# Supplementary Material for "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*

In this supplementary material, we provide additional details to complement the main paper. Specifically, we present:

- Additional implementation details;
- Additional experimentl results including non-logarithmic comparison;
- Explanation of our models with ternary inference;
- A proof sketch for stochastic rounding's convergence guarantee in DQT

## 1. Additional Implementation Details

### 1.1. Model configuration

The detailed configurations for models of different sizes are provided in Table 1. Note that the batch size varies across model sizes during training, but no gradient accumulation is used in any experiment. For each model, the learning rate is selected via grid search over the set {1e-5, 1e-4, 5e-4, 1e-3} using our development set. Regarding the tokenizer, we adopt a publicly released pre-trained one[1] without further updates during training. We fix the random seed to 42 for reproducibility.

### 1.2. Dataset preprocessing

The maximum length of training data for both datasets is set to 512. Texts longer than 512 tokens are split into separate chunks, while shorter texts are padded accordingly. For Wikipedia, this preprocessing results in approximately 14 million sentences derived from the original 6.4 million examples, while for FineWeb, the dataset yields around 33 million sentences. The test set of WikiText-2 is from [2].

1. https://huggingface.co/1bitLLM/bitnet_b1_58-large
2. https://huggingface.co/datasets/Salesforce/wikitext/viewer/wikitext-2-v1/test

| Params | hidden_size | intermediate_size | num_hidden_layers | num_attention_heads | batch_size |
|---|---|---|---|---|---|
| 130M | 768 | 2048 | 12 | 12 | 64 |
| 320M | 1024 | 2048 | 24 | 16 | 32 |
| 1B | 2048 | 3072 | 24 | 32 | 16 |

Table 1: Configuration of different models sizes.

| Model size | FP32 | BF16 | BF16+Adafactor | FP8 | FP8+Adafactor |
|---|---|---|---|---|---|
| 130M | 69327 | 54675 | 53827 | 39276 | 38315 |
| 1B | 76533 | 58345 | 53723 | 40945 | 37669 |

Table 2: Actual GPU memory usage (in MB) of models with different sizes on a single GH200.

### 1.3. Low-precision environments

For FP8 simulation, we choose to use MS-AMP because the GH200 Superchips are equipped with ARM-based CPUs, which are not fully supported by the transformer-engine library currently. As for the reason for using low-precision simulation, hardware and software constraints, such as the inability to modify low-level PyTorch kernels or implement true integer-based weight updates, make it infeasible to perform actual low-bit computation in our environment. These limitations are typical in academic settings, where direct access to low-level hardware accelerators is restricted. As an alternative, we focus on evaluating the practical effectiveness of our method under memory-constrained conditions, simulating the scenarios where computational resources are limited.

### 1.4. Actual Memory Usage

We present the actual memory usage of models with different sizes in Table 2. The values reflect usage on a single GH200 GPU, which has 97,871 MB of available memory. We can also observe exactly how much memory is saved after applying the Adafactor optimizer.

### 1.5. Weight Update Frequency

For BitNet models, we compare quantized weight matrices at adjacent training steps by iterating through all corresponding rows and columns to identify whether each element is updated or not. For DQT models, we can directly compare the weight matrices before and after stochastic rounding. The update percentage is computed as the ratio of changed weight elements to the total number of elements.

## 2. Additional Experimental Results

We provide a more precise and clear comparison in Figure 1 and Figure 2.

## 3. Ternary Inference

Since the straight-through estimator is not employed in our proposed DQT, both the forward and backward operate directly on $n$-bit weight matrices, which means that our DQT models

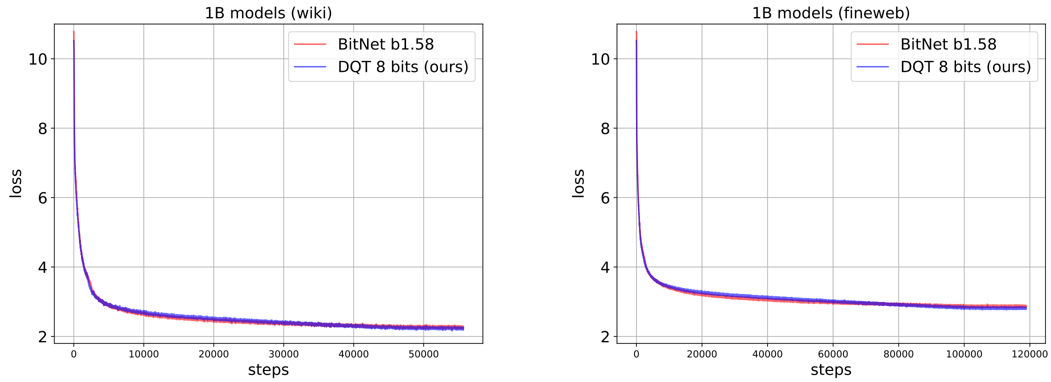


Figure 1: Non-logarithmic training loss comparison of DQT 8 bits and BitNet b1.58 in 1B sizes. Our DQT 8 bits performs slightly better than BitNet b1.58.

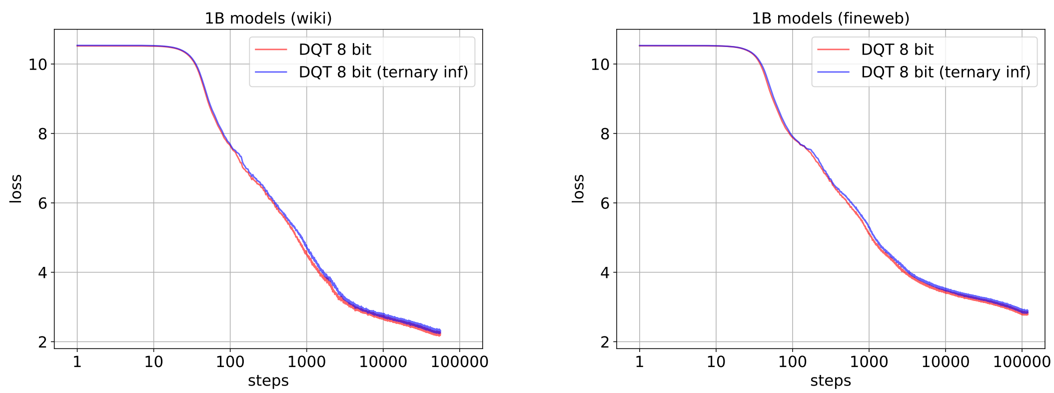


Figure 2: Training loss comparison of DQT 8 bits and DQT 8 bits that utilizes ternary inference. DQT achieves ternary inference with minimal degradation.

trained with larger bit-width are not inherently ternary during inference. To enable a fair comparison with BitNet b1.58, we adapt our models to perform ternary inference. This is achieved by using ternary weights during the forward pass while maintaining $n$-bit weights for the backward pass, updated through the straight-through estimator. The straight-through estimator is employed only to enable ternary inference in $n$-bit DQT models as a variant of our models. We additionally present the training loss for DQT 8 bit and DQT 8 bit with ternary inference in the supplementary material, Figure 2. Note that when ternary inference is applied, the memory usage is the same as BitNet b1.58 during inference.

## 4. A Proof Sketch for Convergence Guarantee in DQT

We consider the simplified optimization problem: $min\ L(\theta)$, where $L(\cdot)$ is a smooth and possibly non-convex loss function and $\theta$ stands for the model parameter.

We use a stochastic gradient method with quantized updates:

$$\theta_{t+1} = SR(\theta - \eta \nabla L(\theta_t)) = \theta_t - \eta \nabla L(\theta_t) + \epsilon_t, \tag{1}$$

where $SR(\cdot)$ denotes the stochastic rounding quantizer (core approach of DQT), $\eta$ is the learning rate and $\epsilon_t$ is the quantization error (noise) which can be represented as:

$$\epsilon_t = SR(\theta - \eta \nabla L(\theta_t)) - (\theta - \eta \nabla L(\theta_t)). \tag{2}$$

It is easy to find that in Equation 1, the latter is in the form of SGD optimization with noise. Let $x = \theta - \eta \nabla L(\theta_t)$, then $\epsilon_t$ can be written into:

$$\epsilon_t = SR(x) - x, \tag{3}$$

where

$$SR(x) = \begin{cases} \lfloor x \rfloor, \text{ with } p = \lceil x \rceil - x \\ \lceil x \rceil, \text{ otherwise} \end{cases}. \tag{4}$$

Now we can calculate the expectation:

$$\mathbb{E}(SR(x)) = (\lceil x \rceil - x) \cdot \lfloor x \rfloor + (x - \lfloor x \rfloor) \cdot \lceil x \rceil = x. \tag{5}$$

Then it would be obvious that

$$\mathbb{E}(\epsilon_t) = \mathbb{E}(SR(x)) - \mathbb{E}(x) = 0. \tag{6}$$

This draws the important conclusion that the noise is **zero-mean and unbiased**.

Next, we continue to calculate the variance of $\epsilon_t$. Assume we quantize $x$ with step size $\Delta$. Define: $q_{low} = \lfloor x/\Delta \rfloor \cdot \Delta$, and $q_{high} = q_{low} + \Delta$. With probability $\alpha$, $SR(x) = q_{high}$ and with probability $1 - \alpha$, $SR(x) = q_{low}$. Then following Equation 3, the noise $\epsilon_t$ can be represented as:

$$\epsilon_t = \begin{cases} q_{low} - x = -\alpha\Delta, \text{ with } p = 1 - \alpha \\ q_{high} - x = (1 - \alpha)\Delta, \text{ with } p = \alpha \end{cases}. \tag{7}$$

The variance of $\epsilon_t$ can be represented using the following equation:

$$\mathbb{V}(\epsilon_t) = (1 - \alpha)(\alpha\Delta)^2 + \alpha[(1 - \alpha)\Delta]^2 = \alpha(1 - \alpha)\Delta^2. \tag{8}$$

Thus, the variance of $\epsilon_t$ is bounded:

$$\mathbb{V}(\epsilon_t) \leq \frac{1}{4}\Delta^2, \tag{9}$$

the maximum is achieved when $\alpha = \frac{1}{2}$.

In conclusion, for an SGD optimization problem with noise, if the noise is **zero-mean and its variance is bounded**, the convergence can be guaranteed.

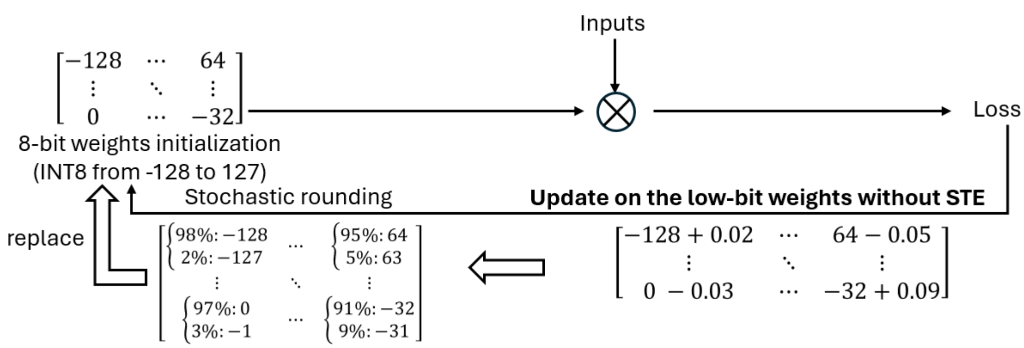


Figure 3: An example of our training process using 8-bit quantization.