Direct Quantized Training of Language Models with Stochastic Rounding

Kaiyan Zhao[1] KAIYAN1006@LOGOS.T.U-TOKYO.AC.JP
Tsuguchika Tabaru[2] TABARU@FUJITSU.COM
Kenichi Kobayashi[2] KENICHI@FUJITSU.COM
Takumi Honda[2] HONDA.TAKUMI@FUJITSU.COM
Masafumi Yamazaki[2] M.YAMAZAKI@FUJITSU.COM
Yoshimasa Tsuruoka[1] TSURUOKA@LOGOS.T.U-TOKYO.AC.JP
[1] *The University of Tokyo, Tokyo, Japan*
[2] *Fujitsu Limited, Kawasaki, Japan*

Editors: Hung-yi Lee and Tongliang Liu

Abstract

Although recent quantized Large Language Models, such as BitNet, have paved the way for significant reduction in memory usage during deployment with binary or ternary weights, training these models still demands substantial memory footprints. This is partly because high-precision (i.e., unquantized) weights required for straight-through estimation must be maintained throughout the whole training process. To address this, we explore directly updating the quantized low-precision weights without relying on straight-through estimation during backpropagation, aiming to save memory usage during training. Specifically, we employ a stochastic rounding technique to minimize the information loss caused by the use of low-bit weights throughout training. Experimental results on our LLaMA-structured models of various sizes indicate that (1) training with only low-precision weights is feasible even when they are constrained to ternary values; (2) extending the bit width to 8 bits achieves performance on par with BitNet b1.58; (3) our models remain robust to precision scaling and memory reduction, showing minimal performance degradation when moving from FP32 to lower-memory environments (BF16/FP8); and (4) our models also support inference using ternary weights, showcasing their flexibility in deployment.[1]

Keywords: Large Language Models; Quantization-Aware Training.

1. Introduction

Large Language Models (LLMs) have become a promising solution for a wide range of Natural Language Processing (NLP) tasks, including machine translation (Xu et al., 2024a; Wu et al., 2024; Miao et al., 2025), reasoning (OpenAI, 2022, 2024; Kojima et al., 2024; Miao et al., 2024) and multimodal understanding (Huang et al., 2025; Mao et al., 2025). However, their development is challenged by the need for vast datasets and substantial computational resources, especially as the size of current LLMs continually grows larger (Duan et al., 2024).

Quantization, which involves converting high-precision parameter matrices into lower-precision formats, has emerged as an effective approach for enabling resource-efficient LLMs. Traditional quantization methods can be divided into two categories: Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT). PTQ reduces the bit precision

1. Code is available at https://github.com/KYuuto1006/DQT.

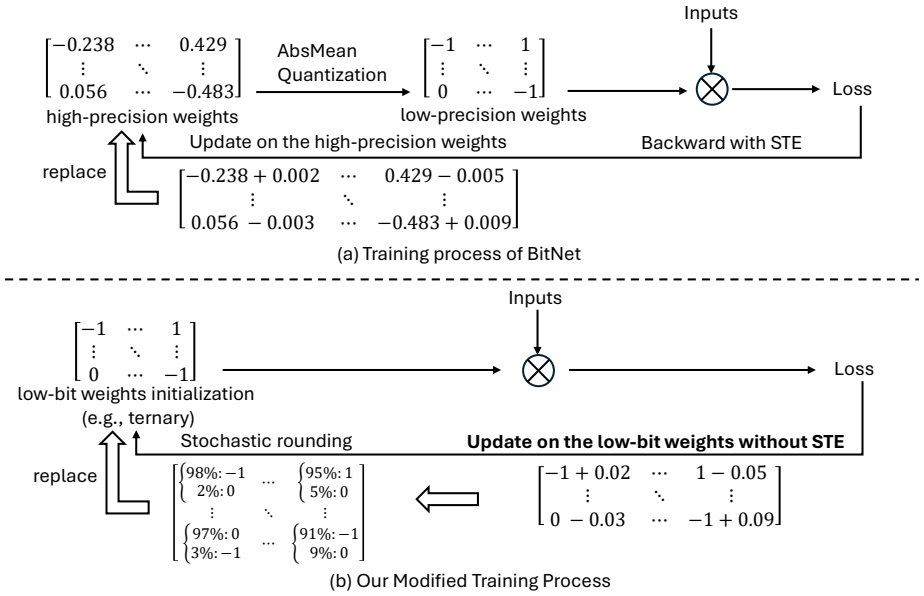

Figure 1: Comparison of the training process for BitNet and our modified one. Upper: The training process for BitNet, where the original high precision weights are updated with the straight-through estimator in backward process. Lower: We directly update the low precision weights with stochastic rounding, eliminating the need to quantize the weight matrices in each training step and keeping weight matrices always at low-bit. We provide an 8-bit example in Supplementary Material, Figure 3.

of weight matrices in an already pre-trained LLM (Banner et al., 2019; Frantar et al., 2023), while QAT incorporates quantization during training, enabling the model to adapt to low-bit precision throughout the learning process (Jacob et al., 2018).

Recently, a QAT method, BitNet (Wang et al., 2023; Ma et al., 2024), has shown the feasibility of quantizing full-precision (FP32) transformers into binary or ternary models from scratch, while maintaining competitive performance with unquantized ones. We illustrate the training process of traditional QAT methods such as BitNet in Figure 1 (a). After computing the loss based on the quantized weights and inputs, the loss is backpropagated to update the original high-precision weights via the straight-through estimator (STE) (Bengio et al., 2013). These weights are then re-quantized in each training step. This iterative process is necessary because quantization itself is not differentiable (Chen et al., 2019), requiring special gradient accumulation methods. As a result, the high-precision weight matrices are always maintained throughout the whole training process, which makes traditional QAT inefficient and takes up a lot of extra memory footprints. For example, maintaining the weights of a 1B LLM alone would require 4GB of memory in the FP32 format, whereas maintaining ternary weights reduces this to 0.2GB. This requirement for memory

footprints limits the accessibility of QAT techniques for researchers and organizations with limited computational resources.

In order to address these challenges, we explore Direct Quantized Training (DQT), a modified QAT approach for language models that maintains only low-precision weights throughout the training process. DQT eliminates the reliance on STE by directly updating the quantized low-precision weights during backpropagation, as shown in Figure 1 (b). Specifically, we use stochastic rounding (Von Neumann and Goldstine, 1947) to preserve the low-precision format of the weight matrices after backpropagation and minimize the information loss caused by using only low-bit weights. The DQT method keeps all weight matrices fixed at n-bit precision (INTn) throughout the entire training process and eliminates the needs to quantize high-precision weight matrices at each training step. More importantly, traditional QAT often introduces extra memory overhead due to the updates on high-precision weights, which limits its practicability in training general LLMs. In contrast, the light-weighted memory dependency of DQT enables quantization's application even for scenarios where computational resources are constrained.

Experimental results on our LLaMA-structured models ranging from 130M, 320M and 1B parameters demonstrate that (1) DQT enables model convergence even when weight matrices are constrained to ternary values; (2) with 8-bit DQT, our models can achieve performance levels competitive with BitNet b1.58, showing the feasibility of the approach; (3) DQT models exhibit robust performance against GPU memory reductions, showing their light-weighted memory dependency; and (4) inference using only ternary weights in DQT remains effective, delivering performance comparable to BitNet. In addition, we conduct an in-depth analysis of stochastic rounding and find that it helps preserve critical update signals and contributes to training stability. We assume that DQT could provide new insights on addressing the computational challenges posed by traditional QAT.

2. Related Works

Efficient learning methods for LLMs have become a critical area of research (Zhao et al., 2025). Quantization for deep neural networks has a history spanning nearly ten years, with researchers initially compressing networks to reduce memory usage and computational load while maintaining accuracy (Rastegari et al., 2016; Hubara et al., 2018). Recent quantization methods for LLMs can be divided into two categories: Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT).

Post-Training Quantization PTQ transforms high-precision parameters into low-bit ones after the parameters are already pre-trained. Some approaches utilize a small set of calibration data to accomplish this transformation while preserving the model's performance (Nagel et al., 2020; Li et al., 2021; Frantar et al., 2023). Others explore methods that eliminate the need for calibration data altogether (Cai et al., 2020). The challenge of PTQ lies in achieving a balance between compression efficiency and minimal accuracy degradation, often involving computational trade-offs that make it a non-trivial task for general-purpose applications. Moreover, the performance of PTQ consistently lags behind that of QAT, as there is a gap between the learned high-precision representations and the constrained bit width (Chen et al., 2024; Liu et al., 2024).

Quantization-Aware Training To bridge the gap between high-precision parameter training and quantization, QAT incorporates the quantization of model parameters during the training process. One of the initial applications of QAT to LLMs comes from Liu et al. (2024), who propose a data-free distillation-based method and quantize the model to 4 bits. Xu et al. (2024b) further expand distillation-based methods to binary quantization by introducing limited trainable vectors. More recently, BitNet (Wang et al., 2023; Ma et al., 2024) is proposed, achieving training from scratch QAT with weight values constrained to $\{-1, 0, 1\}$. However, due to the non-differentiability of the quantization process, special gradient approximation methods like the straight-through estimator (STE) (Bengio et al., 2013) are commonly employed during training. While effective, this approach often results in slower training processes and increased computational memory overhead since high-precision weights are always maintained during the training process. These inefficiencies become particularly pronounced when scaling QAT to larger models, limiting the practical training of QAT in real-world use cases. To address this, in this work, we explore the potential for more efficient and practical QAT methods that have significantly lower memory dependency compared to traditional QAT approaches.

3. Method

In this section, we first introduce stochastic rounding, the core idea of our proposed method that maintains weight matrices at low-precision during training. Then we move on to describe how stochastic rounding is applied in the modified training process for QAT.

3.1. Stochastic Rounding

The idea of stochastic rounding (SR) originates from Von Neumann and Goldstine (1947), and is initially used for reducing bias in numerical computations and recently for deep learning models (Gupta et al., 2015). It is a rounding technique that probabilistically rounds values to the nearest representable precision based on their distance from those values. Given a high precision value x, stochastic rounding can be defined as the following equation (Gupta et al., 2015; Markov et al., 2023; Zhang et al., 2024):

$$\mathrm{SR}(x) = \begin{cases} \lfloor x \rfloor, & \text{with } p = \lceil x \rceil - x \\ \lceil x \rceil, & \text{otherwise} \end{cases}, \tag{1}$$

where p stands for the probability of turning x to floor(x) or ceil(x). In this way, we can naturally quantize high-precision values into low-precision ones.

3.2. Modified Training Process

Next, we continue to explain the details of DQT. As shown in Figure 1 (b), we start training from low-precision weight matrices instead of high-precision ones. We achieve this through utilizing AbsMean Quantization following Ma et al. (2024) on a randomly initiated weight matrix W. The absolute mean value for W can be represented as

$$\mathrm{AbsMean}(W) = \frac{1}{k} \sum_{i=1}^{k} |w_i|, \tag{2}$$

where w_i is the i_{th} element in W. For n-bit quantization, $Q_n = -2^{n-1}$ and $Q_p = 2^{n-1} - 1$ are then determined to constrain the range of quantization, and the scaling factor s can be defined as

$$s = \frac{Q_p}{\text{AbsMean}(W)}. \tag{3}$$

Finally, the quantized weights $\widetilde{W}$ can be expressed in the following equation:

$$\widetilde{W} = \text{Clip}[\text{Round}(W \cdot s), Q_n, Q_p]/s, \tag{4}$$

where the Clip() function assures that all the values are in the range of $[Q_n, Q_p]$ and Round() returns the nearest integer. This allows us to constrain the quantized $\widetilde{W}$ to n-bits (INTn). While for the inputs and activations, we follow the settings introduced in BitNet (Wang et al., 2023; Ma et al., 2024) and quantize them into 8 bits.

After computing the language modeling cross-entropy loss based on $\widetilde{W}$ and inputs in each training step, we first allow the optimizer to calculate W', the dense weight matrix intended for updating. In traditional QAT, the straight-through estimator is applied, and W' replaces the original high-precision W, subsequently undergoing the quantization process from Equation (2) to Equation (4) again in the next training step. However, in our approach, we simplify this process by directly applying stochastic rounding on W' using Equation (1):

$$\widehat{W} = \text{SR}(W'), \tag{5}$$

to make sure it maintains n-bits (INTn) without requiring the retention of high-precision weights, thus getting rid of the straight-through estimator and skipping the process from Equation (2) to Equation (4). During our modified DQT, we directly replace $\widetilde{W}$ with $\widehat{W}$ and proceed to the next training step, ensuring that the weight matrices are always constrained to n-bits throughout training, which makes the biggest difference from traditional QAT.

4. Experiments

In this section, we begin by detailing our training dataset and implementation setup in Section 4.1. Section 4.2 presents a comparison between the training behavior of DQT models and other baselines. To demonstrate the robustness of DQT under reduced GPU memory environments, we evaluate its performance in FP32, BF16, and FP8 formats, as well as with memory-efficient optimizers, in Section 4.3. We then analyze the effect of varying bit widths in DQT in Section 4.4. Finally, Section 4.5 reports evaluation results on various tasks.

4.1. Implementation Setup

Model architecture. We follow the architecture of LLaMA2 (Touvron et al., 2023) and initialize models with sizes ranging from 130M to 320M and 1B parameters. Detailed model specifications and configurations are provided in the Supplementary Material, Section 1.1. For the main experiments, we use AdamW (Loshchilov and Hutter, 2019) as the optimizer. For DQT models, we apply modified versions of the optimizers that incorporate stochastic rounding.

Baselines. We choose two types of baselines for comparison with our DQT. The first is a reproduced full-precision (FP32) model. We compare DQT to it to demonstrate the effectiveness of our approach with reduced bits. The second baseline is the reproduced BitNet b1.58, a well-known QAT method that employs ternary weights at inference but relies on high-precision weights during training.

Datasets We use two types of datasets to pretrain the models. The first is the English Wikipedia dataset (20231101.en)[2]. For the 1B models, we further pre-train them using a larger dataset containing 10B tokens from the FineWeb dataset (Penedo et al., 2024)[3]. We split 1% of the data as the corresponding development set. Models are trained for one epoch with a cosine scheduler applied and a 2000 step warm-up. Refer to Supplementary Material, Section 1.2 for details on how we process the training dataset.

Evaluation. Besides reporting the training loss and perplexity results, we perform zero-shot evaluation on general language modeling tasks using the lm_eval benchmark (Gao et al., 2024). Specifically, we evaluate on WinoGrande (Sakaguchi et al., 2019), ARC (Clark et al., 2018), PIQA (Bisk et al., 2020) and SciQ (Welbl et al., 2017), which span a range of reasoning and question answering tasks across diverse domains.

Hardware. We pre-train the 130M models on 4 NVIDIA A100 80GB GPUs, the 330M models on 8 NVIDIA GH200 Grace Hopper Superchips, and the 1B models on 16 GH200 Superchips. To remain hardware-agnostic, we simulate our quantization approach under FP32, BF16, and FP8 environments. We use MS-AMP[4] for FP8 experiments with optimization level set to MS-AMP O2. While we recognize that FP8 is still not true ternary precision, it offers a practical compromise under current hardware constraints, as done in many prior works.

4.2. Main Results

We first present the training loss for DQT variants, our reproduced LLaMA (FP32) and BitNet b1.58 in Figure 2, including different model sizes and training datasets under the FP32 environment. In all subfigures of Figure 2, the blue line represents the ternary implementation of DQT, where weight matrices are always constrained to $\{-1, 0, 1\}$. The green line denotes the 8-bit (INT8) implementation of DQT (in the format of FP32 due to simulation). The orange line corresponds to the reproduced BitNet b1.58, which utilizes high-precision information during training. Finally, the red line indicates the standard FP32 LLaMA implementation.

We first examine the blue lines: although there remains a performance gap compared to higher-precision models, the ternary DQT implementation still demonstrates the ability to converge. Across all model sizes and training datasets, we observe that standard FP32 models (red lines) consistently achieve the best performance, as they are not subject to any quantization. BitNet (orange lines) consistently delivers performance close to FP32, benefiting from high-precision updates during training. Finally, for our DQT 8 bit implementation (green lines), the performance gap with BitNet b1.58 narrows as model

2. https://huggingface.co/datasets/wikimedia/wikipedia
3. https://huggingface.co/datasets/HuggingFaceFW/fineweb/viewer/sample-10BT
4. https://github.com/Azure/MS-AMP

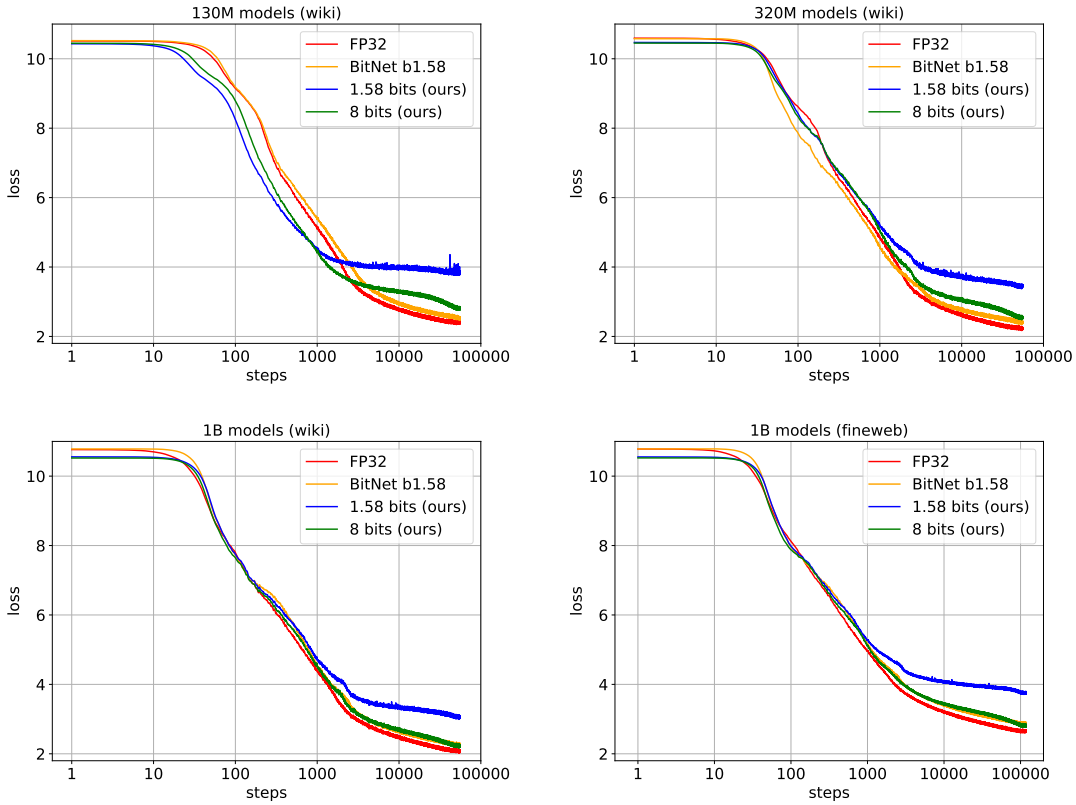

Figure 2: Comparison of our DQT and other baselines across different model sizes and training datasets. The horizontal axis represents the training steps while the vertical axis represents the training loss. As model size increases, the performance of our DQT models, especially DQT 8 bit, become more comparable to and even better than the reproduced BitNet b1.58.

size increases. Particularly, in the 1B models, DQT 8 bit models surpass BitNet b1.58, and this trend persists even when training on the larger FineWeb dataset. A clearer non-logarithmic comparison between the 1B DQT 8-bit model and BitNet b1.58 is provided in the Supplementary Material, Figure 1.

Generally speaking, as model size increases, all models show improved performance, reflecting the general benefits of scaling. However, the performance gains of our DQT models are especially pronounced. This suggests that DQT benefits more from scaling than other quantized approaches, narrowing the gap with other baselines and, in some cases, even surpassing them. Notably, since our DQT models do not rely on high-precision weights during training, even when 8-bit quantization is used, their memory requirements are significantly lower than those of BitNet, which still depends on high-precision updates. In the next section, we demonstrate DQT's low GPU memory dependency in detail.

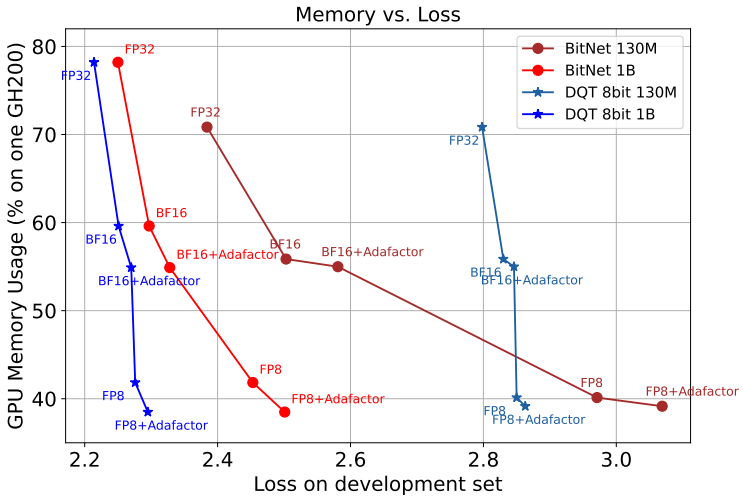

Figure 3: GPU memory usage versus loss on development set. While BitNet suffers significant performance degradation in low-precision formats, DQT demonstrates strong robustness with minimal loss increase.

4.3. Low Memory Experiments

We additionally conduct two types of experiments to evaluate DQT's performance given reduced GPU memory, simulating resource-limited environments. First, we assess performance in low-precision formats using BF16 and FP8 [5]. Second, we examine the effect of memory-efficient optimizers, as standard AdamW maintains two high-precision states (momentum and variance) for each parameter, contributing significantly to memory usage. To demonstrate DQT's low reliance on high-precision information, we specifically choose Adafactor (Shazeer and Stern, 2018), which eliminates the need for fully storing these additional states. Note that we conduct experiments with Adafactor in BF16 and FP8 formats, as its benefits are most pronounced in low-precision environments.

Figure 3 presents the GPU memory usage and corresponding development set loss at the end of training for 130M and 1B models trained on the Wikipedia dataset. The y-axis indicates the percentage of GPU memory used on a single GH200 GPU under our experimental settings. We explain how we acquire the percentage in Supplementary Material, Section 1.4. As shown in the figure, BitNet models experience clear performance degradation as GPU memory usage decreases in BF16 and FP8 settings, for both 130M and 1B scales. This is expected, since BitNet still relies on high-precision information for effective weight updates. In contrast, our DQT 8-bit models maintain robust performance under reduced precision. For both 130M and 1B models, the performance drop is less than 0.1 in loss, demonstrating DQT's resilience to lower GPU memory usage and its minimal dependence on high-precision

5. In our notation,"n-bit" refers to the quantization level of the model weights (INT-n), while the labels such as FP32, BF16, or FP8 refer to the precision of the environment. Note that any "n-bit" can be simulated in these precisions.

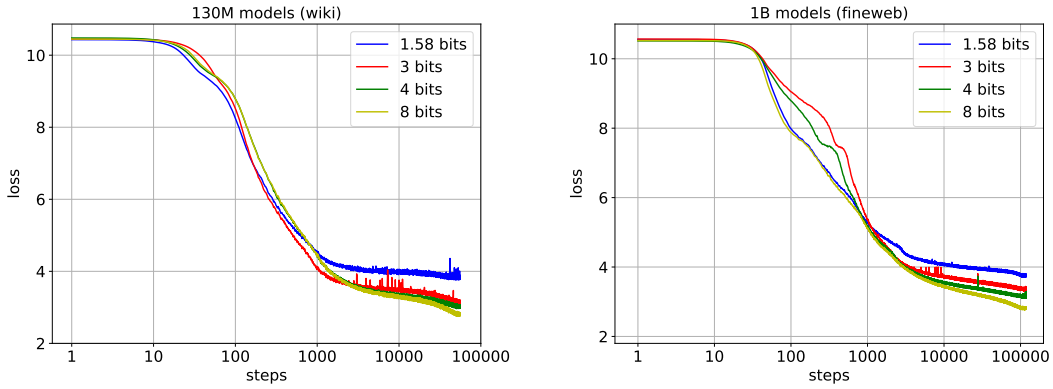

Figure 4: Comparison of bit widths in DQT. Higher n-bit results in better performance.

information. This robustness persists even when applying memory-efficient optimizers such as Adafactor in both BF16 and FP8 settings. These results suggest that DQT models can be effectively trained even in memory-constrained environments, showing consistent performance across both low-precision formats and lightweight optimization methods. We believe that DQT can be further extended to even lower-precision settings while maintaining its effectiveness.

4.4. Impact of the Bit Width in DQT

Next, we proceed to examine the impact of varying the number of bits in DQT. Specifically, we experiment with n in $\{1.58, 3, 4, 8\}$ on 130M models trained with the Wikipedia dataset and 1B models trained with the FineWeb dataset under FP32. The corresponding results are presented in Figure 4. We select the optimal learning rate for each model.

From Figure 4, we can first observe a clear trend: as the number of bits used in DQT increases, the model's performance improves consistently for both 130M and 1B models. Notably, for the relatively lower bits DQT implementations (1.58-bit and 3-bit), we can observe some outliers in the blue line and red line, indicating the difficulty of training low-bit models. In contrast, the higher-bit implementations exhibit more stability throughout the training process.

4.5. Evaluation Results

In this section, we conduct evaluation on the WikiText-2 test set (Merity et al., 2016) and several other zero-shot tasks. We focus on 1B parameter models to best demonstrate the capability of our approach. The results are summarized in Table 1. For consistency, we set the sequence length to 512 across all tasks, matching the configuration used during training. In the table, 'ternary Inf.' refers to DQT models trained with 8-bit weights but evaluated using ternary weights. Details on how ternary inference is implemented for DQT models are provided in Supplementary Material, Section 3. As shown in Table 1, FP32 models achieve the highest overall performance across tasks, followed closely by our DQT 8-bit variants. Notably, except for the WinoGrande task using 1B models trained on the Wikipedia dataset,

Models	Wikitext2(↓)	WinoGrande (↑)	ARC (easy) (↑)	ARC (challenge) (↑)	PIQA (↑)	SciQ (↑)
1B models (wiki)						
FP32	27.03	49.01	39.56	23.29	56.20	70.70
BitNet b1.58	34.90	51.38	36.74	22.95	54.68	69.40
DQT 8 bit	30.94	49.96	37.54	23.55	56.47	69.50
DQT 8 bit (ternary Inf.)	35.51	51.30	36.36	24.23	54.19	68.90
1B models (fineweb)						
FP32	19.99	52.33	48.06	25.00	67.90	80.20
BitNet b1.58	28.20	51.22	44.36	22.53	65.40	75.40
DQT 8 bit	25.43	51.97	45.12	23.63	66.38	75.90
DQT 8 bit (ternary Inf.)	27.32	51.70	45.62	22.87	65.51	75.70

Table 1: Evaluation results on different tasks. Except for WikiText-2, where perplexity is reported, we report the accuracy metric for the remaining tasks.

our DQT 8 bit models outperform BitNet b1.58 across all other benchmarks. These results highlight that DQT 8 bit models more closely approximate the performance of FP32 models compared to BitNet. Moreover, when ternary inference is applied, the performance slightly decreases compared to 8-bit inference but remains on par with BitNet, demonstrating the robustness of our approach with ternary inference and its flexibility in deployment.

5. Analysis

The use of stochastic rounding in DQT can be viewed as a convergence-guaranteed optimization scheme. This follows from the fact that stochastic rounding introduces zero-mean noise with bounded variance, a property well studied in stochastic optimization theory (Gupta et al., 2015; Hubara et al., 2018; Bottou et al., 2018; Ajalloeian and Stich, 2021). As a result, the updates in DQT maintain convergence guarantees despite operating in a quantized space. For completeness, we provide a proof sketch of this guarantee in Supplementary Material, Section 4. In this section, we primarily focus on the empirical analysis, demonstrating how these theoretical properties manifest in practice.

While stochastic rounding offers advantages in keeping low-precision weights, it may bring some problems. For example, small weight updates may be applied with low probability (e.g., a value like $-1 + 0.02$ has only a 2% chance of being rounded to 0). If such rare updates do occur, we are particularly interested in understanding their impact on the overall training dynamics. In this section, we provide a detailed analysis of the effects and implications of using stochastic rounding.

5.1. The Role of Stochastic Rounding in DQT

Firstly, we provide empirical evidence for the critical role of stochastic rounding in preserving gradient information during training with low-precision weights. Particularly, we compare 130M ternary DQT with a variant that simply uses absmax quantization on the updated weight matrices and maintains the weights in ternary format without stochastic rounding. As shown in the left part of Figure 5, the latter fails to converge, despite operating under the same bit budget. This outcome is expected, as absmax quantization can easily ignore small updates on the weights. In contrast, stochastic rounding not only performs quantization

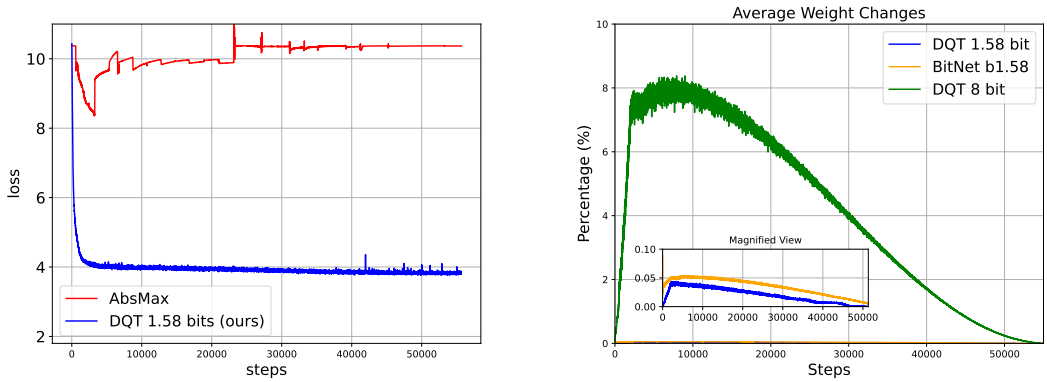

Figure 5: Left: Comparison between DQT 1.58 bits and a variant using absmax quantization for weight updates under the same learning rate. Right: Percentage of updated weights after each training step.

but also facilitates the accumulation of fine-grained updates, enabling effective training even in extremely low-bit regimes.

5.2. Quantifying Weight Update Frequency in DQT

As discussed in Section 5.1, stochastic rounding facilitates training by allowing even small weight updates to take effect. To better understand its impact, we examine how frequently weights are updated during training. Specifically, we analyze 130M-size models to measure the percentage of quantized weights that change after each training step.

We quantify the weight update frequency in the right part of Figure 5 for three variants: DQT 1.58 bit, BitNet b1.58 and DQT 8 bit under the same learning rate and batch size. Note that we show the average percentages of all weight matrices in the model. For BitNet, the peak weight update rate is approximately 0.05% (observed at step 2000, the end of the warm-up phase), meaning only 0.05% of quantized ternary weights change after a single step. Ternary DQT exhibits a similar update rate of around 0.04%, indicating minimal difference between the two in terms of update frequency. In contrast, the 8-bit DQT variant, with weight values ranging from -128 to 127, shows a significantly higher update frequency, reaching up to 8%.

5.3. Impact of Small Weight Updates in DQT Training

Finally, to investigate the impact of small weight updates during training, we rank the absolute value of weight updates at each step and selectively intervene in the lowest 20%. For these bottom 20% updates, we apply one of two interventions: either suppress the update by retaining the original ternary value, or enforce a change by rounding it to a different quantized value, even if the update is small. For example, if an update of $+0.02$ from $-1+0.02$ falls within the smallest 20%, we either keep it at -1 (suppress) or round it to 0 (enforce), depending on the experimental condition. Figure 6 shows the training loss

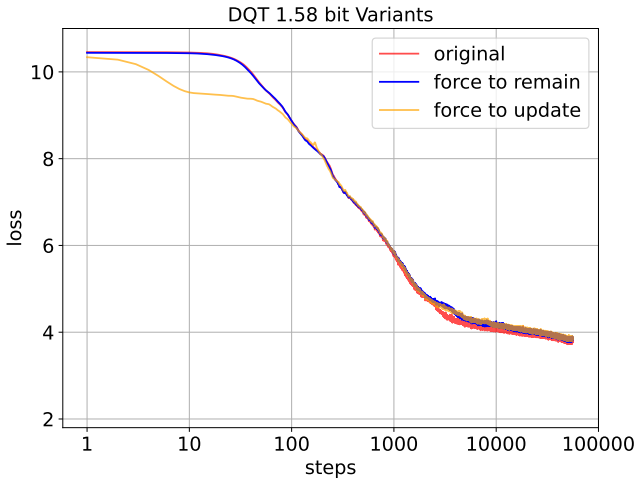

Figure 6: Force to remain refers to the variant that ignores the smallest 20% of weight updates, while force to update enforces changes in this bottom 20%. All variants use the same learning rate.

for these variants, where *force to remain* refers to suppressing the bottom 20% of updates, and *force to update* corresponds to enforcing them to change. From Figure 6, we observe that the original implementation of DQT (1.58-bit) achieves the best performance. Ignoring the smallest 20% of updates has minimal impact, while enforcing these small updates to take effect slightly accelerates convergence. However, the final losses at the end of training are similar across all variants. This may be due to the fact that small updates contribute less to training overall, and even when such updates do occur, stochastic rounding may re-adjust them toward the optimal value in the next training steps.

6. Conclusion

In this work, we explore Direct Quantized Training, a modified QAT method that directly updates low-bit weight matrices without relying on high-precision weights during training. This design enables DQT to operate effectively even under constrained GPU memory settings. Experimental results across different sizes of models demonstrate that DQT enables training without updating on high-precision weights, which are required for straight-through estimation. Moreover, when using 8-bit weights, DQT achieves performance comparable to both FP32 models and BitNet b1.58 during training and inference, demonstrating its effectiveness and practicality for efficient model training.

References

Ahmad Ajalloeian and Sebastian U. Stich. On the convergence of sgd with biased gradients, 2021. URL https://arxiv.org/abs/2008.00051.

Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/c0a62e133894cdce435bcb4a5df1db2d-Paper.pdf.

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

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning, 2018. URL https://arxiv.org/abs/1606.04838.

Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13169–13178, 2020.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. Efficientqat: Efficient quantization-aware training for large language models, 2024. URL https://arxiv.org/abs/2407.11062.

Shangyu Chen, Wenya Wang, and Sinno Jialin Pan. Metaquant: Learning to quantize by learning to penetrate non-differentiable quantization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/f8e59f4b2fe7c5705bf878bbd494ccdf-Paper.pdf.

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

Jiangfei Duan, Shuo Zhang, Zerui Wang, Lijuan Jiang, Wenwen Qu, Qinghao Hu, Guoteng Wang, Qizhen Weng, Hang Yan, Xingcheng Zhang, Xipeng Qiu, Dahua Lin, Yonggang Wen, Xin Jin, Tianwei Zhang, and Peng Sun. Efficient training of large language models on distributed infrastructures: A survey, 2024. URL https://arxiv.org/abs/2407.20018.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training compression for generative pretrained transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1737–1746. JMLR.org, 2015.

Lang Huang, Qiyu Wu, Zhongtao Miao, and Toshihiko Yamasaki. Joint fusion and encoding: Advancing multimodal retrieval from the ground up, 2025. URL https://arxiv.org/abs/2502.20008.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18(187):1–30, 2018. URL http://jmlr.org/papers/v18/16-456.html.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9781713871088.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations*, 2021.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. LLM-QAT: Data-free quantization aware training for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 467–484, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.26. URL https://aclanthology.org/2024.findings-acl.26.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

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

Zhuoyuan Mao, Mengjie Zhao, Qiyu Wu, Hiromi Wakaki, and Yuki Mitsufuji. Deepresonance: Enhancing multimodal music understanding via music-centric multi-way instruction tuning, 2025. URL https://arxiv.org/abs/2502.12623.

Ilia Markov, Adrian Vladu, Qi Guo, and Dan Alistarh. Quantized distributed training of large models with convergence guarantees. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 24020–24044. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/markov23a.html.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843.

Zhongtao Miao, Kaiyan Zhao, and Yoshimasa Tsuruoka. Improving arithmetic reasoning ability of large language models through relation tuples, verification and dynamic feedback, 2024. URL https://arxiv.org/abs/2406.17873.

Zhongtao Miao, Qiyu Wu, Masaaki Nagata, and Yoshimasa Tsuruoka. Improving word alignment using semi-supervised learning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19871–19888, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1020. URL https://aclanthology.org/2025.findings-acl.1020/.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

OpenAI. Introducing chatgpt, 2022. https://openai.com/blog/chatgpt.

OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale, 2024. URL https://arxiv.org/abs/2406.17557.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 525–542, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46493-0.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR, 2018.

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

John Von Neumann and Herman Heine Goldstine. Numerical inverting of matrices of high order. 1947.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. Bitnet: Scaling 1-bit transformers for large language models, 2023. URL https://arxiv.org/abs/2310.11453.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *NUT@EMNLP*, 2017.

Qiyu Wu, Masaaki Nagata, Zhongtao Miao, and Yoshimasa Tsuruoka. Word alignment as preference for machine translation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3223–3239, Miami, Florida, USA, November 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.emnlp-main.188.

Haoran Xu, Young Jin Kim, Amr Sharaf, and Hany Hassan Awadalla. A paradigm shift in machine translation: Boosting translation performance of large language models. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=farT6XXntP.

Yuzhuang Xu, Xu Han, Zonghan Yang, Shuo Wang, Qingfu Zhu, Zhiyuan Liu, Weidong Liu, and Wanxiang Che. Onebit: Towards extremely low-bit large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=ZwiG9KjfHV.

Zhenyu Zhang, Ajay Jaiswal, Lu Yin, Shiwei Liu, Jiawei Zhao, Yuandong Tian, and Zhangyang Wang. Q-galore: Quantized galore with int4 projection and layer-adaptive low-rank gradients, 2024. URL https://arxiv.org/abs/2407.08296.

Kaiyan Zhao, Qiyu Wu, Zhongtao Miao, and Yoshimasa Tsuruoka. Prompt tuning can simply adapt large language models to text encoders. In Vaibhav Adlakha, Alexandra Chronopoulou, Xiang Lorraine Li, Bodhisattwa Prasad Majumder, Freda Shi, and Giorgos Vernikos, editors, *Proceedings of the 10th Workshop on Representation Learning for NLP (RepL4NLP-2025)*, pages 38–50, Albuquerque, NM, May 2025. Association for Computational Linguistics. ISBN 979-8-89176-245-9. doi: 10.18653/v1/2025.repl4nlp-1.3. URL https://aclanthology.org/2025.repl4nlp-1.3/.