BitDistiller: Unleashing the Potential of Sub-4-Bit LLMs via Self-Distillation

Dayou Du[1], Yijia Zhang[2], Shijie Cao[3], Jiaqi Guo[3], Ting Cao[3], Xiaowen Chu[1], Ningyi Xu[2]
[1]The Hong Kong University of Science and Technology (Guangzhou)
[2]Shanghai Jiao Tong University
[3]Microsoft Research Asia
ddu487@connect.hkust-gz.edu.cn, {zhangyijia, xuningyi}@sjtu.edu.cn,
{shijiecao, jiaqiguo, ting.cao}@microsoft.com, xwchu@ust.hk

Abstract

The upscaling of Large Language Models (LLMs) has yielded impressive advances in natural language processing, yet it also poses significant deployment challenges. Weight quantization has emerged as a widely embraced solution to reduce memory and computational demands. This paper introduces BitDistiller, a framework that synergizes Quantization-Aware Training (QAT) with Knowledge Distillation (KD) to boost the performance of LLMs at ultra-low precisions (sub-4-bit). Specifically, BitDistiller first incorporates a tailored asymmetric quantization and clipping technique to maximally preserve the fidelity of quantized weights, and then proposes a novel Confidence-Aware Kullback-Leibler Divergence (CAKLD) objective, which is employed in a self-distillation manner to enable faster convergence and superior model performance. Empirical evaluations demonstrate that BitDistiller significantly surpasses existing methods in both 3-bit and 2-bit configurations on general language understanding and complex reasoning benchmarks. Notably, BitDistiller is shown to be more cost-effective, demanding fewer data and training resources. The code is available at https://github.com/DD-DuDa/BitDistiller.

1 Introduction

Scaling up model sizes has been pivotal to the success of large language models (LLMs), yielding unprecedented performance across diverse natural language processing tasks (Brown et al., 2020; Touvron et al., 2023; Kaplan et al., 2020). However, such escalating model size poses significant challenges in deployment, particularly on resource-constrained devices, due to the substantial memory footprint and computational requirements.

Weight quantization has emerged as a popular strategy to enhance the efficiency and accessibility of LLMs by reducing model size with minimal performance loss (Gholami et al., 2022). In practice, 4-bit quantization has been widely adopted, offering a balance between a considerable compression ratio and the preservation of LLM capabilities (Lin et al., 2023; Frantar et al., 2022; Liu et al., 2023a).

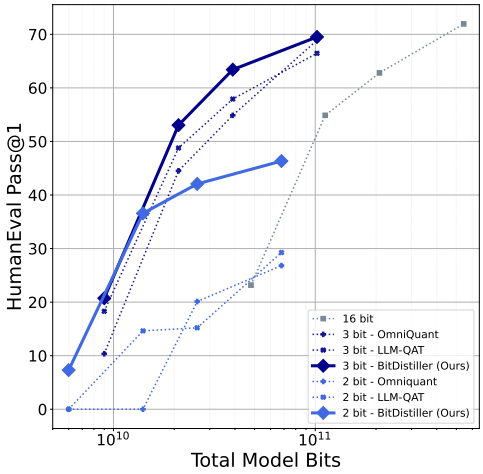


Figure 1: Bit-Level scaling laws for code generation performance for 3B to 34B parameter coder models. BitDistiller outperforms existing QAT methods in both 3-bit and 2-bit settings. Details in Table 2.

However, sub-4-bit quantization significantly degrades the fidelity of model weights, leading to deteriorated model performance, especially in smaller models or tasks requiring complex reasoning (Dettmers and Zettlemoyer, 2023). To address this, researchers have developed various Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT) methods (Chee et al., 2023; Shao et al., 2023). PTQ, while appealing without retraining, struggles to preserve model performance at very low precisions. In contrast, QAT incorporates quantization into the training loop, enabling dynamic adaptation to reduced precision and thus maintaining higher accuracy (Liu et al., 2023b; Kim et al., 2023a). Despite its early promise, two fundamental challenges are essential for achieving high model performance in extreme low-bit QAT: how to maximally preserve weight fidelity during

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 102–116

quantization, and how to effectively learn low-bit representations during training.

In this work, we present BitDistiller, a novel framework that synergizes QAT with Knowledge Distillation (KD) to significantly boost the performance of sub-4-bit quantized LLMs. To minimize quantization error, BitDistiller employs a tailored asymmetric quantization and clipping strategy to maintain the capabilities of the full-precision model as much as possible, particularly at ultra-low-bit levels. For efficient and effective low-bit representation learning, BitDistiller leverages a simple yet effective self-distillation approach, wherein the full-precision model acts as its own teacher to refine the low-bit student model. Notably, BitDistiller innovates with a Confidence-Aware Kullback-Leibler divergence (CAKLD) objective that optimizes knowledge transferring efficacy, enabling faster convergence and enhanced model performance.

Our empirical evaluations, conducted on a diverse suite of general language understanding and complex reasoning tasks including mathematics and coding, demonstrate that BitDistiller significantly outperforms existing PTQ and QAT methods in the realm of sub-4-bit quantization. As illustrated in Figure 1, BitDistiller achieves the most favorable scaling law in both 3-bit and 2-bit configurations on the code reasoning benchmark. Moreover, BitDistiller is demonstrated to be more cost-effective, requiring less training data and fewer training resources, thereby marking a significant advancement toward deploying robust Large Language Models on resource-constrained devices.

2 Background and Related Work

2.1 Weight Quantization for LLMs

PTQ and QAT PTQ is directly applied to pre-trained models without additional training. PTQ for LLMs typically employs techniques that either adjust quantization error (Frantar et al., 2022; Chee et al., 2023) or prioritize salient weights (Dettmers et al., 2023b; Lin et al., 2023; Kim et al., 2023b). However, the lack of retraining with PTQ may cause notable decreases in model performance at extremely low precisions. In contrast, QAT integrates quantization into the training phase, enabling the model to learn better representations for low-bit weights, as demonstrated by approaches like LLM-QAT (Liu et al., 2023b), OmniQuant (Shao et al., 2023), PB-LLM (Shang et al., 2023), and BitNet (Wang et al., 2023). Despite improved model performance, QAT is still challenged by the need of extensive training and data, with significant potential for further optimization and enhancement. In this work, we harness the synergy of QAT and KD to enhance the performance of quantized LLMs, especially at sub-4-bit settings.

Granularity and Format Optimizations Extensive research indicates that adopting finer-grained quantization approaches, such as group-wise quantization, can achieve higher accuracy compared to layer-wise or channel-wise methods (Shen et al., 2020; Frantar et al., 2022). Floating-point formats (FP8/FP4/NF4) have been demonstrated to deliver superior accuracy compared to integer formats (INT8/INT4) in LLM quantization (Kuzmin et al., 2022; Dettmers and Zettlemoyer, 2023; Zhang et al., 2023b). Notably, asymmetric quantization methods, particularly for floating-point formats, outperform their symmetric counterparts by better accommodating the distribution of model weights (Zhang et al., 2023a). BitDistiller aligns with these insights, employing finer granularity and asymmetric techniques for quantization.

2.2 Knowledge Distillation for LLMs

In the realm of LLMs, white-box knowledge distillation (KD) has become increasingly prevalent due to the accessible distribution of the teacher model, which facilitates the transmission of knowledge representations to the student model (Hinton et al., 2015; Zhu et al., 2023). Notably, MINILLM (Gu et al., 2023) utilizes the reverse KLD to ensure the accuracy and fidelity of language generation. GKD (Agarwal et al., 2023) has explored alternative divergences and addressed the distribution mismatch by sampling outputs from the student model during training.

To attain exceedingly high compression ratios, a promising method is to combine KD with model quantization, where KD can be effectively used to mitigate the accuracy decline of quantized models (Zhang et al., 2020; Kim et al., 2022). In cutting-edge research applying QAT-based KD for LLMs, TSLD (Kim et al., 2023a) considers risks of overfitting and conducts logit distillation with ground truth loss. Similarly, LLM-QAT leverages randomly teacher-generated data for data-free distillation. In distinction from TSLD and LLM-QAT, we achieve better performance and cost-efficiency in the extremely low-bit quantization level.

3 Methodology

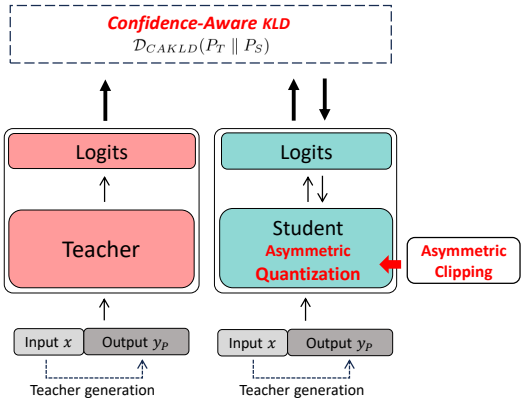


Figure 2: Depiction of the QAT-based KD framework of BitDistiller.

In this section, we introduce BitDistiller, a QAT with self-distillation framework for LLMs, as illustrated in Figure 2. To maximally preserve weight fidelity during quantization, we first present an asymmetric quantization and clipping method (see Section 3.1). Second, to counteract the performance degradation caused by precision reduction, we adopt Knowledge Distillation and propose a novel Confidence-Aware KL divergence (CAKLD) objective, in which the full-precision model acts as a teacher and the low-precision one plays a student (see Section 3.2).

Algorithm 1 outlines the process of BitDistiller. Given the full-precision weight w, BitDistiller adopts the asymmetric clipping to alleviate outliers in w (Line 1), prior to the training loop. Then, in each training step, BitDistiller forwards the model with the quantized weights (w_Q^t), computes the loss with the proposed CAKLD objective (Line 4-5), and updates the full-precision weights (Line 6-7) (Bengio et al., 2013). When the training finishes, BitDistiller returns the final quantized weights.

3.1 Asymmetric Quantization and Clipping

The adoption of finer granularities, or smaller group sizes, in weight quantization of LLMs inherently leads to asymmetrical distributions and the presence of outliers in weight groups. Proper management of asymmetry is crucial to maintaining model performance in low-bit PTQ regimes. Our investigation reveals that the effects of asymmetry are more prominent in extremely low-bit QAT, such as 3-bit and 2-bit configurations, necessitating tailored strategies to address these challenges.

Algorithm 1 BitDistiller

Input: Full-precision weight w, Dataset $\mathbb{D} = \{(x, y)\}$, Learning rate η, Training step T
Require: Clipping function $Clip$, Quantization function Q, Loss function $\mathcal{D}_{\text{CAKLD}}$
Output: Low-precision weight w_Q^T

1: $w^1 = Clip(w)$;
2: **for** $t = 1$ **to** T **do**
3: Sample a batch of data $\mathcal{B}$ from $\mathcal{D}$;
▷ Forward with quantized weight
4: $w_Q^t = Q(w^t)$;
5: Compute $\mathcal{D}_{\text{CAKLD}}(P^w \parallel P^{w_Q^t})$ on $\mathcal{B}$;
▷ Backward on full-precision weight w^t
6: Compute gradients $\frac{\partial \mathcal{D}_{\text{CAKLD}}}{\partial w^t}$;
7: $w^{t+1} = \text{Update}(w^t, \frac{\partial \mathcal{D}_{\text{CAKLD}}}{\partial w^t}, \eta)$;
8: **end for**
9: $w_Q^T = Q(w^T)$;

Therefore, in BitDistiller, we adopt asymmetric quantization techniques coupled with asymmetric clipping strategies to enhance the representational fidelity of quantized weights and maximally preserve the capabilities of the full-precision model.

Asymmetric Quantization Previous studies have shown that floating-point formats (e.g., FP, NF) often outperform integer formats (INT) in LLM quantization (Dettmers et al., 2023a; Liu et al., 2023a). However, as the quantization level falls to 2-bit, we observed a notable decline in the effectiveness of FP/NF formats. This advantage of FP/NF formats is attributed to their non-uniform nature, which can capture a wider range of values. Such a non-uniform distribution aligns better with the natural distribution of weight tensors in LLMs. In 2-bit cases, the limited representational capacity, offering only four distinct values, undermines the benefits of non-uniform distribution and impedes the efficient utilization of each numerical value. In light of these findings, we employ NF formats for quantization above 2-bit, while opting for the INT format at the 2-bit level.

For NF formats (e.g., NF3), we adopt the AFPQ method (Zhang et al., 2023a) to enable asymmetric quantization, which establishes separate scales, s_{pos} for positive weights w_{pos} and s_{neg} for negative weights w_{neg}, as shown in Equation 1. For INT formats (e.g., INT2), we utilize conventional asymmetric methods with a single scale and a designated zero point, as detailed in Equation 2.

$$NF\text{-}Asym : Q(w) = \begin{cases} \lfloor \frac{w_{pos}}{s_{pos}} \rceil, & \text{if } w > 0 \\ \lfloor \frac{w_{neg}}{s_{neg}} \rceil, & \text{if } w \leq 0 \end{cases} \quad (1)$$

$$INT\text{-}Asym : Q(w) = \lfloor \frac{w - z}{s} \rceil \quad (2)$$

Asymmetric Clipping The strategy of clipping, which involves constraining the range of weight values, has been recognized for its contribution to maintaining high accuracy after quantization (Sakr et al., 2022; Shao et al., 2023). However, naive clipping methods often fall short in effectiveness, while advanced clipping techniques come at a high computational cost which is prohibitive for practical QAT use (Li et al., 2019; Jung et al., 2019). To circumvent these limitations, we propose the use of asymmetric clipping solely during the initial phase, prior to the commencement of QAT. Asymmetric clipping at initialization provides a good starting point that significantly contributes to the final overall quantized model accuracy without incurring the prohibitive costs associated with iterative clipping optimization.

To enable asymmetric clipping for QAT initialization, given input features X cached from a small calibration set, we conduct an automatic search for two optimal clipping values, α and β, for each layer of the model. These values aim to minimize the output difference after quantization. Formally, the objective is to optimize the following:

$$\begin{aligned} \alpha^*, \beta^* &= \underset{\alpha,\beta}{\text{argmin}} ||Q(w_c)X - wX|| \\ w_c &= Clip(w, \alpha, \beta) \\ &\begin{cases} \alpha \in [\min\_val, 0) \\ \beta \in (0, \max\_val] \end{cases} \end{aligned} \quad (3)$$

To demonstrate the efficacy of asymmetric quantization and clipping, we conduct a tensor-wise analysis. We selected a random weight tensor from the LLaMa-2-7B model and focused on a single output channel. As illustrated in Figure 3, our approach to asymmetric quantization and clipping achieves higher fidelity preservation compared to symmetric quantization. A more detailed ablation study on the impact of asymmetric quantization and clipping on model performance is presented in Table 3 in Section 4.4.

3.2 Self Distillation with CAKLD

To better counteract the performance degradation caused by precision reduction, we propose to adopt

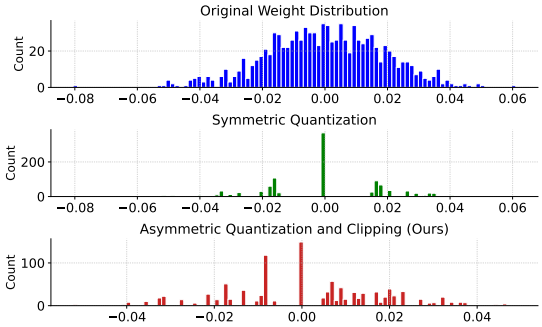


Figure 3: (Top) The original weight distribution of a single output channel in the final down projection layer of LLaMA-2-7B. (Middle&Bottom) The weight distribution after symmetric quantization and asymmetric quantization and clipping, both using 3-bit quantization with the group size of 128.

Knowledge Distillation (KD) in QAT, where the full-precision model acts as a teacher and its quantized variant plays a student:

$$\mathcal{L} = \mathcal{D}(P_T \parallel P_S), \quad (4)$$

where $\mathcal{D}$ is a divergence measure of two distributions. P_T and P_S denote the full-precision and quantized model, respectively.

The intuition for KD is two-fold. First, learning the token-level probability distributions potentially helps the quantized model better imitate its full-precision counterpart (Hinton et al., 2015), thereby re-gaining the strong downstream performance. Second, owing to the generative nature of LLM, it is easy to scale up the data size for QAT with the full-precision model.

The divergence $\mathcal{D}$ chosen for distillation plays a crucial role. Agarwal et al. (2023) find that the mode-seeking behavior advocated by the Reverse KL divergence (i.e., $\mathcal{D}_{KL}(P_S \parallel P_T)$) leads to better performance than Forward KL (i.e., $\mathcal{D}_{KL}(P_T \parallel P_S)$) on instruction tuning (Chung et al., 2022), while Forward KL promotes mode-covering and is superior on general text generation tasks like summarization (Narayan et al., 2018). To provide a general receipt for QAT, we aim to seek a way to trade off the mode-seeking and mode-covering behaviors automatically, instead of manual selection according to some empirical understanding of downstream tasks.

To this end, we propose a novel **Confidence-Aware KL divergence**, shorted as CAKLD. It blends the Reverse KL and Forward KL with a coefficient γ estimated by the averaged token probability, so that the mode-seeking and mode-covering

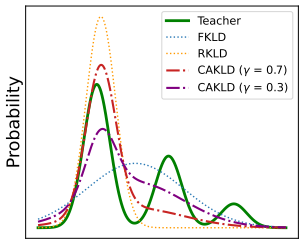


Figure 4: Comparison of Reverse KL, Forward KL and CAKLD, when a Gaussian distribution tries to fit a Gaussian mixture (Teacher).

behaviors can be automatically traded off according to the full-precision model's confidence on the training data:

$$\begin{aligned}\mathcal{D}_{CAKLD}(P_T \parallel P_S) &= \gamma \mathcal{D}_{KL}(P_S \parallel P_T) \\ &\quad + (1-\gamma)\mathcal{D}_{KL}(P_T \parallel P_S) \\ \mathcal{D}_{KL}(P_T \parallel P_S) &= \mathbb{E}_{(x,y)\sim\mathbb{D}}[\frac{1}{|\{y\}|}\sum_{i=1}^{|\{y\}|} \\ &\quad \mathbb{E}_{c\sim P_T(\cdot|x,y_{<i})}[\log \frac{P_T(c|x,y_{<i})}{P_S(c|x,y_{<i})}]] \\ \gamma &= \mathbb{E}_{(x,y)\sim\mathbb{D}}[\frac{1}{|\{y\}|}\sum_{i=1}^{|\{y\}|} P_T(y_i|x,y_{<i})]\end{aligned} \tag{5}$$

Intuitively, when the full-precision model is confident on the training data, CAKLD will prefer more on the mode-seeking behaviors. Otherwise, CAKLD will advocate more on the mode-covering behaviors, as the full-precision model is not certain about the data and modeling its single mode is sub-optimal. Figure 4 visualizes the difference between Reverse KLD, Forward KLD and CAKLD when a Gaussian distribution tries to fit a Gaussian mixture. It is clear that CAKLD manages to trade off mode-seeking and mode-covering behaviors with the coefficient. For a detailed performance comparison and in-depth analysis, please refer to Figure 6 and Appendix A.2

4 Experiments

We evaluate BitDistiller on the LLaMA-2 (Touvron et al., 2023) families and domain-specific LLMs with sub-4–bit quantization. We have set up comparative experiments to demonstrate the proficiency of our method against existing PTQ and QAT methods. Our findings illustrate that BitDistiller substantially enhances both the general language performance and the accuracy of reasoning tasks.

4.1 Experimental Settings

Tasks and Models Following (Frantar et al., 2022; Lin et al., 2023), we benchmark LLaMA-2 (Touvron et al., 2023) on general language tasks, including language modeling tasks (WikiText-2 (Merity et al., 2016)), common sense QA benchmarks (PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC (Clark et al., 2018)) and in-context learning ability (MMLU (Hendrycks et al., 2020)) under a few-shot setting. We also consider the complex reasoning tasks and evaluate various sizes of domain-specific LLMs, including WizardCoder (Luo et al., 2023) on LLM-Humaneval-Benchmarks (Chen et al., 2021) in the setting of greedy decode, and MetaMath (Yu et al., 2023) on GSM8K (Cobbe et al., 2021). To evaluate the domain-specific LLMs of smaller sizes, we finetune OpenLLaMA-3B (Geng and Liu, 2023) with domain-specific datasets.

Baselines PTQ baselines include vanilla round-to-nearest (RTN), GPTQ (Frantar et al., 2022), AWQ (Lin et al., 2023), SpQR (Dettmers et al., 2023b) and QuIP (Chee et al., 2023). QAT baselines include Omniquant (Shao et al., 2023), LLM-QAT (Liu et al., 2023b) and TSLD (Kim et al., 2023a). Detailed PTQ and QAT settings can be found in appendix A.1.

Quantization and Distillation We focus on 3-bit/2-bit group-wise quantization, with a group size of 128 (represented as 'g') as the default setting except for the 3B models with a group size of 64 because of the dimension constraint. Following (Liu et al., 2023b; Kim et al., 2023a), we utilize logits distillation. Prior to QAT, the coefficient γ, key for CAKLD, is pre-calculated from a subset of $\mathbb{D}$. The implementation details and example analysis of CAKLD are available in Appendix A.2.

Training Datasets We use the instruction-tuning data from Alpaca (Taori et al., 2023) and the training set of WikiText-2 for general language tasks. For code understanding and generation, we use Evol-Instruct-Code (Rosh, 2023). For math reasoning we use MetaMathQA (Yu et al., 2023).

Given the instruction prompt x, sequence $s = \{x, y\}$ where output $y \sim p(\cdot|x)$ have three different choices: Ground Truth y_g, Student-generated Output y_q and Teacher-generated Output y_p. As suggested by (Agarwal et al., 2023; Zhou et al., 2023), we opt to generate the Teacher-generated Output y_p using sampling with a temperature of 0.7 (Yuan et al., 2023). We conduct experiments in Section 4.4 for an ablation study on the choices of output y. (See Appendix A.3 for more details of

LLaMA-2-7B		PPL ↓	MMLU (5s)	PIQA	Hella.	Wino.	ARC-c	Avg
BF16		5.47	46.45	77.86	57.14	68.35	43.34	58.63
3 Bits g128	RTN	6.65	38.65	75.24	53.70	67.32	38.56	54.69
	GPTQ	6.38	39.57	75.46	51.68	67.16	38.39	54.45
	AWQ	6.71	39.68	76.27	55.14	67.56	40.61	55.85
	OmniQuant	6.10	41.22	77.47	54.41	67.09	39.08	55.85
	LLM-QAT	6.02	41.32	77.26	54.74	68.35	40.61	56.46
	BitDistiller (ours)	**5.97**	**43.65**	76.99	55.38	68.35	41.21	**57.12**
2 Bits g128	RTN	3453	24.12	53.43	26.33	49.96	21.58	35.08
	GPTQ	NaN	23.12	49.51	25.04	49.57	22.69	33.99
	AWQ	2.2e5	25.38	52.39	25.70	50.12	21.33	34.98
	OmniQuant	12.84	25.42	58.92	29.20	50.83	19.45	36.76
	LLM-QAT	9.30	23.62	70.08	43.79	61.64	29.09	45.64
	BitDistiller (ours)	**8.08**	**29.25**	73.61	48.70	61.09	33.27	**49.18**

Table 1: **General language task** results of BitDistiller versus established PTQ and QAT methods on LLaMA-2-7B Model. Our method achieves leading performance in both 3-bit and 2-bit quantization.

training datasets composition).

Training Implementation We leverage DeepSpeed (Rasley et al., 2020) and HuggingFace repository (Wolf et al., 2020) to devise a QAT-based KD framework enabling the distillation of models. The model optimization is facilitated through the AdamW optimizer (Loshchilov and Hutter, 2017), applied with zero weight decay. We initialize the constant learning rate to 8e-6 and set the sequence length to 1024 for the code-related task and 512 for others.

4.2 Evaluation on Language Modeling Tasks

Table 1 presents a comparative analysis of BitDistiller's performance against previous PTQ and QAT methods on general language tasks. BitDistiller surpasses competing methods in terms of WikiText-2 perplexity and MMLU (5-shot) accuracy. Furthermore, BitDistiller demonstrates consistent performance across various QA benchmarks. Notably, in 2-bit weight quantization, BitDistiller substantially increases the average accuracy by +3.54% over LLM-QAT (Liu et al., 2023b) and by +12.43% compared to the leading PTQ method (Shao et al., 2023). Similar results on LLaMA-2-13B and LLaMA-2-70B can be found in the Appendix A.4.

4.3 Evaluation on Reasoning Tasks

Table 2 demonstrates the superior performance of BitDistiller on reasoning-based benchmarks, including HumanEval and GSM8K, across a range of domain-specific language model families. BitDistiller achieves improvements over other methods in both 3-bit and 2-bit quantization. Especially in 2-bit quantization, while other methods exhibit significant performance drops, BitDistiller maintains a commendable level of accuracy. Detailedly, our method outperforms LLM-QAT by a remarkable margin of 24.69%, achieving an accuracy of 61.33% on complex mathematical reasoning tasks. These outcomes bolster the potential for implementing ultra-low-precision inference deployment in practical reasoning tasks without substantially compromising performance.

4.4 Ablation Studies

Asymmetric Quantization and Clipping In this ablation study, we evaluate the efficacy of quantization strategies on the LLaMA-2-7B model. Our approach examines the impact of asymmetric quantization and clipping techniques within QAT. We specifically assess the 3-bit and 2-bit quantization levels, reporting our findings in terms of Perplexity (PPL) and MMLU (5-shot).

As demonstrated in Table 3, asymmetric quantization significantly enhances model performance. Notably, under a 2-bit configuration, PPL can be reduced from 3.4e2 to 16.94 in post-training. Furthermore, the application of asymmetric clipping during initialization yields additional performance gains upon training completion. See Appendix A.5 for integration with other PTQ methods.

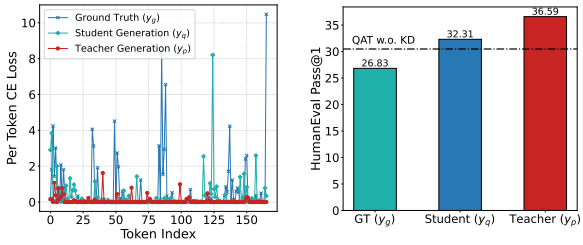
Data Generation In our analysis, we meticulously evaluated the logit information of the teacher model by computing the cross-entropy

Domain-specific LLMs		HumanEval @WizardCoder				GSM8K @MetaMath		
		3B	7B	13B	34B	3B	7B	13B
BF16		23.17	54.88	62.80	71.95	36.40	66.41	72.30
3 Bits g128	RTN	4.27	34.15	50.00	33.54	17.50	59.30	68.51
	GPTQ	4.30	46.34	55.48	63.41	6.72	62.11	68.75
	AWQ	16.46	45.73	53.04	67.07	21.87	62.34	68.67
	OmniQuant	10.36	44.51	54.88	68.90	23.67	61.70	68.28
	LLM-QAT	18.29	48.78	57.92	66.46	26.25	60.78	66.62
	BitDistiller (ours)	**20.73**	**53.66**	**63.41**	**69.51**	**32.50**	**64.38**	**69.69**
2 Bits g128	RTN	0.0	0.0	0.0	0.61	0.0	0.0	7.89
	GPTQ	0.0	0.0	1.83	3.65	0.0	0.0	11.43
	AWQ	0.0	0.0	0.0	0.0	0.0	0.0	7.89
	OmniQuant	0.0	0.0	20.12	26.83	0.0	0.0	9.45
	LLM-QAT	0.0	14.63	15.21	29.27	6.56	23.13	36.64
	BitDistiller (ours)	**7.31**	**36.59**	**42.07**	**46.34**	**16.09**	**51.02**	**61.33**

Table 2: **Reasoning task** results of BitDistiller versus established PTQ and QAT methods on domain-specific LLMs. Our method achieves leading performance in both 3-bit and 2-bit quantization.

LLaMA-2-7B		PPL ↓ (start ↦ end)	MMLU (5s) ↑ (start ↦ end)
3 Bits g128	NF-Sym	6.45 ↦ 6.10	38.28 ↦ 39.27
	→ NF-Asym	6.30 ↦ 6.01	41.53 ↦ 42.61
	+ Clip-Asym	6.08 ↦ **5.97**	42.90 ↦ **43.65**
2 Bits g128	INT-Sym	2.4e5 ↦ 2.5e5	24.95 ↦ 26.03
	→ INT-Asym	3.4e2 ↦ 16.94	24.12 ↦ 24.82
	+ Clip-Asym	17.98 ↦ **8.08**	26.75 ↦ **29.25**

Table 3: Ablation study of asymmetric quantization and clipping on WikiText2 perplexity and MMLU (5-shot). The "start ↦ end" notation denotes the metric values before and after training.



(a) (b)

Figure 5: Comparative analysis of using various data generation methods on WizardCoder-7B. (a) shows the per-token cross-entropy loss. (b) presents the HumanEval Pass@1. ('QAT w.o. KD' indicates the baseline where only the ground truth dataset is used for supervised fine-tuning, without knowledge distillation.)

loss (CELoss) for various outputs y. Figure 5a illustrates that the data generated by the teacher model y_p exhibits low CELoss, indicative of a high-confidence logit distribution, which in turn facilitates better convergence with our proposed CAKLD. The comparative performance results depicted in Figure 5b reveal that the use of teacher-generated data in conjunction with CAKLD yields superior outcomes when compared to employing either a fixed dataset or student-generated data y_q.

Distillation Objectives In Figure 6, we demonstrate the effectiveness of our proposed Confidence-Aware KL Divergence (CAKLD) by showcasing performance indicators for reasoning tasks under different objective functions. Our findings show that CAKLD outperforms other objective functions. Though JSD also has a bounded coefficient for interpolation, in practice we observe that it has a weak ability to converge for QAT.

4.5 Analysis and Discussion

Comparison with SpQR SpQR automatically identifies sensitivity outliers and stores them in sparse matrices at higher precision while compressing all other weights to lower bits. For a fair comparison, we experimented with the configuration of SpQR to achieve an average bit rate similar to BitDistiller. As shown in Table 4, BitDistiller consistently outperforms SpQR across various benchmarks. Since sparse representation is orthogonal to QAT with distillation, we plan to investigate how QAT can effectively manage outliers using higher

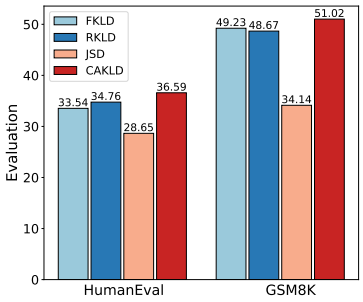


Figure 6: Performance comparison between different objective functions on WizardCoder-7B and MetaMath-7B with domain-specific tasks.

precision in sparse matrices.

Model	Avg bits	Method	PPL ↓	MMLU (5s)	QA-avg
LLama-2-7B	3.16	SpQR	6.06	40.96	59.61
	3.15	BitDistiller	5.97	43.65	60.48
	2.36	SpQR	11.99	26.35	48.47
	2.15	BitDistiller	8.08	29.25	54.17
LLama-2-13B	3.16	SpQR	5.28	52.42	63.23
	3.15	BitDistiller	5.20	53.21	63.90
	2.28	SpQR	10.54	28.87	49.69
	2.15	BitDistiller	6.78	37.50	57.63

Table 4: Performance comparison of quantized models using SpQR and BitDistiller on LLaMA-2-7B and LLaMA-2-13B.

Comparison with QuIP QuIP enhances 2-bit PTQ for LLMs through incoherence processing. Its subsequent iteration, QuIP#[1], refines this approach by shifting from scalar quantization to vector quantization via lattice codebooks, significantly narrowing the performance gap with 16-bit models. For a consistent comparison, we utilize the BF16 pretrained model and then apply Quip(#) and BitDistiller. As shown in Table 5, our BitDistiller surpasses QuIP across all benchmarks. In comparison with QuIP#, BitDistiller retains its superior performance in language modeling and programming, while QuIP# outperforms in mathematical reasoning. Being orthogonal to QAT with distillation, PTQ incorporating incoherence processing and vector quantization could potentially serve as an effective initialization method for BitDistiller. We intend to explore whether the integration of QuIP(#) into BitDistiller can further improve the performance of low-bit models.

Comparison with TSLD Prior work (Kim et al., 2023a) introduced Token-Scaled Logit Distillation

[1] https://cornell-relaxml.github.io/quip-sharp/

Method		LLaMA-2-7B PPL↓	LLaMA-2-7B MMLU (5s)	LLaMA-2-7B QA-avg	WizardCoder-7B HumanEval	MetaMath-7B GSM8K
BF16		5.47	46.45	61.67	54.88	66.41
2 Bits	Quip	55.65	24.70	39.23	0.0	0.0
	Quip#	8.97	30.90	52.40	12.96	60.00
	BitDistiller	8.08	29.25	54.17	36.58	51.02

Table 5: Performance comparison of 2-bit quantized models using QuIP, QuIP#, and BitDistiller on LLaMA-2-7B, WizardCoder-7B, and MetaMath-7B.

(TSLD) to alleviate overfitting during QAT. To facilitate a direct and fair comparison between TSLD and our CAKLD, we incorporate TSLD into the BitDistiller framework by replacing CAKLD with TSLD while keeping all other settings unchanged. As depicted in Figure 7, CAKLD not only converges more rapidly but also delivers superior overall performance compared to TSLD.

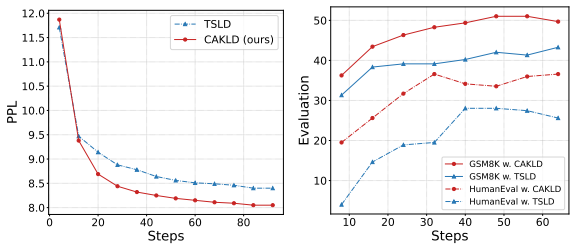


Figure 7: Comparison of TSLD and CAKLD on perplexity (left) and reasoning tasks performance (right).

Effectiveness of Self-Distillation Table 6 compares 2-bit QAT performance using the LLaMA-2-7B or larger LLaMA-2-13B as the teacher model. Surprisingly, in practice the larger 13B model didn't improve accuracy, hinting that a teacher with the same model architecture as the student may enhance weight alignment and probability distribution matching, thereby improving model effectiveness. Further investigation and deeper analysis are needed in future work to fully understand the implications of different teacher-student sizes and architectures in QAT.

LLaMA-2-7B	Quantized Student	Teacher	PPL ↓	MMLU (5s) ↑
2 Bits	7B	13B	8.12	28.27
g128	7B	7B	8.08	29.25

Table 6: Performance comparison of 2-bit quantized models using LLaMA-2-13B and LLaMA-2-7B as the teacher model.

Training Efficiency Table 7 highlights the efficiency of BitDistiller compared to LLM-QAT (Liu et al., 2023b) in quantizing the WizardCoder-7B

model. The results demonstrate a dramatic reduction in the total time required for quantization: BitDistiller completes the process in approximately 3 hours on a single A100-80G GPU, as opposed to the hundreds of GPU hours required by LLM-QAT. (Original LLM-QAT uses 64 GPUs. For a direct and fair comparison, we evaluate the GPU hours needed for LLM-QAT on a single GPU.)

Method	Devices	#Data	Time (Hours)			
			Data Gen	Quant Init	QAT	Total
LLM-QAT	1 * A100 80G	100K	270	0	10.64	280.64
BitDistiller		2K	1.47	0.63	0.92	3.02

Table 7: Time required for LLM-QAT and BitDistiller to quantize WizardCoder-7B on a NVIDIA A100-80G.

5 Conclusion

BitDistiller leverages QAT with self-distillation to boost sub-4-bit LLM performance. The asymmetric quantization and clipping strategies, coupled with the innovative CAKLD objective, facilitate faster learning and superior performance. BitDistiller outperforms existing PTQ and QAT methods, achieving notable improvements in 3/2-bit settings across diverse language and reasoning tasks. Moreover, BitDistiller is more cost-efficient with fewer data and training resources required.

Limitations

Despite the promising results demonstrated by BitDistiller, it is important to acknowledge certain limitations and areas for future investigation.

A key limitation lies in the empirical nature of our findings. For instance, the reason behind the counterintuitive outcome where a 7B model outperforms a 13B model as a teacher during the distillation of a 2-bit 7B student model. Having the same model architecture may be the reason but not detailed explained and understood. This highlights the need for a deeper investigation and theoretical exploration to complement our empirical observations.

Looking ahead, we aim to extend BitDistiller to the realm of 1-bit (binary) quantization. While this presents a more challenging scenario, it also offers the potential for significant advancements in efficient LLM inference as binary weights enables computation with only additions and without multiplications.

Moreover, the current iteration of BitDistiller applies exclusively to scalar quantization. As future work, we plan to explore the adaptation of BitDistiller to vector quantization. Preliminary research in this area indicates that vector quantization could yield substantial benefits, and incorporating it into our framework represents a natural and promising progression of our research.

Acknowledgements

This work was partially supported by Hong Kong CRF grant under Grant NO. C70004-22G. And we would like to thank the HPC-AI-Integrated Intelligent Computing center of HKUST(GZ) for providing some of the hardware platforms in this project.

References

Rishabh Agarwal, Nino Vieillard, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. 2023. Gkd: Generalized knowledge distillation for auto-regressive sequence models. *arXiv preprint arXiv:2306.13649*.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

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.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan

Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

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*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

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*.

Tim Dettmers and Luke Zettlemoyer. 2023. The case for 4-bit precision: k-bit inference scaling laws. In *International Conference on Machine Learning*, pages 7750–7774. PMLR.

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*.

Xinyang Geng and Hao Liu. 2023. Openllama: An open reproduction of llama.

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

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*.

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*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. 2019. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4350–4359.

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

Minsoo Kim, Sihwa Lee, Sukjin Hong, Du-Seong Chang, and Jungwook Choi. 2022. Understanding and improving knowledge distillation for quantization-aware training of large transformer encoders. *arXiv preprint arXiv:2211.11014*.

Minsoo Kim, Sihwa Lee, Janghwan Lee, Sukjin Hong, Du-Seong Chang, Wonyong Sung, and Jungwook Choi. 2023a. Token-scaled logit distillation for ternary weight generative language models. *arXiv preprint arXiv:2308.06744*.

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*.

Andrey Kuzmin, Mart Van Baalen, Yuwei Ren, Markus Nagel, Jorn Peters, and Tijmen Blankevoort. 2022. Fp8 quantization: The power of the exponent. *Advances in Neural Information Processing Systems*, 35:14651–14662.

Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. 2019. Fully quantized network for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2810–2819.

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*.

Shih-yang Liu, Zechun Liu, Xijie Huang, Pingcheng Dong, and Kwang-Ting Cheng. 2023a. Llm-fp4: 4-bit floating-point quantized transformers. *arXiv preprint arXiv:2310.16836*.

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*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct.

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

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Nick Rosh. 2023. Evol-teacher: Recreating wizardcoder. https://github.com/nickrosh/evol-teacher.

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.

Charbel Sakr, Steve Dai, Rangha Venkatesan, Brian Zimmer, William Dally, and Brucek Khailany. 2022. Optimal clipping and magnitude-aware differentiation for improved quantization-aware training. In *International Conference on Machine Learning*, pages 19123–19138. PMLR.

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*.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8815–8821.

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.

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*.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, JamesT. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*.

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*.

Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. 2020. Ternarybert: Distillation-aware ultra-low bit bert. *arXiv preprint arXiv:2009.12812*.

Yijia Zhang, Sicheng Zhang, Shijie Cao, Dayou Du, Jianyu Wei, Ting Cao, and Ningyi Xu. 2023a. Afpq: Asymmetric floating point quantization for llms.

Yijia Zhang, Lingran Zhao, Shijie Cao, Wenqiang Wang, Ting Cao, Fan Yang, Mao Yang, Shanghang Zhang, and Ningyi Xu. 2023b. Integer or floating point? new outlooks for low-bit quantization on large language models. *arXiv preprint arXiv:2305.12356*.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. 2023. Distillspec: Improving speculative decoding via knowledge distillation.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2023. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*.

A Appendix

A.1 Details of PTQ and QAT Configuration

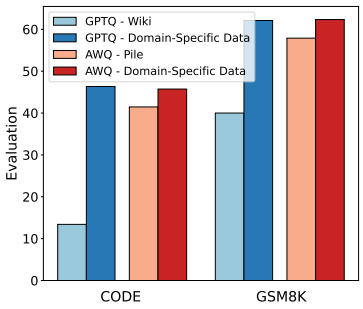
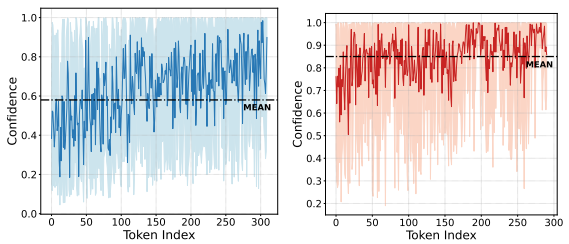


Figure 8: Comparative Evaluation of PTQ Methods Using Various Calibration Datasets

We evaluate PTQ methods by examining the impact of different calibration dataset distributions. Illustrated in Figure 8, calibrating with domain-specific data significantly enhances task-specific performance. For a fair comparison, all PTQ methods utilize the default calibration datasets for general language tasks and domain-specific calibration datasets (Rosh, 2023; Yu et al., 2023) for reasoning tasks.

Regarding QAT methods, it should be noted that the use of symmetric quantization in LLM-QAT results in degradation when grouped quantization is applied. To ensure a fair comparison, we replicate the approach with our setup and employ asymmetric uniform quantization.

A.2 Implementation Details and Analysis of Confidence-Aware KLD



(a) Text Generation Task (b) Reasoning Task

Figure 9: Per-token confidence scores when teacher model (full-precision) conducting text generation task and reasoning task.

We use a straightforward method in the pre-calculation of the coefficient γ. We utilize ten batches of training data to perform forward passes without updating parameters. Subsequently, we obtain the logits from the teacher model to compute the average token probability. In Figure 9, we have conducted analysis by examining the confidence scores of the teacher model in various tasks during next-word prediction. This analysis reveals that confidence levels can vary in text generation tasks, in contrast to reasoning tasks where each step is critical. Notably, in text generation tasks using LLMs, relying solely on the highest conditional probability through Greedy Search may result in local optima, overlooking more optimal sequences. These observations advocate for a mean-seeking Kullback-Leibler (KL) approach, encouraging the student model to encompass all potential modes of the teacher, thereby more effectively capturing the teacher's general generative capabilities. In reasoning tasks, where the teacher model shows high confidence in next-word predictions, the student model should concentrate on learning the predominant mode from the teacher. Our proposed method, CAKLD, is designed to balance these two distinct modes effectively.

A.3 Training Datasets Examples

For general language tasks, we mix token sequences from Alpaca and WikiText-2 datasets with a ratio of 2:1. Since WikiText-2 lacks explicit instructions, we utilize the first 128 tokens from the corpus as the input prompt for the teacher model's generation process, setting the temperature to 0.7. For tasks related to code understanding and generation, we employ the Evol-Instruct-Code dataset. For mathematical reasoning, we utilize MetaMathQA. Examples of the training data utilized are shown in Table 9.

It is essential to highlight that our self-distillation process utilizes only a small portion of the involved datasets.

A.4 Evaluation of General Language Tasks on LLaMA-2-13B and LLaMA-2-70B

Additional results for the General Language Tasks for LLaMA-2-13B and LLaMA-2-70B are presented in Table 10 and Table 11, respectively.

A.5 Integration with AWQ For Quantization Strategies

As shown in Table 8, we explore the efficacy of combining asymmetric clipping with AWQ during the self-distillation process. Our results indicate

LLaMA-2-7B		PPL ↓ (start ↦ end)
3 Bits g128	INT-Asym	6.65 ↦ 6.15
	AWQ	6.48 ↦ 6.09
	Clip-Asym	6.21 ↦ 6.00
	AWQ + Clip-Asym	6.18 ↦ 6.00
2 Bits g128	INT-Asym	3.4e2 ↦ 16.94
	AWQ	2.2e5 ↦ Inf
	Clip-Asym	17.98 ↦ 8.08
	AWQ + Clip-Asym	16.61 ↦ 8.13

Table 8: Results of quantization initialization for QAT combining with AWQ on PPL of WikiText-2.

that asymmetric clipping significantly enhances robustness in sub-4-bit quantization scenarios. For instance, at the 2-bit quantization level, both INT-Asym and AWQ methods are unable to complete the task. Conversely, Clip-Asym not only succeeds but also achieves a marked improvement in perplexity. It is also noteworthy that while integrating AWQ prior to QAT yields improvements initially, there is no additional performance gain after training. This suggests that a straightforward clipping approach is sufficiently effective for initializing QAT.

Tasks	Resources	Number	Instruction	Teacher Response
General Language Tasks	Alpaca	4K	Find the five steps to meditate	1. Find a quiet place to meditate\n2. Sit in a comfortable position\n3. Close your eyes and take a deep breath\n4. Focus your attention on your breath\n5. When thoughts invade your mind, gently bring your attention back to your breath
General Language Tasks	WikiText	2K	In December 1856 , Rice brought forward two bills in Congress : an enabling act that would allow Minnesota to form a state cons	itution, and a bill that would allow Missouri to secede from the Union. The bills were defeated, and Rice was removed from the chairmanship of the House Judiciary Committee. .....
Code-related Tasks	Evol-Instruct-Code	2K	Create a nested loop to print every combination of numbers between 0-9, excluding any combination that contains the number 5. Additionally, exclude any combination that contains a repeating digit. Implement the solution without using any built-in functions or libraries to check for repeating digits.	Here's one way to implement the solution:\n“‘ python\r\n for i in range(10):.....
Math-related Tasks	MetaMathQA	2K	A store is comparing their profits throughout the year. They had profits of $1,500 in the first quarter of the year, $3,000 in the third quarter, and $2,000 in the fourth quarter. If their annual profits are $8,000, how much profit, in dollars, did they make in the second quarter?	\nThe annual profit is the sum of the profits in each quarter, so $8,000 = $1,500 + $3,000 + $2,000 + x, where x is the profit in the second quarter.\nTo find x, we need to isolate it on one side of the equation.....

Table 9: The Training Dataset examples for different tasks.

LLaMA-2-13B		PPL ↓	MMLU (5s)	PIQA	Hella.	Wino.	ARC-c	Avg
BF16		4.88	55.54	79.16	60.13	72.14	48.12	63.02
3 Bits g128	RTN	5.52	50.74	78.35	57.75	71.11	43.86	60.36
	GPTQ	5.41	50.63	77.26	56.84	70.72	42.83	59.66
	AWQ	5.47	49.64	77.09	57.52	70.32	43.86	59.69
	OmniQuant	5.48	48.97	77.64	57.08	70.88	44.28	59.77
	LLM-QAT	5.32	51.60	78.29	58.45	70.56	44.62	60.70
	BitDistiller (ours)	**5.20**	**53.21**	78.67	58.66	71.59	46.67	**61.76**
2 Bits g128	RTN	109.21	24.74	57.56	32.56	50.75	21.84	37.49
	GPTQ	15.08	23.70	56.04	30.99	51.22	19.28	36.25
	AWQ	1.2e5	27.04	53.16	25.82	51.70	23.04	36.15
	OmniQuant	25.69	26.09	61.81	31.92	51.38	22.27	38.69
	LLM-QAT	7.80	29.37	74.10	49.49	63.14	33.87	49.99
	BitDistiller (ours)	**6.78**	**37.50**	75.84	51.30	65.90	37.46	**53.60**

Table 10: **General language task** results of BitDistiller versus established PTQ and QAT Methods on LLaMA-2-13B Model. Our method achieves leading performance in both 3-bit and 2-bit quantization.

LLaMA-2-70B		PPL ↓	PIQA	Hella.	Wino.	ARC-c	Avg
BF16		3.32	82.32	64.68	77.74	54.18	69.73
2 Bits	RTN	28.45	94.96	40.11	55.17	26.45	46.67
	GPTQ	8.35	62.79	36.54	57.30	23.97	45.15
	AWQ	7e5	52.29	25.82	48.15	22.78	37.18
	OmniQuant	9.77	70.72	45.18	56.20	31.40	50.88
	BitDistiller (ours)	**5.54**	79.54	60.38	75.37	47.10	**65.60**

Table 11: **General language task** results of BitDistiller versus established PTQ and QAT Methods on LLaMA-2-70B Model. Our method achieves leading performance in 2-bit quantization.