Exploring Post-training Quantization in LLMs from Comprehensive Study to Low Rank Compensation

Zhewei Yao, Xiaoxia Wu, Cheng Li, Stephen Youn, Yuxiong He
Microsoft
{zheweiyao, xiaoxiawu, chengli1, stephen.youn, yuxhe}@microsoft.com

Abstract

Post-training quantization (PTQ) has emerged as a promising technique for mitigating memory consumption and computational costs in large language models (LLMs). However, a systematic examination of various quantization schemes, model families, and quantization bit precision has been absent from the literature. In this paper, we conduct a comprehensive analysis of these factors by investigating the effects of PTQ on weight-only, activation-only, and weight-and-activation quantization using diverse methods such as round-to-nearest (RTN), GPTQ, ZeroQuant, and their variants. We apply these methods to two distinct model families with parameters ranging from 125M to 176B. Our contributions include: (1) a sensitivity analysis revealing that activation quantization is generally more susceptible to weight quantization, with smaller models often outperforming larger models in terms of activation quantization; (2) an evaluation and comparison of existing PTQ methods to optimize model size reduction while minimizing the impact on accuracy, revealing that none of the current methods can achieve the original model quality for quantization with either INT4-weight or INT4-weight-and-INT8-activation; (3) based on these insights, we propose an optimized method called Low-Rank Compensation (LoRC), which employs low-rank matrices to enhance model quality recovery with a minimal increase in model size.

1 Introduction

As the progression of hardware bandwidth lags behind that of computational demand [13], the resource demands of extra-large models such as MT-NLG-530B [25]—which necessitates the deployment of multiple nodes for operation—escalate, adding to the complexities of cross-node communication. This has emphasized the urgency to curtail both the size and computational expense of Large Language Models (LLMs). An increasingly effective solution to these issues is post-training quantization (PTQ). This method aids in the reduction of training prerequisites while simultaneously lowering the bit precision of weights and activations to either INT4 or INT8.

While the effectiveness of post-training quantization (PTQ) has been underscored in a number of recent studies [31, 11, 30, 7], a comprehensive, systematic investigation into several key dimensions of this technique remains to be undertaken. Specifically, the extant literature falls short in providing thorough coverage of the functionality of various PTQ methods or the sensitivity of disparate models. Moreover, despite current quantization methods demonstrating promising results in the reduction of model sizes, the question persists as to whether these methods are achieving their optimal potential in minimizing Large Language Models (LLMs) sizes.

*Equal Contribution. Code will be released as a part of https://github.com/microsoft/DeepSpeed

With these observations in mind, our study sets forth to address two salient questions: (1) When subjected to quantization, do LLMs of varying sizes and pretraining data exhibit similar behavior? (2) Are existing quantization methods truly leveraging their full potential in reducing the sizes of LLMs?

**Contribution.** To elucidate these queries, we undertake an exhaustive examination of the impact of PTQ on weight-only, activation-only, and combined weight-and-activation quantization. This investigation incorporates a range of PTQ methods, including round-to-nearest (RTN), GPTQ [11], ZeroQuant [31], and their respective variants. To broaden the scope of our analysis, we focus on two distinct model families, OPT [35] and BLOOM [23], spanning model sizes from 125M to a massive 176B. Our code will be made available for reproduction. In summary, we make the following contributions:

1. **Sensitivity Analysis and Comparison Between Existing Methods**

   There are mainly two categories of PTQ for LLMs, i.e., weight-only quantization [11] and weight-and-activation quantization [6, 31, 30]. In the latter, it is uniformly observed across all studies that activation quantization demonstrates greater sensitivity than weight quantization. However, prior research tends to concentrate on a single (family) model to emphasize the necessity of their proposed quantization technique. A comprehensive and systematic evaluation of this PTQ methodology, particularly the sensitivity of weight/activation quantization for varying model sizes and distinct model families, has not yet to be undertaken. Hence, we conduct an examination on both the OPT [35] and BLOOM [23] families.

   **Sensitivity setting.** We use the zero-shot validation perplexity (PPL) differential on three datasets, namely, Wikitext-2 [20], PTB [19], and C4 [22], before and after the quantization of these LLMs to

<table>
<thead>
<tr>
<th>Class</th>
<th>Class-1</th>
<th>Class-2 &amp; ≤0.5</th>
<th>&gt;0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPL Degradation</td>
<td>≤0.1</td>
<td>&gt;0.1 &amp; ≤0.5</td>
<td>&gt;0.5</td>
</tr>
</tbody>
</table>

   **Figure 1:** The model size and quality trade-off of different quantization methods on models from OPT families. Here PTQ (with fine-grained quantization) represents the method from [31, 11]. RTN means the naive round-to-nearest baseline (fine-grained quantization), and FP16/INT8 is used as the no-accuracy-loss baseline. LoRC is our proposed method that works seamless with PTQ.

   **2 Sensitivity Analysis and Comparison Between Existing Methods**

   Table 1: Classification of quantization sensitivity (or quantization loss). The sensitivity increases from **Class-1** to **Class-3**.

   - For larger models (>10B), fine-grained (block size 64–256) 4-bit weight quantization plus 8-bit activation quantization (block size 64–256) with PTQ can be used for real deployment; (2) For middle-size models (<10B and >1B), per-row INT8 quantization plus fine-grained (block size 64–256) INT8 activation quantization can be used with PTQ from [11, 31]; (3) For smaller models (<1B), per-row W8A8 (INT8 weight and INT8 activation) RTN is enough based on [31].
illustrate their sensitivity, as PPL is significantly correlated to zero-shot/few-shot accuracy measurement [7]. Specifically, a higher PPL drop indicates enhanced quantization sensitivity. For simplicity, we also categorize quantization sensitivity (or quantization loss) into three different classes as depicted in Table 1. Notably, the threshold is chosen because when the model size approximately doubles (e.g., 13B vs. 30B, and 30B vs. 66B), the PPL improvement is about 0.5 (see Table 2). The sensitivity (or loss) incrementally increases as the class number ascends. From a practical standpoint, we favor lowering quantization sensitivity (accuracy loss), making Class-1 the optimal-loss PTQ.

Robustness of Weight-only Quantization for Large Models. The results of weight-only quantization in OPT and BLOOM models are summarized in Table 2. INT8 weight-only quantization, either symmetric or asymmetric, results in negligible accuracy loss (less than 0.05, i.e., Class-1). Consequently, for tasks oriented towards generation, FP16 weight can simply be replaced with INT8 weight to reduce memory usage. For INT4 quantization, the asymmetric method outperforms the symmetric approach in accuracy, attributable to its superior utilization of the quantization range. Interestingly, larger models exhibit better tolerance to low-precision quantization (i.e., INT4) than smaller models, with a few exceptions such as OPT-66B. Particularly, BLOOM-176B shows PPL degradation (around 0.3 points) in Class-2, which could explain why the large GLM-130B [34] can operate with INT4 weight-only quantization out of the box with acceptable accuracy impact.

<table>
<thead>
<tr>
<th>Class</th>
<th>W16-A16</th>
<th>W8sym-A16</th>
<th>W4sym-A16</th>
<th>W16-A8sym</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPT-6.7b</td>
<td>11.90</td>
<td>13.44</td>
<td>12.62</td>
<td>12.62</td>
</tr>
<tr>
<td>OPT-66b</td>
<td>10.33</td>
<td>31.52</td>
<td>561.35</td>
<td>561.35</td>
</tr>
<tr>
<td>BLM-7.1b</td>
<td>14.96</td>
<td>15.90</td>
<td>15.14</td>
<td>15.14</td>
</tr>
<tr>
<td>BLM-176b</td>
<td>10.90</td>
<td>11.20</td>
<td>11.62</td>
<td>11.62</td>
</tr>
</tbody>
</table>

Table 2: Average PPL of OPT & BLOOM.

Challenge Encountered in Activation Quantization for Large Models. Activation quantization has consistently proven more difficult than weight quantization [31, 6], as illustrated in Table 2. When compared to weight-only quantization, activation-only quantization indicates that asymmetric quantization can significantly improved performance over symmetric quantization. Moreover, contrary to weight-only quantization, smaller models typically exhibit better tolerance to activation quantization, as their hidden dimension is smaller and the activation dynamic range is also narrower than larger models [31]. It should be noted that for models larger than 10B, all fall into Class-3, indicating a degradation of >0.5 PPL.

Comparison between GPTQ and ZeroQuant. Numerous lightweight optimization-based methods have been proposed, which update the model weights during quantization. These methods such as [31, 11, 30], unlike quantization-aware training, only require a small portion of the training data and a limited training time. Particularly, GPTQ [11] and ZeroQuant [31], have proven to be effective and efficient in terms of GPU resources, time cost, and data usage for INT4 weight quantization. We focus on the variants of GPTQ and ZeroQuant as well as the most straightforward baseline, round-to-nearest neighborhood (RTN). Due to space limit, we defer the experimental results in Appendix C. The main conclusions are: (1) GPTQ typically performs better for weight-only quantization, while ZeroQuant yields superior results for weight and activation quantization. (2) The tested optimization-based methods cannot achieve Class-1 quantization error for either INT4 weight-only or W4A8 quantization with the exception of GPTQ on OPT-30B with weight-only quantization.

3 Proposed Method to Further Push the Limit of Post-training Quantization

Building on the investigation and conclusions drawn from previous sections, it has become apparent that there is still a need for an advanced methodology to further refine the existing methods, with the objective of fully realizing the original fp16 PPL quality. In this section, we introduce a simple yet effective method called LoRC (Low Rank Compensation) to optimize the current existing quantization error and further bridge the gap between the quality of the original model and its quantized counterparts. LoRC is inspired by the employment of low-rank matrix factorization on the quantization error matrix $E := W − \hat{W}$, where $W$ represents the original weight and $\hat{W}$ is the quantized weight. LoRC approximates the error $E$ with $\hat{E} = \hat{U}\hat{V}$ by using two low-rank matrices $\hat{U}$ and $\hat{V}$. This results in a more accurate approximation of the original weight matrix $W$ by $W_{\text{rec}} = \hat{W} + \hat{E}$, reducing errors: $\| W - \hat{W} \| \geq \| W - W_{\text{rec}} \|$. LoRC consists of two steps:

\footnote{\textsuperscript{2}We tested the method proposed by [30] but did not find it better than others for INT4 weight quantization.}
We provide ablation study for the low-rank dimension $n$ in Appendix D.

4 Discussion

Conclusion. In this work, we provide a comprehensive study of post-training quantization (PTQ) on large language models with different PTQ methods (e.g., RTN, GPTQ, ZeroQuant), and with different quantization coverage (weight-only and weight-and-activation quantization), etc. We find that PTQ methods are critical to improving the quantized model quality, and that fine-grained quantization (FGQ) can bring acceptable accuracy and model size trade-off. Finally, we introduced an optimization technique called Low Rank Compensation (LoRC), which works synergistically with PTQ and FGQ, playing a crucial role in enhancing full model quality recovery with a minimal increase in model size.

Limitation. Despite quantizing over 10,000 experiments, our study was constrained by our computing resources. This restriction made us choose between diversifying the model sizes and varying the tasks. We strategically limited our datasets to WikiText, PTB, and C4 to concentrate on a broad range of quantization methods. Consequently, our general findings are more robust concerning the two model families and three datasets examined in this paper. However, caution should be exercised when generalizing these findings to tasks that are dissimilar to those covered in this study.

Table 3: W\textsuperscript{#sym-A16} quantization with # being 4-bit, 3-bit and 2-bit on OPT and BLOOM (BLM).

<table>
<thead>
<tr>
<th>Bits</th>
<th>LoRC</th>
<th>Coarse-grained weight quantization (per-row block-size)</th>
<th>Fine-grained quantization on weight (256 block-size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPT-6.7b</td>
<td>OPT-13b</td>
<td>OPT-30b</td>
<td>OPT-66b</td>
</tr>
<tr>
<td>W8A16</td>
<td>✓</td>
<td>11.90</td>
<td>11.22</td>
</tr>
<tr>
<td>W4A16</td>
<td>✓</td>
<td>12.28</td>
<td>11.42</td>
</tr>
<tr>
<td>W3A16</td>
<td>✓</td>
<td>14.18</td>
<td>12.43</td>
</tr>
<tr>
<td>W2A16</td>
<td>✓</td>
<td>120.56</td>
<td>40.17</td>
</tr>
</tbody>
</table>

Step I: Implement Singular Value Decomposition (SVD) on the error matrix $E = USV$, where $U \in \mathbb{R}^{d_u \times d_u}$ and $V \in \mathbb{R}^{d_v \times d_v}$ are unitary matrices, and $\Sigma \in \mathbb{R}^{d_u \times d_v}$ is a diagonal matrix with its diagonal elements ordered in a descending manner.

Step II: We formulate the matrix $\hat{E} = \hat{U} \hat{V}$ where $\hat{U} = U_m (\Sigma_m)^{\frac{1}{2}}$ and $\hat{V} = (\Sigma_m)^{\frac{1}{2}} V_m$. Here, $U_m = U_{1:m} \in \mathbb{R}^{d_u \times m}$, $V_m = V_{1:m} \in \mathbb{R}^{m \times d_v}$, and $\Sigma_m = \Sigma_{1:m,1:m} \in \mathbb{R}^{m \times m}$.

The objective of LoRC is to achieve a good approximation of the error matrix $E$ using low-rank matrices, with minimal impact on the increase in model size. For instance, consider the standard transformer models [27], where each layer is comprised of a multi-headed attention (MHA) module and a multi-linear perception (MLP) module.

Significantly, LoRC can be viewed as a supplementary feature to existing quantization methodologies such as RTN, GPTQ, and ZeroQuant-Local/Global, and can be seamlessly integrated with FGQ. We have conducted experiments to evaluate the performance of LoRC on both OPT and BLOOM, applying 4-bit, 3-bit, and 2-bit weights by setting the activation to FP16. Based on the discoveries in the preceding sections, we utilize the GPTQ quantization strategy. To gain a comprehensive understanding of LoRC, we include the results with and without the application of FGQ. The datasets and hyperparameters are consistent with those detailed in earlier sections.

Evaluation Results. The findings are showcased in Table 3, split into two sections: coarse-grained weight quantization (per-row) and fine-grained quantization (block-size 256). Notably, we observe that the two low-rank matrices, $\hat{U}$ and $\hat{V}$, can be quantized to 8-bit without any performance discrepancy (Table 9). Thus, the two low-rank matrices for LoRC in Table 3 are INT8 with a low-rank dimension of $m = 8$. Several key observations can be made. Firstly, LoRC consistently achieves best performance across all bit sizes and block sizes, as indicated by the lower perplexity scores when LoRC is activated. Secondly, the enhancement brought about by LoRC becomes more substantial as the bit size diminishes, especially noticeable for W2A16, which displays a markedly greater performance across all bit sizes and block sizes, as indicated by the lower perplexity scores when LoRC is activated. Lastly, the combination of fine-grained quantization with LoRC yields the most impressive results, underscoring the efficacy of LoRC when integrated with FGQ. Overall, the results emphasize the benefits of using LoRC for enhanced performance in weight quantization and its compatibility with FGQ. Notably, recovering the last 0.05-0.1 perplexity can be challenging, but with LoRC, we nearly recover the original model quality.

We provide ablation study for the low-rank dimension $n$ in Appendix D.
References


A Related Work

Different quantization methods [24, 33, 9, 36, 1, 8, 26, 17] for transformer-based models [27] have been explored for a while. However, most of those works need quantization-aware finetuning or even expensive quantization-aware knowledge distillation [15]. Due to the cost of training/finetuning LLMs [21, 16, 26, 29, 28], it is a challenge for practitioners/researchers to do finetuning/distillation on those LLMs, particularly for models like GPT-3-175B [4] and BLOOM-176B [23].

Post-training quantization (PTQ) [32, 3] is an alternative way to quantize the model with no/minimal finetuning requirement. Along this line, several recent works focus on LLMs (beyond the million-parameter scale). [31] proposes vector-based INT8 quantization with layer-by-layer knowledge distillation to overcome the training cost and quantization error introduced by LLMs. [6] uses similar vector-based INT8 quantization weight plus mixed-precision (INT8/FP16) quantization for activation to overcome the sensitivity of activation quantization. However, the inference speed of [6] is generally even slower than FP16 baseline [2] due to the difficulty of implementing mixed-precision calculation within a single tensor. More recently, [11] extends OBQ [10, 14, 18] on LLMs for INT4 weight-only quantization and shows great efficiency on quantization and latency, and [30] shows the outliers from activations can be smoothed out by migrating the quantization difficulty from activations to its associated weights. However, [30] can only work for W8A8 quantization as lower weight precision (INT4) itself already leads to significant accuracy degradation, and the accuracy drop is larger than 0.1 PPL points, which as discussed in the later section is sub-optimal. [7] shows the scaling law of weight-only quantization with the simplest round-to-nearest baseline, but it does not consider the weight-and-activation quantization and/or the above PTQ optimization methods. As can be seen from Figure 1, by using PTQ optimization methods, the model quality can be significantly improved.

Different than existing works, our paper extensively tests the effect of (1) different quantization schemes, e.g., symmetric and asymmetric quantization, (2) different PTQ methods, e.g., [31, 11], (3) different model families, e.g., [23, 35], (4) different quantization coverage, e.g., weight-only and weight-and-activation quantization, and (5) other discussions, e.g., the effect of quantization granularity. As such, we provide a much more comprehensive understanding of post-training quantization for large language models compared to the previous works.

B Background of Quantization

Quantization maps floating point (e.g., FP16/FP32) numbers to integer numbers (e.g., INT4/INT8) so that lower memory usage (weight quantization) and faster integer arithmetic (weight-and-activation quantization) can be achieved compared to the floating point format. In this work, we are focusing on uniform quantization, i.e.,

\[ Q(x) = \text{INT} \left( \frac{x - Z}{S} \right) - Z, \]

where \( Q \) is the quantization function, \( x \) is a floating point input vector/tensor, \( S \) is a real valued scaling factor, and \( Z \) is an integer zero point. Based on different settings, the quantization method can be viewed as (1) symmetric vs. asymmetric quantization (\( Z = 0 \) or not), (2) fine-grained vs. coarse-grained quantization (how to partition the input \( x \) and get its associated scaling factor, e.g., matrix wise or row wise). See [12] for more details.

Throughout this work, we focus on post-training quantization (PTQ), i.e., no or minimal training effort is applied after quantization, for which large accuracy degradation usually exhibits for coarse-grained quantization (per matrix/tensor) due to their large quantization error. As such, we focus on fine-grained quantization. Particularly, we use the per-row quantization (one row of the weight matrix or one token for the activation) from [31] as our coarsest-grained quantization method, and we use block-k quantization (for every \( k \) elements, they have their own scaling factor and/or zero point) as our finer-grained quantization scheme.

C Are existing quantization methods optimally harnessing the potential to minimize LLMs sizes?

Numerous lightweight optimization-based methods have been proposed, which update the model weights during quantization. These methods such as [31, 11, 30], unlike quantization-aware training, only require a small portion of the training data and a limited training time. Particularly, GPTQ [11]
we set a fixed random seed for our experiments. In the context of post-training quantization, varying
we find the best configurations for ZQ-Local and ZQ-Global and denote them as ZQ-Local
∗ will be passed to those columns not yet quantized. See[10, 11] for more details.

weight matrix is performed column-/row-wisely and the quantization errors from previous columns
(e.g., learning rate and its scheduler) provided in the original work [31] are sub-optimal. In this work,

Table 4: The evaluation results of different PTQ methods on OPT and BLOOM (BLM) with
asymmetric quantization on weight or (and) activation.

<table>
<thead>
<tr>
<th>Precision</th>
<th>Method</th>
<th>OPT-6.7b</th>
<th>OPT-13b</th>
<th>OPT-30b</th>
<th>OPT-66b</th>
<th>BLM-1.7b</th>
<th>BLM-3b</th>
<th>BLM-7.1b</th>
<th>BLM-176b</th>
</tr>
</thead>
<tbody>
<tr>
<td>W16A16</td>
<td>RTN</td>
<td>13.44</td>
<td>12.09</td>
<td>11.52</td>
<td>31.52</td>
<td>22.47</td>
<td>19.01</td>
<td>15.90</td>
<td>11.20</td>
</tr>
<tr>
<td></td>
<td>GPTQ</td>
<td>12.28</td>
<td>11.42</td>
<td>10.78</td>
<td>10.52</td>
<td>21.58</td>
<td>18.33</td>
<td>15.50</td>
<td>11.02</td>
</tr>
<tr>
<td></td>
<td>ZQ-Local∗</td>
<td>12.46</td>
<td>11.64</td>
<td>11.05</td>
<td>10.79</td>
<td>21.70</td>
<td>18.50</td>
<td>15.55</td>
<td>11.11</td>
</tr>
<tr>
<td></td>
<td>ZQ-Global∗</td>
<td>12.38</td>
<td>11.62</td>
<td>11.04</td>
<td>10.68</td>
<td>21.38</td>
<td>18.33</td>
<td>15.52</td>
<td>11.05</td>
</tr>
<tr>
<td>W4A16</td>
<td>RTN</td>
<td>14.80</td>
<td>26.36</td>
<td>86.26</td>
<td>815.00</td>
<td>22.75</td>
<td>19.17</td>
<td>16.19</td>
<td>12.22</td>
</tr>
<tr>
<td></td>
<td>GPTQ</td>
<td>13.88</td>
<td>17.28</td>
<td>20.71</td>
<td>648.69</td>
<td>21.71</td>
<td>18.44</td>
<td>15.75</td>
<td>11.86</td>
</tr>
<tr>
<td>W4A8</td>
<td>RTN</td>
<td>12.46</td>
<td>11.64</td>
<td>11.05</td>
<td>10.79</td>
<td>21.70</td>
<td>18.50</td>
<td>15.55</td>
<td>11.11</td>
</tr>
<tr>
<td></td>
<td>GPTQ</td>
<td>12.38</td>
<td>11.62</td>
<td>11.04</td>
<td>10.68</td>
<td>21.38</td>
<td>18.33</td>
<td>15.52</td>
<td>11.05</td>
</tr>
</tbody>
</table>

and ZeroQuant [31], have proven to be effective and efficient in terms of GPU resources, time cost,
and data usage for INT4 weight quantization. In this work, we focus on the variants of GPTQ and
ZeroQuant as well as the most straightforward baseline, round-to-nearest neighborhood (RTN).

RTN directly applies PTQ on the trained data and follows the procedure detailed in Section B to
perform the quantization. Specifically, for asymmetric quantization, we set \( S = \max(x) - \min(x) \)
and \( Z = \min(x) \).

GPTQ extends the OBQ [10]. It tries to optimize the following non-linear least square problem,
\[
\min_{\hat{W}} \| Wx - \hat{W}x \|_2^2
\]
where \( W \) is the weight, \( x \) is the activation, and \( \hat{W} \) is a quantized weight. GPTQ
employs second-order methods to obtain a closed-form solution. In addition, the quantization for each
weight matrix is performed column-/row-wisely and the quantization errors from previous columns
will be passed to those columns not yet quantized. See[10, 11] for more details.

ZQ-Global is the original method proposed in [31], where authors treat each layer as a small neural
network (a.k.a., subnetwork) and use the FP16 subnetwork as the teacher model to distill the quantized
one with a few hundred iterations, i.e., \( \min_{\theta} \| f_\theta(x) - f_\hat{\theta}(x) \|_2^2 \), where \( \theta \) is a set of weights, \( \hat{\theta} \)
the quantized version, \( f_\theta \) is the subnetwork with parameters \( \theta \), and \( x \) is the input. Thus, it can
significantly reduce the GPU resource requirement and time cost.

ZQ-Local is an extension mode of ZQ-Global for further GPU requirement reduction and training
cost reduction. Particularly, instead of using each transformer layer as the subnetwork, we treat each
linear layer as the subnetwork. This method can be viewed as an iterative first-order optimization
method (e.g., SGD) to solve \( \min_{\hat{W}} \| Wx - \hat{W}x \|_2^2 \).

Experimental Setup. We compare the four methods mentioned above on weight-only and weight-
and activation quantization. As weight quantization is always static (i.e., it does not change during
inference), there is virtually no system performance difference between symmetric and asymmetric
quantization. We use asymmetric quantization for better accuracy, and the conclusions would hold
similarly for symmetric quantization. An interesting finding for ZeroQuant is that the hyperparameters
(e.g., learning rate and its scheduler) provided in the original work [31] are sub-optimal. In this work,
we find the best configurations for ZQ-Local and ZQ-Global and denote them as ZQ-Local∗ and
ZQ-Global∗, respectively, with the best tuned results. To ensure consistent and comparable results,
we set a fixed random seed for our experiments. In the context of post-training quantization, varying
the random seed has minimal impact on the final results.

Evaluation of Weight-only Quantization. The results from weight-only quantization using OPT and
Bloom are presented in Table 4. The findings indicate that the larger models tend to be less sensitive
to INT4 weight-only quantization. This observation holds true across all methods (RTN, GPTQ,
ZQ-Local∗, and ZQ-Global∗) with the exception of OPT-66B, which shows greater degradation than
OPT-30B. It is noteworthy that light-weight optimization-based methods significantly outperform the
RTN baseline in terms of accuracy. For instance, these methods substantially reduce the degradation
in perplexity of OPT-30B/66B compared to baseline. Most quantized models with parameters greater
than 6.7B fall under Class II, indicating their potential for real-world applications. For instance, the
quality of INT4 OPT-30B (66B) is superior to that of INT8 OPT-13B (30B).

Evaluation of Weight and Activation Quantization. The evaluation results for existing methods
using W4A8 quantization are presented in Table 4. The three light-weight optimization-based

---

\(^3\)We tested the method proposed by [30] but did not find it better than others for INT4 weight quantization.

\(^4\)The bias term (a.k.a., the zero point) can be simply fused into the previous activation quantization kernel [31].
methods outperform RTN significantly, underscoring their efficacy. However, all of the results fall into either Class-2 or Class-3. This suggests that for certain applications, it might be more beneficial to use smaller models with fewer parameters rather than larger, quantized models.

Among quantization-based methods, ZQ-Global∗ and ZQ-Local∗ generally outperform GPTQ, which is anticipated given that GPTQ was originally designed for weight-only quantization. ZQ-Global∗ performs better than ZQ-Local∗ in most cases except for the two largest models, OPT-66B and Bloom-176B, despite having larger trainable parameters in one step. This again signifies the need for a more suitable and advanced optimization method for large language models (LLMs).

Finding 2 on Comparisons. (1) GPTQ typically performs better for weight-only quantization, while ZeroQuant (including both ZQ-Global∗ and ZQ-Local∗) yields superior results for weight and activation quantization. (2) The tested optimization-based methods cannot achieve Class-1 quantization error for either INT4 weight-only or W4A8 quantization with the exception of GPTQ on OPT-30B with weight-only quantization.

C.1 Fine-grained Quantization and Its Evaluation

With PTQ and row-wise quantization, achieving Class-1 quantization error is challenging for both weight-only and weight-and-activation quantization. Generally, utilizing a smaller model with INT8 weight is more advantageous than employing a model that is twice as large with INT4 weight.

One potential solution to this issue is the implementation of finer-grained quantization schemes [5], where every $k$ elements possess their own scaling factor and/or zero point. This approach can significantly reduce quantization error. In the extreme case, where every single element has its own scaling factor, the original FP16 number can be precisely recovered. Importantly, block-$k$ quantization can be implemented on modern GPUs, one of the most prevalent deep learning architectures, since the compute unit (streaming multiprocessor) of GPUs processes tiles of data (e.g., 128 by 128 tiling size) for matrix computation.

Although fine-grained quantization can substantially narrow the gap between the quantized tensor and its floating-point counterpart, the application of RTN still results in a non-trivial accuracy gap. Consequently, we build upon fine-grained quantization by employing existing optimization-based methods to further enhance accuracy. Specifically, we utilize GPTQ and ZQ-Global for all models and settings and apply ZQ-Local to OPT-66B and Bloom-176B. For the hyperparameters used in ZQ-Global and ZQ-Local, we select the top three identified in Section C for all models, except for Bloom-176B, for which we only use the top-performing hyperparameter to reduce training costs.

4-bit Weight Quantization. We hereby present the W4A16 results for OPT and BLOOM, as delineated in Table 5, corresponding to an array of quantization block sizes. The performance sees a significant improvement with smaller block sizes compared to per-row quantization. The point of diminishing returns, however, varies for different model sizes. For example, smaller models (such as OPT-6.7B and BLOOM-1.7b) continue to see substantial gains until the block size reduces to 32. In contrast, for larger models (those exceeding 10B, with OPT-66B as the exception), the benefits derived from smaller block sizes wane rapidly around block-256/512. Most crucially, for models equal to or larger than 13B, a smaller quantization block size results in quantization error being classified under Class-1, indicating virtually negligible degradation in accuracy.

Table 6: OPT W4asym-A8 with various block-size out of the best result from GPTQ, ZQ-Local, and ZQ-Global on OPT and BLOOM (BLM).

| Precision block-size (W|A) | OPT-6.7b | OPT-13b | OPT-30b | OPT-66b | BLM-1.7b | BLM-3b | BLM-7.1b | BLM-176b |
|---------------------------|----------|---------|---------|---------|----------|--------|---------|----------|
| W4A16                     | 128 | NA      | 10.28   | 10.44   | 20.92    | 18.39  | 15.17   | 10.94    |
| W4A8                      | 12.29   | 11.45   | 10.80   | 10.61   | 21.39    | 18.31  | 15.52   | 11.03    |
| Case-2: per-row | 128 | 12.04   | 11.31   | 10.75   | 10.45    | 21.27  | 17.86   | 15.19    | 10.96    |
| Case-3: 128 | 128 | 12.04   | 11.31   | 10.75   | 10.45    | 21.27  | 17.86   | 15.19    | 10.96    |

Activation Quantization (W4A8). To comprehend the benefits of fine-grained quantization, we present the BLOOM-176B with different quantization block sizes on activation. Here weight is asymmetrically quantized with block size 128.
Table 5: Results of W4\textsuperscript{asym}-A16 quantization with various block-size out of the best result from optimization-based methods on OPT and BLOOM (BLM). N/A means that the block size is not divisible by the hidden size.

<table>
<thead>
<tr>
<th>Block-size</th>
<th>OPT-6.7b</th>
<th>OPT-13b</th>
<th>OPT-30b</th>
<th>OPT-66b</th>
<th>BLM-1.7b</th>
<th>BLM-3b</th>
<th>BLM-7.1b</th>
<th>BLM-176b</th>
</tr>
</thead>
<tbody>
<tr>
<td>W16A16</td>
<td>11.90</td>
<td>11.22</td>
<td>10.70</td>
<td>10.33</td>
<td>20.43</td>
<td>17.58</td>
<td>14.96</td>
<td>10.90</td>
</tr>
<tr>
<td>Per-row</td>
<td>12.28</td>
<td>11.42</td>
<td>10.78</td>
<td>10.52</td>
<td>21.38</td>
<td>18.33</td>
<td>15.50</td>
<td>11.02</td>
</tr>
<tr>
<td>1024</td>
<td>12.16</td>
<td>11.36</td>
<td>10.75</td>
<td>10.52</td>
<td>31.03</td>
<td>N/A</td>
<td>15.24</td>
<td>10.96</td>
</tr>
<tr>
<td>512</td>
<td>12.08</td>
<td>11.32</td>
<td>10.73</td>
<td>10.52</td>
<td>20.93</td>
<td>17.99</td>
<td>15.20</td>
<td>10.95</td>
</tr>
<tr>
<td>256</td>
<td>12.05</td>
<td>11.28</td>
<td>10.74</td>
<td>10.50</td>
<td>20.95</td>
<td>17.97</td>
<td>15.18</td>
<td>10.95</td>
</tr>
<tr>
<td>128</td>
<td>12.10</td>
<td>11.28</td>
<td>10.74</td>
<td>10.44</td>
<td>20.92</td>
<td>17.90</td>
<td>15.17</td>
<td>10.94</td>
</tr>
<tr>
<td>32</td>
<td>12.03</td>
<td>11.28</td>
<td>10.72</td>
<td>10.41</td>
<td>20.82</td>
<td>17.88</td>
<td>15.16</td>
<td>10.95</td>
</tr>
</tbody>
</table>

Quantization on activation, we analyze the quantization between per-row and a block size of 128, with INT4 weight, as highlighted in Table 6. For models of considerable size, specifically those equal to or exceeding 1B, the application of such fine-grained activation quantization (Case-1) results in a substantial reduction in quantization error compared to per-row activation (Case-2). By implementing fine-grained activation quantization with weight quantization (Case-3), we are able to almost restore the performance to the level of their W4A16 counterparts.

Furthermore, we detail the impacts of varying activation quantization block sizes in Table 7 on BLOOM-176B, with INT4 weight. A trend of superior accuracy is observed with smaller block sizes in contrast to larger ones. However, the enhancement in performance reaches a saturation point when the size smaller or equal to 256, which corresponds to the range of values INT8 can represent. Despite INT8’s capability to signify 256 distinct values, activation quantization errors persist due to the application of uniform quantization.

**Finding 3 on FGQ.**

1. Larger models (≥10B) are capable of attaining Class-1 error for 4-bit quantization. These models can leverage low-precision quantization as the model size with INT4 is similar to an INT8 model that is half its size, with improved accuracy. On the other hand, smaller models (≤10B) typically reach only Class-2 or Class-3 error levels.
2. For larger models (>10B), the difference between fine-grained weight-and-activation quantization and fine-grained weight-only quantization is insignificant.
3. The advantage of fine-grained activation quantization fades for larger models when the block size reaches 256.

### D Additional Results for LoRC

The objective of LoRC is to achieve a good approximation of the error matrix $E$ using low-rank matrices, with minimal impact on the increase in model size. For instance, consider the standard transformer models [27], where each layer is comprised of a multi-headed attention (MHA) module and a multi-linear perception (MLP) module. Let $h$ represent the hidden dimension and $l$ the number of layers. The total number of parameters is $12lh^2$ as each layer contains $4h^2$ for MHA (for key, query, value, and projection matrices), and $8h^2$ for MLP (two matrices of sizes $h \times 4h$ and $4h \times h$). With the addition of low-rank LoRC to the six matrices in each layer, the total number of parameters for $l$ layers would amount to $18hnl$. Consequently, the ratio of parameters added to the existing model is $3m/2h$. It’s important to note that the low-rank dimension $m$ can be as small as 4 or 8 (which we will discuss in detail in a later section) while the standard hidden dimension $h \geq 768$, making the number $3m/2h \leq 0.016$.

---

\[ ^5 \text{In the MHA module, LoRC contributes } 2hm \text{ to each of key, query, value, and the projection matrices. In the MLP module, LoRC contributes } 8hm \text{ and } 2hm \text{ respectively to the matrices of dimensions } h \times 4h \text{ and } 4h \times h. \]
Table 9: Results of W4sym A16 quantization with LoRC approximating $\hat{E} = \hat{U}\hat{V}$ on OPT model family. $\hat{U}$ and $\hat{V}$ can be represented with FP16 or INT8, of which the performance are represented below. There is hardly any difference between FP16 and INT8.

<table>
<thead>
<tr>
<th>LoRC</th>
<th>Coarse-grained weight quantization</th>
<th>Fine-grained weight Quantization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6.7b 13b 30b 66b</td>
<td>6.7b 13b 30b</td>
</tr>
<tr>
<td>FP16</td>
<td>12.08 11.35 10.76 10.31</td>
<td>11.993 11.290 10.703</td>
</tr>
<tr>
<td>INT8</td>
<td>12.10 11.36 10.76 10.34</td>
<td>11.987 11.290 10.700</td>
</tr>
</tbody>
</table>

Table 8: W4A16 quantization with LoRC by varying the low-rank dimension $m$.

<table>
<thead>
<tr>
<th>LoRC-dim $m$</th>
<th>OPT-1.3b</th>
<th>OPT-6.7b</th>
<th>OPT-30b</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m = 0$ baseline</td>
<td>15.95</td>
<td>12.06</td>
<td>10.73</td>
</tr>
<tr>
<td>$m = 1$</td>
<td>15.93</td>
<td>12.01</td>
<td>10.73</td>
</tr>
<tr>
<td>$m = 4$</td>
<td>15.73</td>
<td>12.00</td>
<td>10.72</td>
</tr>
<tr>
<td>$m = 8$</td>
<td>15.76</td>
<td>11.99</td>
<td>10.70</td>
</tr>
<tr>
<td>$m = 16$</td>
<td>15.74</td>
<td>12.00</td>
<td>10.69</td>
</tr>
<tr>
<td>$m = 32$</td>
<td>15.71</td>
<td>12.01</td>
<td>10.69</td>
</tr>
</tbody>
</table>

Ablation Study on the Low Rank Dimension $m$. An essential aspect of the LoRC method is on the optimal low-rank dimension, denoted as $m$, explained in Step II. To explore this, we varied $m$ in the range of 1, 4, 8, 16, and 32 for OPT-1.3b/6.7b/30b models, and applied W4A16 GPTQ quantization. The outcomes are depicted in Table 8, indicating that the enhancements achieved through LoRC begin to plateau as the dimension $m$ surpasses 4. The most optimal performance for OPT-6.7b is realized when $m = 8$.

This observation may seem counterintuitive initially, as one might anticipate that larger LoRC dimensions would yield more significant improvements. To gain a more comprehensive understanding, we conducted an analysis of the eigenvalues of the actual error matrix $E = W - \hat{W}$ for each matrix. By randomly selecting 20 matrices from MHA and MLP layers, we plotted the eigenvalues of $E$ as a curve, depicted in Figure 2. The two plots reveal a rapid flattening of eigenvalues after index 8, which elucidates why increasing the LoRC dimension does not considerably enhance performance. Hence, a sensible dimension for $\hat{U}$ and $\hat{V}$ in the LoRC methodology could be 8.$^6$

$^6$Please note that this observation is only true for PTQ. If one uses quantize-aware training (QAT) and let $\hat{U}$ and $\hat{V}$ updated during QAT, we arrive at contrasting conclusions.