EfficientQAT: Efficient Quantization-Aware Training for Large Language Models

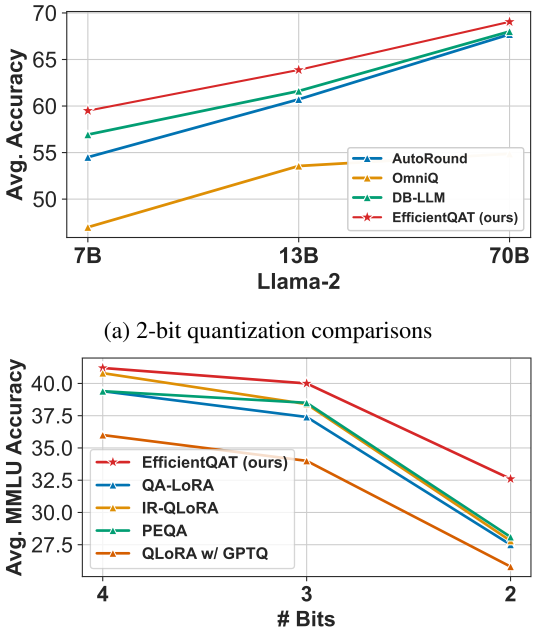
Anonymous ACL submission

Abstract

Large language models (LLMs) are crucial in modern natural language processing and artificial intelligence. However, they face challenges in managing their significant memory requirements. Although quantization-aware training (QAT) offers a solution by reducing memory consumption through low-bit representations with minimal accuracy loss, it is impractical due to substantial training resources. To address this, we propose Efficient Quantization-Aware Training (EfficientQAT), a more feasible QAT algorithm. EfficientQAT involves two consecutive phases: Block-wise training of all parameters (Block-AP) and end-to-end training of quantization parameters (E2E-QP). To the best of our knowledge, Block-AP is the first method to enable direct training of all parameters in a block-wise manner, reducing accuracy loss in low-bit scenarios by enhancing the solution space during optimization. E2E-QP then trains only the quantization parameters (step sizes) end-to-end, further improving the performance of quantized models by considering interactions among all sub-modules. Extensive experiments demonstrate that EfficientQAT outperforms previous quantization methods across a range of models, including base LLMs, instruction-tuned LLMs, and multimodal LLMs, with scales from 7B to 70B parameters at various quantization bits. For instance, EfficientQAT obtains a 2-bit Llama-2-70B model on a single A100-80GB GPU in 41 hours, with less than 3 points accuracy degradation compared to the full precision (69.48 vs. 72.41).

1 Introduction

Recent advancements in large language models (LLMs) (Touvron et al., 2023; Bubeck et al., 2023; Chiang et al., 2023; Xu et al., 2023a; Ying et al., 2024) have demonstrated impressive capabilities in diverse language tasks such as reasoning (Clark et al., 2018, 2019; Zellers et al., 2019), cognitive processing (Fu et al., 2023; Xu et al., 2023a), and agent-based applications (Qin et al., 2023a,b). However, these models are characterized by their extensive parameters, which pose significant challenges for memory footprint and bandwidth (Kim et al., 2023b; Xu et al., 2024a).



(a) 2-bit quantization comparisons

(b) Q-PEFT comparisons

Figure 1: **(a)** EfficientQAT significantly surpasses existing uniform quantization methods, and is either superior to or comparable with vector quantization techniques. **(b)** EfficientQAT markedly outperforms existing Q-PEFT methods.

Quantization-aware training (QAT) is a highly effective quantization technique that minimizes quantization errors by incorporating quantization constraints during training. For example, BitNet b1.58 (Ma et al., 2024) can achieve nearly lossless ternary quantization. The precision of QAT is due to two main factors: 1) Fully trainable parameters allow for enough optimized space for gradient descent optimization; 2) End-to-end training accounts for interactions among all sub-modules in the models. Despite its performance benefits, QAT demands significant training resources, such as time and GPUs, as well as extensive training data. For instance, BitNet b1.58 requires retraining LLMs from scratch using the entire pre-trained dataset. Therefore, this approach is impractical for extremely large models and has only been verified on 3B models with 100B training tokens.

In optimizing quantization for LLMs, current methods emphasize either fine-grained reconstruction or reducing trainable parameters. While these approaches improve efficiency, they significantly degrade accuracy in low-bit scenarios. Mainstream post-training quantization (PTQ) methods (Lin et al., 2023; Frantar et al., 2022; Shao et al., 2023) focus on block-wise reconstruction (Li et al., 2021). They also restrict the optimization space to alleviate overfitting risk by only training rounding parameters (Nagel et al., 2020; Cheng et al., 2023), clipping thresholds (Shao et al., 2023), or step sizes (Esser et al., 2019; Ding et al., 2023). However, these methods not only limit optimizable parameters but also overlook cross-block interactions, leading to notable accuracy degeneration in low-bit scenarios, as shown in Figure 1a. Conversely, quantized parameter-efficient fine-tuning (Q-PEFT) methods (Dettmers et al., 2023a; Kim et al., 2023a) reduce training costs by freezing quantized parameters and only training a few continuous floats. For example, PEQA (Kim et al., 2023a) and QA-LoRA (Xu et al., 2023b) focus on training continuous quantization parameters. Despite this, their performance remains poor, as depicted in Figure 1b, because the severe performance loss in low-bit scenarios (2-bit and 3-bit) cannot be fully recovered with limited trainable parameters.

To address these challenges, we introduce a novel quantization-aware training framework called EfficientQAT. This framework combines the advantages of fully trainable parameters and end-to-end training, similar to native QAT (Ma et al., 2024), while maintaining the training efficiency of PTQ (Cheng et al., 2023; Shao et al., 2023) and Q-PEFT (Xu et al., 2023b). EfficientQAT introduces block-wise training of all parameters (Block-AP) to enhance the optimizable space and mitigate quantization accuracy loss. Block-AP sequentially trains all parameters, including original full-precision weights and quantization parameters (step sizes and zero points), within each transformer block. Several works have been developed based on block-wise reconstruction. However, previous approaches focus on designing additional trainable parameters, such as clipping thresholds for OmniQuant (Shao et al., 2023), weight rounding for AutoRound (Cheng et al., 2023) and BRECQ (Li et al., 2021), or LoRA (Hu et al., 2021) parameters for CBQ (Ding et al., 2023). Our Block-AP is the first to directly train all parameters during block-wise reconstruction, achieving superior performance compared to previous methods (see Table 5). Block-AP successfully demonstrates that complex trainable parameter design is unnecessary for effective block-wise reconstruction in LLMs quantization. Furthermore, we introduce end-to-end training of quantization parameters (E2E-QP) to account for inter-block interactions. E2E-QP keeps the quantized weights fixed and trains only the quantization parameters (step sizes) end-to-end.

Thanks to the integration of the proposed Block-AP and E2E-QP, EfficientQAT characterizes itself as a fast-converging, memory-efficient, and high-performing quantization technique. For instance, EfficientQAT can obtain a 2-bit Llama-2-70B model on a single A100-80GB GPU in just 41 hours, with less than 3 points accuracy degradation on 5 zero-shot common-sense tasks compared to its full-precision counterpart (69.48 vs. 72.41). We also evaluate EfficientQAT across scenarios involving model compression and instruction-tuning. In model compression, as illustrated in Figure 1a, EfficientQAT significantly outperforms existing uniform quantization methods by approximately 5 points on accuracy in the challenging 2-bit quantization setting. In terms of instruction tuning, as shown in Figure 1b, EfficientQAT consistently outperforms existing Q-PEFT methods, including QLoRA (Dettmers et al., 2023a), QA-LoRA (Xu et al., 2023b), and PEQA (Kim et al., 2023a). For instance, EfficientQAT surpasses PEQA (Kim et al., 2023a) with 4.5 points MMLU accuracy when fine-tuning with Alpaca dataset.

2 Related Works

Post-Training Quantization of LLMs. PTQ is a pivotal technique for accelerating and deploying LLMs. Quantization approaches generally fall into two categories: weight-only quantization (Frantar et al., 2022; Dettmers et al., 2023b; Lee et al., 2023a; Kim et al., 2023b) and weight-activation quantization (Xiao et al., 2023; Liu et al., 2023c; Wei et al., 2022, 2023; Yuan et al., 2023; Zhao et al., 2023; Ashkboos et al., 2023; Li et al., 2023a; Ashkboos et al., 2024). Weight-only quantization focuses on compressing weights into low-bit formats, reducing memory demands and enhancing the efficiency of memory-bounded computations in LLMs (Lin et al., 2024; Yuan et al., 2024). Conversely, weight-activation quantization compresses both weights and activations, thus further decreasing the overhead associated with matrix multiplications (Lin et al., 2024). Recent advancements in weight-only quantization include the introduction of vector quantization methods by QUIP#(Tseng et al., 2024) and AQLM(Egiazarian et al., 2024). These methods have shown promising performance but also introduce significant overhead (Gong et al., 2024). Our research continues to explore uniform quantization, which is preferred for its compatibility with hardware implementations.

Quantization-Aware Training of LLMs. QAT can enhance the performance of quantized models beyond what PTQ offers. However, QAT has been less explored in LLMs due to the significant training costs involved. Studies such as LLM-QAT (Liu et al., 2023e) and BitDistiller (Du et al., 2024) investigate the application of knowledge distillation within QAT contexts. Techniques like BitNet b1.58 (Ma et al., 2024) and OneBit (Xu et al., 2024b) employ QAT to achieve extreme binary or ternary quantization levels. Although BitNet b1.58 demonstrates near-lossless performance on models up to 3 billion parameters and 100 billion training tokens with ternary quantization, its applicability to larger models or datasets remains uncertain due to prohibitive training expenses.

Quantized Parameter-Efficient Fine-Tuning of LLMs. Techniques like QLoRA (Dettmers et al., 2023a), INT2.1 (Chai et al., 2023), LQ-LoRA (Guo et al., 2023), and LoftQ (Li et al., 2023b) quantize model parameters to low-bit representations followed by the addition of LoRA (Hu et al., 2021) modules for fine-tuning. However, these methods require merging the LoRA modules into quantized weights, resulting in the model reverting to the FP16 format. Addressing this issue, QA-LoRA (Xu et al., 2023b) redesigns the LoRA module to merge seamlessly into the zero points. The approach most similar to ours is PEQA (Kim et al., 2023a), which uses a round-to-nearest (RTN) method for low-bit quantization and fine-tunes step sizes for task adaptation. However, PEQA experiences significant performance degradation due to limited trainable parameters, which hinders recovery from quantization information loss.

3 EfficientQAT

3.1 Method Overview

In this section, we introduce **EfficientQAT**, a novel quantization-aware training framework for LLMs that enhances memory efficiency. As illustrated in Figure 2, traditional QAT approaches train the full-precision weights $\mathbf{W}$ and quantization parameters s (step sizes) and z (zero points) simultaneously in an end-to-end manner, which significantly increases the memory requirements due to the large number of parameters involved. To address this issue, EfficientQAT adopts a two-stage strategy: block-wise training of all parameters (Block-AP) and end-to-end training of quantization parameters (E2E-QP). In the Block-AP phase, model parameters and quantization parameters are trained block-by-block using reconstruction loss, which not only allows for precise calibration with full training but also reduces memory consumption (Li et al., 2021; Shao et al., 2023) by block-wise training. Following this, the E2E-QP phase fixes the quantized weights and trains the step sizes exclusively on target datasets, thus achieving inter-block interaction in a memory-efficient way. Details on Block-AP and E2E-QP are further described in Sections 3.2 and 3.3, respectively.

3.2 Block-Wise Training of All Parameters

In this section, we introduce the Block-Wise Training of All Parameters (Block-AP) approach, designed to efficiently provide an effective initialization for following end-to-end training.

Quantization and Dequantization. Specifically, Block-AP begins with a standard uniform quantization method:

$$\mathbf{W}_{int} = \mathrm{clamp}(\lfloor \frac{\mathbf{W}}{s} \rceil + z, 0, 2^N - 1), \quad (1)$$

where $\lfloor \cdot \rceil$ represents the rounding operation. N is the target bit number. $\mathbf{W}_{int}$ and $\mathbf{W}$ denote the quantized integer and full-precision weights, respectively. s is the scaling factor and z is the zero point. In the forward propagation, the quantized weights are converted back to full precision as follows:

$$\widehat{\mathbf{W}} = (\mathbf{W_{int}} - z) \cdot s. \quad (2)$$

Here, $\widehat{\mathbf{W}}$ refers to the dequantized weights used in the forward computation. The processes of quantization (Eq.(1)) and dequantization (Eq.(2)) are integrated within the computation graph and can be op-

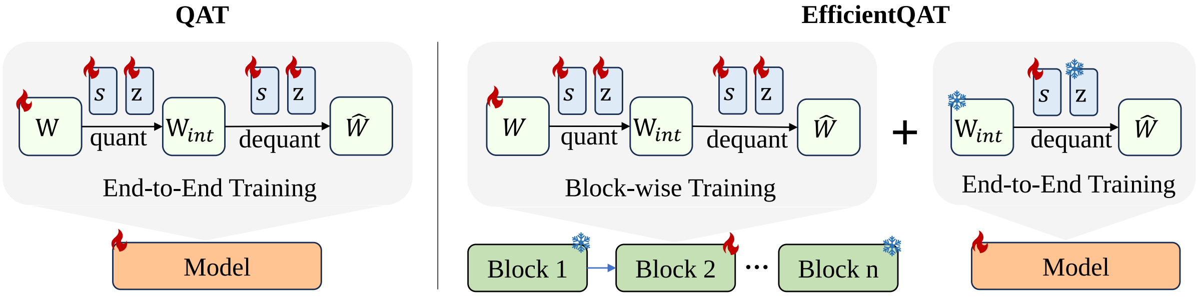


Figure 2: **The overall pipeline of naive QAT and proposed EfficientQAT.** EfficientQAT introduces two novel processes: Block-wise Training of All Parameters (Block-AP) and End-to-End Training of Quantization Parameters (E2E-QP).

timized through gradient descent in a quantization-aware manner.

Blcok-wise Quantization-aware Training. Traditional QAT methods (Ma et al., 2024; Esser et al., 2019; Liu et al., 2023e) train the entire network using Eq.(1) and Eq.(2) in an end-to-end fashion, which typically requires substantial computational resources and extensive data to prevent overfitting. Here we aim to enhance the training efficiency of QAT. Previous studies, such as BRECQ (Li et al., 2021), have demonstrated that block-wise training achieves faster convergence and requires less training time, data, and memory than end-to-end training given a pre-trained model. Following the methodologies in BRECQ (Li et al., 2021) and OmniQuant (Shao et al., 2023), Block-AP sequentially conducts quantization-aware training within one transformer block before moving on to the next under a block-wise reconstruction framework.

Full Training of Model Weights and Quantization Parameters. Unlike previous methods which optimize several quantization parameters such as rounding parameters (Nagel et al., 2020; Cheng et al., 2023; Lee et al., 2023b), clipping parameters (Shao et al., 2023), and step sizes (Esser et al., 2019; Ding et al., 2023), Block-AP behaves like QAT, training all inherent parameters from Eq.(1) and Eq.(2), including scaling factor s, zero point z, and model weights $\mathbf{W}$.

In our Block-AP approach, a straightforward full-training regimen outperforms existing partial-training variants (Nagel et al., 2020; Li et al., 2021; Ding et al., 2023) with intricate designs. Traditional training methods involving rounding parameters (Nagel et al., 2020; Li et al., 2021; Ding et al., 2023) serve as regularization techniques, constraining the update range of integral weights to $(-1, +1)$ to mitigate overfitting. However, this approach limits the solution space, potentially hindering the final performance of quantized models. Our empirical findings demonstrate the superiority of full training within our Block-AP over existing partial-training variants (Nagel et al., 2020; Li et al., 2021; Ding et al., 2023), as shown in Table 5.

Following block-wise training, we obtain the quantized model which includes quantized weights $\mathbf{W}_q$, step sizes s, and zero points z for each quantization group. The weights $\mathbf{W}_q$ and zero points z are stored in a low-bit format, while step sizes s are stored in FP16. Note that s and z are shared within their respective quantization groups and constitute only a small fraction of the model's parameters, approximately 1.6% for a group size of 64. Moreover, the model's memory footprint is substantially reduced by transitioning from full-precision 16-bit weights to 2/3/4-bit quantized weights.

3.3 End-to-End Training of Quantization Parameters

We further introduce the End-to-End Training of Quantization Parameters (E2E-QP), aimed at efficiently training the entire quantized model on target datasets.

End-to-End Training of step sizes. Unlike traditional Quantization-Aware Training (QAT) methods (Liu et al., 2023e; Ma et al., 2024) that train full-precision weights, E2E-QP begins with $\mathbf{W}_q$ initialized via Block-AP and focuses solely on the training of quantization parameters (s and z). Our findings indicate that training s, z, or both yields similar performance (see Table 6 for details). However, since training z involves converting it from a low-bits format to full-precision, we typically train only s by default unless specified otherwise to avoid additional memory overhead.

Additionally, within E2E-QP, there is no quantization process as per Equation (1); only the dequantization process occurs as described in Equation (2). Thus, the gradient of the trainable parameter s is computed as $\frac{\partial \widehat{w}}{\partial s} = w_q - z$.

Overall, the memory usage for training in E2E-QP is drastically reduced due to the reduced trainable parameter count. Detailed memory footprints for various model sizes and bits under E2E-QP are listed in Table 7. For instance, the Llama-2-70B model can complete 2-bit QAT through E2E-QP using only 34.2GB of memory. Equipped with E2E-QP, EfficientQAT is adaptable to different scenarios by simply changing the training datasets, which includes applications such as continual pre-training and instruction-tuning (Taori et al., 2023).

4 Experiments

This section presents extensive experiments to verify our proposed EfficientQAT. Section 4.1 and Sec 4.2 present the comparisons with quantization methods and Q-PEFT methods respectively. Section 4.4 details the training cost and inference speed-up of the proposed EfficientQAT. Section 4.3 presents the comprehensive ablation studies of the proposed EfficientQAT.

4.1 EfficientQAT for LLMs Quantization

Training. We conduct experiments on the Llama-2 and Llama-3 models. For Block-AP, we use 4096 samples from RedPajama (Computer, 2023) with a context length of 2048. We train each block with batch size as 2 and epochs as 2, setting the learning rate of quantization parameters as 1×10^{-4}, and the learning rate of weights as 2×10^{-5} for 2-bit and 1×10^{-5} for 3/4-bits. For E2E-QP, we also employ 4096 samples from RedPajama (Computer, 2023) but with a context length of 4096. We train the entire model with batch size as 32 and epoch as 1, and set the learning rate of step size as 2×10^{-5} for 2-bit and 1×10^{-5} for 3-bits.

Baseline. We compare our results with quantization methods with uniform quantization such as GPTQ (Frantar et al., 2022), AWQ (Lin et al., 2023), OmniQ (Shao et al., 2023), ApiQ (Liao and Monz, 2024) and AutoRound (Cheng et al., 2023), BitDistiller (Du et al., 2024), and DB-LLM (Chen et al., 2024).

Accuracy results. We evaluate the zero-shot accuracy on five common-sense reasoning tasks using the v0.4.2 lm-evaluation-harness[1]. These tasks include WinoGrande (Sakaguchi et al., 2021), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), Arc-Easy (Clark et al., 2018), and Arc-Challenge (Clark et al., 2018). Table 1 shows that the proposed EfficientQAT significantly outperforms previous methods across the Llama-2 and Llama-3 model families, as well as in both 2-bit and 3-bit quantization settings. The performance gains are particularly notable in extremely low-bit quantization, such as 2-bit. For instance, EfficientQAT achieves a +3.26% accuracy improvement over AWQ in w3g128 quantization with Llama-3-8B. Moreover, EfficientQAT surpasses DB-LLM by +9.02% accuracy in w2g64 quantization.

Perplexity results. We also evaluate perplexity on Wikitext2 and C4 using a 2048 context length, following prior studies (Frantar et al., 2022; Shao et al., 2023). The results align with the accuracy comparison, as EfficientQAT consistently achieves lower perplexity across the Llama-2 and Llama-3 model families in both 2-bit and 3-bit quantization. Notably, the benefits are more pronounced in Llama-3 models, which face greater challenges in quantization (Huang et al., 2024). For example, EfficientQAT reduces perplexity by 0.37 and 4.19 points compared to DB-LLM in Llama-2-7B and Llama-3-8B, respectively.

4.2 EfficientQAT for Instruction Tuning

Training and Evaluation. Following existing works (Xu et al., 2023b; Qin et al., 2024), we train Llama-1 models on the Alpaca dataset (Taori et al., 2023) and assess their performance by measuring average 5-shot MMLU (Hendrycks et al., 2020) accuracy works (Xu et al., 2023b; Qin et al., 2024). The training hyperparameters are identical to those described in Section 4.1, except we replace the RedPajama dataset (Computer, 2023) with Alpaca. In line with QLoRA's methodology (Dettmers et al., 2023a), we adjust the source context length to 384 and the target context length to 128, training for 10,000 steps with a batch size of 16.

Baseline. We benchmark EfficientQAT against several leading methods, including QLoRA (Dettmers et al., 2023a), QA-LoRA (Xu et al., 2023b), PEQA (Kim et al., 2023a), and IR-QLoRA (Qin et al., 2024), across quantization setting of 2, 3, and 4 bits. Consistent with QA-LoRA (Xu et al., 2023b), we also employ GPTQ (Frantar et al., 2022) to quantize the fine-tuned QLoRA models into a low-bit format without FP16 LoRA for equitable comparison.

[1] https://github.com/EleutherAI/lm-evaluation-harness

Table 1: Llama 2 & 3 average zero-shot accuracy on 5 common-sense reasoning tasks (↑). "-" indicates the result is unreachable in the public papers.

Method	Bits	Group	2-7	2-13	2-70	3-8	3-70
FP16	16	-	64.86	67.81	72.41	68.58	75.33
RTN	3	128	62.06	65.77	70.83	58.72	65.29
GPTQ	3	128	62.48	66.18	71.47	60.58	71.28
AWQ	3	128	62.82	66.14	71.41	64.82	73.65
OmniQ	3	128	62.42	66.18	71.07	64.09	71.90
AutoRound	3	128	63.72	66.68	71.24	-	-
EfficientQAT	3	128	**64.02**	**67.28**	**71.76**	**67.35**	**73.96**
OmniQ	2	128	46.98	53.56	54.87	52.66	60.06
AutoRound	2	128	54.50	60.72	67.70	-	-
EfficientQAT	2	128	**59.50**	**63.88**	**68.93**	**59.37**	**67.57**
DB-LLM	2	64	56.93	61.61	68.01	51.74	-
EfficientQAT	2	64	**60.14**	**64.48**	**69.48**	**60.76**	**67.89**

Table 2: Llama 2 & 3 Wikitext2 and C4 perplexity (↓), context length 2048. "-" indicates the result is unreachable in the public papers.

			Wikitext 2					C4				
Method	Bits	Group	2-7	2-13	2-70	3-8	3-70	2-7	2-13	2-70	3-8	3-70
FP16	16	-	5.47	4.88	3.32	6.14	2.85	6.97	6.47	5.52	8.88	6.73
GPTQ	3	128	6.29	5.42	3.85	9.58	5.25	7.89	7.00	5.85	11.66	8.64
AWQ	3	128	6.24	5.32	3.74	8.16	4.69	7.84	6.94	5.81	11.49	7.91
OmniQ	3	128	6.03	5.28	3.78	8.27	4.99	7.75	6.98	5.85	11.66	7.97
BitDistiller	3	128	5.97	-	-	-	-	-	-	-	-	-
EfficientQAT	3	128	**5.81**	**5.12**	**3.61**	**7.09**	**4.21**	**7.34**	**6.73**	**5.71**	**10.06**	**7.46**
OmniQ	2	128	11.06	8.26	6.55	18.50	16.79	15.02	11.05	8.52	22.46	15.06
ApiQ	2	128	8.25	6.71	-	-	-	12.04	9.13	-	-	-
BitDistiller	2	128	8.08	-	-	-	-	-	-	-	-	-
EfficientQAT	2	128	**7.19**	**6.08**	**4.61**	**9.80**	**6.38**	**8.79**	**7.75**	**6.48**	**13.22**	**9.53**
ApiQ	2	64	7.59	6.44	-	-	-	10.56	8.92	-	-	-
CBQ	2	64	8.01	-	-	-	-	11.30	-	-	-	-
DB-LLM	2	64	7.23	6.19	4.64	13.60	-	9.62	8.38	6.77	19.20	-
EfficientQAT	2	64	**6.86**	**5.96**	**4.52**	**9.41**	**6.07**	**8.50**	**7.59**	**6.38**	**12.77**	**9.23**

Results. Both Table 3 and Figure 1b indicate that EfficientQAT significantly outperforms existing Q-PEFT methods. For instance, in channel-wise quantization (group size of -1), EfficientQAT achieves more than 3% higher accuracy than PEQA (Kim et al., 2023a). In the 2-bit quantization scenario, the superiority of EfficientQAT is even more pronounced, surpassing QA-LoRA (Xu et al., 2023b) by 5.1% and 4.0% in 7B and 13B models, respectively, and outperforming PEQA by 4.5% and 8.7% in the same models. Moreover, Table 3 also demonstrates that EfficientQAT outperforms both QA-LoRA and QLoRA with GPTQ in smaller model memory footprint (larger group size).

4.3 Ablation Analysis

The EfficientQAT algorithm is comprised of two main components: Block-AP and E2E-QP. This section evaluates the effectiveness, trainable parameters, and training sample requirements of each component. We present the average perplexity for WikiText2 and C4 datasets, and the average accuracy for five zero-shot reasoning tasks, similar to Table 1.

Table 3: Llama-1 average MMLU accuracy (5-shot) about instruction-tuning on Alpaca dataset.

Method	Bits	Group	7B	13B
-	16	-	34.6	46.3
PEQA	4	-1	35.8	45.0
EfficientQAT	4	-1	**38.8**	**48.2**
QLoRA	4+16	-	38.4	48.4
QLoRA w/GPTQ	4	32	36.0	48.0
QA-LoRA	4	32	39.4	49.2
PEQA	4	64	39.4	47.4
IR-QLoRA	4	64	40.8	49.3
EfficientQAT	4	64	**41.2**	**49.5**
QLoRA w/ GPTQ	3	32	34.0	46.1
QA-LoRA	3	32	37.4	47.3
IR-QLoRA	3	64	38.4	-
PEQA	3	64	38.5	46.3
EfficientQAT	3	64	**40.0**	**48.2**
QLoRA w/ GPTQ	2	32	25.8	30.9
QA-LoRA	2	32	27.5	36.9
IR-QLoRA	2	64	27.8	-
PEQA	2	64	28.1	32.2
EfficientQAT	2	64	**32.6**	**40.9**

Table 4: Effectiveness of each component on Llama-2-7B w2g64 quantization.

Block-AP	E2E-QP	Avg. PPL	Avg. Acc.
✗	✗	453.49	40.69
✓	✗	8.53	58.99
✗	✓	9.33	55.71
✓	✓	7.68	60.14

Table 5: W2g64 Llama-2-7B performance with different trainable parameters in the block-wise training (w/o E2E-QP). "#" indicates trainable parameters count in a block.

Param.	#	Memory	Avg. PPL	Avg. Acc.
clipping	6.3M	6.4GB	11.28	53.20
s,z	6.3M	6.4GB	10.26	55.20
round	202.4M	8.6GB	15.50	45.32
s,z,round	208.7M	9.3GB	9.17	57.14
s,z,$\mathbf{W}$	208.7M	8.5GB	8.53	58.99

Table 6: Llama-2-7B w2g64 quantization with different trainable parameters for E2E-QP (w/ Block-AP).

Param.	Avg. Bits	Avg. PPL	Avg. Accuracy
s	2.28	7.68	60.14
z	2.50	7.69	60.08
s, z	2.50	7.68	60.18

Effectiveness of each component. As indicated in Table 4, both the Block-AP and E2E-QP components significantly enhance performance, with their combination yielding the best results. Notably, Block-AP outperforms E2E-QP, aligning with findings from BRECQ (Li et al., 2021).

Trainable parameters of Block-AP. Block-AP trains all parameters, including original weights and quantization parameters. Previous methods have introduced various training strategies to mitigate overfitting, such as trained rounding (Nagel et al., 2020; Cheng et al., 2023), clipping thresholds (Shao et al., 2023), and step sizes (Esser et al., 2019; Ding et al., 2023). We compare Block-AP with these methods by modifying only the trainable parameters of Block-AP. As shown in Table 5, Block-AP (training s, z, $\mathbf{W}$) performs best with an acceptable training cost. Additionally, the memory footprint of directly training $\mathbf{W}$ is even smaller than that of training the rounding operation, which requires an additional copy of rounding parameters.

Trainable parameters of E2E-QP. We further examine the trainable parameters within E2E-QP. Table 6 shows that training s, z, or both yields similar performance. However, given that converting z from an original low-bit representation to a trainable FP16 format increases the average bit count, we opt to train only s by default.

Samples number of Block-AP. We assess the number of training samples for Block-AP, noting that E2E-QP trains all parameters, which may lead to overfitting. To address this, we introduce an additional 64 unseen samples from ReadPajama to evaluate the overfitting issue. We adjust the training epochs to ensure a similar total training time, allowing for fair comparisons across different sample sizes. As illustrated in Figure 3, increasing the number of training samples significantly reduces the gap between training loss and validation loss from 1.07 to 0.06. This reduction corresponds to an increase in the average accuracy for zero-shot tasks from 57.14% to 58.99%. Consequently, we set the default number of training samples for E2E-QP at 4096, as this maintains a minimal gap between training and validation losses.

Samples number of E2E-QP. In the E2E-QP, we train the model for 1 epoch to avoid over-fitting. Our examination of the training sample sizes for E2E-QP, detailed in Table 8, reveals that average perplexity consistently improves as sample sizes increase from 128 to 32,674. However, there is no significant improvement in average accuracy beyond 4096 samples. Therefore, we set the training

Table 7: The detailed training time and training memory of EfficientQAT across different model size and quantization bits on a single A100-80GB GPU.

Llama-2	Block-AP		E2E-QP		
	Time	Memory	Time	Memory (4-/3-/2-bits)	Total Time
7B	3.3h	8.5GB	∼1.5h	7.0/6.4/5.6GB	4.8h
13B	5.6h	10.3GB	∼2.9h	11.7/10.6/9.1GB	8.5h
70B	26.6h	29.9GB	∼14.3h	48.4/42.0/34.2GB	40.9h

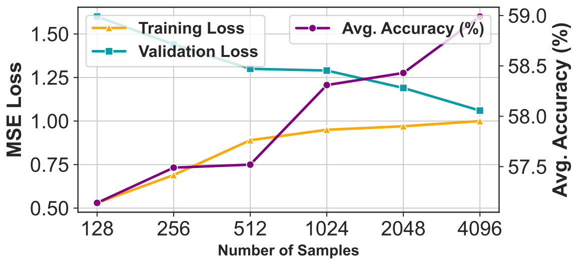


Figure 3: Illustration of training loss, validation loss and average accuracy of w2g64 Llama-2-7b with different training samples size for Block-AP (w/o E2E-QP).

Table 8: Llama-2-7B w2g64 quantization performance with different sample numbers for E2E-QP (w/ Block-AP).

# Samples	Avg. PPL	Avg. Accuracy
128	8.09	59.03
512	7.88	59.81
2048	7.75	60.13
4096	7.68	60.14
8192	7.63	60.19
32764	**7.50**	**60.31**

sample size for E2E-QP at 4096 by default to balance efficiency and performance. Nonetheless, it is possible to further enhance the performance of EfficientQAT by increasing the sample size.

4.4 Efficiency of EfficientQAT

Training Efficiency Table 7 illustrates the required memory and time for training Lllama-2 models using EfficientQAT. The results indicate that the model completes training rapidly, taking 4.8 hours for the 7B model and 40.9 hours for the 70B model. we further compare the training time with other QAT methods, including BitDistiller, and DB-LLM. As shown in Table 9, the training time of EfficientQAT is significantly lower than that of existing methods. For example, the tuning time of EfficientQAT is only 50% of DB-LLM. Additionally, for quantizing a 70B model, the full process of EfficientQAT can be completed on a single A100-80GB GPU. However, other methods require at least 4 A100-80GB GPUs to quantize a model of this size. Therefore, EfficientQAT is both a time-efficient and memory-efficient QAT method.

Inference Efficiency Due to the leverage of standard uniform quantization, the quantized models of EfficientQAT can also achieve speedup through a lot of toolboxes, such as MLC-LLM (team, 2023), AWQ (Lin et al., 2023), and BitBLAS (Wang et al., 2024), T-MAC (Wei et al., 2024), Marlin (Frantar et al., 2024), *etc*. For example, Table 10 shows that INT2 quantization of EfficientQAT can enhance the forward-pass speed by approximately 2.9x to 4.4x through BitBLAS (Wang et al., 2024).

Table 9: Comparisons of training time with existing methods in Llama-2-70B.

Method	One A100-80GB?	GPU hours (h)
LLM-QAT	✗	900
BitDistiller	✗	64
DB-LLM	✗	82
EfficientQAT	✓	**41**

5 Conclusion

In this study, we introduce EfficientQAT, a novel method that completes QAT with improved efficiency in both memory usage and training time. Through comprehensive testing, EfficientQAT proves superior to existing PTQ, QAT, and Q-PEFT methods in terms of versatility and performance across various models and quantization levels. Additionally, EfficientQAT leverages a standard uniform quantization, which simplifies deployment using popular toolboxes. We anticipate that EfficientQAT will stimulate further research and improve the compression of Large Language Models (LLMs), making them more efficient and widely accessible.

6 Limitation

EfficientQAT achieves impressive results in low-bit quantization scenarios, but there remains a performance gap compared to full-precision (FP16) models, particularly in 2-bit settings. Reducing this gap without sacrificing efficiency remains a challenge. Additionally, the method depends on the availability of high-quality and diverse datasets, requiring 4096 samples for effective training in both the Block-AP and E2E-QP phases. The performance of the quantized models can vary significantly based on the size and distribution of the training data. This reliance may limit its effectiveness in data-scarce or domain-specific applications.

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*.

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*.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *ArXiv*, abs/1308.3432.

Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. 2020. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2978–2985.

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*, 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*.

Hong Chen, Chengtao Lv, Liang Ding, Haotong Qin, Xiabin Zhou, Yifu Ding, Xuebo Liu, Min Zhang, Jinyang Guo, Xianglong Liu, et al. 2024. Db-llm: Accurate dual-binarization for efficient llms. *arXiv preprint arXiv:2402.11960*.

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

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

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*.

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

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*.

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

Dayou Du, Yijia Zhang, Shijie Cao, Jiaqi Guo, Ting Cao, Xiaowen Chu, and Ningyi Xu. 2024. Bitdistiller: Unleashing the potential of sub-4-bit llms via self-distillation. *arXiv preprint arXiv:2402.10631*.

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*.

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*.

Elias Frantar, Roberto L Castro, Jiale Chen, Torsten Hoefler, and Dan Alistarh. 2024. Marlin: Mixed-precision auto-regressive parallel inference on large language models. *arXiv preprint arXiv:2408.11743*.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Zhenyu Qiu, Wei Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, and Rongrong Ji. 2023. Mme: A comprehensive evaluation benchmark for multimodal large language models. *ArXiv*, abs/2306.13394.

Ruihao Gong, Yang Yong, Shiqiao Gu, Yushi Huang, Yunchen Zhang, Xianglong Liu, and Dacheng Tao. 2024. Llm-qbench: A benchmark towards the best practice for post-training quantization of large language models. *arXiv preprint arXiv:2405.06001*.

Han Guo, Philip Greengard, Eric P Xing, and Yoon Kim. 2023. Lq-lora: Low-rank plus quantized matrix decomposition for efficient language model finetuning. *arXiv preprint arXiv:2311.12023*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

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

Wei Huang, Xudong Ma, Haotong Qin, Xingyu Zheng, Chengtao Lv, Hong Chen, Jie Luo, Xiaojuan Qi, Xianglong Liu, and Michele Magno. 2024. How good are low-bit quantized llama3 models? an empirical study. *arXiv preprint arXiv:2404.14047*.

Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. 2023a. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. *arXiv preprint arXiv:2305.14152*.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. 2023b. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*.

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

Jung Hyun Lee, Jeonghoon Kim, Se Jung Kwon, and Dongsoo Lee. 2023b. Flexround: Learnable rounding based on element-wise division for post-training quantization. In *International Conference on Machine Learning*, pages 18913–18939. PMLR.

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*.

Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. 2023b. Loftq: Lora-fine-tuning-aware quantization for large language models. *arXiv preprint arXiv:2310.08659*.

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*.

Baohao Liao and Christof Monz. 2024. Apiq: Finetuning of 2-bit quantized large language model. *arXiv preprint arXiv:2402.05147*.

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*.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. 2024. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. *arXiv preprint arXiv:2405.04532*.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2023a. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023b. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*.

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

Yuanzhan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. 2023d. Mmbench: Is your multi-modal model an all-around player? *ArXiv*, abs/2307.06281.

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

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and A. Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *ArXiv*, abs/2209.09513.

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*.

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, Xudong Ma, Xingyu Zheng, Xiaoyang Li, Yang Zhang, Shouda Liu, Jie Luo, Xianglong Liu, and Michele Magno. 2024. Accurate lora-finetuning quantization of llms via information retention. *arXiv preprint arXiv:2402.05445*.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shi Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bo Li, Ziwei Tang, Jing Yi, Yu Zhu, Zhenning Dai, Lan Yan, Xin Cong, Ya-Ting Lu, Weilin Zhao, Yuxiang Huang, Jun-Han Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. 2023a. Tool learning with foundation models. *ArXiv*, abs/2304.08354.

Yujia Qin, Shi Liang, Yining Ye, Kunlun Zhu, Lan Yan, Ya-Ting Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Marc H. Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023b. Toolllm: Facilitating large language models to master 16000+ real-world apis. *ArXiv*, abs/2307.16789.

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

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

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

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

MLC team. 2023. MLC-LLM.

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

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. 2024. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *arXiv preprint arXiv:2402.04396*.

Lei Wang, Lingxiao Ma, Shijie Cao, Quanlu Zhang, Jilong Xue, Yining Shi, Ningxin Zheng, Ziming Miao, Fan Yang, Ting Cao, Yuqing Yang, and Mao Yang. 2024. Ladder: Enabling efficient low-precision deep learning computing through hardware-aware tensor transformation. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 307–323, Santa Clara, CA. USENIX Association.

Jianyu Wei, Shijie Cao, Ting Cao, Lingxiao Ma, Lei Wang, Yanyong Zhang, and Mao Yang. 2024. T-mac: Cpu renaissance via table lookup for low-bit llm deployment on edge. *Preprint*, arXiv:2407.00088.

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.

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.

Peng Xu, Wenqi Shao, Mengzhao Chen, Shitao Tang, Kaipeng Zhang, Peng Gao, Fengwei An, Yu Qiao, and Ping Luo. 2024a. Besa: Pruning large language models with blockwise parameter-efficient sparsity allocation. *arXiv preprint arXiv:2402.16880*.

Peng Xu, Wenqi Shao, Kaipeng Zhang, Peng Gao, Shuo Liu, Meng Lei, Fanqing Meng, Siyuan Huang, Yu Qiao, and Ping Luo. 2023a. Lvlm-ehub: A comprehensive evaluation benchmark for large vision-language models. *arXiv preprint arXiv:2306.09265*.

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

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

Kaining Ying, Fanqing Meng, Jin Wang, Zhiqian Li, Han Lin, Yue Yang, Hao Zhang, Wenbo Zhang, Yuqi Lin, Shuo Liu, et al. 2024. Mmt-bench: A comprehensive multimodal benchmark for evaluating large vision-language models towards multitask agi. *arXiv preprint arXiv:2404.16006*.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. 2023. Mm-vet: Evaluating large multimodal models for integrated capabilities. *ArXiv*, abs/2308.02490.

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*.

Zhihang Yuan, Yuzhang Shang, Yang Zhou, Zhen Dong, Chenhao Xue, Bingzhe Wu, Zhikai Li, Qingyi Gu, Yong Jae Lee, Yan Yan, et al. 2024. Llm inference unveiled: Survey and roofline model insights. *arXiv preprint arXiv:2402.16363*.

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*.

Overview of Appendix

This appendix includes the following sections:

- Sec A gives the reproducibility statement to summarize the information related to the reproduction of our method.
- Sec. B describes the gradient calculation in the Block-AP process.
- Sec. C presents the speedup ratio of uniform quantization using BitBLAS (Wang et al., 2024).
- Sec. D details the sources of results for each comparison method to aid reproduction.
- Sec. E presents the sizes of quantized models.
- Sec. F provides additional ablation studies, including those on group size and training datasets.
- Sec. G applies the proposed EfficientQAT to Llava (Liu et al., 2023b) models.

A Reproducibility Statement

In this section, we summarize the necessary information to reproduce our results. We provide the training and evaluation details at the beginning of each sub-section in Sec. 4. We also provide the source of detailed results for each compared method in Sec.D.

B Gradient of Trainable Parameters in Block-AP

Block-AP, aligned with LSQ+(Bhalgat et al., 2020), uses a straight-through estimator (STE)(Bengio et al., 2013) to facilitate gradient computation through the rounding operation. The gradients of scaling factor s are computed as follows:

$$\frac{\partial \widehat{w}}{\partial s} = \begin{cases} \lfloor \frac{w}{s} \rceil - \frac{w}{s}, 0 \leq \lfloor \frac{w}{s} \rceil + z \leq 2^{N-1}, \\ -z, \lfloor \frac{w}{s} \rceil + z < 0, \\ 2^{N-1} - z, \lfloor \frac{w}{s} \rceil + z > 2^{N-1}. \end{cases} \tag{3}$$

and the gradient with respect to zero point z is:

$$\frac{\partial \widehat{w}}{\partial z} = \begin{cases} 0, 0 \leq \lfloor \frac{w}{s} \rceil + z \leq 2^{N-1}, \\ -1, otherwise, \end{cases} \tag{4}$$

and the full-precision weight $\mathbf{W}$ can also be updated through its gradient[2]:

$$\frac{\partial \widehat{w}}{\partial w} = \begin{cases} 1, 0 \leq \lfloor \frac{w}{s} \rceil + z \leq 2^{N-1}, \\ 0, otherwise, \end{cases} \tag{5}$$

C Speedup with BitBlas

According to Table 10, INT2 quantization enhances the forward-pass speed by approximately 2.9x to 4.4x.

[2] $\widehat{w}$,w is a element from $\widehat{W}$, $\mathbf{W}$

Table 10: Speed of the FP16 linear layer matrix-vector multiplication in PyTorch, and relative INT2 speedups in BitBLAS (Wang et al., 2024). Testing on A100-80GB GPU.

Llama-2	**7B**		13B		70B	
size (out_c × in_c)	4096x4096	11008x4096	5120x5120	13824x5120	8192x8192	28672x8192
FP16	25 us	61 us	38 us	90 us	91 us	286 us
INT2	9 us	21 us	11 us	26 us	24 us	67 us
Speedup	3.1x	2.9x	3.6x	3.5x	3.9x	4.4x

D Results Source of Other Method.

In this study, we present a thorough comparison of our method against existing PTQ techniques, including GPTQ (Frantar et al., 2022), AWQ (Lin et al., 2023), OmniQ (Shao et al., 2023), AutoRound (Cheng et al., 2023), QuIP# (Tseng et al., 2024), and AQLM (Egiazarian et al., 2024). We also compare with existing QAT methods, including LLM-QAT (Liu et al., 2023e), BitDistiller (Du et al., 2024), PB-LLM (Shang et al., 2023) and DB-LLM (Chen et al., 2024). Additionally, we also evaluate quantized parameter-efficient fine-tuning methods such as PEQA (Kim et al., 2023a), QLoRA (Dettmers et al., 2023a), QA-LoRA (Xu et al., 2023b), and IR-QLoRA (Qin et al., 2024). The results we discuss originate from their respective official publications, and other scholarly articles, or are derived from our reproduction. We meticulously document the source of the results for each method as follows:

- GPTQ, AWQ, OmniQ, AutoRound: The zero-shot accuracy results for Llama-2 models using these methods are derived from the AutoRound GitHub repository[3]. The perplexity results for the Llama-2 models using GPTQ, AWQ, and OmniQ are taken from the OmniQ paper (Shao et al., 2023). The results for Llama-3 models using AWQ[4] and GPTQ[5] were obtained through their open-source implementations.

- QuIP#, AQLM: We replicated the results using the official pre-trained models provided by QuIP#[6] and AQLM[7].

- LLM-QAT, BitDistiller: These results are cited from BitDistiller (Du et al., 2024) paper.

- PB-LLM, DB-LLM: These results are cited from recent Llama-3 quantization empirical study (Huang et al., 2024).

- ApiQ: These results are cited from IR-ApiQ (Liao and Monz, 2024) paper.

- PEQA: The per-channel quantization results (g=-1) are cited from their publication (Kim et al., 2023a), and the results for a group size of 64 were produced using our codebase.

- QA-LoRA, QLoRA, QLoRA w/ GPTQ: These results are cited from QA-LoRA (Xu et al., 2023b) paper.

- IR-QLoRA: These results are cited from IR-QLoRA (Qin et al., 2024) paper.

[3] AutoRound: https://github.com/intel/auto-round/blob/main/docs/acc.md
[4] AWQ:https://github.com/mit-han-lab/llm-awq
[5] GPTQ:https://github.com/qwopqwop200/GPTQ-for-LLaMa
[6] https://github.com/Cornell-RelaxML/quip-sharp
[7] https://github.com/Vahe1994/AQLM

Table 11: **Model size of quantized models.** Compression ratio indicates the compression ratio of quantized models compared with FP16 models.

Model	# Bit	Group size	bits/param	size (GiB)	Compression ratio (%)
LLaMA-2-7B	16	-	16	12.55	-
	4	32	4.63	3.98	68.33
	4	64	4.31	3.74	70.20
	4	128	4.16	3.62	71.14
	3	32	3.59	3.35	73.28
	3	64	3.30	3.13	75.08
	3	128	3.15	3.01	75.98
	2	32	2.56	2.42	80.71
	2	64	2.28	2.21	82.40
	2	128	2.14	2.10	83.25
LLaMA-2-13B	16	-	16	24.24	-
	4	32	4.63	7.44	69.30
	4	64	4.31	6.98	71.21
	4	128	4.16	6.75	72.16
	3	32	3.59	6.22	74.33
	3	64	3.30	5.78	76.16
	3	128	3.15	5.56	77.07
	2	32	2.56	4.40	81.87
	2	64	2.28	3.98	83.58
	2	128	2.14	3.77	84.44
LLaMA-2-70B	16	-	16	128.48	-
	4	32	4.63	37.83	70.55
	4	64	4.31	35.34	72.49
	4	128	4.16	34.10	73.46
	3	32	3.59	31.26	75.67
	3	64	3.30	28.87	77.53
	3	128	3.15	27.67	78.46
	2	32	2.56	21.40	83.34
	2	64	2.28	19.16	85.09
	2	128	2.14	18.04	85.96

E Size of Quantized Models

This section illustrates model size reduction achieved through quantization. Models quantized to low-bit representations are more compact.

We implement N-bit quantization with a grouping size of g, where each group of g weights shares the same FP16 step size and an N-bit zero point. Consequently, the average number of bits per parameter is calculated as $N + \frac{N+16}{g}$. It is important to note that only the linear layers within the transformer blocks are quantized; other layers, such as normalization layers, embeddings, and the classification head, remain in FP16 format. Table 11 provides detailed comparisons of quantized model sizes and their compression ratios.

Table 12: Lllma-2-7B 2-bit quantization performance with different group sizes for proposed EfficientQAT.

Group	**Avg. Bits**	**Avg. PPL**	**Avg. Accuracy**
32	2.56	7.59	60.28
64	2.28	7.68	60.14
128	2.10	7.99	59.50
256	2.07	8.18	58.67

F Additional Ablation Analysis

Quantization Group Size. The group size is a crucial hyperparameter in weight-only quantization. A smaller group size offers more granular compression and reduces quantization loss but increases the number of quantization parameters required. As indicated in Table 12, a group size of 64 strikes an optimal balance for 2-bit quantization using EfficientQAT. It outperforms a group size of 128 by achieving a 0.31 lower perplexity and a 0.64% higher accuracy, yet it slightly underperforms compared to a group

Table 13: Block-AP (w/o E2E-QP) results of Llama-2-7B in different calibration datasets.

Bits	Dataset	Wiki PPL	C4 PPL	Avg. Accuracy
w3g128	WikiText2	5.72	7.52	63.24
w3g128	C4	5.92	7.38	63.82
w3g128	Redpajama	5.91	7.41	63.50
w2g64	WikiText2	6.73	9.89	58.26
w2g64	C4	7.87	9.30	59.24
w2g64	Redpajama	7.70	9.36	58.99

size of 32, with a marginal difference of 0.09 in perplexity and 0.14% in accuracy.

Training Dataset. More trainable parameters can increase the risk of overfitting. Previous works (Gong et al., 2024) show that a similar distribution between the calibration dataset and the test dataset can improve test accuracy. RedPajama and C4 datasets are diverse, while WikiText2 is simpler and sourced from Wikipedia. The close distribution of training and test datasets for WikiText2 results in significantly lower WikiText2 perplexity when using it as a calibration dataset. However, the average accuracy of zero-shot tasks in Table R7 shows that Block-AP's generation ability is excellent, with only 0.26% and 1.28% accuracy declines when changing the calibration dataset from RedPajama to WikiText2 for w3g128 and w2g64, respectively. Additionally, using C4 as a calibration dataset can even increase the average accuracy by 0.2-0.3 points. Overall, we recommend using Block-AP with more diverse calibration datasets like C4 or RedPajama.

Table 14: **Results about instruction tuning of large vision-language models.** We following the overall training pipeling of LLaVA-1.5 (Liu et al., 2023a) and just change the fine-tuning methods. 'QLoRA + Block-AP' indicates that we leverage proposed Block-AP to quantized the QLoRA models into low-bits for fair comparisons. † MME's perception scores are normalized to 100 percent.

Model	Method	#Bit		MMbench	MME†	MM-Vet	ScienceQA	Avg.
		Training	Inference					
LLaVA-1.5-7B	LoRA	16	16	66.1	73.8	30.2	68.4	59.6
	QLoRA	4+16	16	64.1	72.8	30.3	68.0	58.8
	QLoRA + Block-AP	4+16	4	63.6	72.0	29.8	67.7	58.3
	EfficientQAT	4	4	64.4	73.2	30.3	68.1	**58.8**(+0.5)
	QLoRA + Block-AP	4+16	3	62.9	71.8	29.7	66.4	57.7
	EfficientQAT	3	3	63.2	71.4	30.9	67.3	**58.2**(+0.5)
	QLoRA + Block-AP	4+16	2	53.7	64.3	28.9	60.7	51.9
	EfficientQAT	2	2	62.3	68.0	27.8	63.4	**55.4**(+3.5)
LLaVA-1.5-13B	LoRA	16	16	68.5	77.1	38.3	71.2	63.8
	QLoRA	4+16	16	67.6	76.9	36.0	69.9	62.7
	QLoRA + Block-AP	4+16	4	67.4	76.6	35.6	69.3	62.4
	EfficientQAT	4	4	67.5	74.8	35.6	70.2	62.0(-0.4)
	QLoRA + Block-AP	4+16	3	66.8	75.5	34.5	68.4	61.3
	EfficientQAT	3	3	67.4	74.8	35.3	69.3	**61.7**(+0.4)
	QLoRA + Block-AP	4+16	2	62.5	72.1	32.5	65.0	58.0
	EfficientQAT	2	2	63.9	73.1	33.9	68.6	**59.9**(+1.9)

G Instruction Tuning for LVLMs.

Traditional Q-PEFT methods only do experiments on the language models. In this section, we further extend proposed EfficientQAT into Large vision-Language models (LVLMs) such as LLaVA (Liu et al., 2023b).

Training and Evaluation. For the fine-tuning of large vision-language models (LVLMs), we largely align with LLaVA1.5 (Liu et al., 2023a), which encompass the training model, datasets, and hyperparameters[8]. Unlike LLaVA1.5, which begins fine-tuning with full-precision Vicuna models using either full

[8] For comprehensive details, please consult the official repository at https://github.com/haotian-liu/LLaVA.

fine-tuning or LoRA-based methods (Hu et al., 2021), EfficientQAT starts with Vicuna models already quantized using our Block-AP method and continues with our E2E-QP fine-tuning approach. The training process involves two steps: initially freezing the LLM and pre-training a projector to align features with a Vision Transformer (ViT), followed by end-to-end fine-tuning of both the LLM and the projector. For EfficientQAT, we modify the learning rates in the second step to 2×10^{-5} for 4-bit and 3×10^{-5} for 2-bit and 3-bit.

Evaluation. Evaluation of the fine-tuned LVLMs are conducted across four benchmarks: MME (Fu et al., 2023), MM-Vet (Yu et al., 2023), MMBench (Liu et al., 2023d), and ScienceQA (Lu et al., 2022).

Baseline. We compare our results with those of QLoRA (Dettmers et al., 2023a), applying our Block-AP method to quantize the QLoRA fine-tuned models to low bits for fair comparison.

Results. As shown in Table 14, EfficientQAT outperforms QLoRA (Dettmers et al., 2023a) in low-bit settings for both LLaVA-1.5-7B and LLaVA-1.5-13B models, consistent with previous results in LMMs. Remarkably, the 2-bit LLaVA-1.5-13B model trained with EfficientQAT achieves an average score of 59.9, surpassing the 59.6 of the FP16 LLaVA-1.5-7B model trained with LoRA. However, there is a slight performance decrease observed in the 4-bit EfficientQAT and 16-bit QLoRA compared to the 16-bit LoRA, indicating that further research is needed to optimize Q-PEFT within LVLMs.