

# Quantize Once, Train Fast: Allreduce-Compatible Compression with Provable Guarantees

Jihao Xin<sup>✉</sup>, Marco Canini<sup>✉</sup>, Peter Richtárik<sup>✉</sup> and Samuel Horváth<sup>✉,\*</sup>

KAUST  
MBZUAI

**Abstract.** Distributed training enables large-scale deep learning, but suffers from high communication overhead, especially as models and datasets grow. Gradient compression, particularly quantization, is a promising approach to mitigate this bottleneck. However, existing quantization schemes are often incompatible with Allreduce, the dominant communication primitive in distributed deep learning, and many prior solutions rely on heuristics without theoretical guarantees. We introduce Global-QSGD, an Allreduce-compatible gradient quantization method that leverages global norm scaling to reduce communication overhead while preserving accuracy. Global-QSGD is backed by rigorous theoretical analysis, extending standard unbiased compressor frameworks to establish formal convergence guarantees. Additionally, we develop a performance model to evaluate its impact across different hardware configurations. Extensive experiments on NVLink, PCIe, and large-scale cloud environments show that Global-QSGD accelerates distributed training by up to 3.51× over baseline quantization methods, making it a practical and efficient solution for large-scale deep learning workloads.

## 1 Introduction

Distributed deep learning has become the standard approach for scaling training across multiple compute nodes, enabling faster convergence on large models and datasets [8]. However, as training scales up, communication overhead increasingly dominates total runtime, particularly in large-scale deployments. For example, Sapio et al. [25] reports that communication accounts for more than 90% of the total training time in deep learning workloads, significantly limiting the benefits of additional computing resources.

Gradient quantization has emerged as a practical solution to alleviate this bottleneck, which reduces the gradient bit-width (e.g., from 32-bit to 8-bit), making it communication-efficient. However, existing quantization methods such as QSGD [2] are often impractical in real-world scenarios due to their incompatibility with *Allreduce*—the dominant communication primitive for distributed deep learning [1, 5, 18, 23, 32]. The key issue arises because quantization is performed locally on each worker. For example, when reducing the precision from 32-bit to 8-bit, the gradients must be scaled using a *norm*, which varies between workers. As a result, gradients are quantized at different scales, and directly aggregating these quantized values via Allreduce leads to numerical inconsistencies.

To address this limitation, we introduce **Global-QSGD**. Instead of using local norms for scaling, Global-QSGD leverages the *global*

**Table 1: Comparison of Allreduce-compatible compressors.**

| Algorithm     | Seamless Integration | w/o Extra Step | Rigorous Guarantees | w/o EF | Tunable Compression |
|---------------|----------------------|----------------|---------------------|--------|---------------------|
| PowerSGD [29] | ✗                    | ✗              | ✗                   | ✗      | ✓                   |
| GradiVeQ [34] | ✗                    | ✗              | ✗                   | ✓      | ✗                   |
| IntSGD [21]   | ✓                    | ✓              | ✗                   | ✓      | ✗                   |
| THC [19]      | ✗                    | ✗              | ✗                   | ✗      | ✓                   |
| Global-QSGD   | ✓                    | ✓              | ✓                   | ✓      | ✓                   |

*norm* computed across all workers, ensuring consistent quantization scales and enabling Allreduce compatibility; that is, quantized gradient values can be directly aggregated in their compressed (quantized) representation without the need (and overhead) of switching numeric representation.

Like other lossy compression methods, quantization inevitably loses information. Traditional quantization applies a linear scaling approach, known as **linear dithering**, which is suboptimal because gradients decrease in magnitude as the model converges. Smaller gradients require finer precision to maintain accuracy. To address this, we leverage **exponential dithering**, which has higher precision for smaller gradient values, improving convergence accuracy.

Unlike previous approaches that rely on heuristics, we provide a rigorous convergence analysis (§5, §6) to establish the theoretical foundations. Using its unbiased nature, we prove that Global-QSGD maintains a bounded variance, ensuring stable convergence and making it a theoretically sound choice. To study the impact across different hardware configurations, we derive a performance model (§7). This paper makes the following key contributions:

- We introduce Global-QSGD, the first quantization method that seamlessly integrates with Allreduce.
- We develop exponential dithering with a custom reduction function to effectively overcome the convergence limitations of traditional linear dithering.
- We provide a comprehensive theoretical framework that rigorously establishes Global-QSGD’s convergence guarantees.
- We evaluate Global-QSGD across several settings, including single-node and cloud environments. Our results show that Global-QSGD accelerates model training by up to 3.51×, significantly reducing communication overhead without sacrificing model accuracy. We openly release our code at <https://github.com/sands-lab/global-qsgd>.

## 2 Related Work

Gradient compression methods can be broadly categorized into three approaches: *sparsification* [1, 3, 17, 20, 27], *decomposition* [29], and *quantization* [2, 13, 22, 26, 30]. Although these methods differ in

\* Corresponding Author. Email: samuel.horvath@mbzuai.ac.ae

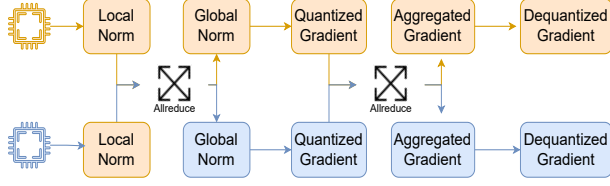


Figure 1: Global-QSGD workflow.

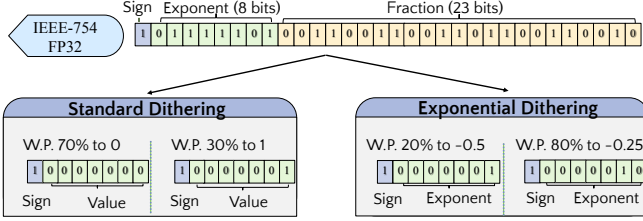


Figure 2: Quantize  $-0.3$  from FP32 to 8-bit.

how they reduce communication overhead, they can also be classified according to their impact on gradient updates, falling into two major theoretical categories: *unbiased* and *biased* compressors. Unbiased compressors, such as quantization schemes like QSGD [2], maintain an expectation-equal gradient estimate with bounded variance, making them easier to analyze in theoretical frameworks [11]. In contrast, biased compressors, including Top- $k$  [3], SignSGD [4], and PowerSGD [29], introduce systematic distortion, requiring correction mechanisms such as error feedback (EF) [15, 24, 27] or induced compressors (IC) [12] to restore convergence guarantees. However, these mechanisms introduce additional memory overhead or rely on an auxiliary unbiased compressor, which limits practicality.

Unfortunately, most of the existing compressors are not natively compatible with Allreduce since the sum of two compressed vectors is not a homomorphic operation; thus, it generally requires an expensive decompress-aggregate-compress operation. This includes greedy and random sparsification,<sup>1</sup> quantization, and sign-based methods.

Several compressors have been proposed to ensure *Allreduce* compatibility, but all rely on some heuristics, so they all lack rigorous theoretical guarantees. GradiVeQ [34] assumes adjacent gradients are linearly correlated, IntSGD [21] requires clipping the communicated integer values, and PowerSGD approximates low-rank decomposition using power iteration while depending on memory-intensive error feedback. THC [19], developed in parallel to our work, also utilizes global norm. The Uniform THC variant is similar to linear dithering in our work. However, THC focuses on system efficiency to heuristically select the quantization scheme. We focus on convergence properties with exponential dithering and provide theoretical guarantees. Table 1 compares our work with the above methods and summarizes the key differences.

### 3 Global-QSGD Overview

Global-QSGD is a quantization operator [2, 13], where the main innovation is that we propose to normalize by the global norm across workers instead of each worker using its local norm. Figure 1 illustrates an example workflow of the Global-QSGD’s two-round Allreduce with 2 workers. Algorithm 1 formalizes the high-level process of Global-QSGD, which can be categorized into 3 main steps:

**Step 1: Obtain global norm (line 1)**, where each worker computes its local norm and then performs an Allreduce operation (max as the

<sup>1</sup> For random sparsification, one can share random seeds across workers to make it Allreduce compatible, e.g., see synchronized random seed [31].

#### Algorithm 1 Global-QSGD

**Input:** Worker count:  $n$ , Gradients:  $\mathbf{x} = [x_1, x_2, \dots, x_n]$ ,  
 $\text{sparse} \in \{\text{True}, \text{False}\}$   
1:  $\|\mathbf{x}\|_{q,p} = \text{Allreduce}(\text{MAX}\{\|x_i\|_{q,p}\})$   
2: **for**  $i \in \{1, 2, \dots, n\}$  **do in parallel**  
3:  $y_i := |x_i| / \|\mathbf{x}\|_{q,p}$   
4:  $z_i := \text{sign}(x_i) \circ \xi_i(y_i)$   
5: **end for**  
6: **if**  $\text{sparse}$  **then**  
7: **return**  $\frac{1}{n} \text{Allgather}(\text{SUM}\{z_i\})$   
8: **end if**  
9: **return**  $\frac{1}{n} \text{Allreduce}(\text{SUM}\{z_i\})$

reduction operator) to obtain the global norm. This ensures that all workers quantize their gradients consistently, preventing local scaling discrepancies. As a one-element Allreduce, the overhead of this step is negligible, and can be hidden by overlapping this step with the gradient communication of a preceding communication round in a sequence of Allreduce invocations.

**Step 2: Quantization (line 2-5)**, where each worker normalizes every gradient element to  $[0, 1]$  while preserving the sign bit, and maps each element to the nearest quantization level  $l_i$  for  $i \in \{0, 1, \dots, s\}$  with stochastic rounding. Figure 2 illustrates an example of how the value  $-0.3$  represented in IEEE FP32 format is converted to an 8-bit representation. §4 details the quantization schemes.

**Step 3: Aggregation (line 9)**, where the workers perform an Allreduce operation, applying the reduction operator directly on the quantized values without requiring changes of numerical representation.

**Optional: Sparsity handling (lines 6-8).** Additionally, we handle sparse gradients differently by only transmitting non-zero elements. Instead of sending full quantized vectors, we use Allgather to efficiently communicate the non-zero values.

## 4 Quantization Schemes

### 4.1 Formulation

To establish a solid theoretical foundation, we first present the mathematical formulation of the Global-QSGD quantization operator, which also serves as the foundation for our subsequent variance analysis (§ 5) and convergence guarantees (§ 6).

We begin with the notations. For a vector  $x_i \in \mathbb{R}^d$ , we define the  $\ell_p$ -norm as  $\|x_i\|_p := (\sum_{j=1}^d |x_{ij}|^p)^{1/p}$  for  $p \in (1, \infty)$ . When  $p = \infty$ ,  $\|x_i\|_\infty$  represents the maximum absolute value among all elements of  $x_i$ . Let  $\mathbf{x} = [x_1, x_2, \dots, x_n] \in \mathbb{R}^{nd}$  be a concatenation of vectors  $x_1, x_2, \dots, x_n \in \mathbb{R}^d$ . We define the  $(q, p)$ -mixed norm in  $\mathbb{R}^{nd}$  as  $\|\mathbf{x}\|_{q,p} := (\sum_{i=1}^n \|x_i\|_q^p)^{1/p}$ . For any vectors  $x_i, x_j$ ,  $x_i \circ x_j$  represents their element-wise multiplication, and, for any vector  $x_i$ ,  $|x_i|$ ,  $\text{sign}(x_i)$  stand for element-wise absolute value and signum operations, respectively. We denote  $[n] := \{1, 2, \dots, n\}$  for any  $n \in \mathbb{N}$ . We summarize the notation in Table 2.

We define the Global-QSGD quantization operator:

**Definition 1** ( $\mathcal{Q}_s^{q,p}$ ). *The Global-QSGD quantization operator with respect to the  $(q, p)$ -mixed norm and with  $s$  levels*

$$0 = l_s < l_{s-1} < l_{s-2} < \dots < l_1 < l_0 = 1,$$

denoted  $\mathcal{Q}_s^{q,p}$ , is defined as follows. Let  $\mathbf{x} = [x_1, x_2, \dots, x_n] \in \mathbb{R}^{nd}$ . Let  $y_i := |x_i| / \|\mathbf{x}\|_{q,p} \in \mathbb{R}^d$  for all  $i \in [1, \dots, n]$ . Then

$$\mathcal{Q}_s^{q,p}(\mathbf{x}) := \|\mathbf{x}\|_{q,p} \frac{1}{n} \sum_{i=1}^n \text{sign}(x_i) \circ \xi_i(y_i), \quad (1)$$

**Table 2: Summary of key notation.**

| Symbol                           | Description                                 |
|----------------------------------|---------------------------------------------|
| $d$                              | Dimension of gradients                      |
| $n$                              | Number of workers                           |
| $s$                              | Number of quantization levels               |
| $l_i$                            | Value of the $i_{th}$ level                 |
| $x_i \in \mathbb{R}^d$           | Gradients of the $i_{th}$ worker            |
| $y_i \in \mathbb{R}^d$           | Normalized gradients of the $i_{th}$ worker |
| $\mathbf{x} \in \mathbb{R}^{nd}$ | Gradients of all $n$ workers                |
| $\mathbf{y} \in \mathbb{R}^{nd}$ | Normalized gradients of all $n$ workers     |
| $\ x_i\ _p$                      | $\ell_p$ norm of $x_i$                      |
| $\ \mathbf{x}\ _{q,p}$           | $(q, p)$ -mixed norm of $\mathbf{x}$        |
| $\xi_i(y_i)$                     | Random rounding of $y_i$                    |
| $\mathcal{Q}_s^{q,p}$            | Global-QSGD compressor in general           |
| $\mathcal{L}_s^{q,p}$            | Global-QSGD with linear dithering           |
| $\mathcal{E}_s^{q,p}$            | Global-QSGD with exponential dithering      |

where  $\xi_i(y_i)$  is an independent element-wise random rounding operator such that

$$(\xi_i(y_i))_j := \begin{cases} l_{u_i^j} & \text{with probability } \frac{(y_i)_j - l_{u_i^j} + 1}{l_{u_i^j} - l_{u_i^j + 1}} \\ l_{u_i^j + 1} & \text{otherwise} \end{cases}, \quad (2)$$

for  $j \in [d]$ , where  $u_i^j \in \{0, 1, 2, \dots, s\}$  and  $l_{u_i^j + 1} \leq (y_i)_j \leq l_{u_i^j}$ .

## 4.2 Linear and Exponential Dithering

We consider two approaches to partitioning the quantization levels  $l_i$  in Global-QSGD:

- **Linear Dithering** ( $\mathcal{L}_s^{q,p}$ ): The straightforward way is to divide the interval  $[0, 1]$  into equal partitions as  $l_i = s^{-i}/s$ . However, as gradients converge to zero during training, this method becomes less accurate since most values fall into the same levels.
- **Exponential Dithering** ( $\mathcal{E}_s^{q,p}$ ): To provide higher precision for near-zero values, we propose a nonuniform interval partitioning approach. Inspired by Horváth et al. [13], we divide intervals exponentially as  $l_s = 0$  and  $l_i = 1/2^{s-i}$ . We use base 2 for natural compatibility with IEEE standard floating point representation.

Supporting 255 levels, we adopt 8-bit as the default quantization bit-width, since most hardware accelerators natively support this numerical representation. For bit-widths without direct hardware support, a manual implementation of bit-wise operations is necessary. To demonstrate that Global-QSGD is theoretically compatible with any bit precision, we also implement the 4-bit version on the NVIDIA Ampere architecture; results are presented in § 9.

The implementation of  $\mathcal{L}_s^{q,p}$  is simple and inherently compatible with Allreduce, since mapping to uniform intervals is a homomorphic operation, allowing efficient aggregation by summing the quantized integers values directly. However, for  $\mathcal{E}_s^{q,p}$ , summation via Allreduce becomes more complex, as the quantized values follow an exponential form  $2^k$ . To ensure compatibility with Allreduce, we introduce stochastic unbiased exponential rounding, denoted as  $\mathcal{C}_{\text{nat}}$ , following the notation in Horváth et al. [13]. This rounding scheme introduces a minor variance increase per step, specifically a factor of  $9/8$ . Over multiple aggregation steps, this accounts for  $(9/8)^{\# \text{aggregation steps}}$ , i.e.,  $(9/8)^{\log(n)} \leq n^{0.17}$  for Tree-Allreduce, resulting in a small increase in variance—for example, only 1.6× for  $n = 16$  and 3.25× for  $n = 1024$ .

Finally, we discuss a strong advantage of  $\mathcal{E}_s^{q,p}$ 's scaling properties with the number of nodes  $n$ , due to better integer overflow resistance. Let us look at the following example, where integer values

## Algorithm 2 Reduce Function for $\mathcal{E}_s^{q,p}$

**Input:**  $(\text{sign}_1, e_1), (\text{sign}_2, e_2), s$

- 1:  $k = -\lfloor \log(2^{-m} + [p - 2^{-m}]_+) \rfloor$
- 2:  $\text{gz}_1 = \mathbf{1}(e_1 > 0)$  {whether  $e_1$  is nonzero}
- 3:  $\text{gz}_2 = \mathbf{1}(e_2 > 0)$  {whether  $e_2$  is nonzero}
- 4:  $\text{sign}_{12} = \text{sign}_1 \cdot \text{sign}_2 \cdot \text{gz}_1 \cdot \text{gz}_2$
- 5:  $\text{diff} = |e_1 - e_2| - (1 - \text{sign}_{12})/2$
- 6:  $\text{smaller} = (\mathbf{1}(e_1 \leq e_2) + \text{gz}_2) \cdot \text{gz}_1$
- 7:  $\text{non\_zero} = 1 - \mathbf{1}(e_1 = e_2 \wedge \text{sign}_{12} = -1)$
- 8:  $\text{sign}_{\text{result}} = \text{sign}_1 \cdot \text{smaller} + \text{sign}_2 \cdot (1 - \text{smaller})$
- 9:  $e_{\text{result}} = (e_1 \cdot \text{smaller} + e_2 \cdot (1 - \text{smaller})) \cdot \text{non\_zero} - \text{sign}_{12} \mathbf{1}(k > \text{diff}) \cdot \text{non\_zero}$
- 10: **return**  $\text{sign}_{\text{result}}, e_{\text{result}}$

are represented with  $A$  bits ( $A \in \mathbb{N}$ ). In the case of linear dithering, the maximum aggregated value is  $ns$ . Besides, one bit is needed to hold the sign. Therefore, we require  $1 + \log(s + 1) + \log(n) \leq A$ , which cannot be satisfied for any  $s$ , for example in the case with  $n = 16$  and  $A = 4$ . On the other hand, the maximum aggregated value with exponential dithering is  $n2^s$ ; but since we only communicate the exponent, we obtain an improved scaling, as the maximum integer communicated is  $s + \log(n)$ . Therefore, we only require  $1 + \log(s + 1 + \log(n)) \leq A$ , which is satisfied for  $s \leq 3$ . This is because exponential dithering scales as  $\log(\log(n))$  with  $n$  instead of just  $\log(n)$ .

## 4.3 Reduce Function for Exponential Dithering

To efficiently implement  $\mathcal{E}_s^{q,p}$ , we introduce the *exponential reduce* function (Algorithm 2), which aggregates values using integer-based arithmetic. This GPU-friendly integer-based and branch-free algorithm ensures efficient parallelization with CUDA.

Since our reduction function acts element-wise, we only consider here a one-dimensional case. Before quantization and aggregation, we divide all values by  $2n$ , where  $n$  is the number of nodes, to ensure that the maximum power of two we encounter during the aggregation is  $-1$  that corresponds to  $1/2 = 2^{-1}$ . Therefore, all the encountered exponents during the aggregation are guaranteed to be negative. In this way, we can represent zero as 0 instead of  $2^0$ .

Next, we define the representation used during the aggregation by our exponential reduce function, which assumes exponential dithering is locally executed prior to its invocation. Let  $\text{sign} \in \{-1, +1\}^2$  be the sign and  $e \in \{\mathbb{N} \cup \{0\}\}$  be the communicated nonnegative integer-valued exponents. Then, then real number  $x$  that corresponds to the pair  $(\text{sign}, e)$  is defined as

$$x = \begin{cases} 0, & \text{if } e = 0, \\ \text{sign } 2^{-e}, & \text{otherwise.} \end{cases} \quad (3)$$

We proceed with the derivation for the reduce function. Let  $x_1, x_2 \in \mathbb{R}$  represented by  $(\text{sign}_1, e_1), (\text{sign}_2, e_2)$ , respectively, be the values to be summed using the reduce function. To facilitate efficient rounding, define the maximum difference  $|e_1 - e_2|$  that can appear during aggregation as

$$k = -\lfloor \log(2^{-m} + [p - 2^{-m}]_+) \rfloor,$$

<sup>2</sup> Zeros have any sign.

where  $p \sim \text{Unif}[0, 1]$  is a sample of the uniform distribution in the interval  $[0, 1]$ ,<sup>3</sup>  $m := s + 1$ , and  $[x]_+ := \max\{0, x\}$ . It holds

$$\Pr(k > b) = \Pr(-k < -b) = \Pr(p < 2^{-b}) = 2^{-b}.$$

Note that the support set for  $k$  is  $\{0, 1, \dots, m\}$ , and for  $b \in \{0, 1, \dots, m-1\}$ . Without loss of generality, we assume that  $\text{sign}_1 = 1$  and  $0 < e_1 \leq e_2$ . We later discuss the case with  $e_1 = 0$  or  $e_2 = 0$ . If  $\text{sign}_2 = 1$ , then

$$C_{\text{nat}}(2^{-e_1} + 2^{-e_2}) = \begin{cases} 2^{-e_1+1}, & \text{w.p. } 2^{e_1-e_2}, \\ 2^{-e_1}, & \text{w.p. } 1 - 2^{e_1-e_2}, \end{cases}$$

where w.p. stands for “with probability.” We note that

$$p < 2^{e_1-e_2} \iff k > e_2 - e_1.$$

Therefore,

$$C_{\text{nat}}(2^{-e_1} + 2^{-e_2}) = \begin{cases} 2^{-e_1+1}, & \text{if } k > e_2 - e_1, \\ 2^{-e_1}, & \text{otherwise.} \end{cases}$$

Analogously, when  $\text{sign}_2 = -1$ , we write  $C_{\text{nat}}(2^{-e_1} - 2^{-e_2})$  as:

$$\begin{cases} 0, & \text{if } e_1 = e_2 \\ \begin{cases} 2^{-e_1-1}, & \text{w.p. } 2^{e_2-e_1-1}, \\ 2^{-e_1}, & \text{w.p. } 1 - 2^{e_2-e_1-1}, \end{cases} & \text{otherwise.} \end{cases}$$

Equivalently, we can write  $C_{\text{nat}}(2^{-e_1} - 2^{-e_2})$  as:

$$\begin{cases} 0, & \text{if } e_1 = e_2 \\ \begin{cases} 2^{-e_1-1}, & \text{if } k > e_2 - e_1 - 1, \\ 2^{-e_1}, & \text{if } k \leq e_2 - e_1 - 1, \end{cases} & \text{otherwise.} \end{cases}$$

In general, we compare  $k$  to the following quantity

$$\text{diff} = |e_1 - e_2| - (1 - \text{sign}_{12}) // 2,$$

where  $\text{sign}_{12} := \text{sign}_1 \text{sign}_2 \in \{-1, 1\}$  and  $//$  corresponds to the integer division. It is easy to check that the above quantity recovers all the above-mentioned cases. To determine which exponent is smaller, we use  $\text{smaller} = \mathbf{1}(e_1 \leq e_2) \in \{0, 1\}$ , where  $\mathbf{1}$  is the indicator function. Finally, to filter out the case of different signs and the same exponent, we use  $\text{non\_zero} := 1 - \mathbf{1}(e_1 = e_2 \text{ and } \text{sign}_{12} = -1) \in \{0, 1\}$ . Then, we obtain the resulting sign and exponent as

$$\begin{aligned} \text{sign}_{\text{result}} &= \text{sign}_1 \cdot \text{smaller} + \text{sign}_2 \cdot (1 - \text{smaller}) \\ e_{\text{result}} &= (e_1 \cdot \text{smaller} + e_2 \cdot (1 - \text{smaller})) \cdot \text{non\_zero} \\ &\quad - \text{sign}_{12} \cdot \mathbf{1}(k > \text{diff}) \cdot \text{non\_zero}. \end{aligned}$$

To incorporate zeros, we first define the following variables to identify if an exponent is greater than zero

$$\begin{aligned} \text{gz}_1 &= \mathbf{1}(e_1 > 0) \\ \text{gz}_2 &= \mathbf{1}(e_2 > 0). \end{aligned}$$

In the case where at least one of the exponents is zero, we do not change the exponents by adding or subtracting zero. This can be achieved by redefining  $\text{sign}_{12} \in \{-1, 0, 1\}$  to

$$\text{sign}_{12} := \text{sign}_1 \cdot \text{sign}_2 \cdot \text{gz}_1 \cdot \text{gz}_2.$$

<sup>3</sup> Note this can be precomputed.

Furthermore, we need to redefine smaller to account for zeros. This can be achieved by

$$\text{smaller} := (\mathbf{1}(e_1 \leq e_2) + !\text{gz}_2) \cdot \text{gz}_1,$$

where  $!$  denotes logical negation. This concludes our construction of the reduce function for the exponential dithering.

## 5 Variance Analysis

We provide the variance analysis of Global-QSGD, which is the key ingredient to ensure convergence in § 6. The theoretical framework extends from [6, 14, 16, 27]. In § 5.1, we extend the concept of *Unbiased Compressors* ( $\mathcal{U}^d(\omega)$ ) to a broader class, *Unbiased Distributed Mean Compressors* ( $\mathcal{U}^{n,d}(\theta)$ ). We then prove that  $\mathcal{Q}_s^{q,p} \in \mathcal{U}^{n,d}(\theta)$ , establishing its unbiasedness. Building upon this property, in § 5.2 we demonstrate that  $\mathcal{Q}_s^{q,p}$  has bounded variance, a crucial condition to ensure convergence. The proofs of all the theorems and lemmas are in [33][Appendix A].

### 5.1 Unbiasedness

We start by defining the Unbiased Compressor as  $\mathcal{U}^d(\omega)$ :

**Definition 2** (Unbiased Compressor). *A randomized mapping  $\mathcal{C}: \mathbb{R}^d \rightarrow \mathbb{R}^d$  is an unbiased compressor if there exists  $\omega \geq 0$  such that  $\forall x \in \mathbb{R}^d$ :*

$$\mathbb{E}[\mathcal{C}(x)] = x, \quad \mathbb{E}[\|\mathcal{C}(x) - x\|_2^2] \leq \omega \|x\|_2^2. \quad (4)$$

If this holds, for simplicity we will write  $\mathcal{C} \in \mathcal{U}^d(\omega)$ .

We now generalize this notion to distributed settings as follows:

**Definition 3** (Unbiased Distributed Mean Compressor). *For any  $x_1, x_2, \dots, x_n \in \mathbb{R}^d$ , let*

$$\mathbf{x} := [x_1, x_2, \dots, x_n] \in \mathbb{R}^{nd}, \quad \mathbf{x} := \frac{1}{n} \sum_{i=1}^n x_i. \quad (5)$$

*A randomized mapping  $\mathcal{G}: \mathbb{R}^{nd} \rightarrow \mathbb{R}^d$  is an unbiased distributed mean compressor if there exists  $\theta \geq 0$  such that  $\forall \mathbf{x} \in \mathbb{R}^{nd}$ :*

$$\mathbb{E}[\mathcal{G}(\mathbf{x})] = \mathbf{x}, \quad \mathbb{E}[\|\mathcal{G}(\mathbf{x}) - \mathbf{x}\|_2^2] \leq \frac{\theta}{n} \|\mathbf{x}\|_{2,2}^2. \quad (6)$$

If this holds, for simplicity we will write  $\mathcal{G} \in \mathcal{U}^{n,d}(\theta)$ .

To show that Definition 3 is more general than Definition 2, we formalize the following lemma:

**Lemma 1** ( $\mathcal{C} \subset \mathcal{U}^{n,d}$ ). *If  $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_n \in \mathcal{U}^n(\omega)$  and they are independent, then  $\mathcal{G}: \mathbb{R}^{nd} \rightarrow \mathbb{R}^d$  defined as Equation 7 and belongs to  $\mathcal{U}^{n,d}(\omega/n)$ .*

$$\mathcal{G}(\mathbf{x}) := \frac{1}{n} \sum_{i=1}^n \mathcal{C}_i(x_i). \quad (7)$$

In the next lemma, we show that  $\mathcal{Q}_s^{q,p} \in \mathcal{U}^{n,d}(\theta)$  has an interesting reduction property that helps us to analyze its theoretical properties using known results for the case  $n = 1$  [2, 13].

**Lemma 2.** *Let  $\mathcal{Q}_s^{q,p}(\mathbf{x}) := \|\mathbf{x}\|_{q,p} \text{sign}(\mathbf{x}) \circ \xi(\mathbf{y})$ , where  $\xi(\mathbf{y}) = [\xi_1(y_1), \xi_2(y_2), \dots, \xi_n(y_n)]$  and  $\xi_i(y_i)$  is defined in (2). Then,*

$$\mathcal{Q}_s^{q,p}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n (\mathcal{Q}_s^{q,p}(\mathbf{x}))_i, \quad (8)$$

where  $(\mathcal{Q}_s^{q,p}(\mathbf{x}))_i$  refers to coordinates  $[(i-1)d+1, \dots, id]$ . Moreover, if  $\mathcal{Q}_s^{q,p} \in \mathcal{C}(\omega)$  then  $\mathcal{Q}_s^{q,p} \in \mathcal{U}^{n,d}(\theta)$  with  $\theta = \omega/n$ .

Note that there is a difference in the dependence on the dimension, i.e.,  $d \rightarrow nd$ , since we work with the concatenated vector  $\mathbf{x}$ .

## 5.2 Variance Bound

Next, we derive the exact bound on the variance of  $\mathcal{L}_s^{q,p}$  and  $\mathcal{E}_s^{q,p}$ . In addition, for the special case of  $p = q = 2$ , we establish an upper bound on sparsity, i.e., the sum of zero norms of the communicated vectors ( $\|y\|_0$  denotes the number of non-zero elements of  $y$ ).

**Theorem 3.** *If  $p, q \geq 2$  then  $\mathcal{L}_s^{q,p} \in \mathcal{U}^{n,d}\left(\frac{\sqrt{d}}{\sqrt{ns}}\right)$  for  $s \leq \sqrt{nd}$ , and  $\mathcal{E}_s^{q,p} \in \mathcal{U}^{n,d}\left(\frac{1}{sn} + \frac{\sqrt{d}}{\sqrt{n2^{s-1}}}\right)$  for  $s \leq 1 + \log(\sqrt{nd})$ . Moreover, if  $p = q = 2$  then for any  $\mathbf{x} \in \mathbb{R}^{nd}$*

$$\sum_{i=1}^n \|\mathcal{L}_s^{2,2}(\mathbf{x})_i\|_0 \leq s^2 + \sqrt{nd},$$

$$\sum_{i=1}^n \|\mathcal{E}_s^{2,2}(\mathbf{x})_i\|_0 \leq 2^{2s-2} + \sqrt{nd}.$$

The above theorem guarantees that we can achieve  $\mathcal{O}(\sqrt{nd})$  compression ratio for  $s = \mathcal{O}(1)$ . Furthermore, we note that the variance bound scales better for exponential dithering with the number of levels  $s$ , which means that exponential dithering exhibits a smaller relative compression error for larger  $s$ . Using these definitions, standard convergence analysis (e.g., [11]) extends naturally to our distributed setting. In particular, employing an unbiased distributed mean compressor  $\mathcal{Q} \in \mathcal{U}^{n,d}(\theta)$  increases the complexity of the iteration by a factor of  $1 + \theta n$  (recovering  $1 + \omega$  for standard compressors).

## 6 Convergence Analysis

We show how the standard analysis for unbiased compressors can be generalized to the proposed unbiased distributed mean compressor. Since we construct compressors to be unbiased, the only extra challenge of the analysis is to bound the variance of the gradient estimator.<sup>4</sup>

**Lemma 4** (Variance bound). *Let  $x \in \mathbb{R}^d$  be deterministic and  $\mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\nabla f_i(x, \xi_i)] = \nabla f_i(x)$  for all  $i \in [n]$ . Then*

$$\mathbb{E} [\|\mathcal{Q}([\nabla f_1(x, \xi_1), \dots, \nabla f_n(x, \xi_n)]) - \nabla f(x)\|_2^2] \leq \frac{\rho}{n} \sum_{i=1}^n \|\nabla f_i(x)\|_2^2 + \left(\rho + \frac{1}{n}\right) \frac{1}{n} \sum_{i=1}^n \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi_i) - \nabla f_i(x)\|_2^2] \quad (9)$$

holds with  $\rho = \frac{\omega}{n}$  for  $\mathcal{Q}([\nabla f_1(x, \xi_1), \dots, \nabla f_n(x, \xi_n)]) := \frac{1}{n} \sum_{i=1}^n \mathcal{C}_i(\nabla f_i(x, \xi_i))$ , where  $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_n \in \mathcal{U}^n(\omega)$  are independent. Furthermore, if  $\mathcal{Q} \in \mathcal{U}^{n,d}(\theta)$  then (9) holds with  $\rho = \theta$ .

In Lemma 4, we show that applying the unbiased distributed mean compressor  $\mathcal{Q} \in \mathcal{U}^{n,d}(\theta)$  yields the same bound on variance<sup>5</sup> as the difference that  $\omega/n$  is replaced with  $\theta$ . The rest of the analysis is the same as the general analysis of Gorbunov et al. [11], which shows that the worst-case increase in the number of iterations to achieve the same precision as the algorithm without compression is by the

<sup>4</sup> The same applies to the analysis of non-smooth functions, where gradients are replaced with subgradients.

<sup>5</sup> We assume that the noise due to sampling and compression are independent.

factor  $1 + \omega$  that translates to  $1 + \theta n$  for the unbiased distributed mean compressors  $\mathcal{Q} \in \mathcal{U}^{n,d}(\theta)$  [11, Theorem 4.1 for Alg. 1: SGD and Alg. 13: Quantized-SGD]. For exponential dithering, this would be a factor of  $1 + n^{1.17} \left( \frac{1}{sn} + \sqrt{d}/\sqrt{n2^{s-1}} \right)$ . In this estimate, we consider all the sources of variance for Allreduce, i.e., variance due to initial quantization and extra variance due to stochastic rounding during Allreduce.

For example, let us consider the setup where we use exponential dithering to decrease communication precision from 32 to 8 bits, the dimensionality of the model is  $d = 10^6$ , and the number of machines is  $n = 16$ . We have to reserve 1 bit for the sign and the other 7 bits for levels. Plugging these numbers into our formula yields the worst-case increment in the expected number of iterations, which is 20%, while we save 75% of communication. Therefore, if communication is a significant bottleneck, our quantization leads to a guaranteed speed-up. Note that this is only the worst case, and in real-world scenarios, this can be much less, e.g., the empirical compression error in our experiments is below 0.5% for exponential dithering.

Furthermore, while our analysis primarily focuses on SGD, the concept of global quantization is intentionally designed to be broadly compatible with various optimizers, including popular ones such as Nesterov momentum and Adam. This compatibility stems from its core property of being unbiased with bounded variance, aligning it with the theoretical foundations underlying these optimizers. Typically, optimizers such as SGD are predicated on the use of stochastic gradients that are unbiased and maintain variance within specific limits—conditions supported by our method. For example, integration with adaptive methods can be directly obtained by generalizing the results of Défossez et al. [9]. Our experiments with Transformer-XL using Adam confirm this compatibility.

In summary, global quantization is compatible with standard optimization methods designed for efficient distributed learning, such as distributed variants of Adam and SGD. By maintaining unbiased gradient estimates with bounded variance, our quantization method preserves convergence guarantees comparable to those of uncompressed methods while significantly reducing communication costs.

## 7 Performance Model

Gradient compression can only be beneficial if the introduced computation overhead can be compensated by communication gains. Thus, we propose a performance model to analyze how Global-QSGD can speed up training.

In practice, there are two popular Allreduce implementations: Ring-based and Tree-based. We adopt Tree-based Allreduce for our evaluation, as it has fewer reduction steps in our 4-GPU setup, where each quantized reduction introduces computation overhead and accuracy loss due to random rounding. The tree Allreduce algorithm first recursively aggregates the gradients, then does a recursive-doubling Allgather. The depth of the tree is  $2 \log(n)$  where  $n$  is the number of workers. The performance of Allreduce is well studied [28] and it is commonly modeled as  $2 \log(n) \alpha + 2 \frac{\log(n)S}{\beta} + \frac{\log(n)S}{\gamma}$ , where  $\alpha$  is the propagation delay in seconds,  $S$  is the size of gradients in bytes,  $\beta$  is the bandwidth (byte/s), and  $\gamma$  is the computation speed (byte/s). The first term represents the propagation delay, the second term is the bidirectional transmission delay, and the last term is the computation cost.

We denote the quantized gradient size as  $\hat{S}$ , and the computation cost with custom reduction as  $\hat{\gamma}$ . The bandwidth and propagation delay remain the same. Denote the quantization and dequantization time as  $\delta$  factor of the gradient size. Then the Allreduce performance

for Global-QSGD is  $2 \log(n)\alpha + 2 \frac{\log(n)\hat{S}}{\beta} + \frac{\log(n)\hat{S}}{\hat{\gamma}} + \delta S$ .

We denote the quantization ratio  $\rho = \frac{S}{\hat{S}}$  (by default  $\rho = 4$ ). We consider the computation cost negligible ( $\delta = 0$ ), as the quantization and dequantization operations are executed only once per Allreduce invocation, which ideally can be overlapped with the communication if the model is communication-bottlenecked. Therefore, the cost of the reduction operation will dominate the performance as the number of workers increases. We denote the computation overhead as  $\omega = \frac{\gamma}{\hat{\gamma}}$ . We empirically measure<sup>6</sup> this overhead to be  $\omega_{\mathcal{L}} = 1$  and  $\omega_{\mathcal{E}} = 79$ , for linear and exponential dithering, respectively. The reduction operation of linear dithering is the native arithmetic summation, which has a similar time for uint8 and float32 data types in our experiments. Instead, the custom reduction used for exponential dithering involves more arithmetic operations, which yield a higher computation overhead.

The condition of performance gain is that per-batch training time after applying Global-QSGD is less than the per-batch training time without quantization, which is:

$$2 \log(N)\alpha + 2 \frac{\log(N)S}{\beta} + \frac{\log(N)S}{\gamma} > 2 \log(N)\alpha + 2 \frac{\log(N)\hat{S}}{\beta} + \frac{\log(N)\hat{S}}{\hat{\gamma}} + \delta S$$

Thus, the condition for Global-QSGD to speed up training is as follows (details of the derivation are in [33][Appendix A.5]):

$$\begin{cases} \beta > \frac{6\gamma}{(\omega-4)}, & \text{if } (\omega < 4), \\ \beta < \frac{6\gamma}{(\omega-4)}, & \text{if } (\omega > 4). \end{cases} \quad (10)$$

With  $\omega_{\mathcal{L}} = 1$ , theoretically linear dithering is guaranteed to speed up training (since  $\beta > 0$ ). In the case of exponential dithering ( $\omega_{\mathcal{E}} = 79$ ), there is a training speed up if the relation  $\beta < 0.08\gamma$  holds. Our evaluation is conducted using A100 GPU ( $\gamma = 2$  TB/s) with P2P ( $\beta = 53.9$  GB/s) or SHM ( $\beta = 5.4$  GB/s). Therefore, both the P2P- and SHM-based communication satisfy the speedup condition.

## 8 Implementation

Our implementation of Global-QSGD supports both linear ( $\mathcal{L}_s^{q,p}$ ) and exponential ( $\mathcal{E}_s^{q,p}$ ) dithering methods, with default configuration using the  $\ell_{inf}$ -norm ( $p = q = \infty$ ) and 8-bit precision ( $s = 255$ ). For ultra-low precision scenarios (4-bit or lower), our implementation can be seamlessly extended without modification when hardware support exists (such as on Hopper GPUs with native 4-bit operations).

In the absence of native hardware support, a custom bit-wise operation is required. We generally encourage users to use the bit-width supported by the hardware, i.e., 8-bit on Ampere and 4-bit on Hopper. We also implemented a 4-bit version  $\mathcal{L}_s^{q,p}$  on Ampere GPUs,<sup>7</sup> where we manually pack 4-bit values into 8-bit. Note that this implementation is used to demonstrate Global-QSGD’s effectiveness in convergence on low-bit. As this is not hardware- nor CUDA-optimized, its performance is not representative.

As discussed in § 4.2, linear dithering ( $\mathcal{L}_s^{q,p}$ ) encounters significant limitations in low-bit large-scale scenarios. For example, with 4-bit quantization, after allocating 1 bit for the sign and 2 bits to handle potential overflow across 4 workers, only a single bit remains for actual gradient values. This severely restricts precision and cannot

**Table 3: Summary of benchmarks used in this work.**

| Model             | Dataset      | Parameter Size | Training Epochs |
|-------------------|--------------|----------------|-----------------|
| DeepLight         | Tiny Criteo  | 607,959,381    | 10              |
| Wide ResNet-101-2 | MiniImageNet | 126,886,696    | 90              |
| TransformerXL     | WikiText-103 | 191,950,298    | 20              |

be scaled further. To address this limitation, we put the averaging approach inside each Allreduce step rather than accumulating sums until the final step. While this introduces some precision loss due to rounding operations at each averaging step, our experiments demonstrate that the method still maintains fair convergence properties.

The algorithm is wrapped in a custom Allreduce module integrated into PyTorch as a DDP hook. Our hook procedure is invoked by the granularity of a gradient bucket, which by default has a size of at least 25 MB. The procedure consists of three steps: Quantization, Allreduce, and Dequantization. In the case of exponential dithering, the Allreduce step uses a custom reduction function. To achieve this, we implement a customized Allreduce algorithm using NCCL’s point-to-point asynchronous communication API. We support both ring and tree Allreduce. We develop the quantization, de-quantization, and the custom reduction for exponential reduction in CUDA to optimize GPU performance.

To use Global-QSGD, the user simply has to register our DDP hook with one line of code like `model.register_comm_hook()`.

## 9 Evaluation

The aim of our evaluation is to illustrate that our proposal is practical and beneficial in a range of scenarios, including with different bandwidths among GPUs. We focus on measuring the speedup of training throughput in three distinct domains of DL applications. Through task-specific metrics on the test set, we show that Global-QSGD does not impair the model’s generalization ability. We further illustrate that, compared to linear dithering, exponential dithering achieves better overall performance due to its smaller error for small values, despite its additional overhead from the exponential reduction function.

**Setup.** Our experiments are primarily run on one ASUS ESC N4A-E11 server that runs Ubuntu 22.04 with CUDA 11.6, and we use PyTorch 1.13.0. The server is equipped with 4 NVIDIA A100 GPUs, each with 40 GB of RAM. GPUs are peer-to-peer connected by 4 NVLink channels (4th generation). To tease out the effects of bandwidth on training speed, we run experiments with two interconnects: P2P, which employs NVIDIA GPU Direct allowing data to transmit via NVLink directly without the interference of CPU and host memory; SHM, which uses the host memory as a middle buffer; thus, the data will transmit through PCIe. The empirically measured unidirectional bandwidth for 25 MB data is 53.9 GB/s and 5.4 GB/s for P2P and SHM, respectively. Details for bandwidth measurements are in [33][Appendix B].

**Baselines.** Our evaluation includes models across different domains, including DeepLight [10] (Recommendation), Wide ResNet-101-2 [35] (Vision), and TransformerXL [7] (Language modeling) as shown in Table 3. We compare Global-QSGD against the no quantization baseline, QSGD (baseline for quantization), and PowerSGD (baseline for Allreduce compatible compression). The evaluation is conducted with 3 random seeds, and the reported results are averages.

**Convergence.** Figure 3 shows that Global-QSGD can achieve the fastest run time while preserving convergence with P2P. Both linear and exponential dithering achieve good convergence, while striking a trade-off between runtime and convergence performance. Specifically, as expected, linear dithering achieves the quickest run time;

<sup>6</sup> Measured on a A100 with 25 MB data (default bucket size in PyTorch).

<sup>7</sup> This implementation idea can be naturally extended to  $\mathcal{E}_s^{q,p}$ .

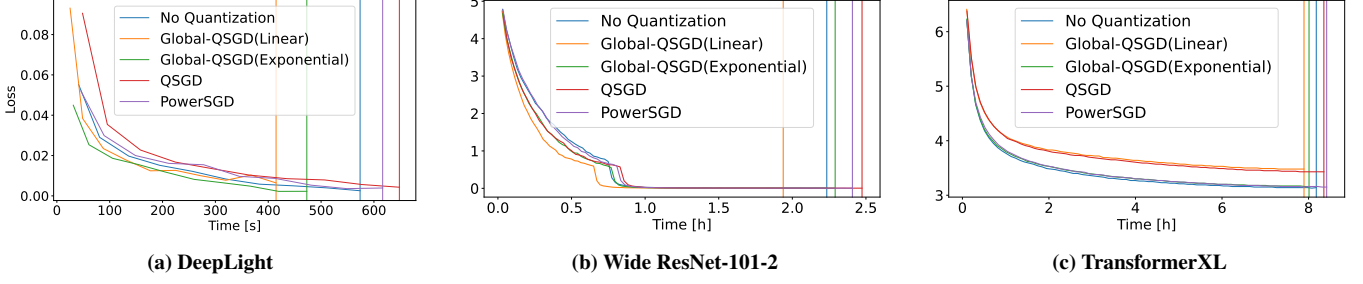


Figure 3: Training loss in P2P. Vertical lines represent the completion times after a fixed iteration count.

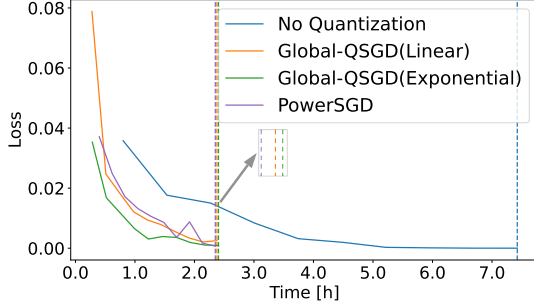


Figure 4: DeepLight training loss in GCP with 64 nodes.

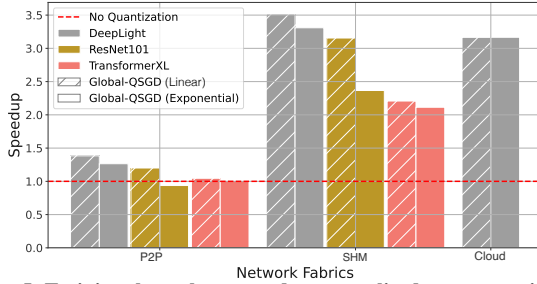


Figure 5: Training throughput speedup normalized to no quantization.

however, exponential dithering is slightly slower but achieves a better convergence similar to no quantization. In particular, exponential dithering can preserve better convergence than other compressors. Given that NVLink P2P provides the highest bandwidth, Global-QSGD demonstrates even more significant speedup with lower-bandwidth network fabrics like SHM or inter-node settings.

**Generalization.** We evaluate models with domain-specific metrics after the fixed-iteration training. Table 4 shows that Global-QSGD can generally preserve generalizability as with no quantization.

**Scalability.** We also run a large-scale experiment with DeepLight on Google Cloud Platform (GCP), with 64 servers, each having one A100 GPU. The servers are connected through a shared network with bandwidth fluctuating from 200 Mbps to 1.5 Gbps. We compare with the PowerSGD baseline as it is Allreduce compatible, while Allgather-based approaches, such as QSGD, are considerably slower and expensive, and thus not included. Figure 4 shows the training curve, where the model experiences more communication bottlenecks, allowing compression to achieve greater time savings.

**Speedup.** Figure 5 shows the speedup obtained by Global-QSGD compared to the no quantization baseline when training different models for a constant number of epochs. The compression method usually obtains a higher speedup with slower network fabrics. Global-QSGD achieves up to  $3.17\times$  speedup in Cloud (GCP) and  $3.51\times$  with SHM. Even when communicating with P2P through NVLink, which has the highest bandwidth, Global-QSGD still achieves  $1.38\times$  speedup. When applying exponential dithering on

Table 4: Model quality after fixed-iteration training.

|                             | DeepLight<br>AUC ( $10^{-1}$ ) | Wide ResNet-101-2<br>Top 5 | TransformerXL<br>PPL |
|-----------------------------|--------------------------------|----------------------------|----------------------|
| No Quantization             | $6.75 \pm 0.10$                | $87.06\% \pm 0.04\%$       | $22.99 \pm 0.11$     |
| $\mathcal{L}_s^{q,p}$ 8-bit | $6.95 \pm 0.03$                | $88.87\% \pm 0.25\%$       | $32.35 \pm 0.71$     |
| $\mathcal{L}_s^{q,p}$ 4-bit | $6.87 \pm 0.11$                | $88.57\% \pm 0.30\%$       | $44.95 \pm 0.93$     |
| $\mathcal{E}_s^{q,p}$ 8-bit | $6.87 \pm 0.03$                | $85.82\% \pm 0.25\%$       | $23.68 \pm 0.12$     |
| QSGD 8-bit                  | $6.78 \pm 0.05$                | $89.39\% \pm 0.32\%$       | $30.82 \pm 0.38$     |
| PowerSGD 32-rank            | $6.76 \pm 0.06$                | $88.19\% \pm 0.22\%$       | $23.36 \pm 0.20$     |

Table 5: Training throughput speedup normalized to no quantization.

|                                  | DeepLight    | Wide ResNet-101-2 | Transformer-XL |
|----------------------------------|--------------|-------------------|----------------|
| <b>P2P Interconnect (NVLink)</b> |              |                   |                |
| $\mathcal{L}_s^{q,p}$ 8-bit      | $1.38\times$ | $1.21\times$      | $1.05\times$   |
| $\mathcal{L}_s^{q,p}$ 4-bit      | $1.65\times$ | $1.05\times$      | $1.02\times$   |
| <b>SHM Interconnect (PCIe)</b>   |              |                   |                |
| $\mathcal{L}_s^{q,p}$ 8-bit      | $3.51\times$ | $2.72\times$      | $1.73\times$   |
| $\mathcal{L}_s^{q,p}$ 4-bit      | $4.94\times$ | $3.68\times$      | $1.89\times$   |

Wide ResNet-101-2 with P2P, a model which is not communication-bottlenecked with the highest bandwidth, Global-QSGD still delivers comparable performance with less than 2% overhead.

**Low bit-width.** Table 5 illustrates the speedup of  $\mathcal{L}_s^{q,p}$  relative to no quantization, highlighting Global-QSGD’s adaptability to low bit-width quantization. Our 4-bit implementation involves the overhead of packing 4-bit values into 8-bit. With dedicated hardware support, these performance gains could be further enhanced.

**Limitations.** The current implementation is not completely optimized. We make use of the NCCL P2P API, and there are synchronous invocations that prevent pipelining opportunities for further overlapping compression and communication.

## 10 Conclusion

We introduced Global-QSGD, a family of quantization operators for distributed learning based on global normalization. This enables the reduction of quantized values without changes of numeric representation (and corresponding overheads) with Allreduce communication and is backed by sound theory. Through extensive experiments, we showed that Global-QSGD yields fast convergence in realistic benchmarks. In particular, we demonstrated that Global-QSGD with exponential dithering achieves the best balance between the variance caused by compression and the speed of convergence.

## Acknowledgments

This publication is based upon work supported by the King Abdulah University of Science and Technology Research Funding (KRF) under Award No ORA-CRG2021-4699.

## References

- [1] S. Agarwal, H. Wang, S. Venkataraman, and D. Papailiopoulos. On the utility of gradient compression in distributed training systems. In *Proceedings of the Conference on Machine Learning and Systems*, 2022.
- [2] D. Alistarh, D. Grubic, J. Z. Li, R. Tomioka, and M. Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2017.
- [3] D. Alistarh, T. Hoefer, M. Johansson, S. Khirirat, N. Konstantinov, and C. Renggli. The convergence of sparsified gradient methods. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2018.
- [4] J. Bernstein, Y. Wang, K. Azizzadenesheli, and A. Anandkumar. SIGNSGD: Compressed optimisation for non-convex problems. In *Proceedings of the International Conference on Machine Learning*, 2018.
- [5] A. Castelló, M. Catalán, M. F. Dolz, E. S. Quintana-Ortí, and J. Duato. Analyzing the impact of the MPI allreduce in distributed training of convolutional neural networks. *Computing*, 2023.
- [6] J.-B. Cordonnier. Convex optimization using sparsified stochastic gradient descent with memory. Technical report, EPFL, 2018.
- [7] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2019.
- [8] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng. Large scale distributed deep networks. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2012.
- [9] A. Défossez, L. Bottou, F. Bach, and N. Usunier. A simple convergence proof of Adam and Adagrad. *Transactions on Machine Learning Research*, 2022.
- [10] W. Deng, J. Pan, T. Zhou, D. Kong, A. Flores, and G. Lin. Deeplight: Deep lightweight feature interactions for accelerating ctr predictions in ad serving. In *International Conference on Web Search and Data Mining*, 2021.
- [11] E. Gorbunov, F. Hanzely, and P. Richtárik. A unified theory of SGD: Variance reduction, sampling, quantization and coordinate descent. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2020.
- [12] S. Horváth and P. Richtárik. A better alternative to error feedback for communication-efficient distributed learning. In *Proceedings of the International Conference on Learning Representations*, 2021.
- [13] S. Horváth, C.-Y. Ho, L. Horváth, A. N. Sahu, M. Canini, and P. Richtárik. Natural compression for distributed deep learning. In *Proceedings of the Conference on Mathematical and Scientific Machine Learning*, 2022.
- [14] M. T. K. Mishchenko, E. Gorbunov and P. Richtárik. Distributed learning with compressed gradient differences. *Optimization Methods and Software*, 2024.
- [15] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *Proceedings of the International Conference on Machine Learning*, 2019.
- [16] A. Koloskova, S. Stich, and M. Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *Proceedings of the International Conference on Machine Learning*, 2019.
- [17] J. Konečný and P. Richtárik. Randomized distributed mean estimation: accuracy vs communication. *Frontiers in Applied Mathematics and Statistics*, 2018.
- [18] M. Langer, Z. He, W. Rahayu, and Y. Xue. Distributed training of deep learning models: A taxonomic perspective. *IEEE Transactions on Parallel and Distributed Systems*, 2020.
- [19] M. Li, R. B. Basat, S. Vargaftik, C. Lao, K. Xu, M. Mitzenmacher, and M. Yu. THC: Accelerating distributed deep learning using tensor homomorphic compression. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, 2024.
- [20] S. Li and T. Hoefer. Near-optimal sparse allreduce for distributed deep learning. In *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2022.
- [21] K. Mishchenko, B. Wang, D. Kovalev, and P. Richtárik. IntSGD: Adaptive floatless compression of stochastic gradients. In *Proceedings of the International Conference on Learning Representations*, 2021.
- [22] T. Na, J. H. Ko, J. Kung, and S. Mukhopadhyay. On-chip training of recurrent neural networks with limited numerical precision. In *Proceedings of the International Joint Conference on Neural Networks*, 2017.
- [23] D. Narayanan, M. Shoeybi, J. Casper, P. LeGresley, M. Patwary, V. Korthikanti, D. Vainbrand, P. Kashinkunti, J. Bernauer, B. Catanzaro, A. Phanishayee, and M. Zaharia. Efficient large-scale language model training on GPU clusters using megatron-LM. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021.
- [24] P. Richtárik, I. Sokolov, and I. Fatkhullin. EF21: A new, simpler, theoretically better, and practically faster error feedback. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2021.
- [25] A. Sapio, M. Canini, C.-Y. Ho, J. Nelson, P. Kalnis, C. Kim, A. Krishnamurthy, M. Moshref, D. Ports, and P. Richtárik. Scaling distributed machine learning with in-network aggregation. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, 2021.
- [26] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *Proceedings of the Annual Conference of the International Speech Communication Association*, 2014.
- [27] S. U. Stich, J.-B. Cordonnier, and M. Jaggi. Sparsified SGD with memory. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2018.
- [28] R. Thakur, R. Rabenseifner, and W. Gropp. Optimization of collective communication operations in MPICH. *The International Journal of High Performance Computing Applications*, 2005.
- [29] T. Vogels, S. P. Karimireddy, and M. Jaggi. PowerSGD: Practical low-rank gradient compression for distributed optimization. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2019.
- [30] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li. TernGrad: Ternary gradients to reduce communication in distributed deep learning. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2017.
- [31] C. Xie, S. Zheng, S. Koyejo, I. Gupta, M. Li, and H. Lin. Cser: Communication-efficient sgd with error reset. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2020.
- [32] J. Xin, S. Bae, K. Park, M. Canini, and C. Hwang. Immediate communication for distributed ai tasks. In *Proceedings of the Workshop on Hot Topics in System Infrastructure*, 2024.
- [33] J. Xin, M. Canini, P. Richtárik, and S. Horváth. Quantize once, train fast: Allreduce-compatible compression with provable guarantees, 2025. arXiv:2305.18627 [cs.LG]; Full version of this paper.
- [34] M. Yu, Z. Lin, K. Narra, S. Li, Y. Li, N. S. Kim, A. Schwing, M. Annavaram, and S. Avestimehr. GradiVeQ: Vector quantization for bandwidth-efficient gradient aggregation in distributed CNN training. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2018.
- [35] S. Zagoruyko and N. Komodakis. Wide residual networks. In *British Machine Vision Conference*, 2016.

## A Proofs

In this section, we include complete proofs of the claims made in the main paper.

### A.1 Proof of Lemma 1

Firstly, we show unbiasedness.

$$\mathbb{E}[\mathcal{Q}(\mathbf{x})] = \mathbb{E}\left[\frac{1}{n} \sum_{i=1}^n \mathcal{C}_i(x_i)\right] = \frac{1}{n} \sum_{i=1}^n \mathbb{E}[\mathcal{C}_i(x_i)] \stackrel{(4)}{=} \frac{1}{n} \sum_{i=1}^n x_i = \mathbf{x}.$$

For the variance,

$$\mathbb{E}[\|\mathcal{Q}(\mathbf{x}) - \mathbf{x}\|_2^2] = \mathbb{E}\left[\left\|\frac{1}{n} \sum_{i=1}^n \mathcal{C}_i(x_i) - x_i\right\|_2^2\right] = \frac{1}{n^2} \sum_{i=1}^n \mathbb{E}[\|\mathcal{C}_i(x_i) - x_i\|_2^2] \stackrel{(4)}{\leq} \frac{\omega}{n} \frac{1}{n} \sum_{i=1}^n \|x_i\|_2^2,$$

where the second equality is due to independence and zero mean of each summand.

### A.2 Proof of Lemma 2

It is easy to see that such operator is unbiased since  $\mathbb{E}[(\xi(y_i))] = (\xi(y_i))_j$  by construction. Therefore,

$$\begin{aligned} \mathbb{E}[\mathcal{Q}_s^{q,p}(\mathbf{x})] &\stackrel{(1)}{=} \mathbb{E}\left[\|\mathbf{x}\|_{q,p} \frac{1}{n} \sum_{i=1}^n \text{sign}(x_i) \circ \xi_i(y_i)\right] \\ &= \|\mathbf{x}\|_{q,p} \frac{1}{n} \sum_{i=1}^n \text{sign}(x_i) \circ \mathbb{E}[\xi_i(y_i)] \\ &\stackrel{(2)}{=} \|\mathbf{x}\|_{q,p} \frac{1}{n} \sum_{i=1}^n \text{sign}(x_i) \circ y_i \\ &= \|\mathbf{x}\|_{q,p} \frac{1}{n} \sum_{i=1}^n \text{sign}(x_i) \circ \frac{x_i}{\|\mathbf{x}\|_{q,p}} = \mathbf{x}. \end{aligned}$$

Furthermore, note that for  $\mathcal{Q}$ , the local compressors do not belong to  $\mathcal{U}^n(\omega)$  for any  $\omega > 0$  due to their dependence on  $\mathbf{x}$ .

To obtain the variance bound, we show that it is sufficient to look at  $n = 1$ , which corresponds to the unbiased compressor (Definition 2), due to the following property.

$$\begin{aligned} &\mathbb{E}[\|\mathcal{Q}_s^{q,p}(\mathbf{x}) - \mathbf{x}\|_2^2] \\ &\stackrel{(1)}{=} \mathbb{E}\left[\left\|\|\mathbf{x}\|_{q,p} \frac{1}{n} \sum_{i=1}^n \left(\text{sign}(x_i) \circ \xi_i(y_i) - \frac{x_i}{\|\mathbf{x}\|_{q,p}}\right)\right\|_2^2\right] \\ &= \frac{1}{n^2} \sum_{i=1}^n \mathbb{E}\left[\left\|\|\mathbf{x}\|_{q,p} \left(\text{sign}(x_i) \circ \xi_i(y_i) - \frac{x_i}{\|\mathbf{x}\|_{q,p}}\right)\right\|_2^2\right] \\ &= \frac{1}{n^2} \mathbb{E}[\|\|\mathbf{x}\|_{q,p} \text{sign}(\mathbf{x}) \circ \xi(\mathbf{y}) - \mathbf{x}\|_2^2] \\ &= \frac{1}{n^2} \mathbb{E}[\|\mathcal{Q}_s^{q,p}(\mathbf{x}) - \mathbf{x}\|_2^2], \end{aligned}$$

where the second inequality is due to independence of  $\{\xi_i\}$ . If we can show that  $\mathcal{Q}_s^{q,p}(\mathbf{x}) \in \mathcal{U}^{1,nd}(\omega)$ , then

$$\begin{aligned} \mathbb{E}[\|\mathcal{Q}_s^{q,p}(\mathbf{x}) - \mathbf{x}\|_2^2] &= \frac{1}{n^2} \mathbb{E}[\|\mathcal{Q}_s^{q,p}(\mathbf{x}) - \mathbf{x}\|_2^2] \\ &\leq \frac{\omega}{n^2} \|\mathbf{x}\|_{2,2}^2 = \frac{\omega}{n} \frac{1}{n} \sum_{i=1}^n \|x_i\|_2^2, \end{aligned}$$

which implies  $\theta = \omega/n$ .

### A.3 Proof of Theorem 3

The results provided in the theorem follow the same logic as we use in Lemma 2, i.e., our compression is equivalent to applying standard compression to concatenated vector  $\mathbf{x} = \mathbf{x} := [x_1, x_2, \dots, x_n] \in \mathbb{R}^{nd}$ .

The rest of the proof uses known one-node results of Alistarh et al. [2], Horváth et al. [13].

For the variance part, we firstly use the second part of Lemma 2, which shows that  $\mathcal{Q}_s^{q,p} \in \mathcal{C}(\omega) \Rightarrow \mathcal{Q}_s^{q,p} \in \mathcal{U}^{n,d}(\theta)$  with  $\theta = \omega/n$ . Secondly, we apply  $n = 1$  results of [13, Theorem 7] and [2, Theorem 3.4] combined with the fact about norm stating that  $\|\mathbf{x}\|_{q,p} \leq \|\mathbf{x}\|_{2,2}$  for  $p, q \geq 2$ .

The claim about sparsity with  $p = q = 2$  follows from the first part of Lemma 2, which implies that the number of non-zero elements of  $\mathcal{Q}_s^{q,p}$  before aggregation is the same as  $\mathcal{Q}_s^{q,p}$ . The rest of the proof for  $\mathcal{L}_s^{2,2}$  follows directly from [2, Lemma 3.1]. For  $\mathcal{E}_s^{2,2}$ , one can also directly apply [2, Lemma 3.1] using the fact that the length of the first segment  $[l_0, l_1]$  is  $1/2^{s-1}$ .

### A.4 Proof of Lemma 4

As discussed in the main part of the paper, the only difference from the standard analysis is the difference in the variance bound. The lemma 4 shows that both bounds are almost equivalent, where they only differ by a constant. Recall  $\mathbb{E}[\cdot]$  denotes the standard expectation.

*Proof.* We assume that the noise due to sampling and compression are independent. For the first case, we have

$$\begin{aligned} & \mathbb{E} \left[ \left\| \frac{1}{n} \sum_{i=1}^n \mathcal{C}_i(\nabla f_i(x, \xi_i)) - \nabla f(x) \right\|_2^2 \right] \\ &= \frac{1}{n^2} \sum_{i=1}^n \mathbb{E} [\|\mathcal{C}_i(\nabla f_i(x, \xi_i)) - \nabla f_i(x)\|_2^2] + \frac{1}{n^2} \sum_{i \neq j} \mathbb{E} [\langle \mathcal{C}_i(\nabla f_i(x, \xi_i)) - \nabla f_i(x), \mathcal{C}_j(\nabla f_j(x, \xi_j)) - \nabla f_j(x) \rangle] \\ &= \frac{1}{n^2} \sum_{i=1}^n \mathbb{E} [\|\mathcal{C}_i(\nabla f_i(x, \xi_i)) - \nabla f_i(x)\|_2^2] + \frac{1}{n^2} \sum_{i \neq j} \langle \mathbb{E} [\mathcal{C}_i(\nabla f_i(x, \xi_i)) - \nabla f_i(x)], \mathbb{E} [\mathcal{C}_j(\nabla f_j(x, \xi_j)) - \nabla f_j(x)] \rangle. \end{aligned}$$

Using tower property, we get

$$\begin{aligned} \mathbb{E} [\mathcal{C}_i(\nabla f_i(x, \xi_i)) - \nabla f_i(x)] &= \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\mathbb{E}_{\mathcal{C}_i} [\mathcal{C}_i(\nabla f_i(x, \xi_i)) - \nabla f_i(x) | \xi_i]] \\ &= \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\nabla f_i(x, \xi_i) - \nabla f_i(x)] = 0. \end{aligned}$$

Furthermore, using the same technique and the unbiasedness and bounded variance of the compression operator yields

$$\begin{aligned} & \mathbb{E} [\|\mathcal{C}_i(\nabla f_i(x, \xi_i)) - \nabla f_i(x)\|_2^2] = \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\mathbb{E}_{\mathcal{C}_i} [\|\mathcal{C}_i(\nabla f_i(x, \xi_i)) - \nabla f_i(x)\|_2^2 | \xi_i]] \\ &= \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\mathbb{E}_{\mathcal{C}_i} [\|\mathcal{C}_i(\nabla f_i(x, \xi_i))\|_2^2 | \xi_i]] - 2\mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\mathbb{E}_{\mathcal{C}_i} [\langle \mathcal{C}_i(\nabla f_i(x, \xi_i)), \nabla f_i(x) \rangle | \xi_i]] + \|\nabla f_i(x)\|_2^2 \\ &= \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\mathbb{E}_{\mathcal{C}_i} [\|\mathcal{C}_i(\nabla f_i(x, \xi_i))\|_2^2 | \xi_i]] - \|\nabla f_i(x)\|_2^2 \\ &= \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\mathbb{E}_{\mathcal{C}_i} [\|\mathcal{C}_i(\nabla f_i(x, \xi_i)) - \nabla f_i(x, \xi_i) + \nabla f_i(x, \xi_i)\|_2^2 | \xi_i]] - \|\nabla f_i(x)\|_2^2 \\ &= \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\mathbb{E}_{\mathcal{C}_i} [\|\mathcal{C}_i(\nabla f_i(x, \xi_i)) - \nabla f_i(x, \xi_i)\|_2^2 | \xi_i]] + \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi_i)\|_2^2] - \|\nabla f_i(x)\|_2^2 \\ &\quad + 2\mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\mathbb{E}_{\mathcal{C}_i} [\langle \mathcal{C}_i(\nabla f_i(x, \xi_i)) - \nabla f_i(x, \xi_i), \nabla f_i(x, \xi_i) \rangle | \xi_i]] \\ &= \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\mathbb{E}_{\mathcal{C}_i} [\|\mathcal{C}_i(\nabla f_i(x, \xi_i)) - \nabla f_i(x, \xi_i)\|_2^2 | \xi_i]] + \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi_i)\|_2^2] - \|\nabla f_i(x)\|_2^2 \\ &\quad + 2\mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\langle \mathbb{E}_{\mathcal{C}_i} [\mathcal{C}_i(\nabla f_i(x, \xi_i))] - \nabla f_i(x, \xi_i), \nabla f_i(x, \xi_i) \rangle | \xi_i]] \\ &= \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\mathbb{E}_{\mathcal{C}_i} [\|\mathcal{C}_i(\nabla f_i(x, \xi_i)) - \nabla f_i(x, \xi_i)\|_2^2 | \xi_i]] + \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi_i)\|_2^2] - \|\nabla f_i(x)\|_2^2 \\ &\quad + 2\mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\langle 0, \nabla f_i(x, \xi_i) \rangle | \xi_i]] \\ &\leq (\omega + 1)\mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi_i)\|_2^2] - \|\nabla f_i(x)\|_2^2 \\ &= (\omega + 1)\mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi_i) - \nabla f_i(x) + \nabla f_i(x)\|_2^2] - \|\nabla f_i(x)\|_2^2 \\ &= (\omega + 1)\mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi_i) - \nabla f_i(x)\|_2^2] + 2(\omega + 1)\langle \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\nabla f_i(x, \xi_i)] - \nabla f_i(x), \nabla f_i(x) \rangle + \omega\|\nabla f_i(x)\|_2^2 \\ &= (\omega + 1)\mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi_i) - \nabla f_i(x)\|_2^2] + \omega\|\nabla f_i(x)\|_2^2, \end{aligned}$$

which concludes the first part of the proof. For the second part, we proceed analogously and have that for  $\mathcal{Q} \in \mathcal{U}^{n,d}(\theta)$

$$\begin{aligned}
& \mathbb{E} [\|\mathcal{Q}([\nabla f_1(x, \xi_1), \dots, \nabla f_n(x, \xi_n)]) - \nabla f(x)\|_2^2] \\
&= \mathbb{E}_{\{\xi_i \sim \mathcal{D}_i\}_{i=1}^n} \left[ \mathbb{E}_{\mathcal{Q}} \left[ \left\| \mathcal{Q}([\nabla f_1(x, \xi_1), \dots, \nabla f_n(x, \xi_n)]) - \frac{1}{n} \sum_{i=1}^n \nabla f_i(x, \xi_i) + \frac{1}{n} \sum_{i=1}^n \nabla f_i(x, \xi_i) - \nabla f(x) \right\|_2^2 \middle| \{\xi_i\}_{i=1}^n \right] \right] \\
&= \mathbb{E}_{\{\xi_i \sim \mathcal{D}_i\}_{i=1}^n} \left[ \mathbb{E}_{\mathcal{Q}} \left[ \left\| \mathcal{Q}([\nabla f_1(x, \xi_1), \dots, \nabla f_n(x, \xi_n)]) - \frac{1}{n} \sum_{i=1}^n \nabla f_i(x, \xi_i) \right\|_2^2 \middle| \{\xi_i\}_{i=1}^n \right] \right] \\
&\quad + \mathbb{E}_{\{\xi_i \sim \mathcal{D}_i\}_{i=1}^n} \left[ \left\| \frac{1}{n} \sum_{i=1}^n \nabla f_i(x, \xi_i) - \nabla f(x) \right\|_2^2 \right] \\
&\quad + \mathbb{E}_{\{\xi_i \sim \mathcal{D}_i\}_{i=1}^n} \left[ \left\langle \mathbb{E}_{\mathcal{Q}} [\mathcal{Q}([\nabla f_1(x, \xi_1), \dots, \nabla f_n(x, \xi_n)]) | \{\xi_i\}_{i=1}^n] - \frac{1}{n} \sum_{i=1}^n \nabla f_i(x, \xi_i), \frac{1}{n} \sum_{i=1}^n \nabla f_i(x, \xi_i) - \nabla f(x) \right\rangle \right].
\end{aligned}$$

The first element of the scalar product is zero since  $\mathcal{Q}$  is unbiased. Therefore

$$\begin{aligned}
& \mathbb{E} [\|\mathcal{Q}([\nabla f_1(x, \xi_1), \dots, \nabla f_n(x, \xi_n)]) - \nabla f(x)\|_2^2] \\
&= \mathbb{E}_{\{\xi_i \sim \mathcal{D}_i\}_{i=1}^n} \left[ \mathbb{E}_{\mathcal{Q}} \left[ \left\| \mathcal{Q}([\nabla f_1(x, \xi_1), \dots, \nabla f_n(x, \xi_n)]) - \frac{1}{n} \sum_{i=1}^n \nabla f_i(x, \xi_i) \right\|_2^2 \middle| \{\xi_i\}_{i=1}^n \right] \right] \\
&\quad + \mathbb{E}_{\{\xi_i \sim \mathcal{D}_i\}_{i=1}^n} \left[ \left\| \frac{1}{n} \sum_{i=1}^n \nabla f_i(x, \xi_i) - \nabla f(x) \right\|_2^2 \right] \\
&\leq \frac{\theta}{n} \sum_{i=1}^n \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi_i)\|_2^2] + \mathbb{E}_{\{\xi_i \sim \mathcal{D}_i\}_{i=1}^n} \left[ \left\| \frac{1}{n} \sum_{i=1}^n \nabla f_i(x, \xi_i) - \nabla f(x) \right\|_2^2 \right].
\end{aligned}$$

For  $\mathbb{E}_{\{\xi_i \sim \mathcal{D}_i\}_{i=1}^n} \left[ \left\| \frac{1}{n} \sum_{i=1}^n \nabla f_i(x, \xi_i) - \nabla f(x) \right\|_2^2 \right]$ , we have

$$\begin{aligned}
& \mathbb{E}_{\{\xi_i \sim \mathcal{D}_i\}_{i=1}^n} \left[ \left\| \frac{1}{n} \sum_{i=1}^n \nabla f_i(x, \xi_i) - \nabla f(x) \right\|_2^2 \right] \\
&= \frac{1}{n^2} \sum_{i=1}^n \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi_i) - \nabla f_i(x)\|_2^2] + \frac{1}{n^2} \sum_{i \neq j} \langle \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\nabla f_i(x, \xi_i)] - \nabla f_i(x), \mathbb{E}_{\xi_j \sim \mathcal{D}_j} [\nabla f_j(x, \xi_j)] - \nabla f_j(x) \rangle \\
&= \frac{1}{n^2} \sum_{i=1}^n \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi_i) - \nabla f_i(x)\|_2^2]
\end{aligned}$$

Finally, for the  $\mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi_i)\|_2^2]$ , we have

$$\begin{aligned}
& \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi_i)\|_2^2] = \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi_i) - \nabla f_i(x) + \nabla f_i(x)\|_2^2] \\
&= \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi_i) - \nabla f_i(x) + \nabla f_i(x)\|_2^2] + \|\nabla f_i(x)\|_2^2 + 2\langle \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\nabla f_i(x, \xi_i)] - \nabla f_i(x), \nabla f_i(x) \rangle \\
&= \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi_i) - \nabla f_i(x) + \nabla f_i(x)\|_2^2] + \|\nabla f_i(x)\|_2^2.
\end{aligned}$$

Putting them all together

$$\begin{aligned}
& \mathbb{E} [\|\mathcal{Q}([\nabla f_1(x, \xi_1), \dots, \nabla f_n(x, \xi_n)]) - \nabla f(x)\|_2^2] \\
&\leq \frac{\theta}{n} \sum_{i=1}^n \mathbb{E}_{\xi_i \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi_i)\|_2^2] + \mathbb{E}_{\{\xi_i \sim \mathcal{D}_i\}_{i=1}^n} \left[ \left\| \frac{1}{n} \sum_{i=1}^n \nabla f_i(x, \xi_i) - \nabla f(x) \right\|_2^2 \right] \\
&= \frac{\theta}{n} \sum_{i=1}^n \mathbb{E} [\|\nabla f_i(x)\|_2^2] + \left( \theta + \frac{1}{n} \right) \frac{1}{n} \sum_{i=1}^n \mathbb{E} [\|\nabla f_i(x, \xi_i) - \nabla f_i(x)\|_2^2]
\end{aligned}$$

yields the desired bound.  $\square$

For the adaptive methods, we show that the assumptions in Section 2.3 of [9] remain applicable when combining stochastic gradients with global quantization. The only assumption related to the stochastic gradients is that the  $\ell_\infty$  norm of the stochastic gradients is almost surely bounded.

In our case, the stochastic estimator has the following form

$$\mathcal{Q}([\nabla f_1(x, \xi_1), \dots, \nabla f_n(x, \xi_n)]).$$

We assume that  $\|\nabla f_i(x, \xi_i)\|_\infty \leq R$ , for all  $i \in [n]$ . Furthermore, for both linear and exponential global quantization, we have

$$\begin{aligned} \|\mathcal{Q}([\nabla f_1(x, \xi_1), \dots, \nabla f_n(x, \xi_n)])\|_\infty &= \left\| \frac{1}{n} \sum_{i=1}^n (\mathcal{Q}_s^{q,p}([\nabla f_1(x, \xi_1), \dots, \nabla f_n(x, \xi_n)]))_i \right\|_\infty \\ &\leq \frac{1}{n} \sum_{i=1}^n \left\| (\mathcal{Q}_s^{q,p}([\nabla f_1(x, \xi_1), \dots, \nabla f_n(x, \xi_n)]))_i \right\|_\infty \\ &\leq \|\nabla f_1(x, \xi_1), \dots, \nabla f_n(x, \xi_n)\|_{p,q} \\ &= \left( \sum_{i=1}^n \|\nabla f_i(x, \xi_i)\|_q^p \right)^{1/p} \\ &\leq \left( \sum_{i=1}^n (d \|\nabla f_i(x, \xi_i)\|_\infty)^{p/q} \right)^{1/p} \leq n^{1/p} d^{1/q} R. \end{aligned}$$

Therefore, global quantization preserves the bound of the  $\ell_\infty$  norm of the stochastic gradients up to a constant. This implies that all the convergence guarantees of Défossez et al. [9] also apply to our setting with global quantization.

#### A.5 Proof for Equation 10

$$\begin{aligned} 2\log(N)\alpha + 2\frac{\log(N)S}{\beta} + \frac{\log(N)S}{\gamma} &> 2\log(N)\alpha + 2\frac{\log(N)\hat{S}}{\beta} + \frac{\log(N)\hat{S}}{\hat{\gamma}} + \delta S \\ 2\frac{S}{\beta} + \frac{S}{\gamma} &> 2\frac{\hat{S}}{\beta} + \frac{\hat{S}}{\hat{\gamma}} \\ 2\frac{\rho}{\beta} + \frac{\rho}{\gamma} &> 2\frac{1}{\beta} + \frac{1}{\hat{\gamma}} \\ 2\rho\gamma\hat{\gamma} + \rho\beta\hat{\gamma} &> 2\gamma\hat{\gamma} + \beta\gamma \\ \rho\beta\hat{\gamma} - \beta\gamma &> 2\gamma\hat{\gamma} - 2\rho\gamma\hat{\gamma} \\ \beta(\rho\hat{\gamma} - \gamma) &> 2\gamma\hat{\gamma} - 2\rho\gamma\hat{\gamma} \end{aligned}$$

Since  $\hat{\gamma}$  is the computation speed (byte/s), so  $\hat{\gamma} \geq 0$ . We divide both sides of the inequality by  $\hat{\gamma}$ , and replace  $\omega = \frac{\gamma}{\hat{\gamma}}$ :

$$\begin{aligned} \beta(\rho - \omega) &> 2\gamma - 2\rho\gamma \\ \beta(\rho - \omega) &> 2\gamma(1 - \rho) \end{aligned}$$

Then we have:

$$\begin{cases} \beta > \frac{2\gamma(1-\rho)}{(\rho-\omega)}, & \text{if } (\rho > \omega), \\ \beta < \frac{2\gamma(1-\rho)}{(\rho-\omega)}, & \text{if } (\rho < \omega). \end{cases}$$

## B Bandwidth Measurement

We begin by describing our hardware setup. Our experiments were conducted on a node equipped with four A100-SXM4-80GB GPUs. According to NVIDIA's specifications, each A100 GPU can achieve up to 300 GB/s (600 GB/s bi-directional) bandwidth using 12 channels of 4th-generation NVLink.

To verify the bandwidth, we ran the following command:

```
1 nvidia-smi nvlink --status
2 GPU 0: NVIDIA A100-SXM4-80GB (UUID: GPU-d9dba6f4-6927-ad43-418f-6d87f2265a79)
3 Link 0: 25 GB/s
4 Link 1: 25 GB/s
5 Link 2: 25 GB/s
6 Link 3: 25 GB/s
7 Link 4: 25 GB/s
8 Link 5: 25 GB/s
9 Link 6: 25 GB/s
10 Link 7: 25 GB/s
11 Link 8: 25 GB/s
12 Link 9: 25 GB/s
13 Link 10: 25 GB/s
14 Link 11: 25 GB/s
```

The output confirms that each GPU is connected to 12 NVLink channels, each providing 25 GB/s of single-direction bandwidth, resulting in a total of 300 GB/s.

Next, we used the following command to examine the peer-to-peer (P2P) connections between GPUs:

```
1 nvidia-smi topo -m
2 GPU0 GPU1 GPU2 GPU3 NIC0 NIC1 CPU Affinity NUMA Affinity GPU NUMA ID
3 GPU0 X NV4 NV4 NV4 SYS SYS 0-127 0 N/A
4 GPU1 NV4 X NV4 NV4 SYS SYS 0-127 0 N/A
5 GPU2 NV4 NV4 X NV4 SYS SYS 0-127 0 N/A
6 GPU3 NV4 NV4 NV4 X PHB PHB 0-127 0 N/A
7 NIC0 SYS SYS SYS PHB X PIX
8 NIC1 SYS SYS SYS PHB PIX X
```

This confirms that the GPUs are connected via NVLink with 12 channels distributed among the three peers, resulting in each GPU pair being connected with four NVLink channels, accounting for a 100 GB/s single-direction bandwidth.

We measured the GPU-to-GPU bandwidth with the following command:

```
1 mpirun -np 10 ./build/all_reduce_perf -b 25M -e 25M -g 4
```

The bandwidth is obtained using the NVIDIA NCCL Test suite, a standard tool for testing bandwidth with NVIDIA devices. We performed the NCCL Allreduce operation with a message size of 25 MB, consistent with our deep learning workload communication settings (the default bucket size in PyTorch DDP). It is important to note that Allreduce involves multi-stage collective communication, which does not always saturate the bandwidth and therefore does not achieve peak P2P throughput. Therefore, we empirically measure the bandwidth rather than using theoretical limits.

We conducted measurements under two settings: With Nvidia GPU Direct (P2P) enabled, where data transfer occurs over NVLink. Using Shared Memory (SHM), where data is transferred through the host memory before reaching the destination. We controlled these settings by modifying NCCL's environment variable NCCL\_P2P\_DISABLE. The measured results were 53.9 GB/s for P2P and 5.4 GB/s for SHM.