# Dual Grained Quantization: efficient fine-grained quantization for LLM

Anonymous ACL submission

## Abstract

Large Language Models (LLMs) demonstrate considerable potential across a range of tasks; however, they pose significant challenges due to their extensive memory requirements and computational demands. Fine-grained quantization effectively preserves model performance during aggressive weight compression, yet its inefficiency on hardware platforms hinders its applicability in real-world production environments. To enhance hardware efficiency while preserving the performance of fine-grained quantization, we propose a novel quantization framework, Dual Grained Quantization (DGQ), employing a W4A8 configuration specifically tailored for LLMs. By employing a dual-phase search strategy, DGQ minimizes quantization error without significantly extending quantization time. To improve the accuracy of W4A8-configured LLMs, we introduce aggressive selective equalization. This approach is grounded in the observation that key weights and outliers frequently coexist within the same channels. Comprehensive experiments with our W4A8 CUDA kernel highlight DGQ's exceptional performance, delivering speedups of 1.37× and 2.5× over standard INT8 and FP16 kernels, respectively, while preserving the superior performance of fine-grained quantization.

## 1 Introduction

Large Language Models (LLMs) such as GPT-4 (Bubeck et al., 2023) and LLaMA (Touvron et al., 2023a,b) have excelled in comprehending and generating natural language. However, these models have become much larger. For instance, LLaMA-65B (Touvron et al., 2023a) is approximately 190× larger than Bert-Large (Devlin et al., 2018), necessitating around 130 GB of memory storage, which requires 2×80GB NVIDIA A100 during inference. The deployment of LLMs presents notable challenges, particularly in the allocation of substantial computing and storage resources.

Given the challenges outlined, network quantization (Krishnamoorthi, 2018) propose to map weights and/or activations to lower-bit representations, significantly reduces memory footprint and boosts inference. For LLMs, the quantization schemes that have received the most attention include W4A16 (Lin et al., 2023; Frantar et al., 2022) and W8A8 (Shao et al., 2023; Xiao et al., 2023; Yao et al., 2022). W4A16 reduces the memory usage and transfer durations by converting weight into 4-bit, albeit introducing dequantization overhead and retaining FP16 calculations. W8A8 accelerates the matrix multiplication(MatMul) by leveraging INT8 kernels, yet has a minimal impact on reducing the memory footprint associated with weight storage. To harness the benefits of both, more compact W4A8 (Yao et al., 2023; Li et al., 2023b,a) and W4A4 with mixed precision (Ashkboos et al., 2023; Zhao et al., 2023) are proposed. To preserve model performance, W4A8 methods apply fine-grained quantization, inserting FP16 accumulation within integer MatMul. Concurrently, W4A4 with mixed precision introduces a dynamic retention strategy, selectively keeping critical activations and weights in high precision to preserve model efficacy. Despite these advancements, such methods lead to slower inference times by interrupting the efficient integer MatMul computation pipeline within existing computing units. Figure 1 illustrates the performance across various quantization schemas, indicating that current methods struggle with calculations across diverse token sequences.

In this paper, we propose **D**ual **G**rained **Q**uantization (DGQ), a hardware-efficient fine-grained quantization framework for LLMs. To eliminate float addition within integer MatMul for group-wise quantization, we address the group-wise quantization challenge by implementing group-wise quantization of weights prior to the MatMul process by converting 4-bit weights back to 8-bit representations. Following this adjust-

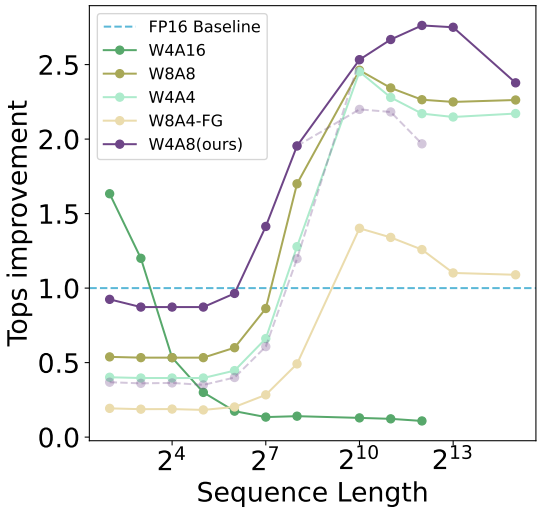


Figure 1: **Speed Comparison of different quantization schemes to FP16 kernels.** The curve of DGQ is formed by joining two distinct lines: the former represents the kernel for the pre-fill phase, and the latter illustrates the kernel for the self-decoding phase.

ment, the modified procedure facilitates the execution of continuous INT8 MatMul and channel-wise dequantization. To minimize the quantization error and eliminate quantization clipping overhead, we propose a dual-phase search algorithm tailored to hardware constraints. Additionally, to further reduce the overhead associated with group-wise weight dequantization, we have developed two W4A8 kernels, each optimized for handling either long or short sequence lengths.

To maintain the model accuracy, we propose aggressive selective equalization(ASE), enhancing the performance of static quantization for activation. Our method is based on the observation that salient weight and outliers often manifest in the same channel. Smoothing the outliers could reduce the relative quantization error of salient weight, benefiting the quantization for both activations and salient weights. To reduce the impact on less significant weights, we strategically reduce the smoothing channel, exclusively choosing channels containing outliers for the smoothing process.

In summary, our key contributions include:

- The Dual Grained Quantization (DGQ) framework enhances W4A8 LLM inference through an optimized dual-grained schema, two inference-stage kernels, and a rapid, gradient-free method.

- Aggressive Selective Equalization strategy that selectively smoothens channels.

- Extensive DGQ evaluation shows significant performance and efficiency improvements, achieving up to 2.5× and 1.37× faster speeds than FP16 and W8A8 implementations with negligible loss of accuracy, respectively.

## 2 Related Works

### 2.1 Model Quantization

Quantization falls into two primary categories: Quantization-Aware Training (QAT) (Martinez et al., 2018; Esser et al., 2019) and Post-training Quantization (PTQ) (Choukroun et al., 2019; Li et al., 2021; Wei et al., 2022). QAT fine-tunes quantized models with the full dataset, maintaining accuracy but requiring complex computations, thus being less ideal for LLMs. Conversely, PTQ applies quantization with minimal data and computational demand. Techniques such as AdaRound (Nagel et al., 2020) and Adaquant (Hubara et al., 2020) optimize quantization parameters and distill quantized models layer by layer. Some approaches (Qin et al., 2022; He et al., 2022) push the boundaries by compressing transformations into binary values.

### 2.2 Quantization on LLMs.

Given that QAT for LLM demands substantial data and computational resources, the predominant quantization techniques for LLM lean towards PTQ. Current research primarily explores two quantization scenarios: (1) weight and activation quantization (Yao et al., 2022; Xiao et al., 2023; Wei et al., 2023; Liu et al., 2023b; Chai et al., 2023; Yuan et al., 2023; Li et al., 2023b; Liu et al., 2023a; Ashkboos et al., 2023), (2) weight only quantization (Frantar et al., 2022; Lin et al., 2023; Dettmers et al., 2023a,b; Lee et al., 2023). Notably, QLLM (Liu et al., 2023a) and QUIK (Ashkboos et al., 2023) push the limit of LLM quantization into 4 bits without significant quantization errors. Both LLM.int8() (Dettmers et al., 2022) and QUIk (Ashkboos et al., 2023) preserve the outliers with high precision to maintain model performance. Maintaining outliers at high precision incurs additional overhead during inference. For W4A8 quantization, ZeroQuantV2 (Yao et al., 2023) concludes, based on extensive experimentation, that fine-grained quantization is crucial for preserving model performance in W4A8 quantization schemes. ZeroQuantFP (Wu et al., 2023) employs float quantization within a W4A8 framework, and FPTQ (Li et al., 2023b) presents an adap-

tive quantization granularity strategy for activation quantization, aimed at minimizing inference overhead. Accelerating fine-grained weight quantization remains a challenging problem that requires further investigation.

QLoRA (Dettmers et al., 2023a) introduces the NF4 double quantization technique, which improves memory utilization by converting FP32 group-wise quantization scales to FP8 and employing a channel-wise FP32 scale. This approach prioritizes memory efficiency over computational efficiency illustrating that direct conversion of quantization scales to FP8 does not significantly impact accuracy. In our proposed methodology, we place significant emphasis on enhancing inference efficiency. This involves initially dequantizing UINT4 weights back to INT8 weights, thereby facilitating INT8 General Matrix Multiply (GEMM) operations to expedite the inference process.

## 3 Preliminary

Quantization is an essential process that converts high-precision values into low-precision precision formats. In our research, we focus on the application of uniform integer quantization, which is advantageous due to its superior hardware support and increased computational efficiency. The process of asymmetric quantization is mathematically represented as:

$$\begin{aligned} \widehat{\mathbf{X}} &= \mathbf{clamp}(\left\lfloor \frac{\mathbf{X}}{\mathbf{s}} \right\rceil + \mathbf{zp}, 0, 2^N - 1), \\ \mathbf{zp} &= \left\lfloor \frac{\mathbf{X_{min}}}{\mathbf{s}} \right\rceil, \quad \mathbf{s} = \frac{\mathbf{X_{max}} - \mathbf{X_{min}}}{2^N - 1} \end{aligned} \tag{1}$$

Here, $\mathbf{X}$ denotes the full-precision tensor, $\widehat{\mathbf{X}}$ represents its quantized version, $s$ is the quantization scale, and $N$ signifies the bit width. We denote the quantization process as $\mathcal{Q}(\mathbf{X}, \mathbf{s}, \mathbf{zp})$, where $\mathbf{s}$ represents the scale and $\mathbf{zp}$ the zero point. For symmetric cases, the zero-point $\mathbf{ZP}$ is assigned a value of zero, and the quantization scale $\mathbf{s}$ is calculated using the formula $\mathbf{max}(|\mathbf{X}|)/(\mathbf{2^{N-1}} - \mathbf{1})$.

**Different Levels of Quantization Granularity.** For weight quantization, granularity is differentiated into two distinct levels: channel-wise and group-wise quantization, as depicted in Figure 2 (a) and (b). Compared to channel-wise quantization, vanilla group-wise quantization requires float addition calculation across groups, thereby diminishing efficiency. For activation quantization, two distinct granularities are identified: token-wise and tensor-wise. Token-wise quantization requires the dynamic computation of the quantization scale for each token, which introduces additional overhead during inference. In our framework, we use static quantization for efficient inference.

**Different Inference Phase of LLMs.** There are two phases for LLM inference, the pre-fill phase (calculating key-value (KV) cache with long sequence input) and the self-decoding phase (generating the word with KV cache). In the pre-fill phase, the LLMs usually deal with long-sequence context while the LLMs deal with only one word during the self-decoding phase.

**Scaling Equalization.** Migrating quantization difficulty from activation/weight to weight/activation reduces quantization error in coarse-grained quantization. This is captured by the equation:

$$\mathbf{X}' = \boldsymbol{diag}(\mathbf{s})^{-1}\mathbf{X}, \mathbf{W}' = \boldsymbol{diag}(\mathbf{s})\mathbf{W} \tag{2}$$

## 4 Dual Grained Quantization

### 4.1 Inference Schema

To enhance the hardware efficiency of fine-grained quantization and avoid group-wise float scales, we propose dual-grained quantization (DGQ), see Figure 2 (c). This approach incorporates a pivotal dequantization step preceding the GEMM operation. It entails the conversion of UINT4 weights into INT8 weights, thereby facilitating the execution of INT8 GEMM operations. DGQ employs a two-step quantization process: initially, quantization is performed using a channel-wise FP16 scale $\mathbf{s^{(1)}}$, followed by a group-wise INT8 scale $\mathbf{S^{(2)}}$ for the second step. Furthermore, the second-step dequantization can be integrated with the process of loading UINT4 weights into INT8 kernels, streamlining the procedure. For the scenario described, involving hidden states $\mathbf{X}_{s8} \in \mathbb{N}_{s8}^{b \times h}$ and a weight matrix$\mathbf{W}_{u4} \in \mathbb{N}_{u8}^{h \times o}$, the quantization methodology is characterized by the following equations:

$$\begin{aligned} \mathbf{s}_{f16} &= \mathbf{s_X} \cdot \mathbf{s^{(1)}}, \\ \mathbf{W}_{s8} &= \mathbf{S^{(2)}} \cdot (\mathbf{W}_{u4} + \mathbf{ZP}_{u4}), \\ \mathbf{O}_{f16} &= \mathbf{s}_{f16} \cdot (\mathbf{X}_{s8}\mathbf{W}_{s8}), \end{aligned} \tag{3}$$

Here, $\mathbf{S^{(2)}} \in \mathbb{N}_{s8}^{g \times o}$ represents the group-wise quantization scales, $\mathbf{ZP}_{u4} \in \mathbb{N}_{u4}^{g \times o}$ represents the group-wise quantization zero points and $\mathbf{s^{(1)}} \in \mathbb{R}_{f16}^{o}$ represents channel-wise quantization scales. $o$ is the number of output channels, $\mathbf{n_g}$ signifying the number of groups in group-wise quantization,

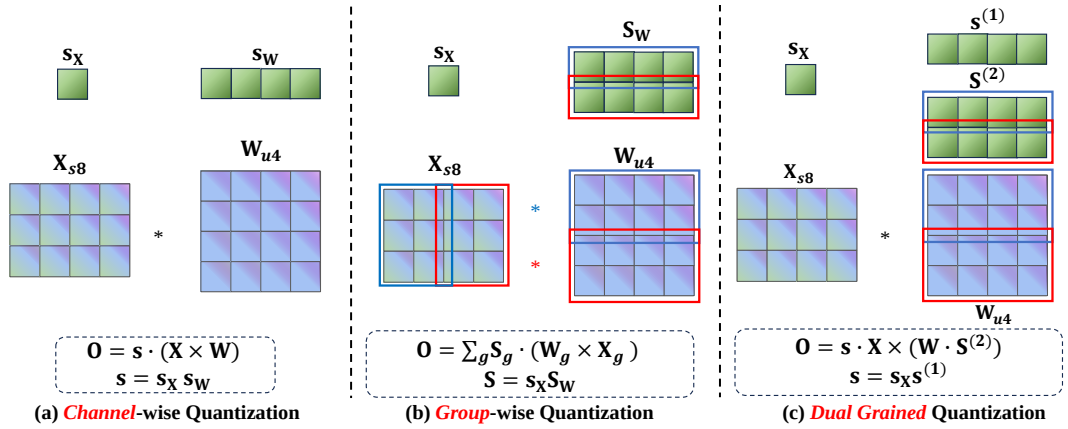


Figure 2: **Different grained quantization method.** Unlike channel-wise and DGQ quantization, group-wise quantization tends to result in FP16 accumulation, which conflicts with typical hardware design.

and $\mathbf{g}$ denoting the size of each group within the group-wise quantization (where the product of $\mathbf{n_g}$ and $\mathbf{g}$ equals $\mathbf{h}$, the number of input channels).

### 4.2 Dual-phase Search

DGQ requires to quantize the group-wise FP16 scales into channel-wise FP16 scales $\mathbf{s^{(1)}}$ and group-wise INT8 scales $\mathbf{s^{(2)}}$. However, directly quantizing the FP16 scales to INT8 following Dai et al. (2021) results in a significant quantization error, as demonstrated in Table 1. The primary reason for this limitation is that the number of quantization bits allocated for the second step of fine-grained quantization is restricted to prevent overflow during the dequantization process. To address this limitation, we propose a novel dual-phase method to search proper dual-grained quantization scales.

Following prior studies (Choukroun et al., 2019; Lin et al., 2023), our method also employs the Mean Squared Error (MSE) of the output to calibrate, as delineated in equation 4. Given the extensive search space of quantization parameters, it is imperative to first determine an optimal fine-grained scale $\mathbf{S'}$ to serve as a constraint, thereby narrowing the search space. To further refine the model, we compute the MSE for each weight and activation group $\mathbf{W}_k$, $\mathbf{X}_k$, following the approach described in Equation 5.

| PPL ↓ | 1-7B | 1-13B | 1-30B | 2-7B | 2-13B |
|---|---|---|---|---|---|
| Original | 6.03 | 5.39 | 4.43 | 5.87 | 5.23 |
| VS-Quant | 6.93 | 6.00 | 4.83 | 2155.54 | 5.48 |

Table 1: **Wikitext2 Results for W4A8 LLaMA family: Original Group-Wise Quantization vs. VS-Quant.** Here, we use 4bit for $\mathbf{S_W}$ to avoid overflow.

$$\arg\min_{\mathbf{S'},\mathbf{ZP}} \|\mathbf{XW} - \widehat{\mathbf{X}}\mathcal{Q}(\mathbf{W}, \mathbf{S'}, \mathbf{ZP})\|^2. \quad (4)$$

$$\arg\min_{\mathbf{s'}_k,\mathbf{zp}_k} \|\mathbf{X}_k\mathbf{W}_k - \widehat{\mathbf{X}}_k\mathcal{Q}(\mathbf{W}_k, \mathbf{s'}_k, \mathbf{zp}_k)\|^2. \quad (5)$$

To refine our quantization scales further, we adjust the calibration process by altering the optimization objective to focus on $\mathbf{s^{(1)}}$. The optimization objective for this phase is depicted in Equation 6.

$$\arg\min_{\mathbf{s^{(1)}}} \|\mathbf{XW} - \widehat{\mathbf{X}}\mathcal{Q}_W(\mathbf{W}, \mathbf{S}, \mathbf{ZP})\|^2, \quad \mathbf{S^{(2)}} = \lfloor \frac{\mathbf{S'}}{\mathbf{s^{(1)}}} \rceil, \mathbf{S} = \mathbf{s^{(1)}} \cdot \mathbf{S^{(2)}} \quad (6)$$

For the second phase, our method continues to utilize the absolute maximum of $\mathbf{X}$ for grid search. This strategy is informed by the discrepancy between the absolute maxima of $\mathbf{S'}$ and $\mathbf{X}$, potentially leading to the neglect of relevant sections of the search space. To enhance inference efficiency through the avoidance of clipping, our optimization framework embeds constraints on the datatype's upper and lower limits, as specified in Equation 7.

$$\mathbf{W}_{u4} \in [\mathbf{0}, \mathbf{15}], \mathbf{S^{(2)}} \in [-\mathbf{128}, \mathbf{127}], \quad \mathbf{W}_{s8}\mathbf{S^{(2)}} \in [-\mathbf{128}, \mathbf{127}] \quad (7)$$

The rounding operation in the second phase of quantization, represented by $\lfloor \cdot \rceil$, may result in $\mathbf{S}$ surpassing $\mathbf{S'}$. This can lead to a reduction in the maximum weight values to less than 15. In exploring the search space, it's possible for the range of $\mathbf{S^{(2)}}$ to surpass 15. As a result, we have opted to loosen the constraints on $\mathbf{S^{(2)}}$, concentrating instead on the limitations for $\mathbf{W}_{s8}$ and $\mathbf{W}_{u4}$. Consequently, it suffices to apply clamping to $\mathbf{W}_{u4}$ as

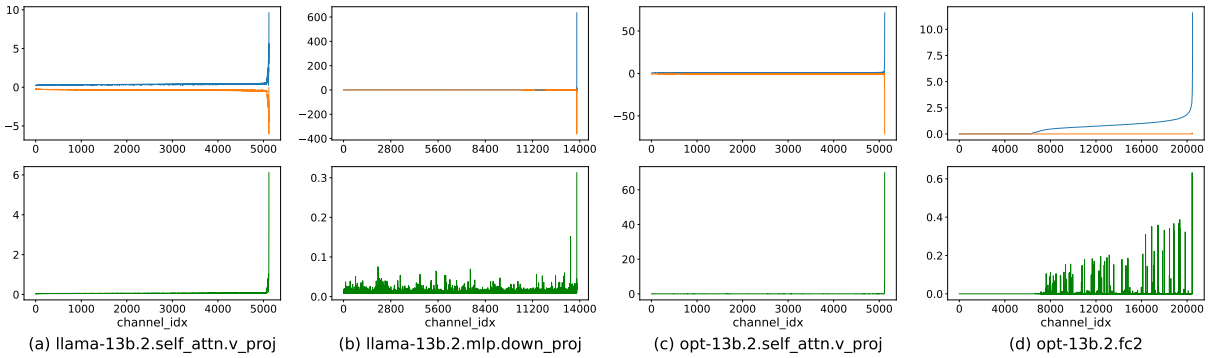


Figure 3: **Figures of maximum, minimum and mean values for different layers of different models.** Blue lines mean the maximum, orange lines present the minimum and green lines signify the mean value.

described in Equation 8.

$$\mathbf{W}_{u4} \in [\mathbf{max}(\mathbf{0}, \lfloor \frac{\mathbf{-127}}{\mathbf{S^{(2)}}} \rceil + \mathbf{ZP}), \mathbf{min}(\mathbf{15}, \lfloor \frac{\mathbf{127}}{\mathbf{S^{(2)}}} \rceil + \mathbf{ZP})] \tag{8}$$

### 4.3 Aggressive Selective Equalization

Unlike W8A8 (Xiao et al., 2023), 4-bit quantized weight is hard to cover the quantization difficulty migrated from activations. To reduce the quantization error associated with weights, our focus intensifies on salient weights (Lin et al., 2023), identified as those exerting the greatest influence on model accuracy. Given that the Hessian matrix of weights, calculated using $\mathbf{H} = \mathbf{X}\mathbf{X}^{\mathbf{T}}$, correlates directly with the absolute values of the inputs, it is possible for outlier activations and salient weights to coexist within the same input channel. To elucidate this point, Figure 3 illustrates both the maximum and mean activation values. For a systematic visualization, we organize the maximum values in ascending order and align the mean values accordingly, using the indices of the maximum values. This organization demonstrates that outliers in both maximum and mean values often correspond to the same channels, highlighting their variability across different models and layers. Therefore, it suffices to consider only the values of outliers when devising our equalization strategy. To prevent disruption to the distribution of other weights and those of preceding layers, our approach selectively smooths the channels containing outliers.

In our equalization strategy, we determine the smoothing scale $\mathbf{k}_j$ using the following formulas:

$$\mathbf{z}_j = \mathbf{max}(|\mathbf{X}_{[j,:]}|) \tag{9}$$

$$\mathbf{k}_j = \mathbf{clamp}(\mathbf{z}_j / \mathbf{max_p}(\mathbf{z}), \text{low} = 1) \tag{10}$$

Here, the top $\mathbf{p}$ of the highest values serve as the threshold for clipping. Outliers exceeding the $\mathbf{p}$ threshold are adjusted to this threshold, and the resulting scale is utilized to boost channels that are particularly sensitive to quantization. We conducted evaluations with various settings of $\mathbf{p}$, ultimately selecting $\mathbf{p} = 1.0\%$ for LLaMA models and $\mathbf{p} = 0.3\%$ for OPT models.

### 4.4 Optimization of Kernel Design

As shown in Figure 4(b), W4A16's dequantization necessitates extra computations prior to the MatMul operation, with weight dequantization occurring right after loading to reduce memory transfer time. This process is applied to each input individually. However, this approach requires repeated dequantization for sequences longer than one, introducing overhead that offsets the benefits of lower memory transfer time and thus impairs performance. Contrastingly, the W4A8 configuration, with fewer bit operations (BOPs) than W4A16, is expected to outperform due to its ability to process longer sequences before hitting computational limits. To address these challenges, we suggest moving the dequantization step to precede the MatMul operation for all inputs.

Figure 4(c) and (d) show our innovative kernel designs tailored for both the self-decoding and prefill phases. During the self-decoding phase, dequantization is seamlessly integrated into the MatMul process, following the vanilla W4A16 kernel design. However, we have increased the sequence length for each computation. While this adjustment may reduce performance in single-size self-decoding, it significantly improves performance for smaller sequences, as depicted in Figure 1. This scenario more accurately reflects typical real-world

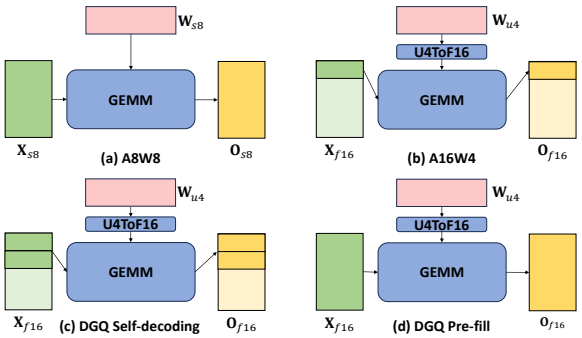


Figure 4: Design of our W4A8 Kernels (c) and (d) compared to W8A8(a) and W4A16(b).

applications. Transitioning to the pre-fill phase, the strategy involves separating dequantization from the MatMul operation. This allows for the dequantization of weights to occur only once per MatMul. Although this change is likely to incur substantial additional memory transfer overhead, it is justified by the pre-fill phase's handling of longer sequences. This overhead is effectively mitigated, facilitating an enhancement in performance.

# 5 Experiments

## 5.1 Experiments Setups

**Baseline.** We benchmark against established baselines within the W4A8 quantization setting, SmoothQuant (Xiao et al., 2023), LLM-QAT (Liu et al., 2023b) and FPTQ (Li et al., 2023b). Since the quantization granularity varies from different quantization methods, we detail the quantization granularity of each in Table 2.

| Method | activation | weight |
|---|---|---|
| SmoothQuant | token-wise | tensor-wise |
| LLM-QAT | token-wise | channel-wise |
| FPTQ | token/tensor-wise | group-wise |
| Ours | tensor-wise | dual-grain |

Table 2: **Quantization granularity of different methods.** FPTQ uses adaptive quantization granularity.

**Models and datasets.** We choose LLaMA[1] (Touvron et al., 2023a) and LLaMA2[2] (Touvron et al., 2023b) families to evaluate our quantization methods. For Common Sense Question Answers evaluation, we employ four zero-shot evaluation tasks: HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), Winogrande (Sakaguchi et al., 2021) and ARC (Clark et al., 2018). The benchmarking for Common Sense Question Answers is conducted using lm-eval (Lin and Chen, 2023). Additionally, we use WiKiText2 (Merity et al., 2016) and C4 (Raffel et al., 2020) to compare the generation ability. For calibration purposes, we select 128 samples from the PTB (Marcus et al., 1994) dataset to serve as the calibration dataset. Furthermore, to demonstrate the broad applicability of our approach, we evaluate its performance on OPT models.

**Implementation.** We implement our methods with Pytorch Huggingface for the proof of concept. We use the CUTLASS[3] GEMM kernels to develop our two-grained quantization kernels. All latency measurements in the paper are tested on NVIDIA RTX3090 24G GPUs.

## 5.2 Performance Evaluation

Table 3 presents the performance of our quantization methods across various tasks, including Wikitext2, C4, and Common Sense QA. Compared to FPTQ (Li et al., 2023b), our methods demonstrate equivalent or superior performance. FPTQ introduces extra dynamic quantization for specific layers and employs naive group-wise quantization approach, introducing extra inference overhead. Compared to LLM-QAT (Liu et al., 2023b), our methods exhibit enhanced performance. Notably, LLM-QAT, as a QAT approach, incurs considerable overhead in terms of computational resources. Table 4 presents a comparison on the MMLU task between our methods and FPTQ, demonstrating our method's superior performance over FPTQ on MMLU task. Additional Perplexity (PPL) test results are available in the Appendix B .

## 5.3 Efficiency Evaluation

**Comparison on End-to-End inference.** In Figure 5, we conduct a comparison of the end-to-end efficiency among LLaMA-7B, LLaMA-13B, and OPT-13B models with different quantization methods: SmoothQuant (W8A8), AWQ (W4A8), and our methods. We conducted tests using an input context length of 1024, generating 128 words across three batch sizes: 2, 4, and 6. In the pre-fill phase, our method achieves the highest acceleration ratio in comparison to both W8A8 and W4A16 configurations, attaining an acceleration ratio of up to 1.67 times relative to FP16. During the self-decoding phase, the W4A16 configuration exhibits superior performance, while our methods

[1]https://huggingface.co/huggyLLaMA/LLaMA-*b
[2]https://huggingface.co/meta-LLaMA/LLaMA-2-*b-hf
[3]https://github.com/NVIDIA/cutlass

| Size | Bit | Method | PPL ↓ | | Zero-Shot CSQA ↑ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | wikitext2 | c4 | PIQA | ARC-e | ARC-c | HellaSwag | Winogrande | **Avg.** |
| **1-7B** | FP16 | | 5.68 | 7.08 | 79.0 | 72.7 | 44.7 | 76.1 | 70.3 | 68.6 |
| | W8A8 | SmoothQuant | 5.89 | 7.23 | 79.0 | 72.2 | 44.5 | 75.2 | 70.1 | 68.2 |
| | W4A8 | LLM-QAT | - | - | 77.5 | 70.2 | 43.3 | 73.5 | 67.7 | 66.4 |
| | W4A8 | FPTQ | 5.95 | 7.37 | 78.4 | 70.8 | - | 74.5 | 70.0 | 67.5 |
| | W4A8 | DGQ | 6.02 | 7.43 | 78.2 | 71.6 | 44.2 | 73.5 | 68.4 | 67.2 |
| **1-13B** | FP16 | | 5.09 | 6.61 | 80.1 | 74.7 | 47.8 | 79.1 | 72.9 | 70.9 |
| | W8A8 | SmoothQuant | 5.21 | 6.72 | 79.7 | 73.9 | 47.6 | 78.3 | 72.1 | 70.3 |
| | W4A8 | LLM-QAT | - | - | 79.1 | 73.0 | 47.1 | 77.5 | 70.6 | 69.5 |
| | W4A8 | FPTQ | 5.35 | 6.83 | 79.3 | 72.7 | - | 77.5 | 72.1 | 69.7 |
| | W4A8 | DGQ | 5.39 | 6.93 | 79.7 | 73.3 | 47.4 | 77.5 | 71.3 | 69.8 |
| **1-65B** | FP16 | | 3.53 | 5.62 | 82.3 | 79.8 | 55.6 | 84.2 | 77.4 | 75.9 |
| | W8A8 | SmoothQuant | 3.73 | 5.93 | 81.1 | 77.4 | 54.4 | 82.3 | 74.9 | 74.04 |
| | W4A8 | FPTQ | 3.88 | 5.85 | 81.4 | 78.4 | - | 83.4 | 75.8 | 74.8 |
| | W4A8 | DGQ | 3.89 | 5.97 | 81.1 | 78.9 | 54.9 | 81.0 | 75.8 | 74.3 |
| **2-7B** | FP16 | | 5.47 | 6.97 | 79.1 | 74.5 | 46.3 | 76.0 | 69.4 | 69.1 |
| | W8A8 | SmoothQuant | 5.80 | 7.24 | 78.0 | 76.0 | 46.4 | 75.9 | 69.1 | 69.1 |
| | W4A8 | FPTQ | 5.85 | 7.35 | 78.0 | 72.8 | - | 74.9 | 69.4 | 67.8 |
| | W4A8 | DGQ | 5.87 | 7.44 | 78.4 | 73.0 | 44.3 | 74.0 | 68.0 | 67.5 |
| **2-13B** | FP16 | | 4.88 | 6.46 | 80.6 | 77.5 | 49.2 | 79.4 | 72.0 | 71.7 |
| | W8A8 | SmoothQuant | 5.04 | 6.55 | 79.5 | 77.3 | 49.1 | 79.3 | 72.1 | 71.5 |
| | W4A8 | FPTQ | 5.19 | 6.83 | 79.4 | 75.8 | - | 78.1 | 70.6 | 70.4 |
| | W4A8 | DGQ | 5.23 | 6.82 | 79.5 | 76.0 | 48.5 | 77.5 | 70.9 | 70.5 |
| **2-70B** | FP16 | | 3.31 | 5.52 | 82.7 | 81.0 | 57.3 | 83.8 | 78.0 | 76.6 |
| | W8A8 | SmoothQuant | 3.46 | 5.61 | 82.4 | 80.7 | 57.2 | 82.6 | 78.1 | 76.2 |
| | W4A8 | FPTQ | 3.64 | 5.78 | 82.4 | 79.9 | - | 82.6 | 77.0 | 75.7 |
| | W4A8 | DGQ | 3.68 | 5.89 | 82.0 | 80.2 | 57.0 | 81.3 | 76.9 | 75.5 |

Table 3: Comparison on WikiText2, C4 and Common Sense QA.

| **Acc↑** | Method | Humans | STEM | Social | Other | **Avg.** |
|---|---|---|---|---|---|---|
| 1-7B | FP16 | 33.60 | 31.10 | 38.20 | 38.40 | 35.20 |
| | FPTQ | 30.20 | 29.95 | 32.76 | 35.87 | 32.02 |
| | FP16 | 31.27 | 30.53 | 36.79 | 35.90 | 33.50 |
| | Ours | 28.96 | 30.22 | 35.91 | 34.24 | 32.20 |
| 1-13B | FP16 | 44.60 | 37.10 | 54.00 | 53.50 | 47.10 |
| | FPTQ | 40.96 | 34.19 | 49.72 | 49.75 | 43.46 |
| | FP16 | 40.73 | 38.31 | 54.60 | 54.04 | 47.06 |
| | Ours | 39.56 | 41.50 | 48.66 | 51.27 | 45.74 |
| 1-65B | FP16 | 61.80 | 52.00 | 73.30 | 67.60 | 63.50 |
| | FPTQ | 59.85 | 49.24 | 71.50 | 65.89 | 61.52 |
| | FP16 | 57.72 | 47.04 | 75.96 | 67.44 | 62.18 |
| | Ours | 55.60 | 44.86 | 72.11 | 63.53 | 59.57 |

Table 4: Comparison on MMLU.

| BS | Kernel | FP16 | W8A8 | W4A4 | Ours |
|---|---|---|---|---|---|
| 1 | (4096,4096) | 0.048 | 0.091 | 0.094 | **0.050** |
| | (5120,5120) | 0.070 | 0.111 | 0.094 | **0.059** |
| | (11008,4096) | 0.132 | 0.095 | 0.118 | **0.111** |
| | (4096,11008) | 0.121 | 0.226 | 0.134 | **0.104** |
| 2048 | (4096,4096) | 1.117 | 0.504 | 0.518 | **0.443** |
| | (5120,5120) | 1.597 | 0.663 | 0.655 | **0.598** |
| | (11008,4096) | 2.963 | 1.181 | 1.157 | **1.154** |
| | (4096,11008) | 3.146 | 1.200 | **0.981** | 1.075 |

Table 5: Single Kernel latency comparison.

surpass the efficiency of the W8A8 schema. Overall, our method achieves the highest acceleration ratio in end-to-end inference. Since our quantization schema is one of the important points in our paper, we only replace the linear layers into quantization layers and adapt the KV cache quantization, the end-to-end latency could be further reduced.

**Comparison of Single Kernel Performance.** Our kernel performance is evaluated in Table 8 against W8A8 (Xiao et al., 2023) and W4A4 with mixed precision kernels (Ashkboos et al., 2023). Considering the specific phases of self-decoding and prefilling, alongside varying kernel sizes in LLaMA models, our experimental analysis encompasses diverse comparisons. The results demonstrate that our kernels significantly outperform both W8A8 and W4A4 kernels. This superiority can be attributed to the fact that W4A4 kernel, when employing mixed precision, encounter delays due to additional dynamic quantization processes.

### 5.4 Ablation Study

**Different quantization granularity for W4A8 LLaMA models.** In our experiments with LLaMA-7b model, we explored different quantization granularity combinations, as detailed in Table (6). The results indicate that fine-grained and coarse-grained quantization methods can lead to up to a 1 PPL (Perplexity) difference. Specifically, in fine-grained

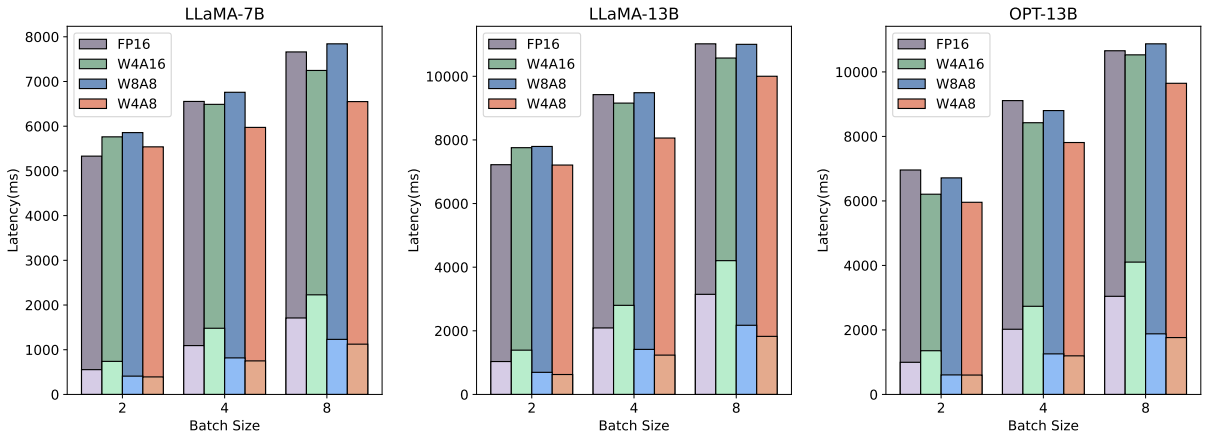


Figure 5: **End to End latency comparison on LLaMA-7B, LLaMA-13B and OPT-13B.** The lower part shows the pre-fill phase time cost and the upper part shows the self-decoding phase time cost.

quantization, we observe that static activation quantization in combination with group-wise weight quantization outperforms dynamic activation quantization coupled with channel-wise weight quantization. Additionally, our dual-grained quantization approach demonstrates that it introduces minimal additional quantization errors compared to group-wise quantization.

| **PPL** ↓ | FP16 | S+CW | S+GW | D+CW | **S+DG** |
|---|---|---|---|---|---|
| WikiText2 | 5.68 | 6.57 | 6.03 | 6.37 | 6.04 |
| C4 | 7.08 | 8.10 | 7.44 | 7.75 | 7.43 |

Table 6: **Comparison for different quantization granularity combinations for W4A8 LLaMA models.** S means static tensor-wise quantization, D means dynamic token-wise quantization, CW means channel-wise quantization, GW means Group-wise quantization and DG means Dula-Grained quantization.

**Effect of $\mathbf{p}$ in Aggressive Selective Equalization.** We evaluate varying $\mathbf{p}$ from 0.1% to 2% as shown in Table 7. When $\mathbf{p}$ exceeds the optimal value, unnecessary equalization distorts the weight distribution. Conversely, a too-small $\mathbf{p}$ fails to include outliers within the equalization range. Incorporating GLU (Dauphin et al., 2017) into LLaMA amplifies outlier effects via element-wise multiplication, raising outlier values. Consequently, LLaMA-13b requires a larger $\mathbf{p}$ to accommodate outliers compared to OPT-13b.

| **PPL** ↓ | LLaMA-13b | LLaMA2-13b | OPT-13b |
|---|---|---|---|
| $\mathbf{p} = 0.1\%$ | 5.55 | 5.32 | 11.04 |
| $\mathbf{p} = 0.3\%$ | 5.45 | 5.27 | 10.92 |
| $\mathbf{p} = 0.5\%$ | 5.45 | 5.25 | 11.12 |
| $\mathbf{p} = 1.0\%$ | 5.41 | 5.23 | 11.26 |
| $\mathbf{p} = 2.0\%$ | 5.44 | 5.29 | 11.21 |

Table 7: Comparison for different $\mathbf{p}$ in ASE.

**Effect of Aggressive Selective Equalization.** We replace the ASE with different equalization methods from SmoothQuant (Xiao et al., 2023) and FPTQ (Li et al., 2023b), to show the efficiency of our approach. Our method surpasses the equalization techniques employed by both SmoothQuant and FPTQ. In the context of OPT models, where outliers are less pronounced, the SmoothQuant strategy proves adequate, and our method achieves comparable performance.

| **PPL** ↓ | task | naive | SQ | FPTQ | Ours |
|---|---|---|---|---|---|
| LLaMA-13b | wikitext | 38.57 | 6.09 | 5.68 | 5.39 |
| | c4 | 9.82 | 7.66 | 7.23 | 6.93 |
| LLaMA2-13b | wikitext | 110.86 | 5.73 | 5.43 | 5.23 |
| | c4 | 33.05 | 7.43 | 7.07 | 6.82 |
| opt-13b | wikitext | 3722.76 | 10.99 | 11.13 | 10.30 |
| | c4 | 4700.02 | 11.89 | 12.13 | 11.89 |

Table 8: **Comparison for different equalization methods via different models.** Quantization granularity is static activation and dual-grained quantization.

## 6 Conclusion

In this paper, we introduce DGQ, an innovative and efficient approach for W4A8 quantization tailored for LLMs. DGQ addresses the inefficiencies of group-wise quantization through a dual-scale strategy, combining fine-grained integer and coarse-grained full-precision quantizations. We enhance the quantization scale search algorithm to fit our scheme and introduce an Aggressive Selective Equalization strategy for smoother scale optimization without extensive searching. Additionally, our optimized kernels delivering speedups of 1.37 × and 2.5 × over standard INT8 and FP16 kernels.

## Limitation

In this paper, we introduce a hardware-efficient W4A8 quantization framework tailored for LLMs. Our initial latency performance evaluation is conducted using CUTLASS kernels as a preliminary test. For deployment in real-world production environments, the development of specialized kernels is necessary to achieve significantly enhanced acceleration performance. Additionally, our testing focused exclusively on the widely-used CUDA computation platform. Future work will explore the adaptation of our framework to other computing platforms, broadening its applicability and performance optimization across diverse hardware environments.

## References


Saleh Ashkboos, Ilia Markov, Elias Frantar, Tingxuan Zhong, Xincheng Wang, Jie Ren, Torsten Hoefler, and Dan Alistarh. 2023. Towards end-to-end 4-bit inference on generative large language models. *arXiv preprint arXiv:2310.09259*.

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*.

Yuji Chai, John Gkountouras, Glenn G Ko, David Brooks, and Gu-Yeon Wei. 2023. Int2. 1: Towards fine-tunable quantized large language models with error correction through low-rank adaptation. *arXiv preprint arXiv:2306.08162*.

Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. 2019. Low-bit quantization of neural networks for efficient inference. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3009–3018. IEEE.

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*, abs/1803.05457.

Steve Dai, Rangha Venkatesan, Mark Ren, Brian Zimmer, William Dally, and Brucek Khailany. 2021. Vsquant: Per-vector scaled quantization for accurate low-precision neural network inference. *Proceedings of Machine Learning and Systems*, 3:873–884.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR.

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. 2023a. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2023b. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

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

Yefei He, Zhenyu Lou, Luoming Zhang, Weijia Wu, Bohan Zhuang, and Hong Zhou. 2022. Bivit: Extremely compressed binary vision transformer. *arXiv preprint arXiv:2211.07091*.

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*.

Raghuraman Krishnamoorthi. 2018. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*.

Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. 2023. Owq: Lessons learned from activation outliers for weight quantization in large language models. *arXiv preprint arXiv:2306.02272*.

Qingyuan Li, Ran Meng, Yiduo Li, Bo Zhang, Liang Li, Yifan Lu, Xiangxiang Chu, Yerui Sun, and Yuchen Xie. 2023a. A speed odyssey for deployable quantization of llms. *arXiv preprint arXiv:2311.09550*.

Qingyuan Li, Yifan Zhang, Liang Li, Peng Yao, Bo Zhang, Xiangxiang Chu, Yerui Sun, Li Du, and Yuchen Xie. 2023b. Fptq: Fine-grained post-training quantization for large language models. *arXiv preprint arXiv:2308.15987*.

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*.

Yen-Ting Lin and Yun-Nung Chen. 2023. Llm-eval: Unified multi-dimensional automatic evaluation for open-domain conversations with large language models. *arXiv preprint arXiv:2305.13711*.

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*.

Mitch Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: Annotating predicate argument structure. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Julieta Martinez, Shobhit Zakhmi, Holger H Hoos, and James J Little. 2018. Lsq++: Lower running time and higher recall in multi-codebook quantization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 491–506.

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

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

Haotong Qin, Yifu Ding, Mingyuan Zhang, Qinghua Yan, Aishan Liu, Qingqing Dang, Ziwei Liu, and Xianglong Liu. 2022. Bibert: Accurate fully binarized bert. *arXiv preprint arXiv:2203.06390*.

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*.

Xiuying Wei, Ruihao Gong, Yuhang Li, Xianglong Liu, and Fengwei Yu. 2022. Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization. *arXiv preprint arXiv:2203.05740*.

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*.

Xiaoxia Wu, Zhewei Yao, and Yuxiong He. 2023. Zeroquant-fp: A leap forward in llms post-training w4a8 quantization using floating-point formats. *arXiv preprint arXiv:2307.09782*.

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

Zhewei Yao, Cheng Li, Xiaoxia Wu, Stephen Youn, and Yuxiong He. 2023. A comprehensive study on post-training quantization for large language models. *arXiv preprint arXiv:2303.08302*.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183.

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

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

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. 2023. Atom: Low-bit quantization for efficient and accurate llm serving. *arXiv preprint arXiv:2310.19102*.

## A More performance results

In this section, we provide a comprehensive presentation of our results across various datasets to complement the main paper. Here, we use dynamic token-wise activation quantization. Specifically, the results include:

- WikiText2 perplexity in the LLaMA families (Table 10).
- C4 perplexity in the LLaMA families (Table 11).
- WikiText2 perplexity in the OPT families (Table 12).
- C4 perplexity in OPT families (Table 13).
- PTB perplexity in OPT families (Table 14).

## B Overhead introduced by Dual-phase search

We evaluate the time cost associated with our dual-phase search algorithm to quantify the overhead introduced by the second phase. The results are presented in the table below, from which we deduce that the second phase of our algorithm does not significantly increase quantization duration.

| Model | $\mathbf{T}$ | $\mathbf{T_2}$ | $\Delta$ |
|---|---|---|---|
| LLaMA-7B | 548.18 | 52.57 | 9.59% |
| LLaMA-13B | 964.44 | 128.81 | 13.04% |

Table 9: **The time consumption of Dual-Phase Search.** Here $\mathbf{T}$ presents the total time and the $\mathbf{T_2}$ is the second phase time.

| LLaMA PPL ↓ | | 1-7B | 1-13B | 1-30B | 1-65B | 2-7B | 2-13B | 2-70B |
|---|---|---|---|---|---|---|---|---|
| FP16 | - | 5.68 | 5.09 | 4.10 | 3.53 | 5.47 | 4.88 | 3.31 |
| W3A16 g128 | RTN | 7.01 | 5.88 | 4.87 | 4.24 | 6.66 | 5.51 | 3.97 |
| | GPTQ | 6.55 | 5.62 | 4.80 | 4.17 | 6.29 | 5.42 | 3.85 |
| | AWQ | 6.46 | 5.51 | 4.63 | 3.99 | 6.24 | 5.32 | - |
| W4A8 g128 | RTN | 8.72 | 7.81 | 6.76 | 6.16 | 5.72 | 44.82 | 8.70 |
| | ZeroQuantv2 | 6.44 | 5.32 | 4.36 | - | - | - | - |
| | SmoothQuant | 6.04 | 5.36 | 4.48 | 3.98 | 5.97 | 5.23 | 3.65 |
| | ZeroQuant-FP | 6.32 | 5.26 | **4.26** | - | - | - | - |
| | **Ours** | **5.85** | **5.21** | 4.28 | **3.71** | **5.64** | **5.01** | **3.45** |
| | **Ours†** | **6.04** | **5.39** | **4.45** | **3.89** | **5.87** | **5.23** | **3.74** |

Table 10: **Quantization Results on Wikitext2 (Merity et al., 2016) with A16W3 and A8W4 LLaMA Models.** † indicates static quantization for activation.

| LLaMA 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 |
| W3A16 g128 | RTN | 8.62 | 7.49 | 6.58 | 6.10 | 8.40 | 7.18 | 6.02 |
| | GPTQ | 7.85 | 7.10 | 6.47 | 6.00 | 7.89 | 7.00 | 5.85 |
| | AWQ | 7.92 | 7.07 | 6.37 | 5.94 | 7.84 | 6.94 | - |
| | OmniQuant | 7.34 | 6.76 | 6.11 | 5.73 | 7.35 | 6.65 | 5.86 |
| W4A8 g128 | RTN | 10.76 | 9.94 | 8.14 | 7.96 | 17.29 | 90.57 | 11.86 |
| | ZeroQuantv2 | 7.79 | 6.78 | 6.16 | - | - | - | - |
| | SmoothQuant | 7.51 | 6.89 | 6.39 | 5.94 | 7.50 | 6.82 | 5.78 |
| | ZeroQuant-FP | 7.51 | 5.73 | **6.09** | - | - | - | - |
| | **Ours** | **7.29** | **6.73** | 6.10 | **5.73** | **7.16** | **6.62** | **5.62** |
| | **Ours†** | **7.43** | **6.93** | **6.31** | **5.97** | **7.44** | **6.82** | **5.89** |

Table 11: **Quantization Results on c4 (Raffel et al., 2020) with A16W3 and A8W4 LLaMA Models**. † indicates static quantization for activation.

| OPT PPL ↓ | | 125M | 1.3B | 2.7B | 6.7B | 13B | 30B | 66B |
|---|---|---|---|---|---|---|---|---|
| FP16 | - | 27.65 | 14.63 | 12.47 | 10.86 | 10.12 | 9.56 | 9.34 |
| W3A16 g128 | RTN | 51.22 | 119.00 | 297.98 | 23.54 | 46.03 | 18.80 | 136.89 |
| | GPTQ | 39.24 | 16.47 | 13.69 | 11.65 | 10.35 | 9.73 | 10.96 |
| | AWQ | 36.74 | 16.32 | 13.58 | 11.41 | 10.68 | 9.85 | 9.60 |
| W4A8 g128 | RTN | 32.21 | 17.33 | 15.51 | 51.57 | 3978.101 | 2407.99 | 2832.57 |
| | ZeroQuantv2 | 31.69 | 15.53 | 13.02 | 11.29 | 10.43 | 9.86 | 9.62 |
| | SmoothQuant | 29.01 | 14.71 | 12.71 | 10.90 | 10.25 | 9.57 | 9.32 |
| | RPTQ | - | 15.39 | - | 11.21 | 10.90 | 10.22 | 9.46 |
| | ZeroQuant-FP | - | 15.32 | - | 10.89 | 10.16 | 9.52 | - |
| | Ours | 29.25 | 14.78 | 12.67 | 10.93 | 10.29 | 9.53 | 9.31 |
| | Ours† | 29.94 | 14.96 | 12.75 | 10.92 | 10.30 | 9.55 | 9.32 |

Table 12: **Quantization Results on Wikitext2 (Merity et al., 2016) with A16W3 and A8W4 OPT Models.** † indicates static quantization for activation.

| **OPT PPL ↓** | | 125M | 1.3B | 2.7B | 6.7B | 13B | 30B | 66B |
|---|---|---|---|---|---|---|---|---|
| FP16 | - | 24.61 | 14.73 | 13.17 | 11.75 | 11.21 | 10.69 | 10.28 |
| W3A16 g128 | RTN | 40.13 | 126.47 | 372.23 | 32.56 | 44.12 | 25.70 | 286.87 |
| | GPTQ | 30.08 | 16.47 | 14.54 | 12.48 | 11.58 | 10.91 | 11.35 |
| | AWQ | 30.39 | 16.27 | 14.19 | 12.30 | 11.61 | 10.96 | 10.53 |
| | OmniQuant | 29.34 | 16.11 | 14.15 | 12.31 | 11.63 | 10.98 | 10.51 |
| W4A8 g128 | RTN | 27.93 | 17.52 | 16.33 | 98.34 | 3926.05 | 3557.30 | 2493.73 |
| | ZeroQuantv2 | 27.19 | 15.73 | 13.82 | 12.19 | 11.64 | 11.00 | 10.63 |
| | SmoothQuant | 25.99 | 15.16 | 13.46 | 11.88 | 11.39 | 10.75 | 10.32 |
| | RPTQ | - | 15.48 | - | 12.11 | 11.62 | 11.01 | 10.57 |
| | ZeroQuant-FP | - | 15.32 | - | 11.95 | 11.30 | 10.75 | - |
| | Ours | 26.03 | 15.10 | 13.42 | 11.87 | 11.40 | 10.74 | 10.33 |
| | **Ours**† | 26.64 | 15.24 | 13.45 | 11.89 | 11.42 | 10.76 | 10.33 |

Table 13: **Quantization Results on c4 (Raffel et al., 2020) with A16W3 and A8W4 OPT Models** † indicates static quantization for activation.

| **OPT PPL ↓** | | 125M | 1.3B | 2.7B | 6.7B | 13B | 30B | 66B |
|---|---|---|---|---|---|---|---|---|
| FP16 | - | 32.55 | 16.97 | 15.11 | 13.09 | 12.34 | 11.84 | 11.36 |
| W3A16 g128 | RTN | 64.67 | 222.13 | 337.75 | 39.90 | 65.33 | 34.27 | 309.69 |
| | GPTQ | 45.17 | 19.90 | 17.06 | 14.24 | 12.84 | 12.54 | 13.27 |
| | AWQ | 44.07 | 19.59 | 16.52 | 13.98 | 12.87 | 66.68 | 3.4e3 |
| | OmniQuant | 45.29 | 20.42 | 17.08 | 14.23 | 13.49 | 12.54 | 12.06 |
| W4A8 g128 | RTN | 38.31 | 20.84 | 19.75 | 65.86 | 3370.84 | 2972.69 | 2556.84 |
| | ZeroQuantv2 | 36.66 | 18.35 | 16.11 | 13.70 | 12.91 | 12.28 | 11.84 |
| | SmoothQuant | 34.32 | 17.37 | 15.27 | 13.27 | 12.55 | 11.93 | 11.42 |
| | RPTQ | - | 17.79 | - | 13.74 | 13.40 | 12.41 | 11.73 |
| | ZeroQuant-FP | - | 18.19 | - | 13.44 | 12.55 | 11.90 | - |
| | Ours | 34.29 | 17.48 | 15.31 | 13.26 | 12.61 | 11.93 | 9.25 |
| | **Ours**† | 35.29 | 17.61 | 15.34 | 13.29 | 12.63 | 11.93 | 11.42 |

Table 14: **Quantization Results on ptb (Marcus et al., 1994) with A16W3 and A8W4 OPT Models** † indicates static quantization for activation.