LFQ: LOGIT-AWARE FINAL-BLOCK QUANTIZATION FOR BOOSTING THE GENERATION QUALITY OF LOW-BIT QUANTIZED LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

As large language models (LLMs) continue to scale, low-bit weight-only post-training quantization (PTQ) offers a practical solution to their memory-efficient deployment. Although block-wise PTQ is capable of matching the full-precision (FP) baseline on basic language modeling and understanding, its quality is degraded for *generative* tasks—especially at longer responses and extended chains of thought, which is critical in boosting task accuracy. We attribute this shortfall to two factors: (i) the omission of the unembedding layer (the LM head) in block-wise optimization and (ii) the reliance on the mean squared error (MSE) objective. Both factors cause the token probability distribution of the quantized model to misalign with that of the FP model, yielding notable accuracy drops on text generation benchmarks. To rectify the discrepancy, we introduce *Logit-aware Final-block Quantization (LFQ)*, a simple yet effective enhancement to block-wise PTQ that quantizes the final Transformer block by minimizing the cross-entropy between the logits of the FP model and those of its quantized counterpart. By aligning token probabilities at the logit level in the final block, LFQ consistently improves the accuracy of complex generation tasks over state-of-the-art block-wise PTQ across diverse model families and text generation tasks, while maintaining parity with FP baselines on language modeling and understanding.

1 INTRODUCTION

The evident success of large language models (LLMs) (Grattafiori et al., 2024; Qwen et al., 2025; Team et al., 2025) based on the decoder-only transformer (Vaswani et al., 2023) is largely attributed to their ever-increasing number of parameters (Kaplan et al., 2020). However, the proportionally increasing memory footprint of the model significantly impedes the cost-effective deployment of LLMs. Not only is a large model difficult to fit in commercial devices, but the serving cost of the model also increases sharply with the model size. To this end, quantization have been widely adopted to increase the inference efficiency of LLMs by employing lower precision data types.

Recently, weight-only quantization (Frantar et al., 2023; Lin et al., 2024) has emerged as a particularly attractive methodology due to its high compression ratio and effective preservation of model quality. By quantizing the LLM weights into low-precision but retaining difficult-to-quantize activations in full precision (FP), memory pressure is effectively relieved while reducing the accuracy degradation. Low-bit weight-only quantization is obtained either via quantization-aware training (QAT) or post-training quantization (PTQ). Although Liu et al. (2025) shows that QAT is capable of restoring the degraded accuracy even under sub-4-bit settings, the computational resource required to conduct QAT makes it prohibitively memory-intensive and time-consuming. On the other hand, layer-wise PTQ can be conducted with a relatively small amount of resources, but suffers from model quality degradation.

Block-wise PTQ (Lee et al., 2023; Shao et al., 2024; Cheng et al., 2024; Lee et al., 2025b; Chen et al., 2025; Park et al., 2025) strikes an effective balance between the two ends, achieving effective and efficient degradation recovery. By minimizing the mean squared error (MSE) between the outputs of an FP Transformer block and those of its quantized counterpart, cross-layer dependencies within each transformer block are accounted for, recovering the performance comparable to FP baselines on tasks such as language modeling (e.g. WikiText2 (Merity et al., 2016)) and general natural language understanding (e.g. MMLU (Hendrycks et al., 2021)).

However, relatively little attention has been shed on the degradation of *generation* quality caused by the low-bit quantization of LLMs. It is particularly alarming considering the increasing trend towards generating longer responses for increased task performance. Emerging large reasoning models (DeepSeek-AI et al., 2025; Aggarwal & Welleck, 2025) scale inference-time compute to produce extended chains of thought, thereby achieving higher accuracy on complex multi-step reasoning tasks. As this trend toward generating more tokens continues in pursuit of increased accuracy, serving costs rise sharply, necessitating a strong demand for an efficient quantization method that can maintain the generation quality of FP baselines.

In this work, we uncover that the standard block-wise PTQ approach—while effective at language modeling and understanding—suffers from the degradation of generation quality. The limitation is attributed to the fact that block-wise PTQ only preserves the quality of output activations of a transformer block, rather than preserving the fidelity of the next-token sampling *distribution*. Specifically, (i) existing block-wise PTQ methods completely ignore the unembedding layer (also known as the LM head), and (ii) rely on the MSE as optimization objective. Even when the MSE between the outputs of a quantized block and its FP counterpart is minimized, the actual probabilities assigned to plausible tokens can be perturbed, producing substantial shifts in distribution. Such misalignment is less observable on natural language understanding tasks, which does not involve autoregressive generation, but becomes pronounced in long-form generation as compounding probability distortions steer the generation trajectory away from the FP baseline.

Motivated by the observations, we propose *Logit-aware Final-block Quantization (LFQ)*, which enables low-bit quantized LLMs to achieve performance close to FP baselines on text generation tasks such as IFEval (Zhou et al., 2023), GSM8K (Cobbe et al., 2021), MATH500 (Lightman et al., 2023), and AIME (AIME, 2024). Unlike standard block-wise PTQ, LFQ quantizes the final Transformer block by minimizing the cross-entropy loss between the logits of the FP model and those of its quantized counterpart. Specifically, all Transformer blocks from the first to the penultimate are quantized by minimizing the MSE between the outputs of the FP and quantized blocks, while the final block is optimized using cross-entropy at the logit level, aligning token probabilities with the FP model and thereby reproducing the token prediction probabilities of the FP model. Thanks to its simple design, LFQ can be seamlessly applied to existing block-wise approaches. Moreover, LFQ consistently improves the generation quality of block-wise PTQ methods, while maintaining performance comparable to FP baselines on language modeling and understanding tasks.

Our contribution is threefold:

- To the best of our knowledge, we are the first to show that the conventional block-wise PTQ objective—minimizing MSE at intermediate outputs—does not align with reproducing the token predictions of FP models, thereby inducing non-negligible accuracy gaps between FP baselines and their low-bit quantized counterparts on text generation tasks.
- We propose Logit-aware Final-block Quantization (LFQ), which quantizes the final Transformer block by minimizing the cross-entropy between the logits of the FP model and its quantized counterpart, consistently improving the generation quality of low-bit quantized LLMs across existing block-wise PTQ methods.
- We validate LFQ across diverse models—including Llama 3.1, Qwen2.5, and large reasoning models (e.g., L1-Max, DeepSeek-R1-Distill)—on text generation benchmarks such as IFEval, GSM8K, MATH500, and AIME 2024. We further evaluate LFQ on WikiText2 and MMLU to ensure that it performs comparably to, and in some cases better than, existing block-wise PTQ techniques on language modeling and understanding tasks as well.

2 Problem Statement

Block-wise PTQ progressively quantizes each Transformer block by minimizing the mean squared error (MSE), from the first to the final block. In this section, we focus our attention on the final block, which is distinctive from the other blocks as it is directly attached to the LM head that produces the token sampling distribution. Below, we provide a brief overview of notations and assumptions used for illustrative purposes throughout this paper.

Let $\boldsymbol{W}_{\text{FP}}, \boldsymbol{W}_q \in \mathbb{R}^{c_{in} \times c_{out}}$ denote the full-precision (FP) final Transformer block and its quantized counterpart, and let $\boldsymbol{X} \in \mathbb{R}^{L \times c_{in}}$ represent the input to the final block, where L is the sequence length. Let $\mathcal{V}$ denote the vocabulary, with size $V = |\mathcal{V}|$. The LM head is then defined as $\boldsymbol{W}_{\text{Head}} \in \mathbb{R}^{c_{\text{out}} \times V}$. For illustrative purposes only, however, we restrict the vocabulary to $\mathcal{V} = \{t_1, t_2\}$, so that $\boldsymbol{W}_{\text{Head}} \in \mathbb{R}^{c_{\text{out}} \times 2}$. Unless otherwise specified, we omit the normalization layer between the final block and the LM head for simplicity.

First, to illustrate that minimizing the MSE between the outputs of a FP final Transformer block and its quantized counterpart can adversely affect the generation quality of low-bit quantized LLMs, we consider the case where $c_{\text{out}} = 2$. The final block is quantized by minimizing $\|\boldsymbol{X}\boldsymbol{W}_{\text{FP}} - \boldsymbol{X}\boldsymbol{W}_q\|_F^2$, yielding $\boldsymbol{X}\boldsymbol{W}_q = [0.7, 0.3]$ for $\boldsymbol{X}\boldsymbol{W}_{\text{FP}} = [0.8, 0.2]$ as an example. However, it is worth noting the following example:

$$\text{When } \boldsymbol{W}_{\text{Head}} = \begin{bmatrix} 0.5 & 0.3 \\ 0.5 & 1.0 \end{bmatrix}, \ \boldsymbol{X}\boldsymbol{W}_{\text{FP}}\boldsymbol{W}_{\text{Head}} = [\mathbf{0.5}, 0.44] \text{ and } \boldsymbol{X}\boldsymbol{W}_q\boldsymbol{W}_{\text{Head}} = [0.5, \mathbf{0.51}]\,.$$

This result implies that the FP model predicts token t_1, while its quantized counterpart instead predicts the opposite token, t_2. Consequently, even if the final block is quantized to minimize the MSE between $\boldsymbol{X}\boldsymbol{W}_{\text{FP}}$ and $\boldsymbol{X}\boldsymbol{W}_q$, ignoring the LM head during block-wise quantization can lead the quantized model to produce different token predictions from the FP model.

Next, even when the LM head is considered, minimizing the MSE between the logits of the FP and quantized models does not guarantee identical token predictions. For example, suppose we obtain

$$\boldsymbol{X}\boldsymbol{W}_q\boldsymbol{W}_{\text{Head}} = \begin{cases} \text{(i)} \ \left[0.4, \mathbf{0.6}\right] \ \text{for } \boldsymbol{X}\boldsymbol{W}_{\text{FP}}\boldsymbol{W}_{\text{Head}} = \left[\mathbf{0.6}, 0.4\right] \\ \text{(ii)} \ \left[\mathbf{0.6}, 0.4\right] \ \text{for } \boldsymbol{X}\boldsymbol{W}_{\text{FP}}\boldsymbol{W}_{\text{Head}} = \left[\mathbf{0.9}, 0.1\right] \end{cases}. \tag{1}$$

Then, the corresponding MSE values are given by

$$\|\boldsymbol{X}\boldsymbol{W}_{\text{FP}}\boldsymbol{W}_{\text{Head}} - \boldsymbol{X}\boldsymbol{W}_q\boldsymbol{W}_{\text{Head}}\|_F^2 = \begin{cases} \text{(i)} \ (\mathbf{0.6} - 0.4)^2 + (0.4 - \mathbf{0.6})^2 = 0.08 \\ \text{(ii)} \ (\mathbf{0.9} - \mathbf{0.6})^2 + (0.1 - 0.4)^2 = 0.18 \end{cases}.$$

Although the first case (i) yields the smaller MSE, it leads to the opposite token prediction, whereas the second (ii)—despite having the larger MSE—produces the same token prediction as the FP model. This therefore demonstrates that minimizing MSE at the logit level does not necessarily align with reproducing the FP model's token predictions. Consistently, Figure 1 (a) illustrates that standard block-wise PTQ achieves a lower MSE yet predicts a different top-1 token than the FP model, leading to an incorrect reasoning trajectory.

3 METHOD

As discussed in Section 2, ensuring that low-bit quantized LLMs reproduce the token predictions of their full-precision (FP) counterparts requires explicitly accounting for the LM head and replacing mean squared error (MSE) in the optimization objective of block-wise post-training quantization (PTQ) methods. To this end, we propose *Logit-aware Final-block Quantization (LFQ)*, which quantizes the final Transformer block by minimizing the cross-entropy loss between the logits of the FP model and those of its quantized counterpart.

3.1 LOGIT-AWARE FINAL-BLOCK QUANTIZATION (LFQ)

Even when the LM head is taken into account, minimizing the MSE does not guarantee that the quantized model will predict the same token as the FP model. Since minimizing cross-entropy is equivalent to minimizing KL divergence, and KL divergence is equal to zero if and only if two distributions are identical, minimizing cross-entropy at the logit level directly encourages the quantized model's token-level distribution to match its FP counterpart. Furthermore, as Bruch (2021) demonstrates, cross-entropy can be used for learning to rank, which would also help the quantized model recover the FP model's top-k token ordering. Accordingly, when optimizing the quantized final Transformer block $\boldsymbol{W}_q$, we minimize the cross-entropy between the FP model's logits and those of the quantized model to align the block-wise PTQ objective with the FP model's token generation. Specifically, while the first through penultimate Transformer blocks are quantized by minimizing the MSE between the outputs of the FP and quantized blocks, the final block is quantized using the following optimization objective:

$$\min_{\boldsymbol{W}_q} \mathcal{L}_{\text{CE}}(\boldsymbol{X}\boldsymbol{W}_{\text{FP}}\boldsymbol{W}_{\text{Head}}, \boldsymbol{X}\boldsymbol{W}_q\boldsymbol{W}_{\text{Head}}) = -\sum_{i=1}^{L}\sum_{j=1}^{V} \text{softmax}(\boldsymbol{X}\boldsymbol{W}_{\text{FP}}\boldsymbol{W}_{\text{Head}})_{i,j} \log(\boldsymbol{X}\boldsymbol{W}_q\boldsymbol{W}_{\text{Head}})_{i,j}, \tag{2}$$

where $\text{softmax}(\boldsymbol{Z}) = \frac{\exp(Z_{i,j})}{\sum_{k=1}^{V}\exp(Z_{i,k})}$ for $\boldsymbol{Z} = [Z_{i,j}]_{i=1,j=1}^{L,V}$. We refer to Eq. 2 as *"LFQ."*

The specific quantization parameters contained in $\boldsymbol{W}_q$ depend on the chosen block-wise PTQ method. For example, when instantiating (1) FlexRound (Lee et al., 2023), (2) OmniQuant (Shao et al., 2024), or (3) Block-AP (Chen et al., 2025), Eq. 2 specializes accordingly as:

$$\text{(1) FlexRound: } \boldsymbol{W}_q = \boldsymbol{s}_1\left\lfloor \frac{\boldsymbol{W}_{\text{FP}}}{\boldsymbol{s}_1 \odot \boldsymbol{S}_2 \odot \boldsymbol{s}_3} \right\rceil \text{ where } \boldsymbol{s}_1, \boldsymbol{s}_3 \in \mathbb{R}_{>0}^{c_{out}\times\frac{c_{in}}{g}}, \text{ and } \boldsymbol{S}_2 \in \mathbb{R}_{>0}^{c_{out}\times c_{in}}, \tag{3}$$

$$\Rightarrow \text{Eq. 2: } \min_{\boldsymbol{W}_q} \mathcal{L}_{\text{CE}}(\boldsymbol{X}\boldsymbol{W}_{\text{FP}}\boldsymbol{W}_{\text{Head}}, \boldsymbol{X}\boldsymbol{W}_q\boldsymbol{W}_{\text{Head}}) = \min_{\boldsymbol{s}_1,\boldsymbol{S}_2,\boldsymbol{s}_3} \mathcal{L}_{\text{CE}}(\boldsymbol{X}\boldsymbol{W}_{\text{FP}}\boldsymbol{W}_{\text{Head}}, \boldsymbol{X}\boldsymbol{W}_q\boldsymbol{W}_{\text{Head}}).$$

$$\text{(2) OmniQuant: } \boldsymbol{W}_q = \boldsymbol{h}\left\lfloor \frac{\boldsymbol{W}_{\text{FP}}}{\boldsymbol{h}} \right\rceil \text{ where } \boldsymbol{h} = \frac{\boldsymbol{\gamma}\max(\boldsymbol{W}_{\text{FP}}) - \boldsymbol{\beta}\min(\boldsymbol{W}_{\text{FP}})}{2^b - 1} \text{ with } \boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}_{[0,1]}^{c_{out}\times\frac{c_{in}}{g}}, \tag{4}$$

$$\Rightarrow \text{Eq. 2: } \min_{\boldsymbol{W}_q} \mathcal{L}_{\text{CE}}(\boldsymbol{X}\boldsymbol{W}_{\text{FP}}\boldsymbol{W}_{\text{Head}}, \boldsymbol{X}\boldsymbol{W}_q\boldsymbol{W}_{\text{Head}}) = \min_{\boldsymbol{\gamma},\boldsymbol{\beta}} \mathcal{L}_{\text{CE}}(\boldsymbol{X}\boldsymbol{W}_{\text{FP}}\boldsymbol{W}_{\text{Head}}, \boldsymbol{X}\boldsymbol{W}_q\boldsymbol{W}_{\text{Head}}).$$

$$\text{(3) Block-AP: } \boldsymbol{W}_q = \boldsymbol{s}\left\lfloor \frac{\boldsymbol{W}_{\text{FP}}}{\boldsymbol{s}} \right\rceil \text{ where } \boldsymbol{s} \in \mathbb{R}_{>0}^{c_{out}\times\frac{c_{in}}{g}}, \tag{5}$$

$$\Rightarrow \text{Eq. 2: } \min_{\boldsymbol{W}_q} \mathcal{L}_{\text{CE}}(\boldsymbol{X}\boldsymbol{W}_{\text{FP}}\boldsymbol{W}_{\text{Head}}, \boldsymbol{X}\boldsymbol{W}_q\boldsymbol{W}_{\text{Head}}) = \min_{\boldsymbol{s},\boldsymbol{W}_{\text{FP}}} \mathcal{L}_{\text{CE}}(\boldsymbol{X}\boldsymbol{W}_{\text{FP}}\boldsymbol{W}_{\text{Head}}, \boldsymbol{X}\boldsymbol{W}_q\boldsymbol{W}_{\text{Head}}).$$

Here, b denotes the low bit-width and g the group size ($g = c_{\text{in}}$ for per-channel quantization, and $g = 128$ for group-wise quantization). Hereafter, we refer to Eq. 3, Eq. 4, and Eq. 5 as "FlexRound+LFQ", "OmniQuant+LFQ", and "Block-AP+LFQ".

Three points are worth highlighting here. First, since LFQ integrates the LM Head and cross-entropy into the loss objective of standard block-wise PTQ, it is agnostic to the underlying block-wise method and thus can be applied seamlessly. Second, because LFQ optimizes only the final Transformer block by minimizing cross-entropy at the logit level, it is memory-efficient and thus can be run on a single GPU like other block-wise PTQ techniques. Third, because LFQ modifies only the optimization objective and leaves the quantization scheme unchanged, LFQ-quantized LLMs remain fully compatible with existing packing/unpacking kernels (e.g., Frantar et al. (2023); Lin et al. (2024); Park et al. (2024)) and can therefore be accelerated without additional effort.

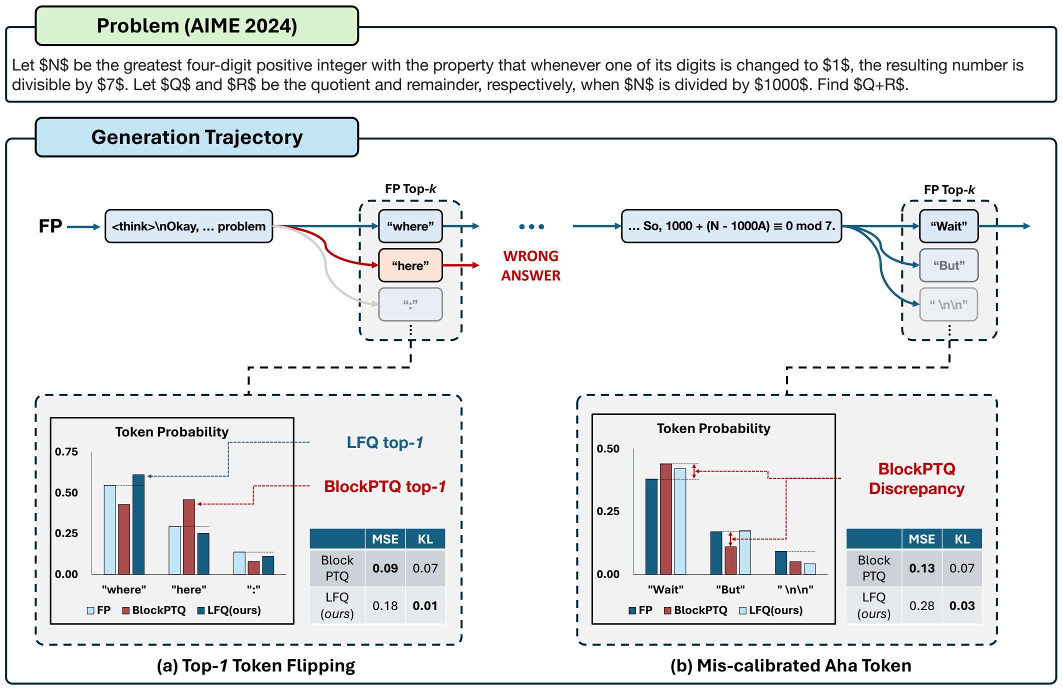
3.2 Effect of LFQ on Token Generation

To illustrate that cross-entropy better reproduces the FP model's token predictions than MSE, we revisit the example 1 in Section 2. The corresponding cross-entropy values are given as follows:

$$-\sum_{i,j=1}^{L,V} \text{softmax}(\mathbf{X}\mathbf{W}_{\text{FP}}\mathbf{W}_{\text{Head}})_{i,j} \log(\mathbf{X}\mathbf{W}_q\mathbf{W}_{\text{Head}})_{i,j} = \begin{cases} \text{(i)} \ -\mathbf{0.6}\log(0.4) - 0.4\log(\mathbf{0.6}) \approx 0.75 \\ \text{(ii)} \ -\mathbf{0.9}\log(\mathbf{0.6}) - 0.1\log(0.4) \approx 0.55 \end{cases}$$

In contrast to the MSE—whose value in case (i) is smaller than in case (ii), even though case (i) predicts the opposite token while case (ii) predicts the same token as the FP model—the cross-entropy assigns a smaller value to case (ii) than to case (i). This observation highlights that minimizing the cross-entropy loss at the logit level is essential for guiding low-bit quantized LLMs to align with the FP model's token predictions.

To make this trend concrete, Figure 1 presents a reasoning trajectory generated by L1-Qwen-7B-Max for Problem 28 of AIME 2024 and compares token-level probability distributions for the FP



Let N be the greatest four-digit positive integer with the property that whenever one of its digits is changed to 1, the resulting number is divisible by 7. Let Q and R be the quotient and remainder, respectively, when N is divided by 1000. Find $Q+R$.

(a)

	MSE	KL
Block PTQ	**0.09**	0.07
LFQ (*ours*)	0.18	**0.01**

(b)

	MSE	KL
Block PTQ	**0.13**	0.07
LFQ (*ours*)	0.28	**0.03**

Figure 1: Reasoning trajectory of L1-Qwen-7B-Max under greedy decoding on AIME 2024 Problem 28. We compare token-level probability distributions for the FP baseline, block-wise PTQ ("blockPTQ" in the figure), and LFQ (ours) at two instants: (a) the first step where block-wise PTQ's top-1 token diverges from the FP baseline, and (b) the first "aha" moment guiding the reasoning onto the correct path. In (a), block-wise PTQ's top-1 (`"here"`) corresponds to the FP baseline's top-2, yielding an incorrect answer, whereas LFQ's top-1 (`"where"`) matches the FP baseline's top-1 and thus reaches the correct answer. In (b), block-wise PTQ is overconfident in (`"Wait"`) and underconfident in the subsequent "aha" token, (`"But"`), while LFQ assigns probabilities to these "aha" tokens that remain closer to the FP baseline.

baseline, block-wise PTQ, and LFQ at two key points: (a) the first instance where block-wise PTQ's top-1 token diverges from the FP baseline, and (b) the first "aha" moment that steers the reasoning onto the correct path. In Figure 1 (a), although block-wise PTQ attains a smaller MSE than LFQ, thanks to minimizing cross-entropy at the logit level, LFQ yields a smaller KL divergence from the FP distribution. Consequently, LFQ reproduces the FP model's top-1 token prediction (i.e., `"where"`) and follows the correct trajectory to the right answer, whereas block-wise PTQ diverges and thus fails to solve the problem.

Moreover, because the occurrence of an "aha" moment is pivotal for re-evaluating and correcting an ongoing reasoning trajectory, the extent to which low-bit quantized models track the FP baseline on such "aha" tokens—e.g., `"Wait"` and `"But"`—is a key determinant of their accuracy on complex reasoning benchmarks. As shown in Figure 1 (b), even when the top-1 token for all three models is `"Wait"`, it is noteworthy that block-wise PTQ is overconfident in `"Wait"`, leaving it underconfident in another "aha" token like `"But"`. By contrast, LFQ allocates these probabilities closer to the FP baseline, resulting in not only a smaller KL divergence but also higher accuracy than block-wise PTQ as reported in Table 2.

4 EXPERIMENT

In this section, we verify the effectiveness of LFQ on Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct (Qwen et al., 2025) under 4-bit per-channel and 3-bit group-wise weight-only quantization settings on IFEval and MATH500 (Lightman et al., 2023). We also evaluate LFQ on large reasoning models—L1-Qwen-7B-Max (Aggarwal & Welleck, 2025) and DeepSeek-R1-Distill-Llama-

Table 1: Performance of Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct with LFQ under block-wise PTQ (FlexRound, OmniQuant, and Block-AP). Within each PTQ method, the best accuracy is shown in **bold**. “W4” and “W3g128” denote 4-bit per-channel weight-only quantization and 3-bit group-wise quantization (group size 128), respectively. LFQ yields consistent gains in generation quality across block-wise PTQ, while preserving the language modeling and understanding performance of existing methods.

		Language Modeling/Understanding		Text Generation	
Method	# Bits	WikiText2 (↓)	MMLU (↑)	IFEval (↑) (greedy)	MATH500 (↑) (greedy)
Qwen2.5-7B-Instruct	BF16	6.85	73.49	70.79	74.2
FlexRound	W4	7.23	**72.50**	69.50	72.6
FlexRound+LFQ (Ours)	W4	**7.21**	72.48	**71.35**	**73.4**
FlexRound	W3g128	7.63	70.13	66.54	65.6
FlexRound+LFQ (Ours)	W3g128	**7.58**	**70.26**	**67.84**	**68.0**
OmniQuant	W4	7.73	**71.00**	68.21	69.8
OmniQuant+LFQ (Ours)	W4	**7.53**	70.99	**69.50**	**71.6**
OmniQuant	W3g128	8.08	**68.43**	68.21	63.6
OmniQuant+LFQ (Ours)	W3g128	**7.91**	68.39	**68.58**	**64.4**
Block-AP	W4	7.87	69.60	66.73	68.0
Block-AP+LFQ (Ours)	W4	**7.77**	**69.94**	**68.02**	**69.0**
Block-AP	W3g128	8.70	**67.09**	61.00	60.0
Block-AP+LFQ (Ours)	W3g128	**8.18**	67.06	**63.77**	**61.8**
Qwen2.5-14B-Instruct	BF16	5.24	78.82	79.85	78.4
FlexRound	W4	5.67	**77.33**	77.82	76.4
FlexRound+LFQ (Ours)	W4	**5.62**	77.31	**78.00**	**77.2**
FlexRound	W3g128	6.15	75.84	75.05	69.6
FlexRound+LFQ (Ours)	W3g128	**6.11**	**75.85**	**77.08**	**71.6**
OmniQuant	W4	5.93	76.64	73.94	73.4
OmniQuant+LFQ (Ours)	W4	**5.89**	**76.66**	**75.23**	**75.2**
OmniQuant	W3g128	6.43	75.62	74.31	**70.4**
OmniQuant+LFQ (Ours)	W3g128	**6.36**	**75.73**	**75.42**	69.8
Block-AP	W4	6.23	76.84	70.79	71.6
Block-AP+LFQ (Ours)	W4	**6.17**	**76.86**	**72.27**	**72.4**
Block-AP	W3g128	6.81	74.58	71.72	67.0
Block-AP+LFQ (Ours)	W3g128	**6.69**	**74.58**	**72.46**	**68.0**

8B (DeepSeek-AI et al., 2025)—using MATH500 and AIME 2024 (AIME, 2024) (AIME′24 for short). Finally, we empirically (i) demonstrate the importance of incorporating the LM head and utilizing cross-entropy in the objective, (ii) validate that quantizing only the final Transformer block via logit-level cross-entropy is sufficient (i.e., it is not a must to quantize multiple final blocks with cross-entropy), and (iii) the comparison of LFQ against LoRA-based quantization error compensation (LQEC). These findings are established on Llama 3.1 8B Instruct (Grattafiori et al., 2024) using IFEval (Zhou et al., 2023) and GSM8K (Cobbe et al., 2021) under 4-bit per-channel weight-only quantization. Unless otherwise noted, we use group-wise quantization with a group size of 128.

We randomly select calibration sequences of length 2048 tokens from the C4 training set (Raffel et al., 2020) for all experiments. We do so to emphasize that LFQ can preserve performance comparable to FP baselines on language modeling (e.g., WikiText-2 (Merity et al., 2016)) and understanding (MMLU (Hendrycks et al., 2021)), while consistently improving the generation quality of block-wise PTQ methods. We report perplexity on WikiText2 using a sequence length of 4096, five-shot accuracy on MMLU, prompt-level strict-accuracy on IFEval (following Qwen et al. (2025)), 8-shot accuracy on GSM8K, and zero-shot accuracy on MATH500 and AIME′24. For generation tasks, we use greedy decoding to ensure a fair comparison between quantized models with and with-

Table 2: Performance of L1-Max-Qwen-7B and DeepSeek-R1-Distill-Llama-8B with LFQ under block-wise PTQ (FlexRound). Within the PTQ method, the best accuracy is shown in **bold**. "W4" and "W3g128" denote 4-bit per-channel weight-only quantization and 3-bit group-wise quantization (group size 128), respectively. LFQ yields consistent gains in generation quality across block-wise PTQ, while preserving the language modeling and understanding performance of existing methods.

		Language Modeling/Understanding		Text Generation		
Method	# Bits	WikiText2 (↓)	MMLU (↑)	MATH500 (↑) (greedy)	AIME′24 (↑) (greedy)	AIME′24 (↑) (pass@8)
L1-Qwen-7B-Max	BF16	29.57	54.58	88.0	46.67	55.30
FlexRound	W4	31.20	**53.43**	86.0	30.00	51.71
FlexRound+LFQ (Ours)	W4	**30.44**	53.10	**87.6**	**43.33**	**55.09**
FlexRound	W3g128	31.45	52.24	85.2	23.33	41.85
FlexRound+LFQ (Ours)	W3g128	**29.46**	**52.53**	**86.4**	**30.00**	**45.18**
DeepSeek-R1-Distill-Llama-8B	BF16	11.85	55.69	70.4	30.00	30.49
FlexRound	W4	12.61	**54.57**	68.2	16.67	27.71
FlexRound+LFQ (Ours)	W4	**12.46**	54.21	**69.8**	**26.67**	**30.07**
FlexRound	W3g128	13.80	53.61	62.8	10.00	15.98
FlexRound+LFQ (Ours)	W3g128	**13.24**	**54.03**	**67.2**	**13.33**	**16.97**

out LFQ. For AIME′24, to estimate pass@8 as well, we additionally use temperature 0.6 and top-p 0.95, and sample 16 responses per question with a maximum generation length of 4096 tokens.

4.1 Qwen2.5 on IFEval and MATH500

To assess whether LFQ can improve low-bit instruction-tuned LLMs on both natural language instruction following and challenging math word problems, we evaluate LFQ for Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct on IFEval and MATH500 using greedy decoding. Table 1 shows that, across different quantization configurations, LFQ consistently improves the generation quality of instruction-tuned models quantized by FlexRound, OmniQuant, and Block-AP on both IFEval and MATH500. Consequently, FlexRound+LFQ narrows the gap between 4-bit per-channel models and their FP baselines to within 1 percentage point (pp) for Qwen2.5-7B-Instruct and within 2 pp for Qwen2.5-14B-Instruct across all benchmarks considered (MMLU, IFEval, and MATH500).

4.2 Large Reasoning Models on MATH500 and AIME 2024

To test whether LFQ can also perform well for large reasoning models that produce long chains of thought by scaling test-time compute, we apply LFQ to L1-Qwen-7B-Max and DeepSeek-R1-Distill-Llama-8B on MATH500 and AIME′24. Given that Table 1 identifies FlexRound+LFQ as the most effective among FlexRound+LFQ, OmniQuant+LFQ, and Block-AP+LFQ, we focus on FlexRound+LFQ here. Table 2 shows that standard block-wise PTQ suffers substantial degradation under greedy decoding on AIME′24. In contrast, LFQ nearly matches the FP baseline on AIME′24, indicating that it restores alignment with the FP model's top-1 token predictions. Furthermore, LFQ raises pass@8 to within 0.5 percentage points of the FP baselines. Taken together, these results suggest that LFQ effectively aligns the token-level probabilities of low-bit quantized LLMs with those of their FP counterparts.

4.3 Ablation Study

Importance of LM Head and cross-entropy. To assess the impact of incorporating the LM head and using cross-entropy in the loss objective when quantizing the final block, we incrementally augment existing block-wise PTQ methods (FlexRound, OmniQuant, and Block-AP) with these components. As shown in Table 3, adding the LM head alone generally improves accuracy on text generation benchmarks (IFEval and GSM8K) as well as on language modeling and understanding. With the LM head in place, employing cross-entropy rather than mean squared error (MSE) yields further gains on text generation tasks. We therefore conclude that leveraging both the LM head and cross-entropy, as in Eq. 2, is essential for boosting the generation quality of low-bit quantized LLMs.

Table 3: Performance of Llama 3.1 8B Instruct when block-wise PTQ methods (FlexRound, OmniQuant, and Block-AP) are incrementally augmented by (i) incorporating the LM head and (ii) using a logit-level cross-entropy objective in order to quantize the final Transformer block. Within each block-wise PTQ method, the best accuracy is shown in **bold** and the second-best is underlined. Here, all results use 4-bit per-channel weight-only quantization. LFQ (with both LM Head and cross-entropy, ours) yields consistent gains in generation quality across block-wise PTQ, while preserving the language modeling and understanding performance of existing methods.

			Language Modeling/Understanding		Text Generation	
Method	LM-Head	Cross-Entropy	WikiText2 (↓)	MMLU (↑)	IFEval (↑) (greedy)	GSM8K (↑) (greedy)
Llama 3.1 8B Instruct	N/A	N/A	6.75	68.34	74.49	84.99
FlexRound	X	X	<u>7.06</u>	66.19	70.24	81.35
FlexRound+LFQ	O	X	7.08	<u>66.75</u>	<u>71.53</u>	<u>81.58</u>
	O	O	**7.06**	**66.97**	**72.09**	**81.80**
OmniQuant	X	X	7.49	<u>64.87</u>	70.61	78.17
OmniQuant+LFQ	O	X	<u>7.48</u>	64.77	<u>71.35</u>	<u>78.32</u>
	O	O	**7.47**	**65.48**	**71.35**	**79.76**
Block-AP	X	X	7.76	63.24	<u>68.58</u>	73.84
Block-AP+LFQ	O	X	<u>7.74</u>	<u>63.54</u>	68.39	<u>74.00</u>
	O	O	**7.69**	**63.77**	**68.76**	**74.45**

Sufficiency of quantizing solely the final block via logit-level cross-entropy. We ask whether applying the logit-level cross-entropy objective to only the final block is sufficient. To test this, we vary the number of topmost Transformer blocks optimized with LFQ (denoted as k) while keeping the remaining blocks quantized via standard MSE reconstruction. For example, when $k = 2$, we apply LFQ sequentially to the penultimate and final blocks. As shown in Figure 2, the average score of IFEval and GSM8K remains almost constant even as k increases; $k = 2$ occasionally yields a marginal gain on that average but at the cost of lower MMLU accuracy. These results indicate that applying LFQ to the final block alone is sufficient and offers the best overall trade-off.

Comparison of LFQ against LQEC. As LQEC has emerged as a promising approach for mitigating memory bottleneck while recovering task accuracy, we compare LFQ with RILQ (Lee et al., 2025a), a state-of-the-art LQEC method. For a fair comparison, we use only the C4 training set as calibration data to initialize LoRA adapters on low-bit quantized models produced by FlexRound, OmniQuant, and Block-AP. Table 4 reports the results. RILQ generally outperforms LFQ on language modeling (WikiText2) and language understanding (MMLU) due to its use of LoRA adapters. Nevertheless, LFQ consistently surpasses RILQ on text generation across all settings. We hypothesize that this stems from the fact that LQEC methods—including RILQ—optimizes MSE, an objective misaligned with matching the FP model's token-level distribution (as elucidated in Section 2).

5 Related Work

Quantization study is typically classified into quantization-aware training (QAT) and post-training quantization (PTQ). As it is well known that QAT can match full-precision (FP) accuracy even under sub-4-bit quantization configurations, it has been applied across domains—from computer vision models (Esser et al., 2020; Lee et al., 2021) to natural language models (Liu et al., 2023; 2025). However, Liu et al. (2025) shows that QAT requires fine-tuning large language models (LLMs) on billions of tokens at least, which is prohibitively memory-intensive and time-consuming. Consequently, research attention has continued to focus more on advancing PTQ.

PTQ is commonly divided into layer-wise and block-wise methods. Layer-wise PTQ (e.g., Frantar et al. (2023); Lin et al. (2024)) can be run fast on a single GPU and typically incurs marginal performance degradation on relatively easy downstream tasks (e.g., commonsense reasoning). However, as these techniques do not involve gradient-based optimization, unless task-specific calibration data

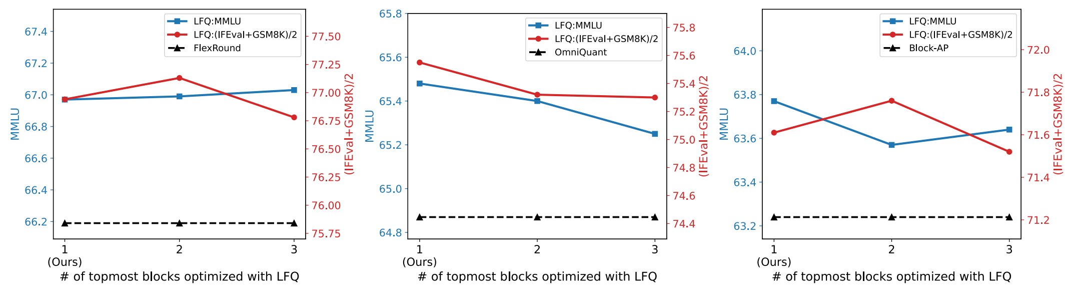


Figure 2: Performance of Llama 3.1 8B Instruct as the number of topmost Transformer blocks optimized with LFQ increases from 1 (ours) to 3, with the remaining blocks quantized via standard MSE reconstruction; shown for FlexRound (left), OmniQuant (center), and Block-AP (right). In each subfigure, the left y-axis shows MMLU accuracy, while the right y-axis reports the IFEval+GSM8K average. All results use 4-bit per-channel weight-only quantization. The average of IFEval and GSM8K (expressed as "(IFEval+GSM8K)/2") stays roughly unchanged, regardless of the number of topmost Transformer blocks optimized with LFQ.

Table 4: Comparison of LFQ (ours) against RILQ, a state-of-the-art LoRA-based quantization error compensation method, on Llama 3.1 8B Instruct using block-wise PTQ (FlexRound, OmniQuant, and Block-AP). Within each PTQ method, the best accuracy is shown in **bold** and the second best is underlined. "W4" denotes 4-bit per-channel weight-only quantization.

		Language modeling/understanding		Text generation	
Method	# Bits	WikiText2 (↓)	MMLU (↑)	IFEval (↑) (greedy)	GSM8K (↑) (greedy)
Llama 3.1 8B Instruct	BF16	6.75	68.34	74.49	84.99
FlexRound	W4	7.06	66.19	70.24	81.35
FlexRound+RILQ	W4	**6.95**	66.86	71.90	80.52
FlexRound+LFQ	W4	7.06	**66.97**	**72.09**	**81.80**
OmniQuant	W4	7.49	64.87	70.61	78.17
OmniQuant+RILQ	W4	**7.24**	**66.07**	71.35	78.85
OmniQuant+LFQ	W4	7.47	65.48	**71.35**	**79.76**
Block-AP	W4	7.76	63.24	68.58	73.84
Block-AP+RILQ	W4	**7.43**	**64.62**	68.58	73.92
Block-AP+LFQ	W4	7.69	63.77	**68.76**	**74.45**

is utilized, they can suffer substantial accuracy degradation on more challenging benchmarks, particularly text generation (Li et al., 2025). On the other hand, block-wise PTQ approaches (Lee et al., 2023; Shao et al., 2024; Cheng et al., 2024; Lee et al., 2025b; Chen et al., 2025) not only account for cross-layer dependencies within a block but also optimize quantization parameters via gradient-based iterative updates, and therefore often outperform layer-wise PTQ. Notwithstanding, we find that existing block-wise PTQ can still exhibit non-negligible degradation in generation quality.

6 Conclusion

We show that block-wise PTQ can degrade generation quality due to (i) omitting the LM head from block-wise optimization and (ii) relying on the MSE objective. To address this, we introduce Logit-aware Final-block Quantization (LFQ), which quantizes the final Transformer block by aligning the quantized model's logits to the FP model's via the cross-entropy loss. Across diverse model families and generation tasks, LFQ consistently improves generation quality over existing block-wise PTQ techniques, while preserving performance on language modeling and understanding.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.04697.

AIME. Aime problems and solutions, 2024, 2024. URL https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions.

Sebastian Bruch. An alternative cross entropy loss for learning-to-rank. In *Proceedings of the Web Conference 2021*, WWW '21, pp. 118–126, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383127. doi: 10.1145/3442381.3449794. URL https://doi.org/10.1145/3442381.3449794.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. EfficientQAT: Efficient quantization-aware training for large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10081–10100, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.498. URL https://aclanthology.org/2025.acl-long.498/.

Wenhua Cheng, Weiwei Zhang, Haihao Shen, Yiyang Cai, Xin He, Lv Kaokao, and Yi Liu. Optimize weight rounding via signed gradient descent for the quantization of LLMs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 11332–11350, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.662. URL https://aclanthology.org/2024.findings-emnlp.662/.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization, 2020. URL `https://arxiv.org/abs/1902.08153`.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers, 2023. URL `https://arxiv.org/abs/2210.17323`.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

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

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Geonho Lee, Janghwan Lee, Sukjin Hong, Minsoo Kim, Euijai Ahn, Du-Seong Chang, and Jungwook Choi. Rilq: Rank-insensitive lora-based quantization error compensation for boosting 2-bit large language model accuracy, 2025a. URL `https://arxiv.org/abs/2412.01129`.

Jung Hyun Lee, Jihun Yun, Sung Ju Hwang, and Eunho Yang. Cluster-promoting quantization with bit-drop for minimizing network quantization loss, 2021. URL `https://arxiv.org/abs/2109.02100`.

Jung Hyun Lee, Jeonghoon Kim, Se Jung Kwon, and Dongsoo Lee. FlexRound: Learnable rounding based on element-wise division for post-training quantization. In Andreas Krause, Emma

Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 18913–18939. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/lee23h.html`.

Jung Hyun Lee, Jeonghoon Kim, June Yong Yang, Se Jung Kwon, Eunho Yang, Kang Min Yoo, and Dongsoo Lee. LRQ: Optimizing post-training quantization for large language models by learning low-rank weight-scaling matrices. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7708–7743, Albuquerque, New Mexico, April 2025b. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.393. URL `https://aclanthology.org/2025.naacl-long.393/`.

Zhen Li, Yupeng Su, Runming Yang, Congkai Xie, Zheng Wang, Zhongwei Xie, Ngai Wong, and Hongxia Yang. Quantization meets reasoning: Exploring llm low-bit quantization degradation for mathematical reasoning, 2025. URL `https://arxiv.org/abs/2501.03035`.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL `https://arxiv.org/abs/2305.20050`.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration, 2024. URL `https://arxiv.org/abs/2306.00978`.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models, 2023. URL `https://arxiv.org/abs/2305.17888`.

Zechun Liu, Changsheng Zhao, Hanxian Huang, Sijia Chen, Jing Zhang, Jiawei Zhao, Scott Roy, Lisa Jin, Yunyang Xiong, Yangyang Shi, Lin Xiao, Yuandong Tian, Bilge Soran, Raghuraman Krishnamoorthi, Tijmen Blankevoort, and Vikas Chandra. Paretoq: Scaling laws in extremely low-bit llm quantization, 2025. URL `https://arxiv.org/abs/2502.02631`.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Gunho Park, Baeseong Park, Minsub Kim, Sungjae Lee, Jeonghoon Kim, Beomseok Kwon, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models. In *ICLR*, 2024. URL `https://openreview.net/forum?id=gLARhFLE0F`.

Seungcheol Park, Jeongin Bae, Beomseok Kwon, Minjun Kim, Byeongwook Kim, Se Jung Kwon, U Kang, and Dongsoo Lee. Unifying uniform and binary-coding quantization for accurate compression of large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28468–28488, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1382. URL `https://aclanthology.org/2025.acl-long.1382/`.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8Wuvhh0LYW.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao, Hongcheng Gao, Peizhong Gao, Tong Gao, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang Guo, Hao Hu, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Chao Hong, Yangyang Hu, Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yanhao Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe Lu, Lijun Lu, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi, Feifan Song, Jianlin Su, Zhengyuan Su, Xinjie Sun, Flood Sung, Heyi Tang, Jiawen Tao, Qifeng Teng, Chensi Wang, Dinglu Wang, Feng Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang, Jinhong Wang, Shengjie Wang, Shuyi Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang, Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Wenhao Wu, Xingzhe Wu, Yuxin Wu, Chenjun Xiao, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Xiaofei Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang Yuan, Mengjie Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang, Yangkun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Haotian Zhao, Yikai Zhao, Huabin Zheng, Shaojie Zheng, Jianren Zhou, Xinyu Zhou, Zaida Zhou, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. Kimi k2: Open agentic intelligence, 2025. URL https://arxiv.org/abs/2507.20534.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

A Additional Ablation Studies of LFQ for Qwen2.5-7B-Instruct and Llama 3.2 3B Instruct

Table 5: Performance of Qwen2.5-7B-Instruct when block-wise PTQ methods (FlexRound, OmniQuant, and Block-AP) are incrementally augmented by (i) incorporating the LM head and (ii) using a logit-level cross-entropy objective in order to quantize the final Transformer block. Within each block-wise PTQ method, the best accuracy is shown in **bold** and the second-best is underlined. Here, all results use 4-bit per-channel weight-only quantization. LFQ (with both LM Head and cross-entropy, ours) yields consistent gains in generation quality across block-wise PTQ, while preserving the language modeling and understanding performance of existing methods.

			Language Modeling/Understanding		Text Generation	
Method	LM-Head	Cross-Entropy	WikiText2 (↓)	MMLU (↑)	IFEval (↑) (greedy)	MATH500 (↑) (greedy)
Qwen2.5-7B-Instruct	N/A	N/A	6.85	73.49	70.79	74.2
FlexRound	X	X	7.23	**72.50**	69.50	72.6
FlexRound+LFQ	O	X	7.26	72.48	71.35	71.4
	O	O	**7.21**	72.48	**71.35**	**73.4**
OmniQuant	X	X	7.73	71.00	68.21	69.8
OmniQuant+LFQ	O	X	**7.29**	**71.02**	68.95	70.6
	O	O	7.53	70.99	**69.50**	**71.6**
Block-AP	X	X	7.87	69.60	66.73	68.0
Block-AP+LFQ	O	X	7.92	69.75	67.28	68.4
	O	O	**7.77**	**69.94**	**68.02**	**69.0**

Table 6: Performance of Llama 3.2 3B Instruct when block-wise PTQ methods (FlexRound, OmniQuant, and Block-AP) are incrementally augmented by (i) incorporating the LM head and (ii) using a logit-level cross-entropy objective in order to quantize the final Transformer block. Within each block-wise PTQ method, the best accuracy is shown in **bold** and the second-best is underlined. Here, all results use 4-bit per-channel weight-only quantization. LFQ (with both LM Head and cross-entropy, ours) yields consistent gains in generation quality across block-wise PTQ, while preserving the language modeling and understanding performance of existing methods.

			Language Modeling/Understanding		Text Generation	
Method	LM-Head	Cross-Entropy	WikiText2 (↓)	MMLU (↑)	IFEval (↑) (greedy)	GSM8K (↑) (greedy)
Llama 3.2 3B Instruct	N/A	N/A	10.14	61.34	71.72	77.48
FlexRound	X	X	10.72	**59.93**	65.80	72.40
FlexRound+LFQ	O	X	10.73	59.66	65.99	**73.09**
	O	O	**10.72**	59.84	**67.28**	73.01
OmniQuant	X	X	11.23	57.78	64.33	71.42
OmniQuant+LFQ	O	X	11.17	58.80	64.33	**71.65**
	O	O	**11.16**	**58.94**	**66.54**	71.49
Block-AP	X	X	11.61	**56.47**	62.29	66.11
Block-AP+LFQ	O	X	11.63	56.44	62.66	66.34
	O	O	**11.61**	56.38	**63.77**	**66.41**

B Experimental Setting of LFQ

We sweep the LFQ learning rate as follows: $\{5e-4, 1e-3\}$ with FlexRound; $\{1.5e-3, 2e-3, 5e-3\}$ with OmniQuant; and $\{1e-5, 2e-5, 3e-5\}$ with Block-AP. Across all block-wise PTQ methods, calibration samples are drawn from C4: 800 for Llama-3.2-3B-Instruct; 600 for Qwen2.5-7B-Instruct and L1-Max-Qwen-7B; 550 for Llama-3.1-8B-Instruct; 512 for DeepSeek-R1-Distill-Llama-8B; and 400 for Qwen2.5-14B-Instruct. For the remaining hyperparameters, we follow the experimental settings recommended in prior work (Lee et al., 2023; Shao et al., 2024; Chen et al., 2025).

On a single A100 GPU, the LFQ process takes approximately 1.5 hours for Qwen2.5-7B-Instruct, L1-Qwen-7B-Max, DeepSeek-R1-Distill-Llama-8B, and Llama 3.1 8B Instruct, and about 2 hours for Qwen2.5-14B-Instruct.

C FURTHER EVALUATION OF LFQ FOR LARGE REASONING MODELS UNDER STOCHASTIC DECODING

Table 7: Avg@8 and standard deviation on MATH500 for L1-Qwen-7B-Max and DeepSeek-R1-Distill-Llama-8B with LFQ under block-wise PTQ (FlexRound). Within the PTQ method, the best accuracy is shown in **bold**. "W4" and "W3g128" denote 4-bit per-channel weight-only quantization and 3-bit group-wise quantization (group size 128), respectively. We use a temperature of 0.6 and a top-p of 0.95.

Method	# Bits	MATH500 (↑) (Avg@8)
L1-Qwen-7B-Max	BF16	89.05 ± 0.74
FlexRound	W4	87.45 ± 0.75
FlexRound+LFQ (Ours)	W4	$\mathbf{88.40} \pm 0.86$
FlexRound	W3g128	85.35 ± 0.64
FlexRound+LFQ (Ours)	W3g128	$\mathbf{86.50} \pm 0.54$
DeepSeek-R1-Distill-Llama-8B	BF16	72.53 ± 1.16
FlexRound	W4	70.10 ± 1.37
FlexRound+LFQ (Ours)	W4	$\mathbf{71.95} \pm 1.20$
FlexRound	W3g128	67.00 ± 0.97
FlexRound+LFQ (Ours)	W3g128	$\mathbf{69.25} \pm 0.85$

To ensure a more robust evaluation of LFQ, we additionally measure Avg@8 on MATH500 for L1-Qwen-7B-Max and DeepSeek-R1-Distill-Llama-8B with a temperature of 0.6 and a top-p of 0.95.

Table 7 shows that LFQ also improves the Avg@K score on MATH across different large reasoning models, demonstrating that LFQ is effective under both greedy and stochastic decoding.

D Importance of Applying LFQ to the Last Block

To emphasize that whether LFQ is applied to the final block is far more important than how many blocks are optimized with LFQ, we conduct the following experiments for Llama 3.1 8B Instruct. When $k = 2$, we apply LFQ to the second-to-last block, while for the last block we only minimize $\|\boldsymbol{X}\boldsymbol{W}_{\text{FP}} - \boldsymbol{X}\boldsymbol{W}_q\|_F^2$ without using either the LM head or cross-entropy (denoted as "$k = 2$ except the last block"). When $k = 3$, we apply LFQ sequentially to the third- and second-to-last blocks, and again, for the last block only, we minimize $\|\boldsymbol{X}\boldsymbol{W}_{\text{FP}} - \boldsymbol{X}\boldsymbol{W}_q\|_F^2$ without employing the LM head or cross-entropy (denoted as "$k = 3$ except the last block"). We then compare these settings with the original $k = 2$ and $k = 3$ configurations in Figure 2.

Table 8: Comparison of $k = 2$ and $k = 3$ except the last block with the original $k = 2$ and $k = 3$ configurations in Figure 2. "LFQ@Last" indicates whether LFQ is applied to the last block. Similarly, "LFQ@Last-1" and "LFQ@Last-2" indicate whether LFQ is applied to the second-to-last and third-to-last blocks, respectively.

Method	LFQ@Last	LFQ@Last-1	LFQ@Last-2	MMLU (↑)	(IFEval+GSM8K)/2 (↑)
Llama 3.1 8B Instruct	N/A	N/A	N/A	68.34	79.74
FlexRound	X	X	X	66.19	75.80
+ $k = 2$ except the last block	X	O	X	66.97	76.17
+ $k = 2$ (Figure 2)	**O**	O	X	**66.99**	**77.13**
+ $k = 3$ except the last block	X	O	O	66.98	76.15
+ $k = 3$ (Figure 2)	**O**	O	O	**67.03**	**76.78**
OmniQuant	X	X	X	64.87	74.39
+ $k = 2$ except the last block	X	O	X	65.29	74.47
+ $k = 2$ (Figure 2)	**O**	O	X	**65.40**	**75.32**
+ $k = 3$ except the last block	X	O	O	**65.29**	74.65
+ $k = 3$ (Figure 2)	**O**	O	O	65.25	**75.30**
Block-AP	X	X	X	63.24	71.21
+ $k = 2$ except the last block	X	O	X	**63.78**	71.30
+ $k = 2$ (Figure 2)	**O**	O	X	63.57	**71.76**
+ $k = 3$ except the last block	X	O	O	**63.81**	71.24
+ $k = 3$ (Figure 2)	**O**	O	O	63.64	**71.52**

As shown in Table 8, the MMLU score remains nearly unchanged regardless of whether LFQ is applied to the final block. In contrast, when LFQ is not applied to the final block, the average of IFEval and GSM8K (i.e., "(IFEval+GSM8K)/2") consistently drops, approaching the performance level of each underlying PTQ technique. These results indicate that, for improving the generation quality of low-bit quantized LLMs, it is far more critical to apply LFQ to the final block than to simply increase the number of blocks optimized with LFQ.

E COMBINATION OF LFQ WITH RILQ

Table 9: Comparison of LFQ (ours) against RILQ, a state-of-the-art LoRA-based quantization error compensation method, on Llama 3.1 8B Instruct using block-wise PTQ (FlexRound, OmniQuant, and Block-AP). Within each PTQ method, the best accuracy is shown in **bold** and the second best is underlined. "W4" denotes 4-bit per-channel weight-only quantization.

		Language modeling/understanding		Text generation	
Method	# Bits	WikiText2 (↓)	MMLU (↑)	IFEval (↑) (greedy)	GSM8K (↑) (greedy)
Llama 3.1 8B Instruct	BF16	6.75	68.34	74.49	84.99
FlexRound	W4	7.06	66.19	70.24	81.35
FlexRound+RILQ	W4	**6.95**	66.86	71.90	80.52
FlexRound+LFQ	W4	7.06	**66.97**	<u>72.09</u>	**81.80**
FlexRound+LFQ+RILQ	W4	<u>6.98</u>	<u>66.96</u>	**72.46**	<u>81.43</u>
OmniQuant	W4	7.49	64.87	70.61	78.17
OmniQuant+RILQ	W4	<u>7.24</u>	**66.07**	71.35	78.85
OmniQuant+LFQ	W4	7.47	65.48	<u>71.35</u>	**79.76**
OmniQuant+LFQ+RILQ	W4	**7.23**	<u>65.82</u>	**71.35**	<u>79.45</u>
Block-AP	W4	7.76	63.24	68.58	73.84
Block-AP+RILQ	W4	<u>7.43</u>	**64.62**	68.58	73.92
Block-AP+LFQ	W4	7.69	63.77	**68.76**	**74.45**
Block-AP+LFQ+RILQ	W4	**7.43**	<u>64.53</u>	<u>68.58</u>	<u>74.22</u>

LFQ underperforms RILQ on WikiText2 perplexity (language modeling) and MMLU accuracy (language understanding), while outperforming RILQ on IFEval and GSM8K (text generation), as shown in Table 4. However, we emphasize that LFQ (quantization objective) and RILQ (LoRA addition) address orthogonal aspects of the problem rather than competing with each other. Because LFQ can be readily combined with RILQ in a complementary manner, we therefore explore their joint application to leverage the strengths of both methods.

LFQ + RILQ performs comparably to RILQ on WikiText2 perplexity and MMLU accuracy, while achieving results close to LFQ on IFEval and GSM8K. This indicates that LFQ + RILQ can effectively inherit the strengths of both techniques.

F Comparison of Block-wise PTQ with AWQ

Table 10: Comparison of block-wise PTQ (FlexRound, OmniQuant, and Block-AP) with AWQ, one of the mainstream layer-wise PTQ techniques. For each task, the worst accuracy is shown in red. "W4" denotes 4-bit per-channel weight-only quantization.

		Language modeling/understanding		Text generation	
Method	# Bits	WikiText2 (↓)	MMLU (↑)	IFEval (↑) (greedy)	GSM8K (↑) (greedy)
Llama 3.1 8B Instruct	BF16	6.75	68.34	74.49	84.99
AWQ	W4	7.96	63.57	67.84	73.54
FlexRound	W4	7.06	66.19	70.24	81.35
OmniQuant	W4	7.49	64.87	70.61	78.17
Block-AP	W4	7.76	63.24	68.58	73.84

As demonstrated by several existing block-wise PTQ studies (Shao et al., 2024; Cheng et al., 2024; Lee et al., 2025b), block-wise PTQ typically outperforms layer-wise PTQ methods such as AWQ across a range of tasks, as shown in Table 10.