficient block-wise optimization strategy to learn scaling factors considering row correlations and angular biases. In addition to above contributions, we further propose a quantization preprocessing paradigm to transform the salient weights into a row-wise pattern which is able to alleviate the difficulty in per-channel quantization. Extensive experiments indicate that **PTQ*1.61*** becomes state-of-the-art extremely low-bit PTQ method for LLMs.

Limitation

Although showcasing superior performance, the preprocessing scheme still has limitations to be reckoned with, which requires more runtime to get a start point before quantization. For example, our runtime reaches 2h on LLaMA-7B, and fortunately, this falls within an acceptable range (OmniQuant reports 1.1h but exhibits worse performance and higher bit-width per weight). Considering that extremely low-bit quantization is the most challenging quantization scenario especially for PTQ, we believe it is worthwhile to sacrifice some computational resources within an acceptable range to pursue higher performance.

In addition, due to the limitation that commercial NVIDIA GPUs do not support such low-bit inference, and designing specific hardware requires larger research teams and financial support, we cannot provide real-world inference evaluation results yet. Our goal is to explore the performance limits of PTQ by fake-quantization before commercial hardware support is available. We believe this will eventually be realized as evidenced by the quick development of GPUs.

Ethics Statement

This paper introduces solutions to the challenges associated with Large Language Models (LLMs) quantization, with the overarching goal of facilitating the widespread adoption and application of LLMs. In the current landscape, ethical concerns tied to LLMs, including the presence of hidden biases encoded in the models, are garnering heightened attention. Following our investigation, we assert that our proposed method does not further amplify the biases and contravene any ethical standards.

Acknowledgment

References

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. LongBench: A bilingual, multi-task benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

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.

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

---

[1]https://huggingface.co/ptq161.

Adrian Bulat and Georgios Tzimiropoulos. 2019. Xnor-net++: Improved binary neural networks. *arXiv preprint arXiv:1909.13863*.

Chee-Yong Chan and Yannis E Ioannidis. 1998. Bitmap index design and evaluation. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 355–366.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. 2024. Quip: 2-bit quantization of large language models with guarantees. *Advances in Neural Information Processing Systems*, 36.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

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*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Together Computer. 2023. Redpajama: an open dataset for training large language models.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*.

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

Xin Ding, Xiaoyu Liu, Yun Zhang, Zhijun Tu, Wei Li, Jie Hu, Hanting Chen, Yehui Tang, Zhiwei Xiong, Baoqun Yin, et al. 2023. Cbq: Cross-block quantization for large language models. *arXiv preprint arXiv:2312.07950*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. 2019. Learned step size quantization. *arXiv preprint arXiv:1902.08153*.

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

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.

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)*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Kun Huang, Bingbing Ni, and Xiaokang Yang. 2019. Efficient quantization for neural networks with binary weights and low bitwidth activations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3854–3861.

Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. 2024. Billm: Pushing the limit of post-training quantization for llms. *arXiv preprint arXiv:2402.04291*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. 2024. Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13355–13364.

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. *arXiv preprint arXiv:2102.05426*.

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

Mingbao Lin, Rongrong Ji, Zihan Xu, Baochang Zhang, Yan Wang, Yongjian Wu, Feiyue Huang, and Chia-Wen Lin. 2020. Rotated binary neural network. *Advances in neural information processing systems*, 33:7474–7485.

Jiawei Liu, Lin Niu, Zhihang Yuan, Dawei Yang, Xinggang Wang, and Wenyu Liu. 2022. Pd-quant: Post-training quantization based on prediction difference metric. *arXiv preprint arXiv:2212.07048*.

Peiyu Liu, Zikang Liu, Ze-Feng Gao, Dawei Gao, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. 2023a. Do emergent abilities exist in quantized large language models: An empirical study. *arXiv preprint arXiv:2307.08072*.

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

Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. 2018. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proceedings of the European conference on computer vision (ECCV)*, pages 722–737.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. 2024. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720.

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

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. 2020. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206. PMLR.

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

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.

Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. 2020. Binary neural networks: A survey. *Pattern Recognition*, 105:107281.

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. *Journal of machine learning research*, 21(140):1–67.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer.

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.

Yuzhang Shang, Zhihang Yuan, Qiang Wu, and Zhen Dong. 2023. Pb-llm: Partially binarized large language models. *arXiv preprint arXiv:2310.00034*.

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

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

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

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

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. 2023. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. *arXiv preprint arXiv:1906.04284*.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. 2023. Bitnet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*.

Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. 2022. Outlier suppression: Pushing the limit of low-bit transformer language models. *Advances in Neural Information Processing Systems*, 35:17402–17414.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

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

Yuzhuang Xu, Xu Han, Zonghan Yang, Shuo Wang, Qingfu Zhu, Zhiyuan Liu, Weidong Liu, and Wanxiang Che. 2024. Onebit: Towards extremely low-bit large language models. *arXiv preprint arXiv:2402.11295*.

Zihan Xu, Mingbao Lin, Jianzhuang Liu, Jie Chen, Ling Shao, Yue Gao, Yonghong Tian, and Rongrong Ji. 2021. Recu: Reviving the dead weights in binary neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5198–5208.

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. *Advances in Neural Information Processing Systems*, 35:27168–27183.

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

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*.

Jiaqi Zhao, Miao Zhang, Chao Zeng, Ming Wang, Xuebo Liu, and Liqiang Nie. 2024. Lrquant: Learnable and robust post-training quantization for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2240–2255.

Appendix

A   Average Bit-width Per Weight of Linear Layer

For a weight in a mix-precision quantized linear layer, its average bit-width b is calculated by following formulation:

$$b = 1 * r_b + b_{salient} * (1 - r_b) + b_{index} + b_{additional}, \quad (8)$$

where r_b is the ratio of binarized weights in the layer and $b_{salient}$ denotes the bit-width of salient weights. The first two item is also called weight bit-width. b_{index} represents the bit-width for index storing using the bitmap mechanism (Chan and Ioannidis, 1998) and $b_{additional}$ is used for saving quantization parameters, *i.e.*, scaling factors.

Assume the weight matrix is 4096×4096 in such layer. For our **PTQ*1.61*** which saves 20% salient weights to 4-bit and binaries the others, the weight bit-width can be effortlessly determined as 1.6-bit and the overall bits number is $4096 \times 4096 \times 0.8 + 4096 \times 4096 \times 0.8 \times 4 = 26,843,545$. In addition, the shape of our 1-bit one-dimensional structured mask is 4096×1, so its r_b is $1 \times 4096 \div 26,843,545 \approx 0.0002$. Moreover, quantization parameters in our method contains 3 low-dimensional scaling factors and a part of zero-points, so $b_{additional}$ will be $(3 \times 4096 \times 1 \times 16 + 0.2 \times 4096 \times 16) \div 26,843,545 \approx 0.008$. Overall, the average bit-width per weight in a layer quantized by our **PTQ*1.61*** is $b = 1.6 + 0.0002 + 0.008 \approx 1.61$.

For PB-LLM which selects 10% salient weights at 8-bit using a 1-bit fine-grained unstructured mask with the same shape as the weight matrix, the obtained average bit-width per weight is $b = 0.1 \times 8 + 0.9 \times 1 + 1 = 2.7$.

BiLLM devises a finer-grained binarization scheme which divides all weights into 3 groups and calculates group-wise scaling factors. Specifically, they propose a structured mask based on Hessian for salient weights and an unstructured mask based magnitude for unsalient weights. From their paper we get that their weight bit-width is 1-bit and $b_{additonal}$ is 0.1-bit then we have $b = 1.0 + 0.1 + 1.0 = 2.1$.
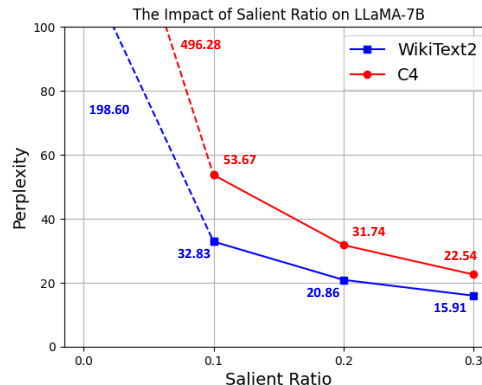

Figure 6: The impact of salient ratio in **PTQ*1.61*** on LLaMA-7B.

B   Structured Mask

B.1   The Impact of Salient Channels Ratio in Proposed Structured Mask

Inspired by PB-LLM which declares the importance of salient weights, we devise a one-dimensional structured mask to preserve top 20% salient channels of weight matrices at 4-bit based on input activations in our **PTQ*1.61***. More comprehensively, we delve into the effects of the salient channels ratio on the quantized model. As illustrated in Figure 6, we compare the performance of **PTQ*1.61*** on pretrained LLaMA-7B with different ratios and the results indicate that higher salient ratios lead to better performance. The reason why we give up the optimal 30% is that the average bit-width per weight in the quantized model nearly approaches 2-bit (1.91-bit), which violates the conditions for extremely low-bit quantization. Therefore, we opt for the second-best performance 20% as our preserved ratio. However, it is crucial to note that the aforementioned experiments do not diminish the significance of our other innovations on non-salient weights binarization. As evident from the Table 3, simply maintaining the salient weights without incorporating block-wise optimization will results in inadequate performance.

B.2   Distinctions with AWQ and OWQ

For AWQ, both their method and our **PTQ*1.61*** take into account the relationship between input activation and weight, and utilize this relationship for subsequent processing. But notably, there is no structured mask in AWQ. They leverage this discovery to perform a grid search based on MSE loss to select appropriate quantization scalars. These scalars are used to scale the corresponding weights

LLaMA	Methods	Bits	WikiText2	C4
1-7	OWQ	2	13.64	**15.78**
	PTQ*1.61*	1.61	**12.50**	17.13
1-13	OWQ	2	10.69	**12.45**
	PTQ*1.61*	1.61	**9.67**	13.51
2-7	OWQ	2	18.78	19.93
	PTQ*1.61*	1.61	**12.70**	**17.73**
2-13	OWQ	2	30.28	36.78
	PTQ*1.61*	1.61	**9.74**	**13.64**

Table 4: PPL comparison between OWQ and **PTQ*1.61*** on LLaMA family.

LLaMA	Methods	Mask	WikiText2	C4
1-7	**PTQ*1.61***	OWQ	22.11	33.77
		Ours	**12.50**	**17.13**
1-13	**PTQ*1.61***	OWQ	57.33	132.08
		Ours	**9.67**	**13.51**
2-7	**PTQ*1.61***	OWQ	NAN	NAN
		Ours	**12.70**	**17.73**
2-13	**PTQ*1.61***	OWQ	1.1e3	2.7e3
		Ours	**9.74**	**13.64**

Table 5: PPL comparison between OWQ and our mask.

up or down. Weights corresponding to larger input activations are assigned larger scalars to increase their magnitude to reduce quantization error. Conversely, our structured mask is designed through rigorous mathematical derivation that input activation has a greater impact on the upper bound of quantization error. This allows us to assess the importance of corresponding weights and avoid unacceptable binarization errors. Furthermore, the results in Table 5 demonstrate that our method outperforms AWQ in extremely low-bit quantization scenarios without requiring additional innovations.

For OWQ, we need to clarify that the two method differ significantly in terms of innovations, motivation, and the tasks they target. In the process of formula derivation, OWQ use Hessian matrix (related to input activation) to decompose quantization errors to determine which weights columns to retain at full precision. However, it is important to emphasize that there are approximation processes in decomposing, such as ignoring the first-order and higher-order terms in Taylor expansion. Moreover, using Cholesky decomposition to simulate the Gaussian elimination of Hessian updating also involves approximations. In contrast, our structured mask does not involve approximation in its mathematical derivation. It directly derives two

key factors that affect the upper bound of quantization errors and combined with visualization, determines that input activation has the greatest impact on the upper bound of quantization errors. This avoids introducing additional errors due to mathematical calculations. Therefore, while there are similarities in form between the structured mask of our **PTQ*1.61*** and OWQ, the motivations are entirely different. Approximation in the Hessian-based structured mask is reasonable for high-bit quantization. However, in extremely low-bit PTQ, especially binarization, these approximations can be infinitely magnified. Thus, directly reducing the upper bound of quantization errors is a method with fewer errors. To prove this, we set up comparative experiments on several models as Table 4, where our **PTQ*1.61*** quantizes the model to 1.61-bit, while OWQ quantizes it to 2-bit. The results show that the models quantized by PTQ1.61 achieve better results under lower bit-width, especially on the LLaMA2. We also design structured masks based on Hessian as OWQ to preserve 20% salient channels, from Table 5 it can be observed that compared with our **PTQ*1.61*** the performance of structured mask proposed by OWQ collapses, which also proves our structured mask has smaller errors and is more suitable for extremely low-bit quantization. Furthermore, it is noteworthy that OWQ needs to store certain weights in FP16 format. Due to the differences in storage methods between INT and FP formats, designing kernel functions is very difficult. In contrast, our weights are all in INT format, making kernel function design much simpler.

C Block-wise Optimization

C.1 Hyperparameters

In addition to the preservation ratio of the structured mask, the remaining hyperparameters are only the learning rates. We conduct extensive experiments with various learning rates from 1e-4 to 1e-2 and finally select the optimal. The limited number of hyperparameters also demonstrate that our **PTQ*1.61*** is not complex and easy to deploy.

C.2 Angular Biases

Except for implict row-wise dependencies, our block-wise strategy also takes angular biases into account. Previous PTQ methods (OmniQuant, GPTQ or other quantization methods for CNN models such as BNN) only focuses on the magnitude gaps between FP models and their quantized

Dataset	Methods	Bits	LLaMA					OPT		
			1-7	1-13	1-30	2-7	2-13	2.7	6.7	13
WikiText2	FP	16	5.68	5.09	4.10	5.47	4.88	12.47	10.86	10.12
	OmniQuant	2	15.47	13.21	8.81	37.37	17.21	1.1e6	9.3e5	4.6e4
	PB-LLM	1.7(+1)	102.19	48.11	26.37	66.30	462.84	238.18	174.76	75.28
	BiLLM	1(+1.1)	35.04	15.14	9.96	32.48	21.77	49.55	45.36	18.22
	PTQ*1.61*	1.61	20.86	14.22	11.84	22.58	15.63	44.10	27.51	22.94
	PTQ*1.61*	1.61	**12.50**	**9.67**	**7.95**	**12.70**	**9.74**	**28.56**	**19.45**	**15.55**
C4	FP	16	7.08	6.61	5.98	6.97	6.46	13.16	11.74	11.19
	OmniQuant	2	24.89	18.31	13.67	90.64	26.76	9.4e5	4.4e6	1.2e5
	PB-LLM	1.7(+1)	67.92	34.20	22.45	66.23	333.54	161.47	102.85	47.50
	BiLLM	1(+1.1)	33.64	14.75	**10.95**	33.72	23.14	40.57	39.58	**17.78**
	PTQ*1.61*	1.61	31.74	20.78	15.53	36.07	22.77	72.64	38.37	29.97
	PTQ*1.61*	1.61	**17.13**	**13.51**	10.98	**17.73**	**13.64**	**33.45**	**22.78**	18.31

Table 6: Perplexities comparison of our **PTQ*1.61***, BiLLM and OmniQuant on pretrained and preprocessed LLaMA and OPT families. * indicates **PTQ*1.61*** without preprocessed.

LLaMA	Mask	WikiText2	C4
1-7B	w/o	13.56	18.13
	w	**12.50**	**17.13**
1-13B	w/o	9.98	13.85
	w	**9.67**	**13.51**
2-7B	w/o	13.69	19.85
	w	**12.70**	**17.73**
2-13B	w/o	10.24	14.26
	w	**9.74**	**13.64**

Table 7: PPL comparison between whether considering angular biases (w) or not (w/o).

LLaMA	Method	GPU Memory	Runtime
7B	OmniQ	13GB	1.1h
	OneBit	360GB	24days
	PTQ*1.61*	15GB	2h
13B	OmniQ	18GB	2.2h
	OneBit	360GB	32days
	PTQ*1.61*	19GB	4.2h

Table 8: Resource requirements comparison.

counterpart, while the inherent directional distinctions, which has been proved to exist by RBNN and LRQuant, cannot be ignored which will not be addressed by traditional scaling factors and MSE loss. Therefore, we introduce loss function considering cosine similarity into block-wise optimization. We provide the detailed formula of dequantized weights considering scaling factors:

$$W_q^{'} = (\alpha_{r_1} \times \alpha_{r_2}) \circ (\alpha_s * \text{sign}(W)). \quad (9)$$

We conduct a comparison on whether this loss function is used or not in our **PTQ*1.61*** in Table 7 and the results demonstrate that with this consideration, the block-wise optimization strategy used in our method becomes more robust and advanced than that in CBQ or OmniQuant.

D More Details on Quantization Preprocessing

D.1 Enhancement on Our PTQ*1.61*

In addition to LLaMA-13B in Table 3, we evaluate the enhancements on our **PTQ*1.61*** brought

by novel quantization preprocessing paradigm on more LLMs and the results are listed in Table 6, from which we confirm that the preprocessing consistently enhances the performance of our **PTQ*1.61*** on each model. Crucially, our results reveal that, apart from preprocessing, our other innovations alone offer comparable performance advantages over existing methods, while attaining a lower weight compression ratio.

Furthermore, the quantization preprocessing also augments the common sense reasoning capabilities of our method as listed in Figure 7.

D.2 Preprocessed Model and FP Model

In Section 3.4, we introduce the rationale behind our quantization preprocessing scheme and emphasize its purpose is to reshape the weight distribution to be more suitable for quantization while not using the preprocessed model directly for inference. As shown in Table 9, it can be observed that after preprocessing the FP16 performance of LLaMA-13B degrades slightly, but after quantization it significantly outperforms the pretrained model, which further proves the effectiveness of our preprocessing method.
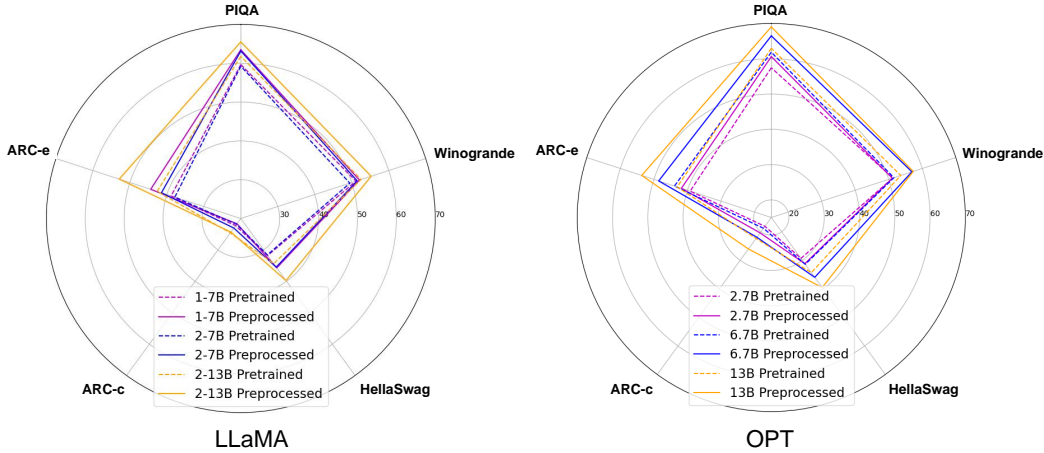
Figure 7: Zero-shot accuracies comparison between pretrained and preprocessed model quantized by our **PTQ*1.61***.

Table 9: Full-precision perplexities comparison of the pretrained and preprocessed LLaMA-13B.

Model	Bits	Dataset	
		WikiText2	C4
Pretrained	16	5.09	6.61
-Quant	1.61	14.22	20.78
Preprocessed	16	9.32	12.33
-Quant	1.61	9.67	13.51

Model	Dataset	
	WikiText2	C4
LLaMA-7B	287.38	829.91
LLaMA-2-7B	NAN	NAN

Table 10: Perplexities on LLMs when setting learnable row-wise mean which is similar to zero-point in QA-LoRA (group-size=1) for binarization.

D.3   Resources Requirement

Due to quantization preprocessing, our **PTQ*1.61*** has higher resource cost compared to other PTQ methods. For example, compared to OmniQuant, as shown in Table 8, our method has slightly higher memory cost. Although this is a limitation of our method, it remains within an acceptable range (compared with OneBit which needs 5 A800 GPUs and over 24days to train a low-bit LLaMA-7B) considering we target the most challenging extremely low-bit scenario (it is really hard to ensure performance especially for PTQ), and the performance is better than OmniQuant which is current SOTA in extremely low-bit PTQ even without quantization preprocessing as indicated in Table 6.

D.4   OPT Results on Other Approaches

The effectiveness of proposed quantization preprocessing scheme on LLaMA families quantized by other existing low-bit PTQ methods has been illustrated in Figure 5. Besides, results on OPT families is available in Figure 8 and the similar phenomenon can be observed.

D.5   Comparison with Post-Quantization PEFT Methods

It is worth mentioning that the restorative LoRA in our quantization preprocessing scheme stands out significant advantages and differences from existing post-quantization PEFT methods (Dettmers et al., 2023; Xu et al., 2023), as shown in Figure 9. For advantages: **(a)** Compared with LoRA and QLoRA (Dettmers et al., 2023): Both methods requires to store additional float-point low-rank matrices, augmenting inference costs. Merging these matrices with the low-bit quantized model will reinstate it to FP16, imposing considerable storage demands. Conversely, our preprocessing approach concurrently optimizes storage and inference. **(b)** Compared with QA-LoRA (Xu et al., 2023): QA-LoRA addresses the aforementioned issues by adjusting the group-wise zero-point in Equation (1), which is similar to the mean used in binarization. Specifically, QA-LoRA partitions each row of the weight matrix into numerous groups (termed group-size) and determines a distinct zero-point for each group, thereby circumventing the need for additional memory during fine-tuning. However, their fine-grained group-size of zero-point is set to 32, which introduces an additional 0.5-bit per weight, leading to extra storage overheads. Our experi-
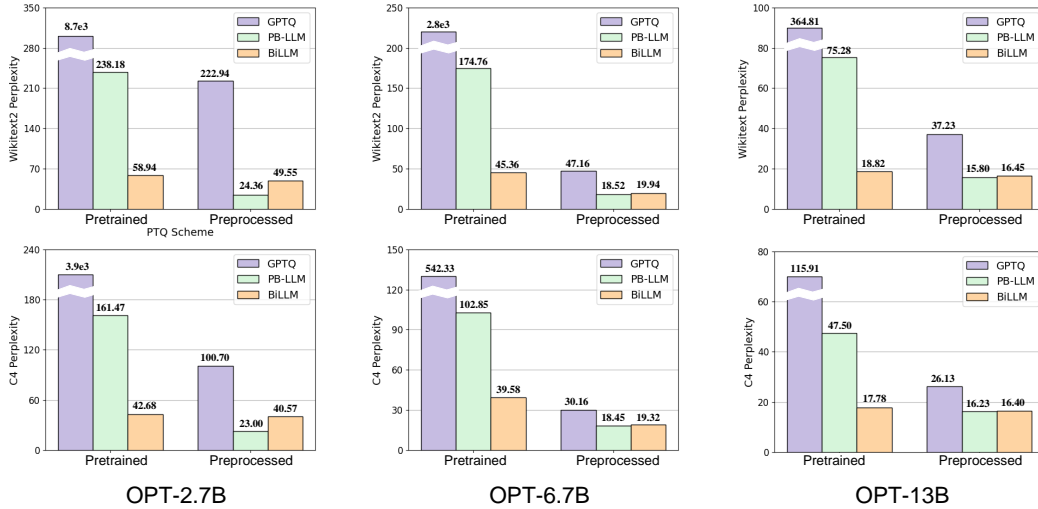
Figure 8: Our novel quantization preprocessing scheme on OPT families quantized other existing PTQ methods.
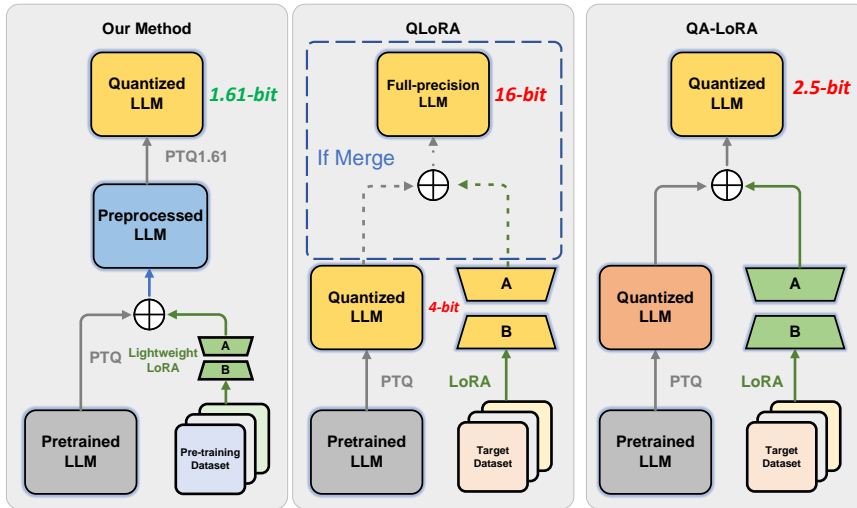

Figure 9: Comparison between our method and post-quantization PEFT methods. The yellow part of each method will be actually deployed and loaded during inference.

ments reveal that setting the group-size to 1 in QA-LoRA as us to learn a row-wise mean significantly compromises performance, as demonstrated in Table 10.

The remaining differences can be summarized as: **(a)** Objective: PEFT aims to enhance model's performance on downstream tasks, whereas restorative LoRA targets to transform salient weights to a concentrated row-wise pattern for better quantization. **(b)** Datasets: PEFT utilizes datasets from the target domain, such as Alpaca (Taori et al., 2023) and FLAN v2 (Longpre et al., 2023), while restorative LoRA tends to leverage the pre-training datasets, *i.e.*, RedPajama (Computer, 2023). **(c)** Time cost: For LLaMA-7B, PEFT usually takes over 10+ hours (even the less QA-LoRA requires over 6 hours), whereas our lightweight restora-

tive LoRA is cost-friendly, requiring less than 1.2 hours.

D.6   Comparison with QAT

As is known to all that QAT frameworks improve quantization performance by directly training a quantized LLM to get optimal weights as well as quantization parameters to accommodate quantization errors at target bit-width, where its strategy for adjusting weights may share some similarities with us. To eliminate this confusion, several important distinctions and limitations need to be highlighted: **(a)** QAT trains all the weights in the LLM, which incurs extremely high training costs, *i.e.*, LLM-QAT (Liu et al., 2023b) requires 3 days to retrain a quantized OPT-1.3B on 8 Nvdia-A100 GPUs and OneBit (Xu et al., 2024) spends over 24 days
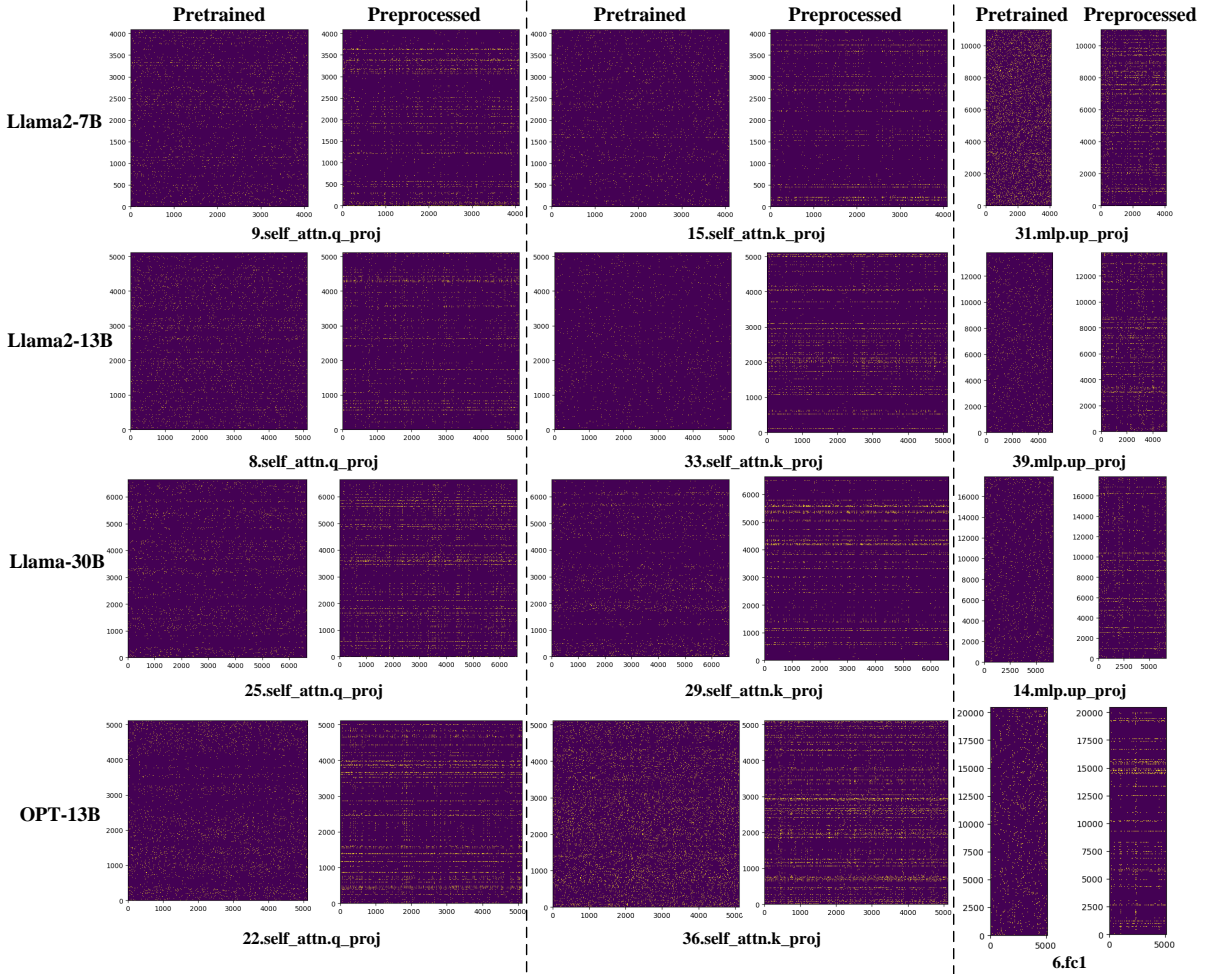
Figure 10: More visualizations which show the impact of our quantization preprocessing method.

to retrain a LLaMA-7B on 5 Nvdia-A800 GPUs. In contrast, our quantization preprocessing only requires to transform a few channels into a PTQ-friendly row-wise format utilizing the low-rank nature of weight compensation, with only a single Nvdia-A100 GPUs for less than 1.2 hours. **(b)** QAT needs to know the target bit-width at the beginning and necessitates retraining from scratch if the bit-width changes, making it inflexible. While our method only requires saving the preprocessed model, which can then be quickly adapted to any target bit-width because the row-wise pattern is universally applicable to per-channel PTQ across various target bit-widths. **(c)** It is well-known that training a binary quantized LLM with gradient descent is highly challenging (Qin et al., 2020). Specifically, QAT introduces the straight-through estimator to approximate quantization gradients. However, under extremely low-bit conditions, the gradient approximation error becomes significantly large, making the training process more unstable

and harder to converge. By comparison, our preprocessing strategy naturally addresses this by placing the restorative LoRA process before quantization, thereby avoiding the associated optimization difficulties.

D.7 More Visualizations

For a deeper understanding of the quantization pre-processing impacts on salient weights distribution, we provide more comprehensive visual data, including a variety of layers and models as Figure 10. The similar channel-wise nature occurs in various preprocessed models and layers which demonstrates that our quantization preprocessing process is essential.

E More Evaluations

E.1 MMLU and GSM8K

In addition to the benchmarks in the content, we also valid the quantization performance on GSM8K and MMLU on several LLMs. However, as the

LLaMA	Method	MMLU	GSM8K
1-7B	PB-LLM	23.0	0.23
	BiLLM	22.9	0
	PTQ*1.61*	23.0	0.15
2-7B	PB-LLM	23.0	0
	BiLLM	22.9	0
	PTQ*1.61*	22.9	0.61
3-8B	PB-LLM	22.9	0.83
	BiLLM	22.8	0.30
	PTQ*1.61*	22.9	0.83

Table 11: Evaluation accuracies on MMLU and GSM8K.

LongBench	PBLLM	BiLLM	PTQ*1.61*
2WikiMQA	5.0	4.1	**12.7**
TriviaQA	9.9	13.2	**34.8**
Multi-News	13.6	11.7	**14.6**
SAMSum	3.3	5.4	**19.5**
QMSum	5.4	8.3	**15.6**
MultiFieldQA-EN	8.4	8.1	**13.2**

Table 12: Evaluation on LongBench.

near-random levels illustrated in Table 11, we observe that under extremely low-bit quantization, all existing PTQ methods nearly make LLMs loss the ability, which is consistent with previous research (Liu et al., 2023a). Considering the disappointing outcomes, we choose not to list the comparison into the content.

E.2 Long Context Understanding

LongBench is a novel benchmark for evaluating long context understanding capability which is a critical measurement for LLMs application. Due to LongBench only supports Chat-LLMs, we select LLaMA2-7b-Chat for evaluation in Table 12. A consistent superior performance proves the effectiveness of our method.

E.3 Throughput and Inference Memory

For real-world system evaluation, current NVIDIA GPUs do not yet support such low-bit inference. Designing specific hardware and operation kernals that meets the inference conditions requires larger research teams and financial support, so our goal is to explore the performance limits of PTQ by fake-quantization before commercial hardware support is available. We believe this will eventually be realized, as evidenced by the latest NVIDIA GPUs now supporting 4-bit inference, whereas only a year ago they were limited to 8-bit.

Compared with PB-LLM and BiLLM which re-

Method	LLaMA-1/2-7B	LLaMA-1/2-13B
PB-LLM	2.36GB	4.49GB
BiLLM	1.83GB	3.50GB
PTQ*1.61*	**1.41GB**	**2.68GB**

Table 13: Inference memory comparison.

quires to load extra unstructured mask during inference, our method is much more efficient. Followed by previous research (Ma et al., 2024), we can obtain information about a 1.58-bit (ours is 1.61-bit) LLaMA-7B achieves a 2.9X speedup in latency and LLaMA2-70B gains an 8.9X increase in throughput (2977 tokens/s).

In addition, we provide the memory usage of LLMs quantized by PB-LLM, BiLLM and **PTQ*1.61*** via calculation considering weight bits, scaling factors and masks. As indicated by Table 13, our **PTQ*1.61*** has an advantage in memory efficiency, which is of practical benefits.

F Discussion on Practically Applicability of Extremely Low-bit Weight Quantization

F.1 Accuracy-Latency Tradeoff Analysis

One important aspect of LLM quantization is the accuracy-latency tradeoff. If speed is the sole priority, an aggressive compression ratio can significantly improve latency, but this often results in unacceptable accuracy degradation. Conversely, adding additional components can help recover accuracy but may hinder system acceleration, leading to slower performance. Therefore, it is crucial to analyze the accuracy-latency tradeoff.

Delve deeper into the calculation process of a quantized model on a GPU. For weight-only quantization, the primary acceleration comes from the transfer of low-bit integer weights from memory to the MAC (Multiply Accumulate) processing unit, which reduces the amount of data transfer compared with FP model. To restore performance, existing quantization methods introduce additional components, such as FP16 channel-wise scaling factors, to reduce quantization errors. For the attention layer of an LLM, the size of the weight matrix is usually 4096×4096, and the scaling factors are a 1×4096 vector. Therefore, in the transfer process mentioned in the previous paragraph, the bad impact on latency from transferring such a small amount of FP16 scaling factors is almost negligible compared to the significant inference acceleration

Model	Method	PIQA	ARC-e	ARC-c	HellaS	WinoG	Race	Avg.
LLaMA-13B	FP	79.16	77.31	46.42	59.90	72.93	39.71	62.57
	PB-LLM	60.55	37.46	18.69	30.79	51.07	30.24	42.75
	SQ(W4A4)	62.45	44.31	24.48	35.63	50.11	31.74	41.45
	PTQ1.61	**68.17**	**58.59**	**27.22**	**40.02**	**58.33**	**34.26**	**47.77**
LLaMA-30B	FP	80.96	80.39	52.73	63.34	75.69	40.57	65.61
	PB-LLM	64.91	46.38	21.33	35.81	61.17	30.91	43.42
	SQ(W4A4)	54.57	28.82	19.45	26.93	49.88	22.52	33.70
	PTQ1.61	**70.24**	**63.64**	**32.17**	**46.82**	**63.61**	**37.13**	**52.27**
LLaMA2-13B	FP	79.05	79.38	48.38	60.04	72.14	40.48	63.25
	PB-LLM	54.46	27.95	19.54	26.74	49.96	26.03	34.11
	SQ(W4A4)	54.18	28.78	19.28	27.82	50.19	27.26	34.59
	PTQ1.61	**66.54**	**56.86**	**26.45**	**40.32**	**55.88**	**33.30**	**46.59**

Table 14: Performance comparison among FP16, SmoothQuant(W4A4), PB-LLM and our **PTQ*1.61***.

benefits brought by transferring the low-bit weight matrices. Considering all above, a small amount of additional components such as scaling factors will not have a significant bad impact on the inference latency in model quantization.

F.2 Compared with Weight-activation Quantization

In order to prove the necessity of extremely low-bit weight PTQ research, we provide the performance gap among FP16 results, W4A4 SmoothQuant and extreme low-bit PTQ methods. The results are shown as Table 14. The results demonstrate the necessity of research into extremely low-bit PTQ from two aspects. Firstly, compared to the results of FP16, previous method (PBLLM) indeed showed a significant gap, but our PTQ1.61 has narrowed the performance gap to an acceptable level. Secondly, compared to SmoothQuant, the most popular and wide applied weight-activation PTQ method, its performance of W4A4, which is currently supported by the latest commercial GPUs, is still inferior to our PTQ1.61, proving the research prospect of extremely low-bit weight quantization is as bright as weight-activation quantization. We are confident that with further advancements, the disparity between extremely low-bit weight quantization and full precision will progressively diminish.