TesseraQ: Ultra Low-Bit LLM Post-Training Quantization with Block Reconstruction

Anonymous ACL submission

Abstract

Large language models (LLMs) have revolutionized natural language processing, albeit at the cost of immense memory and computation requirements. Post-training quantization (PTQ) is becoming the de facto method to reduce the memory footprint and improve the inference throughput of LLMs. In this work, we aim to push the boundary of LLM PTQ by optimizing the weight rounding parameters with the block reconstruction technique, a predominant method in previous vision models. We propose TesseraQ, an advanced PTQ technique, to quantize the weights of LLMs to ultra-low bits. To effectively optimize the rounding in LLMs and stabilize the reconstruction process, we introduce progressive adaptive rounding. This approach iteratively transits the soft rounding variables to hard variables during the reconstruction process. Additionally, we optimize the dequantization scale parameters to fully leverage the block reconstruction technique. We demonstrate that TesseraQ can be seamlessly integrated with existing transformation-based PTQ algorithms such as AWQ/OmniQuant/QuaRot, significantly enhancing their performance. For instance, when compared to AWQ, TesseraQ improves the Wikitext2 perplexity from 14.65 to 6.82 in 2-bit weight quantization.

1 Introduction

Large Language Models (LLMs) have fundamentally transformed natural language processing through their unprecedented capabilities. Modern architectures like GPT-4 (Bubeck et al., 2023) and LLaMA-3 (Meta, 2024) achieve their remarkable performance by leveraging hundreds of billions of parameters. However, this immense scale creates significant deployment challenges (Zhou et al., 2024). The substantial memory and computational requirements make these models impractical for many real-world applications, particularly on consumer devices or in resource-constrained environments (Dettmers et al., 2022). To address these deployment challenges, quantization has emerged as a promising solution. This technique reduces the precision of model parameters and activations from 32-bit floating-point (FP32) to more compact representations such as 8-bit or 4-bit integers (INT8, INT4). By decreasing the model's memory footprint, quantization enables increased I/O throughput while often maintaining performance within acceptable margins.

Post-Training Quantization (PTQ) (Gholami et al., 2022) has perhaps become the most widespread and the easiest way to compress the LLM by reducing the bitwidth of the pretrained model's parameters. For example, with a single GPU and a small number of input sequences, GPTQ (Frantar et al., 2022) can compress an FP16 LLM into INT4 format by deriving the exact solution for quantization error minimization. Recent works like AWQ (Lin et al., 2023), and OmniQuant (Shao et al., 2023) have pushed the compression limit further with INT3 weight-only quantization achieving a small performance gap with respect to the FP16 baseline. However, in a more challenging scenario like INT2 weight-only quantization, these methods still incur a large performance gap compared to the original FP16 model.

We hypothesize that this limitation stems from the restricted optimization space in existing approaches. Most current methods confine their optimization to distribution transformation or weight clipping ranges (Lin et al., 2023; Wei et al., 2023; Shao et al., 2023). While this approach is straightforward, it proves insufficient for ultra-low bit scenarios due to its constrained optimization space. We argue that advancing LLM PTQ performance requires a paradigm shift toward adjusting the entire weight tensor. However, this ambitious approach presents two critical technical challenges. First, the optimization space must be carefully crafted to provide sufficient exploratory freedom while pre-

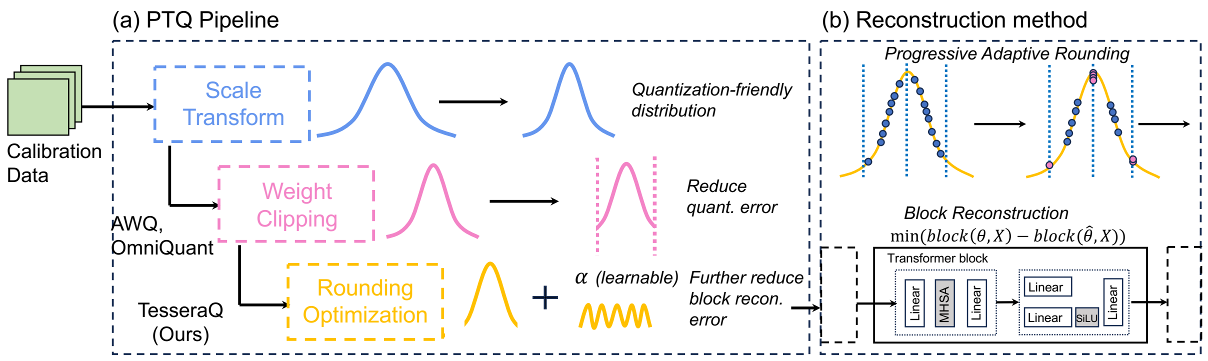

Figure 1: **The overall workflow of our proposed method**. (a) We apply TesseraQ to optimize the weight rounding parameters when the transformation scale and clipping range are determined using prior methods like AWQ/OmniQuant. (b) We propose Progressive Adaptive Rounding (PAR) for block-wise reconstruction, which iteratively hardens some rounding variables and optimizes the rest soft rounding variables till they are all quantized.

serving the model's essential behaviors. Second, optimization through discrete operations necessitates gradient estimation, but current estimation techniques often lead to training instability.

To this end, we propose TesseraQ, a block reconstruction method tailored for LLM PTQ. To improve the existing transformation-based method, we adopt the rounding optimization space (See Sec. 3.1 for details). A key innovation in TesseraQ is the introduction of Progressive Adaptive Rounding (PAR), which eliminates the need for regularization loss present in conventional rounding optimization approaches (Nagel et al., 2020; Li et al., 2021). This advancement is crucial for handling the billions of parameters in modern LLMs. As illustrated in Fig. 1(b), PAR employs an iterative strategy: systematically converting selected rounding variables to binary values while optimizing remaining variables to compensate for quantization errors. To further enhance quantization quality, we complement PAR with adaptive dequantization scale tuning, which significantly improves the expressive power of learned rounding parameters. Our block-wise reconstruction framework enables efficient optimization of individual LLM blocks using a single GPU, making TesseraQ highly practical for real-world applications. We summarize our contributions as follows

1. We propose TesseraQ, a block-wise weight rounding optimization method for LLMs. TesseraQ can be combined with existing transformation or clipping methods like AWQ, OmniQuant, and QuaRot to obtain compelling results.
2. TesseraQ contains Progressive Adaptive Rounding and Dequantization Scale Tuning. Both can stabilize the reconstruction process and effectively optimize post-training performance.
3. Our method obtains excellent performance on both perplexity metric and zero-shot accuracy metric. For example, our method improves OmniQuant perplexity results from 37.4 to 8.0 on LLaMA-2-7B W2A16 quantization. Moreover, TesseraQ+QuaRot improves the average accuracy by 10% on LLaMA-3.1-8B W3A3 quantization as compared to GPTQ+QuaRot.

2 Preliminaries

This section briefly introduces the existing research directions in LLM PTQ. We adopt uniform affine quantization, which essentially discretizes the floating-point representation of weights/activations into low-bit fixed-point representation, given by

$$\mathbf{W}^q = \text{clamp}\left(\left\lfloor \frac{\mathbf{W}}{s} \right\rceil + z, 0, 2^N - 1\right), \quad (1)$$

where $s = \frac{\gamma \max(\mathbf{W}) - \beta \min(\mathbf{W})}{2^N - 1}$ and $z = -\left\lfloor \frac{\beta \min(\mathbf{W})}{s} \right\rceil$ denote the quantization step size and the zero point ($\gamma, \beta \in (0, 1]$ control the clipping range). The resulting $\mathbf{W}^q$ is in the INT-N format. To restore it back to its original range, the dequantization step is given by $\hat{\mathbf{W}} = s \times (\mathbf{W}^q - z)$.

Optimization Objective. The plain rounding-to-nearest (RTN) method directly quantifies the model weights to integers without further optimization. However, this method usually results in significantly low task performance. To improve the LLM PTQ performance, parameters related to quantization are optimized with different objectives. For example, GPTQ (Frantar et al., 2022) and AWQ (Lin et al., 2023) utilize the layer-wise reconstruction objective, given by

$$\min_{\epsilon} \sum_{\ell=1}^{L} ||\hat{\mathbf{W}}^{(\ell)}\mathbf{X}^{(\ell)} - \mathbf{W}^{(\ell)}\mathbf{X}^{(\ell)}||_F^2, \quad (2)$$

where $\ell \in \{1, 2, \dots, L\}$ is the layer index and $\mathbf{X}$ is the input activations. While this layer-wise objective can provide efficient and exact solutions (Frantar et al., 2022), the objective does not consider inter-layer correlation like self-attention and residual connections in LLM. To this end, the block-wise reconstruction objective has been proposed (Li et al., 2021), as

$$\min_{\epsilon} \sum_{b=1}^{B} ||\mathrm{blk}(\hat{\theta}^{(b)}, \mathbf{X}^{(b)}) - \mathrm{blk}(\theta^{(b)}, \mathbf{X}^{(b)})||_F^2. \quad (3)$$

where, blk refers to one decoder block in LLMs comprising self-attention, projection, feed-forward and normalization layers. $\hat{\theta}^{(b)}$ and $\theta^{(b)}$ denote the whole quantized and full-precision parameters within the block. In practice, both layer-wise and block-wise objectives enable efficient calibration on a single GPU due to their local computation attributes. However, block-wise objectives exhibit better performance than layer-wise objectives as they account for contributions from multiple layers.

Optimization Space. Generally, three kinds of optimization spaces are explored in LLM PTQ, (1) the scale transformation, (2) the clipping range (i.e., finding the suitable γ, β), and (3) the weight values. They can be tied with either layer-wise or block-wise objectives. For instance, AWQ (Lin et al., 2023) and OS+ (Wei et al., 2023) optimize transformation and clipping range using Eq. (2), while OmniQuant (Shao et al., 2023) does similar optimization with Eq. (3). Since scale/clipping optimization methods are well-explored, in this paper, we aim to optimize weight values using block-wise objectives to further push the compression limits of LLM PTQ.

3 TesseraQ: Ultra Low-Bit PTQ

3.1 Problem Statement

Previous research has explored various optimization spaces for element-wise weight parameter adjustment. One prevalent approach, adopted by methods like AdaQuant (Hubara et al., 2020) and GPTQ (Frantar et al., 2022), involves learning an unrestricted weight update $\Delta\theta$ combined with rounding-to-nearest (RTN) operations. However, this optimization framework encounters significant limitations with block-wise objectives. AdaQuant's reliance on the Straight-Through Estimator for gradient computation becomes highly unstable at LLM scale (see Appendix A), while GPTQ cannot derive a closed-form solution for a whole block.

Given these challenges, we explore an alternative weight optimization framework based on rounding optimization (Nagel et al., 2020; Li et al., 2021), which offers a fundamentally different optimization space compared to GPTQ. Within our block reconstruction objective (Eq. (3)), we define the quantization function as:

$$\theta^q = \mathrm{clamp}\Big(\lfloor \frac{\theta}{\mathbf{s}} \rfloor + \alpha + \mathbf{z}, 0, 2^N - 1\Big), \quad (4)$$

where $\alpha \in \{0, 1\}$ represents the binary rounding variables. This rounding optimization framework presents both advantages and challenges. On the positive side, it constrains the range of each weight parameter, enabling incremental improvements to existing PTQ models like AWQ, OmniQuant, and QuaRot through fine-grained weight adjustments. However, the framework also faces some significant limitations. For instance, optimizing binary rounding variables necessitates either continuous relaxation with regularization loss (Nagel et al., 2020) or the use of STE (Hubara et al., 2020). These requirements make the optimization of billions of rounding variables particularly challenging. Further evidence of the scaling difficulties with traditional rounding optimization in LLMs is presented in Appendix A.

3.2 Progressive Adaptive Rounding

We introduce Progressive Adaptive Rounding (PAR), a novel differentiable framework for optimizing rounding variables α that overcomes key limitations of previous approaches. At its core, PAR begins by transforming the discrete rounding problem into a continuous optimization task through Sigmoid reparameterization, where we express α as $\alpha = \sigma(\nu)$. We initialize the continuous variable ν using the inverse Sigmoid function applied to the fractional part of the scaled weights: $\nu = \sigma^{-1}(\theta/s - \lfloor\theta/s\rfloor)$. This initialization ensures that our initial quantized weights match the original weights exactly: $\hat{\theta} = \theta$.

PAR's key innovation lies in its dynamic treatment of rounding variables through two complementary sets: $\mathcal{S}_{\text{Hard}}$ and $\mathcal{S}_{\text{Soft}}$. These sets correspond to variables that have been committed to binary values (*hard* rounding) and those that remain continuous (*soft* rounding), respectively. We formalize this dual approach through the following rounding function:

$$\alpha_i = \begin{cases} \sigma(\nu_i) = \frac{1}{1+\exp(-\nu)} & \text{if } i \in \mathcal{S}_{\text{Soft}} \\ \sigma'(\nu_i) = \mathbf{1}_{\nu_i > 0} & \text{if } i \in \mathcal{S}_{\text{Hard}} \end{cases}. \quad (5)$$

The optimization process begins with an empty $\mathcal{S}_{\text{Hard}}$ and proceeds through alternating phases: a *Harden Phase* where selected variables from $\mathcal{S}_{\text{Soft}}$ are committed to binary values by moving them to $\mathcal{S}_{\text{Hard}}$, followed by a *Soften Phase* where the remaining continuous variables are optimized to compensate for any accuracy loss introduced by the hardened variables. We detail these complementary phases in the following subsections.

Harden Phase. The key to effective progressive rounding lies in identifying which variables can be safely committed to binary values while minimizing the impact on reconstruction accuracy. To make this determination systematic, we introduce a scoring metric that quantifies how "ready" a variable is for binary rounding:

$$HS(\nu) = |\sigma(\nu) - 0.5|. \tag{6}$$

This score provides a decent measure of rounding readiness: variables whose soft rounding values ($\sigma(\nu)$) are close to 0.5 receive low scores, indicating that forcing them to binary values would likely cause significant reconstruction error. Conversely, variables already close to binary values (0 or 1) receive high scores, suggesting they are prime candidates for hardening. Leveraging this insight, the Harden Phase proceeds by ranking all parameters according to their HS scores and transferring the top $P\%$ of candidates to $\mathcal{S}_{\text{Hard}}$. The progression rate P follows a carefully designed schedule, starting at 0 and approaching 100 during block reconstruction. This schedule is not uniform: we can advance rapidly in the early stages when many variables are far from 0.5, but must proceed more cautiously in later stages as we handle the more challenging cases with fewer remaining soft variables to compensate for errors. Importantly, our empirical analysis shows that TesseraQ exhibits robust performance across different progression schedules for P, provided they follow the principle of gradually decreasing progression rate. We present a detailed investigation of various scheduling strategies in our ablation study (Sec. 4.3).

Soften Phase. After committing certain variables to binary values in the Harden Phase, we need to optimize the remaining soft variables to compensate for any introduced quantization errors. We formulate this as a gradient-descent optimization problem:

$$\min_{\nu_{i, i\in\mathcal{S}_{\text{Soft}}}} ||\text{blk}(\hat{\theta}, \mathbf{X}) - \text{blk}(\theta, \mathbf{X})||_F^2. \tag{7}$$

Algorithm 1: TesseraQ Calibration process

```
Input: FP16 LLM; Calibration dataset,
        PAR iteration $K$, training steps $T$
for all $b = 1, 2, \ldots, B$-th block do
    Collect input data to the block $\mathbf{X}$, the FP
      output $\text{block}(\theta, \mathbf{X})$ ;
    Initialize $\nu$ and $\mathbf{v}$;
    for all $k = 1, 2, \ldots, K$-iteration do
        Calculate score (Eq. (6)) and
          hard-round the variables with
          highest $P_k\%$ scores;
        for all $t = 1, 2, \ldots, T$ steps do
            Gradient Descend Eq. (7);
            Update $\nu_i$ for $i \in \mathcal{S}_{\text{Soft}}$ and $\mathbf{v}$;
    Hard rounding all $\nu$ and merge them into
      original parameters and apply RTN;
return Quantized model;
```

This minimization objective seeks to maintain the block's output as close as possible to the original unquantized version by adjusting the remaining soft variables. While a straightforward implementation might use masking operations to differentiate between soft and hard rounding variables, this approach would be computationally expensive at LLM scale. Instead, we developed a memory-efficient implementation strategy: we set hard-rounding variables to extreme values (∞ or $-\infty$), leveraging the fact that the sigmoid function naturally produces zero gradients for these values. In general, we found that approximately 200 optimization steps consistently achieve sufficient reduction in block reconstruction error. This empirically determined step count provides a reliable balance between optimization quality and computational efficiency.

Post-Processing. After the entire PAR procedure is finished, we apply hard-rounding $\sigma'(\cdot)$ to all variables and merge their values into the original weights, and then we can use the standard RTN (i.e., Eq. (1)). The merging can be effectively implemented by

$$\theta \leftarrow \theta + \mathbf{s} \times (\sigma'(\nu) - 0.5) \tag{8}$$

We provide a pseudocode for the learning process in Algorithm 1.

3.3 Dequantization Scale Tuning

During the PAR process, the quantized tensor θ^q undergoes continuous changes. To accommodate these dynamic adjustments, we propose Dequantization Scale Tuning (DST), a process that executes concurrently with PAR. Specifically, for the dequantization step, we introduce an additional parameter $\mathbf{v}$ and represent it as

$$\hat{\theta} = 2\sigma(\mathbf{v}) \times \mathbf{s} \times (\theta^q - \mathbf{z}). \tag{9}$$

By initializing $\mathbf{v}$ to zero vectors ($\mathbf{0}$), we begin with a neutral dequantization scale factor ($2\sigma(\mathbf{v})$) of 1, allowing the optimization process to adaptively adjust this value within a controlled range of $(0, 2)$. This sigmoid reparameterization serves two crucial purposes: it ensures smooth training dynamics and reduces the sensitivity to learning rate selection, making the optimization process more robust. Importantly, we deliberately chose to optimize the dequantization scale rather than the quantization scale s in Eq. (1) for two fundamental reasons. First, modifying s would alter the underlying rounding mechanism itself (Nagel et al., 2020), potentially destabilizing the optimization process. Second, optimizing s would necessitate the use of STE, which introduces bias into gradient calculations, potentially leading to suboptimal solutions.

4 Experiments

4.1 Experiements Setup

Most of our experiment setups are similar to OmniQuant (Shao et al., 2023), which also adopts block reconstruction loss function. Specifically, we employ asymmetric uniform quantization with 2/3/4-bit integers. We test both per-group and per-channel weight quantization. For example, we use the notation *W2A16g64* to denote the 2-bit per-group (group size is set to 64) weight-only quantization. In weight-activation quantization experiments, defaults are W4A4, W3A3, and W4A8 with per-channel weight and per-token activation quantization (Dettmers et al., 2022; Shao et al., 2023).

Calibration Data and Comparison. We report two types of evaluation metrics, the perplexity metric for evaluating the upstream datasets like WikiText2 (Merity et al., 2016), C4 (Raffel et al., 2020), and the average accuracy of 5 downstream reasoning tasks including PIQA (Bisk et al., 2020), ARC easy/challenge (Clark et al., 2018), WinoGrande (Sakaguchi et al., 2021) and HellaSwag (Zellers et al., 2019). The perplexity is evaluated with 2048 sequences. We use 512 2048-token segments from the WikiText2 training dataset as calibration data for perplexity comparison and for downstream task comparison, we sample calibration data from the C4 training dataset. We use `lm_eval` (ver0.4.2) to evaluate accuracy.

Training. We set the total PAR number of iterations K to 20 and gradually increase the P_k from 0 to 100%. In each iteration, we optimize the learnable parameters (ν and $\mathbf{v}$) for 250 training steps. We use the Adam optimizer with a fixed learning rate of $1e-3$. The batch size is set to 4. We use AWQ transformation (Lin et al., 2023) to initialize our model since we find AWQ initialization is slightly better than OmniQuant across all configurations except W2A16 quantization.

Models and Baselines. For the upstream tasks, we follow OmniQuant (Shao et al., 2023) to test weight-only quantization results on LLaMA-1-7B/13B/30B/65B (Touvron et al., 2023a), LLaMa-2-7B/13B/70B (Touvron et al., 2023b) and LLaMA-3-8B/70B (Meta, 2024). In this case, we compare GPTQ (Frantar et al., 2022), OmniQuant (Shao et al., 2023), AWQ (Lin et al., 2023), SignRound (Cheng et al., 2023) and GPTQ with QuaRot (Ashkboos et al., 2024). For downstream tasks, we test LLaMA-2-7B, LLaMA-3-8B/70B across 5 downstream tasks. We compare GPTQ, AWQ, OmniQuant, and SignRound.

4.2 Main Results

Perplexity Evaluation. We summarized the Wikitext2 perplexity (PPL) results in Table 1. Our method consistently outperforms existing methods like AWQ and OmniQuant, particularly for the low-bit W2A16 configuration. On LLaMA-2-7B with W2A16 quantization, OmniQuant only obtains 37.37 PPL while our method largely improves this result to **8.05**. In addition, LLaMA-3-8B demonstrates extremely low quantization resiliency, where the AWQ model crashed in W2A16g128 quantization. Our method, on the other hand, significantly improves the perplexity from 334 to 10.03. We observe that in general, the lower the bitwidth, the more improvement we can obtain from TesseraQ. This confirms our initial intuition that extremely low-bit weight quantization requires a thorough adjustment of each weight element. Additionally, the C4 (Raffel et al., 2020) PPL results are provided in Appendix: Table 7. Overall, C4 PPL results concur with the Wikitext2 results, demonstrating a similar trend in performance improvement. For example, TesseraQ improves the PPL of LLaMA-2-7B model from 90.64 to 14.82 with W2A16 quantization.

Table 1: **Weight-only quantization results of LLaMA-1/2/3 models**. We report WikiText2 perplexity (PPL ↓). *, †, ‡ means initialized from AWQ, OmniQuant, and QuaRot, respectively.

LLaMA1&2	Method	L1−7B	L1−13B	L1−30B	L1−65B	L2−7B	L2−13B	L2−70B	L3−8B	L3−70B
FP16	-	5.68	5.09	4.10	3.53	5.47	4.88	3.31	6.14	2.85
W2A16	GPTQ	2.1e3	5.5e3	499.75	55.91	7.7e3	2.1e3	77.95	8.4e4	1.6e4
	GPTQ‡	11.13	9.14	7.04	5.91	18.77	10.84	5.68	24.98	16.29
	AWQ	1.1e5	7002	1.2e5	6.3e6	2.9e6	6.2e3	3973	4.1e5	8.6e4
	OmniQuant	15.47	13.21	8.71	7.58	37.37	17.21	7.81	-	-
	TesseraQ†	**7.56**	**6.56**	**5.75**	**5.21**	**8.05**	**6.55**	**5.26**	**17.88‡**	**11.56‡**
W2A16 g128	GPTQ	44.01	15.60	10.92	9.51	36.77	28.14	NAN	226.7	16.06
	GPTQ‡	16.25	8.14	6.62	5.61	16.10	9.29	5.32	17.43	30.89
	AWQ	13.08	10.02	7.46	6.08	14.65	8.93	5.72	334.1	10.98
	SignRound	641.8	8.36	7.13	5.52	NAN	7.64	NAN	-	-
	OmniQuant	9.72	7.93	7.12	5.95	11.06	8.26	6.55	-	-
	TesseraQ*	**6.92**	**6.07**	**5.26**	**4.83**	**6.82**	**5.92**	**4.73**	**10.03**	**7.47**
W2A16 g64	GPTQ	22.10	10.06	8.54	8.31	20.85	22.44	NAN	86.32	11.78
	GPTQ‡	11.44	7.70	6.23	5.26	15.30	9.17	5.19	16.58	21.50
	AWQ	10.65	8.66	6.65	5.58	11.87	7.81	5.30	53.07	9.04
	OmniQuant	8.90	7.34	6.59	5.65	9.62	7.56	6.11	-	-
	TesseraQ*	**6.78**	**5.97**	**5.18**	**4.70**	**6.67**	**5.81**	**4.60**	**9.28**	**6.96**
W3A16	GPTQ	8.06	6.76	5.84	5.06	8.37	6.44	4.82	16.84	18.94
	GPTQ‡	6.15	5.45	4.53	4.01	6.13	5.35	3.72	7.54	5.22
	AWQ	8.49	6.38	5.89	6.03	14.17	6.42	4.22	11.79	12.28
	OmniQuant	6.49	5.68	4.74	4.04	6.58	5.58	3.92	-	-
	TesseraQ*	**5.99**	**5.35**	**4.44**	**3.89**	**5.84**	**5.16**	**3.68**	**7.46**	**5.12**
W3A16 g128	GPTQ	6.55	5.62	4.80	4.17	6.29	5.42	3.85	9.58	5.25
	GPTQ‡	6.07	5.41	4.48	3.92	5.99	5.28	3.65	7.42	4.98
	AWQ	6.38	5.52	4.59	3.92	6.19	5.30	3.72	8.24	4.63
	SignRound	6.28	5.45	4.50	3.90	8.09	5.23	3.68	-	-
	OmniQuant	6.15	5.44	4.56	3.94	6.03	5.28	3.78	-	-
	TesseraQ*	**5.95**	**5.32**	**4.40**	**3.82**	**5.71**	**5.11**	**3.61**	**6.90**	**4.13**
W4A16	GPTQ	6.13	5.40	4.48	3.83	5.83	5.13	3.58	7.28	4.94
	GPTQ‡	**5.78**	5.20	4.24	3.65	5.61	5.00	3.42	6.57	3.59
	AWQ	5.99	5.24	4.30	3.71	5.82	5.07	3.49	7.09	5.19
	OmniQuant	5.86	5.21	4.25	3.71	5.74	5.02	3.47	-	-
	SignRound	5.93	5.21	4.23	3.65	5.81	5.00	**3.40**	-	-
	TesseraQ*	**5.78**	**5.17**	**4.20**	**3.63**	**5.56**	**4.96**	**3.40**	**6.48**	**3.33**

Weight-Activation Quantization Evaluation. We test weight-activation quantization scenarios with per-channel weight quantization and per-token activation quantization. We experiment with W4A4, and W4A8 quantization and compare with three baselines, Atom, QuaRot, and QuaRot+GPTQ, which are best practice utilizing rotation. We also test our method on QuaRot transformations. The results are provided in Table 2, which summarizes the perplexity on WikiTex2, C4 and average accuracy on downstream tasks. We observe a slightly better performance of our method with TesseraQ compared to GPTQ. Additionally, on the more challenging 3-bit quantization scenario, TesseraQ also exceeds GPTQ by 10% accuracy on the 8B model. Detailed accuracy results are given in Appendix B.

4.3 Ablation Studies

All ablation studies are conducted on LLaMA-2-7B model with W2A16g128 quantization.

Calibration Data. In this section, we compare the performance of different calibration datasets and sizes. We sample calibration data from either WikiText2 (Merity et al., 2016) or C4 (Raffel et al., 2020) training dataset. We also experiment with the different sample sizes, ranging from 128 to 512. Meanwhile, we change the batch size during rounding optimization, ranging from 1 to 4.

Table 3 demonstrates the task performance (PPL and average accuracy metric) as well as the calibration costs (algorithm runtime and GPU memory footprint). First, we find that the source of calibration data will impact the perplexity evaluation. The performance benefits if evaluation data and calibration data are from the same dataset. For example, the C4-calibrated model has 1.2 higher WikiText2 PPL than the WikiText2-calibrated model. Second, increasing the number of samples and the batch size consistently improves the task performance. However, it may also lead to higher runtime and GPU memory consumption, which may be alleviated via multi-GPU calibration. Nevertheless, it is worthwhile to note that even with 128 samples and

Table 2: **W4A4/W3A3 quantization results of LLaMA-1/2/3**. We use per-channel weight quantization and per-token activation quantization *, † means initialized from AWQ, QuaRot.

Bitwidths	Methods	LLaMA-7B			LLaMA-2-7B			LLaMA-3-8B		
		WT2(↓)	C4(↓)	Avg. (↑)	WT2(↓)	C4(↓)	Avg. (↑)	WT2(↓)	C4(↓)	Avg. (↑)
FP16	Pretrained	5.68	7.08	62.30	5.47	6.97	64.87	6.24	9.54	69.25
W4A4	Atom (W4A4g128)	6.16	7.70	60.17	6.14	-	-	-	-	-
	QuaRot	8.37	11.44	55.38	14.19	19.72	47.57	17.83	28.08	51.83
	GPTQ†	6.10	8.01	61.37	6.00	8.19	61.45	7.85	12.70	62.87
	TesseraQ†	**6.07**	**8.00**	**61.92**	**5.92**	**8.03**	**61.75**	**7.45**	**11.35**	**65.12**
W3A3	Atom (W3A3g128)	11.77	15.43	49.28	-	-	-	-	-	-
	QuaRot	2315	1665	35.53	10996	10940	35.18	91551	65662	35.25
	GPTQ†	11.57	13.89	50.82	13.90	15.08	44.62	79.06	106.23	37.87
	TesseraQ†	**10.79**	**13.68**	**51.10**	**9.09**	**12.76**	**50.13**	**27.80**	**30.81**	**47.33**

Table 3: **Ablation studies of calibration data source and data sizes**. We report the LLaMA-2-7B W2A16g128 quantization results with task performances and calibration costs.

#Samples	Batch Size	Runtime/ GPU Mem.	Calib. Data: WikiText2			Calib. Data: C4		
			WikiText2(↓)	C4(↓)	Avg.(↑)	WikiText2(↓)	C4(↓)	Avg.(↑)
128	1	3.2h/17.5GB	7.33	11.39	56.58	8.54	10.83	56.87
256	2	3.9h/28.6GB	7.10	11.16	57.17	8.32	10.66	57.85
512	2	4.0h/40.4GB	7.14	11.22	57.42	8.22	10.47	58.56
512	4	6.0h/65.4GB	**6.82**	10.77	58.35	8.05	**10.29**	**59.27**

a batch size of 1, our TesseraQ can significantly improve the baseline AWQ results.

PAR Schedule. We investigate how to adjust the P during progressive adaptive rounding. In our implementation, we use a handcrafted design, which manually decreases the soft rate (i.e., the percentage of soft rounding variable) as shown in Fig. 2. Our handcrafted design gradually decays the soft rate. To demonstrate that our PAR is quite robust to the schedule of soft rate, we also test several rule-based adjustments, which adjust the soft rate as $\frac{1}{\exp(tx)}$, where $x \in (0, 1]$ is the scaled iteration number and t is the temperature hyper-parameter. We test $t = \{2, 3, 4, 5, 6, 7\}$ and compare it with our handcrafted implementation with LLaMA-2-7B W2A16g128 quantization. The results in Fig. 2 show that $t = 4, 5$ and our handcrafted adjustments obtain the best performance. Overall, we find that our algorithm is not sensitive to the scheduling, and has consistently superior performance than the AWQ initialized model.

Algorithm choices. We also test the algorithm choices in TesseraQ . To be more specific, we experiment with block reconstruction with or without progressive adaptive rounding (PAR) and dequantization scale tuning (DST) and compare their final task performance. As shown in Table 4, both PAR and DST contribute a lot to the final perplexity metric (denoted by WT2 (WikiText2) and C4) and average accuracy (denoted by Avg.). Remarkably, applying one of them solely can also improve the AWQ baseline (first row) results by a large margin.

Table 4: TesseraQ Algorithm choices.

PAR	DST	WT2	C4	Avg.
✗	✗	14.65	18.67	50.52
✓	✗	7.72	11.95	56.79
✗	✓	8.58	13.14	54.45
✓	✓	**6.82**	**10.77**	**58.35**

4.4 Hardware Evaluation

To demonstrate the weight compression effect and the inference throughput change, we test LLaMA-3-8B/70B under different GPU environments, kernel backend and different bitwidths. Table 5 summarizes the results of inference throughput (generated token per second) with batch size 1 or 16. Remarkably, W2A16g128 reduces the weight memory of the 70B model from 132GB to 21GB. For inference speed, the INT2 dequantization kernel (in Triton (JonathanSalwan) support) is currently less optimized, especially for larger models, expending lower throughput compared to FP16 on 70B model. Remarkably, on smaller scale like 8B, INT2 can significantly increase the inference speed. We find that INT4 with Exllama kernel can increase the throughput when batch size is 1 and achieve similar throughput with FP16 model when batch size is 16. Nonetheless, it is worthwhile to note that our TesseraQ complies with standard uniform quantization formats and can be deployed with var-

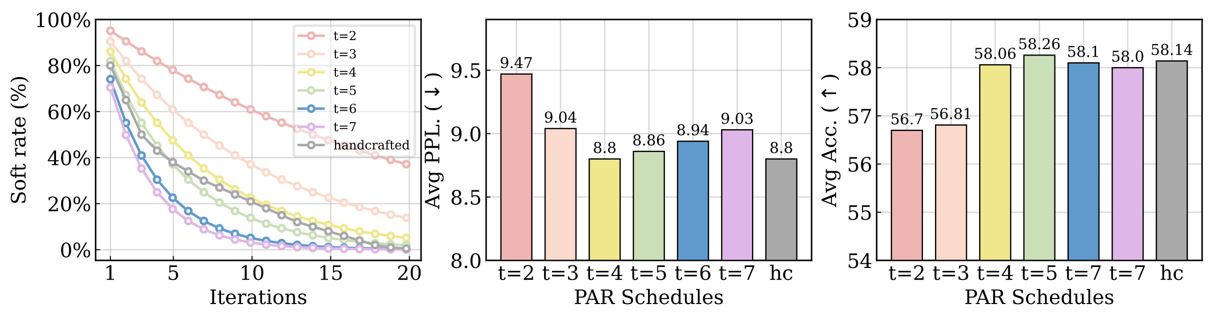

Figure 2: **Ablation study of PAR schedule.** We experiment several rule-based P adjustments and one handcrafted adjustment. *(AWQ baseline results: average PPL: 16.66, average acc.: 50.52).*

Table 5: **Comparison of weight memory compression and inference throughput**. We measure LLaMA-3 series model under various bitwidth/backend. WM stands for weight memory, TP_n denotes inference throughput with a batch size of n (output token/s).

Model	BitWidth	Backend	WM	TP_1	TP_{16}
L3-8B	FP16	Pytorch	15GB	49.23	358.1
	W4A16g128	Exllama	5.5GB	57.54	361.1
	W2A16g128	Triton	3.9GB	165.3	545.5
L3-70B	FP16	Pytorch	OOM	N/A	N/A
	W4A16g128	Exllama	39GB	26.23	86.94
	W2A16g128	Triton	21GB	4.93	54.35

ious kernels that support uniform quantization on various devices.

5 Related Work

Quantization has been a primary method to compress and accelerate off-the-shelf large models. Survey papers by (Gholami et al., 2022) and (Nagel et al., 2021) have systematically summarized the progress of quantization. Here, we list several major quantization works, especially for LLMs.

Post-Training Quantization for LLMs. While Quantization aware Training (QAT) guarantees better task performance in low-bit quantization, PTQ is more suitable for LLM due to its less reliance on computing resources and training data. PTQ methods like (Frantar et al., 2022; Lin et al., 2023; Wei et al., 2022, 2023; Shao et al., 2023; Chee et al., 2023; Liu et al., 2023a) improve the uniform quantization performance by optimizing weights, transformation scales, and clipping ranges. Our method continues improving the uniform quantization effect by incorporating rounding optimization. Other works try to improve PTQ in LLMs in different ways. For example, AQLM and GPTVQ (Egiazarian et al., 2024; van Baalen et al., 2024) explore non-uniform quantization schemes for weight-only quantization, which may better match the distribution of weights. LLM.int8 (Dettmers et al., 2022), BiLLM (Huang et al., 2024a), SiLLM (Huang et al., 2024b) apply mixed-precision quantization to keep salient weights in high precision and maintain the accuracy. However, these methods cannot be applied to quantize activations and thus cannot support integer MatMul. QuaRot (Ashkboos et al., 2024), SpinQuant (Liu et al., 2024) target activation outliers and eliminate them through the rotation matrix. We have demonstrated that our method can also be combined with them.

QAT for LLM. Recent works also explore QAT-based quantization for LLMs. To reduce data access, LLM-QAT (Liu et al., 2023b) generates language data for data-free QAT. To prevent massive weight memory usage, Q-LoRA (Dettmers et al., 2023) applies quantization-aware low-rank adaptation for finetuning. Recently, BitNet and BitNet b.158 (Wang et al., 2023; Ma et al., 2024) trained a 1-bit and 1.58-bit model from scratch, enabling multiplication-free LLM. However, these methods are hard to scale up due to the massive memory and computation requirements, especially for more than 70B models. As a result, they only focus on 1B$\sim$3B-scale models.

6 Conclusion

In this paper, we have proposed TesseraQ, a PTQ method for effectively calibrating large language models. Based on block reconstruction, TesseraQ optimizes weight rounding through a progressive approach that iteratively hardens and softens the rounding variables. Together with dequantization scale tuning, TesseraQ can be seamlessly combined with other PTQ methods like transformation, clipping, and rotation, to reach compelling performance. We demonstarte TesseraQ's superiority on open source LLaMA models. TesseraQ pushes the performance boundaries of quantized LLMs, in terms of perplexity, downstream accuracy, and hardware performance.

Limitations

TesseraQ shares some limitations in terms of algorithm runtime, which may require longer processing time than existing baselines. For example, the LLaMA-2-7B takes 3∼6 hours to finish the calibration process, while for AWQ/GPTQ, the calibration time is around 0.5 hours. Nevertheless, compared to QAT, our method still exhibits remarkable resource efficiency in required data and GPU memory. We leave how to accelerate rounding optimization in our future directions.

References

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. 2024. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*.

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

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. 2023. Quip: 2-bit quantization of large language models with guarantees. *arXiv preprint arXiv:2307.13304*.

Wenhua Cheng, Weiwei Zhang, Haihao Shen, Yiyang Cai, Xin He, Kaokao Lv, and Yi Liu. 2023. Optimize weight rounding via signed gradient descent for the quantization of llms. *arXiv preprint arXiv:2309.05516*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*.

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

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. 2024. Extreme compression of large language models via additive quantization. *arXiv preprint arXiv:2401.06118*.

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

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2022. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC.

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

Wei Huang, Haotong Qin, Yangdong Liu, Yawei Li, Xianglong Liu, Luca Benini, Michele Magno, and Xiaojuan Qi. 2024b. Slim-llm: Salience-driven mixed-precision quantization for large language models. *arXiv preprint arXiv:2405.14917*.

Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. 2020. Improving post training neural quantization: Layer-wise calibration and integer programming. *arXiv preprint arXiv:2006.10518*.

JonathanSalwan. Jonathansalwan/triton: Triton is a dynamic binary analysis library. build your own program analysis tools, automate your reverse engineering, perform software verification or just emulate code.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. 2021. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*.

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

Jing Liu, Ruihao Gong, Xiuying Wei, Zhiwei Dong, Jianfei Cai, and Bohan Zhuang. 2023a. Qllm: Accurate and efficient low-bitwidth quantization for large language models. *arXiv preprint arXiv:2310.08041*.

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

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. 2024. Spinquant–llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*.

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

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

Meta. 2024. Introducing llama 3.1: Our most capable models to date.

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

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. 2021. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*.

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

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

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

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

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

Mart van Baalen, Andrey Kuzmin, Markus Nagel, Peter Couperus, Cedric Bastoul, Eric Mahurin, Tijmen Blankevoort, and Paul Whatmough. 2024. Gptvq: The blessing of dimensionality for llm quantization. *arXiv preprint arXiv:2402.15319*.

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

Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. 2023. Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *arXiv preprint arXiv:2304.09145*.

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

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, et al. 2024. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*.

A Ablation Study on Rounding Optimization

In order to demonstrate the effectiveness of our proposed PAR, we compare our method with the several rounding optimization variants here.

AdaRound (Nagel et al., 2020). For AdaRound, the optimization is formulated by

$$\begin{aligned}&\min_{\nu} \ ||\hat{\mathbf{W}}\mathbf{X} - \mathbf{W}\mathbf{X}||_F^2 + \lambda \sum_{i,j} 1 - |2\sigma(\nu_{i,j}) - 1|^{\beta},\\ &\text{s.t. } \hat{\mathbf{W}} = \mathbf{s} \times (\mathbf{W}^q - \mathbf{z}),\\ &\mathbf{W}^q = \text{clamp}\Big(\lfloor \frac{\mathbf{W}}{\mathbf{s}} \rfloor + \sigma(\nu) + \mathbf{z}, 0, 2^N - 1\Big).\end{aligned} \tag{10}$$

This method utilizes the layer-wise reconstruction objective and a regularization loss. Both λ and β control the strength of the regularization loss during optimization, which encourages the rounding variables to move towards 0 and 1.

AdaQuant (Hubara et al., 2020). This method directly utilizes the STE method to optimize the weighs, given by

$$\begin{aligned}&\min_{\mathbf{V}} \ ||\hat{\mathbf{W}}\mathbf{X} - \mathbf{W}\mathbf{X}||_F^2,\\ &\text{s.t. } \hat{\mathbf{W}} = \mathbf{s} \times (\mathbf{W}^q - \mathbf{z}),\\ &\mathbf{W}^q = \text{clamp}\Big(\lfloor \frac{\mathbf{W}+\mathbf{V}}{\mathbf{s}} \rceil + \mathbf{z}, 0, 2^N - 1\Big),\\ &\frac{\partial \lfloor x \rceil}{\partial x} = 1.\end{aligned} \tag{11}$$

Note that AdaRound and AdaQuant have not implemented their method on LLMs before. Therefore, we implement their method on our own and use the default hyper-parameters in their paper. Specifically, we experiment with the LLaMA-2-7B W2A16g128 quantization case, where the model is uniformly initialized from the AWQ checkpoint. Each weight tensor will be optimized for 5000 iterations for a fair comparison. We compare 3 methods, AdaRound, AdaQuant, and our PAR, with either layer-wise objective (Eq. (2)) or block-wise objective (Eq. (3)). For AdaRound, we set the learning rate the same as our method and while for AdaQuant the learning rate was 1e-5. The results are shown in the Table below.

Generally, we find that PAR consistently outperforms the other two rounding methods regardless of which objective. We think the reason is that we explicitly control the hardness of rounding variables through the progressive approach. While AdaRound and AdaQuant, they are less optimized on LLMs and may require more hyper-parameter search.

Table 6: **Ablation study on rounding method**. The results are reported on LLaMA-2-7B W2A16g128 quantization.

Rounding Method	Objective	WT2(↓)	C4(↓)
None (AWQ)	Layer	14.65	18.67
AdaRound	Layer	10.68	15.67
AdaQuant	Layer	16.78	21.34
PAR	Layer	9.43	12.79
None (OmniQuant)	Block	11.06	16.34
AdaRound	Block	9.05	11.45
AdaQuant	Block	10.05	14.87
PAR	Block	6.82	10.77

B More Experimental Results

In this section, we include additional experimental results from the main section.

B.1 Detailed Accuracy of W4A4/W3A3 Quantization

Table 10 provides the detailed accuracy of each zero-shot tasks in W4A4/W3A3 quantization.

B.2 Results on C4

We demonstrate the perplexity results on the C4 datasets in Table 7. Note that the OmniQuant results are re-evaluated using the official checkpoint, which is slightly higher than the original paper results (Shao et al., 2023). Since the evaluation protocol can be different across different papers, we ensure use of the same evaluation protocol to compare different methods. Note, we restrict all models here from using the WikiText2 calibration data as the calibration data will affect the perplexity metric as shown in our ablation study. The improvements of our method over existing approaches are consistent with the results on the WikiText2 dataset.

Results on Smaller-Size LLM for Edge Inference. In addition to LLMs that are deployed on GPUs, we also test the performance of smaller-size LLMs geared for edge devices. We test LLaMA-3.2-1/3B models and compare them with AWQ in Table 6. We observe that our method significantly outperforms AWQ across different bitwidths in WikiText2 perplexity and average downstream task performance.

B.3 W4A8 Quantization

We also provide the W4A8 quantization in Table 8. Overall we find a small difference in W4A8 quantization due to the 8-bit per-token activation quantization.

Table 7: **Weight-only quantization results of LLaMA-1 and LLaMA-2 Models**. We report C4 perplexity in this table. *, † means initialized from AWQ, and OmniQuant, respectively.

LLaMA1&2 / PPL↓		1−7B	1−13B	1−30B	1−65B	2−7B	2−13B	2−70B
FP16	-	7.08	6.61	5.98	5.62	6.97	6.46	5.52
W2A16	RTN	1.3e5	5.6e4	2.7e4	2.2e4	4.8e4	7.2e4	2.4e4
	GPTQ	689.13	2.5e3	169.80	40.58	NAN	323.12	48.82
	OmniQuant	26.03	18.94	14.55	11.47	90.64	26.76	13.33
	TesseraQ†	**13.28**	**11.43**	**10.81**	**8.52**	**14.82**	**11.96**	**9.15**
W2A16 g128	RTN	1.0e3	447.64	99.45	17.15	4.9e3	139.65	42.13
	GPTQ	27.71	15.29	11.93	11.99	33.70	20.97	NAN
	AWQ	16.35	12.93	10.07	8.78	18.67	11.88	8.49
	OmniQuant	14.06	11.27	10.37	8.65	16.34	12.14	9.33
	TesseraQ*	**10.64**	**9.36**	**8.36**	**7.64**	**10.77**	**9.48**	**7.63**
W2A16 g64	RTN	151.43	76.00	30.07	11.34	475.35	28.69	13.43
	GPTQ	17.71	11.70	9.92	10.07	19.40	12.48	NAN
	AWQ	13.47	11.35	9.12	8.11	15.13	10.85	7.77
	OmniQuant	12.79	10.60	9.46	8.18	13.79	11.02	8.61
	TesseraQ*	**10.32**	**9.05**	**8.18**	**7.48**	**10.50**	**9.23**	**7.44**
W3A16	RTN	28.26	13.22	28.66	12.79	402.35	12.51	10.02
	GPTQ	9.49	8.16	7.29	6.71	9.81	8.02	6.57
	AWQ	11.16	8.37	7.91	8.62	16.25	8.90	6.50
	OmniQuant	8.73	7.68	6.86	6.31	9.24	7.89	6.31
	TesseraQ*	**8.15**	**7.38**	**6.60**	**6.16**	**8.30**	**7.41**	**6.08**

Table 8: **Weight-activation quantization Results of various LLMs**. We report the accuracy of 5 reasoning tasks (↑).

Models	**Bitwidths**	**Methods**	**PiQA**	**ArcE**	**ArcC**	**HellaSwag**	**WinoGrande**	**Avg.**
LLaMA-7B	FP16	-	77.47	52.48	41.46	73.00	67.07	62.30
	W4A8	SmoothQuant	75.19	70.45	37.45	51.06	64.87	59.81
		OS+	78.42	74.49	40.61	55.53	69.37	63.75
		AWQ	77.63	73.31	41.89	55.50	69.85	63.65
		TesseraQ*	**78.89**	**75.33**	**41.55**	**56.11**	**69.14**	**64.21**
LLaMA-2-7B	FP16	-	78.07	76.34	43.51	57.17	69.21	64.87
	W4A8	SmoothQuant	75.24	70.95	38.39	51.30	63.85	59.95
		Outlier Supp.+	77.09	74.74	42.57	56.37	68.51	63.86
		AWQ	77.09	74.36	42.32	56.25	69.53	63.91
		TesseraQ*	**77.42**	**76.26**	**41.63**	**56.42**	**69.22**	**64.19**
LLaMA-3-8B	FP16	-	79.54	80.09	50.17	60.13	73.24	68.64
	W4A8	SmoothQuant	71.98	66.37	34.55	50.46	67.40	58.16
		Outlier Supp.+	77.91	78.78	48.03	58.83	72.53	67.22
		AWQ	79.00	78.40	48.63	58.81	72.45	67.46
		TesseraQ*	**78.99**	**79.88**	**47.61**	**59.09**	**72.77**	**67.67**
Mistral-8B	W4A8	SmoothQuant	79.59	77.56	46.50	57.62	71.11	66.48
		OS+	80.35	79.04	48.03	60.18	72.45	68.02
		AWQ	79.92	79.79	47.35	58.80	74.26	68.03
		TesseraQ*	**80.36**	**79.92**	**49.57**	**60.54**	**73.79**	**68.84**

Table 9: **Weight-activation quantization Results of Mistral-7B**. We report the accuracy of 5 reasoning tasks (↑).

Models	**Bitwidths**	**Methods**	**PiQA**	**ArcE**	**ArcC**	**HellaSwag**	**WinoGrande**	**Avg.**
Mistral-7B	FP16	-	80.68	80.93	50.42	61.26	73.79	69.42
	W2A16 g128	GPTQ	64.20	45.74	22.35	36.68	55.02	44.80
		AWQ	68.44	56.73	27.44	40.60	56.03	49.06
		SignRound	75.84	70.88	30.73	50.87	62.90	58.24
		TesseraQ*	**76.87**	**71.67**	**39.59**	**54.09**	**68.11**	**62.07**
	W3A16 g128	GPTQ	79.70	78.70	48.41	59.15	71.98	67.19
		AWQ	80.19	78.62	45.56	58.28	71.58	66.85
		SignRound	79.54	78.70	46.33	59.60	72.85	67.40
		TesseraQ*	**79.59**	**78.36**	**47.44**	**59.87**	**71.98**	**67.45**
	W4A4	SmoothQuant	57.94	35.14	21.75	30.51	48.30	38.73
		OS+	66.70	56.73	30.20	42.39	52.01	49.61
		AWQ	66.26	54.16	30.80	43.45	53.67	49.67
		TesseraQ*	**72.19**	**65.90**	**33.78**	**49.02**	**57.61**	**55.71**

Table 10: **Detailed W4A4/W3A3 quantization results on each commonsense tasks of LLaMA-1/2/3**. We use per-channel weight quantization and per-token activation quantization *, † means initialized from AWQ, QuaRot.

Models	**Bitwidths**	**Methods**	**PiQA**	**ArcE**	**ArcC**	**HellaSwag**	**WinoGrande**	**Avg.**
LLaMA-7B	FP16	-	78.67	75.33	41.80	56.96	69.85	64.53
	W4A4	SmoothQuant	55.49	31.22	21.16	27.31	49.88	37.02
		OS+	67.46	57.74	31.05	41.83	54.38	50.50
		AWQ	65.56	57.36	26.10	9.02	53.98	48.41
		OmniQuant	66.15	45.20	31.14	56.44	53.43	50.47
		QLLM	68.77	45.20	31.14	57.43	56.67	51.84
		TesseraQ*	**71.98**	**64.77**	**32.67**	**47.59**	**60.22**	**55.45**
		Atom (W4A4g128)	76.28	52.10	38.99	69.81	63.69	60.17
		QuaRot	71.70	64.81	30.88	48.25	61.24	55.38
		GPTQ†	**76.55**	72.60	37.11	53.67	**66.93**	61.37
		TesseraQ†	76.22	**73.31**	**39.25**	**54.45**	66.38	**61.92**
	W3A3	Atom (W3A3g128)	65.56	41.41	30.72	53.19	55.56	49.28
		QuaRot	52.72	26.09	20.82	26.06	51.93	35.53
		GPTQ†	**68.98**	**58.92**	26.87	**43.90**	55.40	50.82
		TesseraQ‡	68.93	57.78	**27.30**	43.24	**58.24**	**51.10**
LLaMA-2-7B	FP16	-	78.07	76.34	43.51	57.17	69.21	64.87
	W4A4	SmoothQuant	53.04	25.71	20.22	25.71	51.77	35.29
		OS+	66.86	56.52	29.60	41.93	56.19	50.23
		AWQ	64.80	53.87	30.20	43.11	57.93	49.99
		OmniQuant	65.94	43.94	30.80	53.53	55.09	49.86
		QLLM	67.68	44.40	30.89	58.45	56.59	51.60
		TesseraQ*	**70.89**	**63.34**	**32.93**	**48.28**	**60.14**	**55.12**
		QuaRot	66.54	55.51	25.76	37.80	52.25	47.57
		GPTQ†	75.89	71.96	39.85	**54.12**	65.43	61.45
		TesseraQ*	**76.22**	**74.20**	**39.50**	53.80	**65.03**	**61.75**
	W3A3	QuaRot	51.74	25.54	22.86	25.84	49.88	35.18
		GPTQ†	64.31	47.21	22.18	36.08	53.27	44.62
		TesseraQ‡	**68.28**	**56.82**	**28.58**	**41.96**	**55.01**	**50.13**
LLaMA-3-8B	FP16	-	79.54	80.09	50.17	60.13	73.24	68.64
	W4A4	SmoothQuant	54.24	27.90	19.79	26.87	51.61	36.09
		OS+	57.34	40.99	20.22	33.19	51.77	40.71
		AWQ	59.68	44.90	22.09	34.53	51.30	42.51
		TesseraQ*	**67.08**	**59.09**	**27.13**	**43.88**	**57.14**	**50.87**
		QuaRot	69.85	58.03	28.07	43.37	59.82	51.83
		GPTQ†	76.22	73.94	41.21	55.47	67.48	62.87
		TesseraQ†	**77.64**	**77.27**	**44.80**	**56.03**	**69.85**	**65.12**
	W3A3	QuaRot	52.28	26.59	20.56	26.11	50.67	35.25
		GPTQ†	56.96	33.62	20.47	28.87	49.40	37.87
		TesseraQ†	**66.05**	**51.59**	**24.40**	**40.59**	**53.98**	**47.33**
LLaMA-3-70B	FP16	-	83.13	87.12	60.92	66.47	79.56	75.44
	W4A4	SmoothQuant	57.45	38.46	24.23	30.22	54.93	41.06
		OS+	53.04	25.79	22.01	25.88	48.85	35.12
		AWQ	69.91	61.71	34.04	47.98	54.61	53.65
		TesseraQ*	**78.29**	**69.15**	**38.12**	**53.74**	**61.16**	**60.09**
		QuaRot	57.88	36.36	19.02	28.13	53.19	38.92
		GPTQ†	79.76	80.17	50.59	60.71	73.08	68.87
		TesseraQ†	**81.84**	**82.64**	**54.07**	**63.90**	**65.64**	**69.62**
	W3A3	QuaRot	52.06	24.87	20.05	25.55	49.25	34.26
		GPTQ†	55.98	34.80	19.45	28.38	51.46	38.02
		TesseraQ†	**74.80**	**66.03**	**36.42**	**51.34**	**58.43**	**57.42**