Communication-Efficient Distributed Learning via Sparse and Adaptive Stochastic Gradient

Journal:	*Transactions on Big Data*
Manuscript ID	TBD-2023-10-0656
Manuscript Types:	Regular Paper
Keywords:	Distributed algorithm, efficient communication, adaptive aggregation, gradient sparsification, stochastic gradient descent

Communication-Efficient Distributed Learning via Sparse and Adaptive Stochastic Gradient

Xiaoge Deng, Dongsheng Li, Tao Sun and Xicheng Lu

***Abstract*—Gradient-based optimization methods implemented on distributed computing architectures are increasingly used to tackle large-scale machine learning applications. A key bottleneck in such distributed systems is the communication overhead for exchanging information such as stochastic gradients between workers. The inherent causes of this bottleneck are the frequent communication rounds and the full model gradient transmission in every round. In this paper, we propose a communication-efficient distributed algorithm named SASG, which enjoys the advantages of sparse communication and adaptive aggregated stochastic gradients. By determining the workers who need to communicate based on an adaptive aggregation rule and sparsifying the transmitted information, the SASG algorithm reduces both the overhead of communication rounds and the number of communication bits in the distributed system. Furthermore, we define a new Lyapunov function to prove that the communication-efficient algorithm is convergent. The convergence result is identical to the sublinear rate of stochastic gradient descent, and our result also reveals that SASG scales well with the number of distributed workers. Finally, experiments on training deep neural networks indicate that the proposed algorithm can significantly reduce communication overhead compared to previous methods.**

***Index Terms*—Distributed algorithm, efficient communication, adaptive aggregation, gradient sparsification, stochastic gradient descent.**

I. Introduction

OVER the past few decades, the scale and complexity of machine learning (ML) models and datasets have significantly increased [1], [2], leading to greater computational demands and spurring the development of distributed training [3]–[5]. A large number of distributed machine learning tasks can be described as

$$\min_{\omega\in\mathbb{R}^d} F(\omega) := \frac{1}{M}\sum_{m\in\mathcal{M}} \mathbb{E}_{\xi_m\sim\mathcal{D}_m}\left[f_m(\omega;\xi_m)\right], \qquad (1)$$

where ω is the parameter vector to be learned, d is the dimension of parameters, and $\mathcal{M} := \{1,\dots,M\}$ denotes the set of distributed workers. $\{f_m\}_{m=1}^M$ are smooth (unnecessary to be convex) loss functions kept at worker m, and $\{\xi_m\}_{m=1}^M$ are independent random data samples associated with probability distribution $\{\mathcal{D}_m\}_{m=1}^M$. For simplicity, we define $F_m(\omega) := \mathbb{E}_{\xi_m\sim\mathcal{D}_m}\left[f_m(\omega;\xi_m)\right]$. Let ω^* be the optimal solution and $F^* := F(\omega^*)$. Problem (1) is encountered in a broad range of distributed machine learning tasks, from linear models to deep neural networks [3], [6].

Stochastic gradient descent (SGD) is a widely used optimization algorithm for solving problem (1). Given a learning rate γ, at each iteration t, the algorithm performs as

$$\omega^{t+1} = \omega^t - \gamma\cdot\frac{1}{M}\sum_{m\in\mathcal{M}}\nabla f_m(\omega^t;\xi_m^t), \qquad (2)$$

where ξ_m^t is a mini-batch data selected by worker m at the t-th iteration. In the context of distributed machine learning, parameter server (PS) is a common architecture that has been studied extensively in [4], [7]. In this architecture, M workers parallelly compute the gradients based on their local data, and a centralized server updates the global variable ω. The distributed SGD in PS operates as follows. At iteration t, the server broadcasts the current model ω^t to all workers, and each worker $m\in\mathcal{M}$ computes the local gradient $\nabla f_m(\omega^t;\xi_m^t)$ with a randomly selected mini-batch of samples $\xi_m^t\sim\mathbb{D}_m$ and uploads it to the server. The server aggregates the received gradients and updates ω^t accordingly via (2). It is worth noting that the convergence of this distributed SGD algorithm has been proved under mild assumptions [8].

Intuitively, multi-processor collaborative training for one task can speed up the training process and reduce training time. However, the associated communication costs typically hinder the scalability of distributed systems [9]. More specifically, at each iteration of PS training (2), the server needs to communicate with all workers to obtain fresh gradients $\{\nabla f_m(\omega^t;\xi_m^t)\}_{m=1}^M$, which may lead to unaffordable communication overhead in systems constrained by limited communication resources such as energy and bandwidth. Even worse, when the computation-to-communication ratio is low, e.g., using high-speed computing devices with low-speed interconnect to train a ML model, parallelization across multiple processors can actually result in lower performance than training on a single processor [10]. Therefore, communication overhead between distributed workers and the server poses a significant bottleneck.

Various techniques have been proposed to alleviate this issue, including sparse communication [11]–[14] and adaptive aggregation methods [15]–[17]. The former is dedicated to compressing each gradient information $\nabla f_m(\omega^t;\xi_m^t)$, while the latter focuses on reducing the number of communications M. Since these two techniques are orthogonal, an important question arises: *can we integrate both approaches to obtain a superior mechanism for problem* (1)*?*

Xiaoge Deng, Dongsheng Li, Tao Sun and Xicheng Lu are with the National Key Laboratory of Parallel and Distributed Computing, College of Computer, National University of Defense Technology, Changsha, 410073, Hunan, China. E-mail: dengxg@nudt.edu.cn, dsli@nudt.edu.cn, nudtsuntao@163.com

Manuscript received April 19, 2021; revised August 16, 2021.

This study provides a positive answer. Specifically, we propose a communication-efficient distributed algorithm that simultaneously reduces the number of communication rounds and bits, without compromising on the convergence rate.

Contributions. This paper focuses on reducing the worker-to-server uplink communication overhead in the PS architecture, which we also refer to as upload. Unlike the server-to-worker downlink communication (i.e., broadcast the same parameter ω), which can be performed concurrently or implemented in a tree-structured manner as in many MPI implementations, the server has to receive gradients of workers sequentially to avoid interference from other workers, resulting in extra latency. Consequently, uploads constitute the primary communication overhead in PS and are the subject of numerous related works [11], [15], [18]–[21]. Throughout this paper, one communication round signifies one upload from a worker. The contributions of the paper are summarized below.

- We propose a communication-efficient algorithm for distributed learning that can reduce both the number of communication rounds and the number of bits.
- We define an auxiliary sequence and devise a new Lyapunov function to establish the convergence analysis of the proposed algorithm. The theoretical results are consistent with the sublinear convergence rate of SGD and exhibit scalability to the number of distributed workers.
- We conduct extensive experiments to demonstrate the superiority of the proposed SASG algorithm.[1]

Notation. Throughout the paper, $\mathbb{E}[\cdot]$ denotes the expectation taken for randomly selected data samples. For a vector $\mathbf{x} \in \mathbb{R}^d$ and a scalar $a \in \mathbb{R}$, $\|\mathbf{x}\|$ and $|a|$ represent the ℓ_2-norm and the absolute value, respectively. $[d]$ is the set $\{1, 2, \cdots, d\}$, and $|\mathcal{S}|$ represents the number of elements in the set $\mathcal{S}$.

II. Related work

Numerous communication-efficient distributed learning approaches have been proposed to fully utilize the computational power of distributed clusters [15], [18]–[20], [22]–[26]. More information can be found in a comprehensive survey [27]. This paper briefly reviews two kinds of related work: 1). from the perspective of what to communicate, the number of transmitted bits per communication round can be reduced via quantization or sparsification, and 2). from the perspective of when to communicate, some communication rules are used to save the number of communication rounds.

A. Communication bit reduction

This research category mainly revolves around quantization and sparsification. The quantization approach compresses information by transmitting lower bits instead of the data originally represented by 32 bits on each dimension of the transmitted gradient. Quantized stochastic gradient descent (QSGD) [18] utilizes an adjustable quantization level, allowing additional flexibility to control the trade-off between the per-iteration communication cost and the convergence rate. TernGrad [20] reduces the communication data size by using ternary gradients. 1-bit quantization method was developed in [28], [29], which reduces each component of the gradient to just its sign (one bit). Adaptive quantization methods are also investigated in [30] to reduce the communication cost.

Sparsification methods reduce the number of elements transmitted at each iteration. Such methods can be divided into two main categories: random and deterministic sparsification. Random sparsification [26], [31] selects some entries randomly for communication. This ideology is named random-k, with k being the number of selected elements. This random choice method is usually an unbiased estimate of the original gradient, making it quite friendly for theoretical analysis. Unlike random sparsification, deterministic sparsification [11]–[14] preserves only a few coordinates of the stochastic gradient with the largest magnitudes. This ideology is also known as top-k. In contrast to the unbiased scheme, it is clear that this approach needs to work with the error feedback or accumulation procedure [14], [32], [33] for convergence.

B. Communication round reduction

Reducing the number of communication rounds to improve communication efficiency is another research focus. Shamir et al. [34] leveraged higher-order information (newton-type method) instead of traditional gradient information, thereby reducing the number of communication rounds. Hendrikx et al. [35] proposed a distributed preconditioned accelerated gradient method to reduce the number of communication rounds. Novel aggregation techniques, including periodic aggregation [19], [24], [36] and adaptive aggregation [15], [16], [21], are also investigated and used to skip certain communications. Among them, local SGD [24], [36] allows every worker to perform local model updates independently, and the resultant models are averaged periodically. Lazily aggregated gradient (LAG) [15] updates the model on the server side, and workers only adaptively upload information that is determined to be informative enough. Unfortunately, while the original LAG has good performance in the deterministic settings (i.e., with full gradient), its performance in the stochastic setting degrades significantly [17]. Recent efforts have been made toward adaptive uploading in stochastic settings [17], [37]. Communication-censored distributed stochastic gradient descent (CSGD) [37] algorithm increases the batch size to alleviate the effect of stochastic gradient noise. Lazily aggregated stochastic gradient (LASG) [17] designed a set of new adaptive communication rules tailored for stochastic gradients, achieving remarkable empirical performance.

We have sourced some ideas from LASG. The significant difference is that LASG reduces communication round overhead only, while our SASG method extends this idea to reduce the overhead of both communication rounds and bits in the stochastic setting. Besides, SASG reduces the memory overhead required for adaptive aggregation technique by storing only the sparsified data on the server side.

III. Algorithm development

This section introduces SASG, a communication-efficient distributed algorithm that reduces both the number of communication rounds and bits. Firstly, we describe the motivation

[1]The code is available at https://github.com/xiaogdeng/SASG

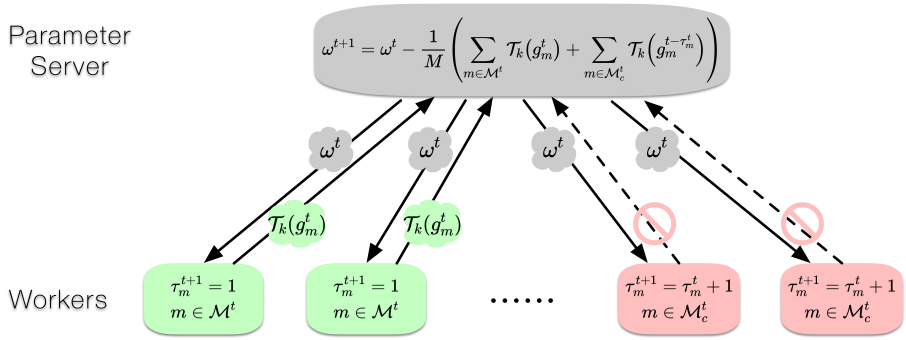


Fig. 1: SASG overview. At the t-th iteration, the parameter server broadcasts ω^t to all workers, and workers in $\mathcal{M}^t$ (marked in green) will upload the spar- sified gradient information $\mathcal{T}_k(g_m^t)$ and reset $\tau_m^{t+1} = 1$; while workers in $\mathcal{M}_c^t$ (marked in pink) will increase the staleness by $\tau_m^{t+1} = \tau_m^t + 1$ and upload nothing; then the server updates parameter ω via (8).

and challenges for developing this method, followed by the implementation details of the proposed algorithm.

A. Motivation and challenges

In the distributed learning system, an important finding is that not all communication rounds between the server and workers contribute equally to the state of the global model [15]–[17], [38]. Then one can design an aggregation rule to skip the inefficient communication rounds. LAG has developed an adaptive selection rule that can detect workers with slowly-varying gradients and triggers the reuse of outdated gradients to improve communication efficiency. However, this technique only reduces the number of communication rounds and requires additional memory overhead on the server side, which is far from enough in resource-limited scenarios. Actually, we can do more beyond LAG, e.g., further reduce the transmitted bits of uploaded information rather than just the number of rounds, and reduce the storage overhead on the server side.

There are two lines of transmission bit reduction: quantization and sparsification. We employ the latter as quantization methods can only reach a compression rate up to 32× (using 1-bit quantization method) in the commonly used single-precision floating-point algorithms. When the dimension d of the model is large, i.e., $d \gg 32$, the sparsification method will far outperform the quantization approach, where the maximum compression rate of $d\times$ can be achieved.

While random sparsification methods guarantee the unbiasedness of the compression operator and facilitate theoretical analysis, the top-k sparsification method tends to achieve better practical results [14], [31]. We will apply the top-k sparsification operator, denoted by $\mathcal{T}_k(\cdot)$. This operator retains only the largest k components (in absolute value) of the gradient vector, and the specific definition is as follows.

***Definition** 1:* For a parameter $1 \le k < d$, the sparsification operator $\mathcal{T}_k(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$, is defined for $\mathbf{x} \in \mathbb{R}^d$ as

$$(\mathcal{T}_k(\mathbf{x}))_{\pi(i)} := \begin{cases} (\mathbf{x})_{\pi(i)}, & \text{if } i \le k, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

where π is a permutation of set $[d]$ such that $|(\mathbf{x})_{\pi(i)}| \ge |(\mathbf{x})_{\pi(i+1)}|$ for $i = 1, \ldots, d-1$.

Nevertheless, these motivations encounter two significant challenges: 1). How to design an effective communication criterion in the sparsification setting? That is, in this composite situation, we need to determine which communications are non-essential exactly. 2). How can we guarantee model convergence when updating the parameters with only sparsified information from a subset of workers? In particular, the only partial information we have is still biased. Our proposed approach will address these challenges.

B. SASG method

To tackle the first challenge, we need to determine an effective selection criterion. Intuitively, if the difference between two consecutive gradients from a worker is small, it is safe to skip the redundant uploads and reuse the previous one in the server. The difference is defined as

$$\Delta_m^t := \nabla f_m(\omega^t; \xi_m^t) - \nabla f_m(\omega^{t-\tau_m^t}; \xi_m^{t-\tau_m^t}), \tag{4}$$

where τ_m^t is the delay count. Additionally, we need to pick a threshold that varies with iteration t to adaptively measure the magnitude of the difference Δ_m^t. Following the idea of LAG, we have

$$\Delta_m^t \le \frac{1}{M^2} \sum_{d=1}^{D} \alpha_d \left\|\omega^{t+1-d} - \omega^{t-d}\right\|^2, \tag{5}$$

where $\{\alpha_d \ge 0\}_{d=1}^D$ are constant weights. This rule is a direct extension of LAG to the stochastic settings. Unfortunately, (5) is ineffective due to the non-diminishing variance of the stochastic gradients (4). Specifically, the variance introduced by the randomly selected samples ξ_m^t and $\xi_m^{t-\tau_m^t}$ in the two iterations makes the value of Δ_m^t almost never been small, rendering criterion (5) less effective. Our SASG method takes advantage of the adaptive selection rule of LASG [17], formulated as

$$\begin{aligned} &\left\|\nabla f_m(\omega^t; \xi_m^t) - \nabla f_m(\omega^{t-\tau_m^t}; \xi_m^t)\right\|^2 \\ &\le \sum_{d=1}^{D} \frac{\alpha_d}{M^2} \left\|\omega^{t+1-d} - \omega^{t-d}\right\|^2. \end{aligned} \tag{6}$$

Algorithm 1 SASG algorithm.

Initialize: Errors $e_m^0 = 0$, delay counters $\tau_m^0 = 1$, $\forall m \in \mathcal{M}$

Input: Learning rate $\gamma > 0$, maximum delay D, Constant weights $\{\alpha_d\}_{d=1}^D$

for $t = 0, 1, \ldots, T$ **do**
  Server broadcasts ω^t to all workers.
  for worker $m = 1, \ldots, M$ **do**
    Compute $g_m^t = \gamma \nabla f_m(\omega^t; \xi_m^t) + e_m^t$.
    Divide the workers into $\mathcal{M}^t$ and $\mathcal{M}_c^t$ according to the selection rule (6).
    if worker $m \in \mathcal{M}^t$ or $\tau_m^t \geq D$ **then**
      Worker m uploads $\mathcal{T}_k(g_m^t)$ to the server.
      Set $e_m^{t+1} = g_m^t - \mathcal{T}_k(g_m^t), \quad \tau_m^{t+1} = 1$.
    else
      Worker m uploads nothing.
      Set $e_m^{t+1} = e_m^t, \quad \tau_m^{t+1} = \tau_m^t + 1$.
    end if
  end for
  Server updates parameter ω according to (8).
end for

This condition is evaluated on the same data ξ_m^t at two different iterations t and $t - \tau_m^t$. In the following, we provide a novel insight to demonstrate the suitability of the criterion for our method. According to the Lipschitz continuous property of ∇f_m,

$$\begin{aligned} &\left\| \nabla f_m(\omega^t; \xi_m^t) - \nabla f_m(\omega^{t-\tau_m^t}; \xi_m^t) \right\|^2 \\ &\leq L_m \left\| \omega^t - \omega^{t-\tau_m^t} \right\|^2, \end{aligned} \tag{7}$$

where L_m is the Lipschitz constant. As the iterative sequence $\{\omega^t\}_{t=0,1,\ldots}$ converges, the right-hand side of (7) diminishes, and thus the left-hand side of (6) diminishes, which eliminates the inherent variance caused by stochastic data ξ_m^t and $\xi_m^{t-\tau_m^t}$.

At each training iteration t, selection rule (6) divides the worker set $\mathcal{M}$ into two disjoint sets, $\mathcal{M}^t$ and $\mathcal{M}_c^t$. The parameter server only needs to receive the new gradient information from $\mathcal{M}^t$ and reuse the outdated gradients (stored on the server side) from $\mathcal{M}_c^t$, which scale down the per-iteration communication rounds from M to $|\mathcal{M}^t|$. After the adaptive selection procedure, selected workers will send the sparse information derived by the top-k operator to the parameter server, which reduces the communication bits and the memory overhead on the server side. However, the biased nature of the top-k operator breaks the algorithmic convergence, exacerbating challenge two.

To address this issue, we utilize error feedback techniques (also known as error accumulation or memory) [14], [32] in our SASG algorithm. Specifically, the SASG algorithm first sparsifies the gradient information g to obtain $\mathcal{T}_k(g)$. When uploading the sparsified information $\mathcal{T}_k(g)$, we calculate the compression error $e = g - \mathcal{T}_k(g)$ and store it. The next iteration involves adding the saved error e to the gradient information and compressing them jointly. Eventually, all the gradient information will be transmitted, despite the delay caused by error feedback. We defined an auxiliary sequence $\{\nu^t\}_{t=0,1,\ldots}$, which can be regarded as an error approximation of $\{\omega^t\}_{t=0,1,\ldots}$. With the recursive properties of this sequence, we prove that SASG is convergent in Section IV.

In summary, the iterative scheme of the SASG algorithm can be expressed as

$$\omega^{t+1} = \omega^t - \frac{1}{M} \left[\sum_{m \in \mathcal{M}^t} \mathcal{T}_k(g_m^t) + \sum_{m \in \mathcal{M}_c^t} \mathcal{T}_k(g_m^{t-\tau_m^t}) \right], \tag{8}$$

where

$$g_m^t := \gamma \nabla f_m(\omega^t; \xi_m^t) + e_m^t, \quad e_m^{t+1} := g_m^t - \mathcal{T}_k(g_m^t). \tag{9}$$

Here $\mathcal{M}^t$ and $\mathcal{M}_c^t$ denote the sets of workers that do and do not communicate with the server at the t-th iteration, respectively. Staleness τ_m^t is determined by the selection of the subset $\mathcal{M}^t$ at the t-th iteration. For worker $m \in \mathcal{M}^t$, the server resets $\tau_m^{t+1} = 1$, and the worker uploads the local gradient information. Otherwise, the server increases staleness by $\tau_m^{t+1} = \tau_m^t + 1$, and worker m uploads nothing. SASG is illustrated in Figure 1 and summarized in Algorithm 1. Specifically, during each iteration $t = 0, 1, 2, \ldots$,

i. server broadcasts the learning parameter ω^t to all workers;
ii. worker m calculates the local gradient $\nabla f_m(\omega^t; \xi_m^t)$ and an auxiliary gradient $\nabla f_m(\omega^{t-\tau_m^t}; \xi_m^t)$ for computing the selection rule (6);
iii. worker in $\mathcal{M}^t$ selected by condition (6) will sparsify the local information $g_m^t = \gamma \nabla f_m(\omega^t; \xi_m^t) + e_m^t$ and upload $\mathcal{T}_k(g_m^t)$ to the server;
iv. server aggregates the fresh sparsified gradients $\mathcal{T}_k(g_m^t)$ from the selected workers $\mathcal{M}^t$ and the outdated gradient information $\mathcal{T}_k(g_m^{t-\tau_m^t})$ (stored in the server) from $\mathcal{M}_c^t$ to update the parameter ω via (8).

As shown in Table I, we only need to upload the sparse gradient information of the selected worker, thus significantly saving the communication overhead ($\mathcal{M}^t$ selected in SASG and LASG is not the same due to different update information). Furthermore, SASG stores the sparsified gradient on the server side, which reduces the memory overhead of traditional adaptive aggregation methods.

IV. Theoretical analysis

This section presents a detailed theoretical analysis of the SASG algorithm. In contrast to LASG, the SASG algorithm employs a biased top-k sparsification operator, and the introduced compression error complicates the analysis. To tackle this challenge, we first prove that the residuals of the compression operator are bounded and define a crucial auxiliary variable ν^t to couple the model parameters ω^t with the compression error e_m^t. After that, we introduce a Lyapunov function with respect to the auxiliary variable ν^t, which is also compatible with the selection criterion (6). The descent lemma on the Lyapunov function derives the convergence theorem of the SASG algorithm, and its convergence rate matches that of

TABLE I: The number of communication rounds per iteration, communication bits per upload, and the total communication overhead to complete T iterations for each algorithm when training the d-dimensional parametric model with M distributed workers (k is the sparsification level defined in (3), which satisfies $k < d$).

Method	# Round	# Bit	Total Overhead
SGD	M	$32d$	$32dMT$
Sparse	M	$32k$	$32kMT$
LASG	$\|\mathcal{M}^t\|$	$32d$	$32d \cdot \sum_{t=1}^{T} \|\mathcal{M}^t\|$
SASG	$\|\mathcal{M}^t\|$	$32k$	$32k \cdot \sum_{t=1}^{T} \|\mathcal{M}^t\|$

the original SGD. All proof details are included in the Proofs section V.

A. Assumptions

Our analysis is based on the following assumptions, which are standard in analyzing SGD and its variants. Functions f_m, F_m and F have been specified in Section I.

***Assumption** 1 (Smoothness):* The loss function $f_m : \mathbb{R}^d \to \mathbb{R}$ is L_m-smooth, and function $F : \mathbb{R}^d \to \mathbb{R}$ is L-smooth, i.e., $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,

$$\begin{aligned} \|\nabla f_m(\mathbf{y}) - \nabla f_m(\mathbf{x})\| &\le L_m \|\mathbf{y} - \mathbf{x}\|, \\ \|\nabla F(\mathbf{y}) - \nabla F(\mathbf{x})\| &\le L \|\mathbf{y} - \mathbf{x}\|. \end{aligned} \tag{10}$$

***Assumption** 2 (Moment boundedness):* The data sampled in each iteration, i.e., $\xi_m^1, \xi_m^2, \ldots$ are independent, and the stochastic gradient $\nabla f_m(\omega; \xi_m^t)$ satisfies

$$\begin{aligned} &\mathbb{E}[\nabla f_m(\omega; \xi_m^t)] = \nabla F_m(\omega), \\ &\mathbb{E}\left[\|\nabla f_m(\omega; \xi_m^t) - \nabla F_m(\omega)\|^2\right] \le \sigma_m^2. \end{aligned} \tag{11}$$

***Assumption** 3 (Gradient boundedness):* For a given parameter $\omega \in \mathbb{R}^d$, the local gradient $\nabla f_m(\omega; \cdot)$ is bounded, i.e., there exists a constant $B_m \in \mathbb{R}$ such that

$$\mathbb{E}\left[\|\nabla f_m(\omega; \cdot)\|\right] \le B_m. \tag{12}$$

B. Technical lemmas

We start our analysis with an important property of the top-k sparsification operator $\mathcal{T}_k(\cdot)$.

***Lemma** 1:* The operator $\mathcal{T}_k(\cdot)$ is a δ-approximate compressor, i.e., there exists a constant $\delta \in (0, 1)$ such that

$$\|\mathcal{T}_k(\mathbf{x}) - \mathbf{x}\|^2 \le (1 - \delta)\|\mathbf{x}\|^2, \forall \mathbf{x} \in \mathbb{R}^d. \tag{13}$$

With the gradient boundedness assumption, we are prepared to present a critical lemma, which demonstrates how to bound the residual error in Algorithm 1.

***Lemma** 2:* Under Assumption 3, at any iteration t of the SASG algorithm, the residual error in worker m, i.e., e_m^t, is bounded:

$$\mathbb{E}\left[\|e_m^t\|^2\right] \le \frac{4(1-\delta)}{\delta^2} \gamma^2 B_m^2. \tag{14}$$

Lemma 2 illustrates that the residual errors maintained in Algorithm 1 do not accumulate too much. In the following, we define a crucial auxiliary sequence for the analysis.

***Definition** 2:* Let ω^t be the model generated by running SASG for t iterations, and the auxiliary variable ν^t is defined as

$$\nu^t := \omega^t - \frac{1}{M} \sum_{m \in \mathcal{M}} e_m^t. \tag{15}$$

The sequence $\{\nu^t\}_{t=0,1,\ldots}$, which can be considered as an error-corrected sequence of $\{\omega^t\}_{t=0,1,\ldots}$, has the following property (Δ_m^t is defined in (4))

$$\nu^{t+1} - \nu^t = -\frac{\gamma}{M} \sum_{m \in \mathcal{M}} \nabla f_m(\omega^t; \xi_m^t) + \frac{\gamma}{M} \sum_{m \in \mathcal{M}_c^t} \Delta_m^t, \tag{16}$$

Based on this property, we provide the following descent lemma for the loss function with respect to the auxiliary variable ν^t.

***Lemma** 3:* Under Assumptions 1 and 2, the sequence $\{\nu^t\}_{t=0,1,\ldots}$ is defined in (15), then the objective function value satisfies (with $\sigma^2 := \sum_{m=1}^M \sigma_m^2$)

$$\begin{aligned} \mathbb{E}[F(\nu^{t+1})] - \mathbb{E}[F(\nu^t)] \le &-(\gamma - \frac{5L+4}{2}\gamma^2)\mathbb{E}\left[\|\nabla F(\omega^t)\|^2\right] \\ &+ \sum_{d=1}^{D} \left[\frac{\alpha_d}{2M^2 L} + \frac{L}{2} + \frac{2\gamma^2(L+1)\alpha_d}{M^2}\right] \|\omega^{t+1-d} - \omega^{t-d}\|^2 \\ &+ \frac{L^2}{2M^2}\mathbb{E}\left[\Big\| \sum_{m \in \mathcal{M}} e_m^t \Big\|^2\right] + [10(L+1) + 2DL]\frac{\gamma^2\sigma^2}{M}. \end{aligned} \tag{17}$$

It is worth noting that all terms on the right-hand side of inequality (17) exist in SGD analysis except $\|\omega^{t+1-d} - \omega^{t-d}\|^2$ and $\|\sum_{m \in \mathcal{M}} e_m^t\|^2$, which arise due to stale information and compression errors. To deal with these terms, we will introduce an associated Lyapunov function. With F^* denoting the optimal value of problem (1), the Lyapunov function is defined as

$$\mathcal{L}^t := \mathbb{E}[F(\nu^t)] - F^* + \sum_{d=1}^{D} \beta_d \|\omega^{t+1-d} - \omega^{t-d}\|^2, \tag{18}$$

where $\{\beta_d \ge 0\}_{d=1}^D$ are constants that will be determined later. The Lyapunov function is coupled with the selection rule (6) that contains the parameter difference terms. We also highlight that in the definition (18), $\mathcal{L}^t > 0$ for any $t \in \mathbb{N}$. A direct extension of Lemma 3 gives the following descent lemma.

***Lemma** 4:* Let Assumptions 1-3 hold, and denote $\sigma^2 := \sum_{m=1}^M \sigma_m^2$, $B^2 := \sum_{m=1}^M B_m^2$. If the learning rate γ and constant weights $\{\alpha_d\}_{d=1}^D$ are chosen properly, the Lyapunov function satisfies

$$\begin{aligned} \mathcal{L}^{t+1} - \mathcal{L}^t \le &- c_f \mathbb{E}[\|\nabla F(\omega^t)\|^2] + a\frac{\gamma^2\sigma^2}{M} + b\frac{\gamma^2 B^2}{M} \\ &- \sum_{d=1}^{D} c_d \|\omega^{t+1-d} - \omega^{t-d}\|^2, \end{aligned} \tag{19}$$

where $c_f, a, b, c_1, \ldots, c_D \ge 0$ depend on the learning rate γ, constants D, L, and $\{\alpha_d, \beta_d\}_{d=1}^D$. More specific explanatory notes can be found in the proof details.

C. Main results

In conjunction with the lemmas introduced above, we are ready to present the main convergence results of our SASG algorithm.

Theorem 1: Under Assumptions 1, 2, and 3, let the sequence $\{\omega^t\}_{t=0,1,\dots}$ be generated by the SASG algorithm; if constant weights $\{\alpha_d\}_{d=1}^D$ are selected properly, and the learning rate is chosen as $\gamma = \min\{1/(5L+4+16\beta_1), c_\gamma/\sqrt{T}\}$, where $c_\gamma > 0$ is a constant and β_1 is defined in (18). We then have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla F(\omega^t)\|^2\right] \le \frac{2(5L+4+16\beta_1)(F(\omega^0)-F^*)}{T} + \frac{2(F(\omega^0)-F^*)}{c_\gamma\sqrt{T}} + \frac{2c_\gamma(a\sigma^2+bB^2)}{M\sqrt{T}}. \tag{20}$$

Theorem 1 shows that our algorithm is guaranteed to converge and achieve a sublinear convergence rate $\mathcal{O}(1/\sqrt{T})$ despite skipping many communication rounds and performing communication compression. In other words, the SASG algorithm using a well-designed adaptive aggregation rule and sparse communication techniques can still achieve an order of convergence rate identical to the SGD method. On the other hand, the convergence result of the algorithm is controlled by the average variance of the distributed system and is not effected by the number of workers, making the SASG algorithm scales well to large-scale distributed systems.

V. Proofs

This section contains proof details of Lemmas and Theorem in Section IV, and all the theoretical proofs are based on the standard Assumptions 1, 2, and 3.

A. Proof of Lemma 1

Given a vector $\mathbf{x} \in \mathbb{R}^d$, from Definition 1, we known that only the first k largest coordinates of the vector are retained. That is

$$\|\mathcal{T}_k(\mathbf{x})\|^2 \ge \frac{k}{d}\|\mathbf{x}\|^2.$$

Due to that $\langle \mathcal{T}_k(\mathbf{x}) - \mathbf{x}, \mathcal{T}_k(\mathbf{x})\rangle = 0$,

$$\|\mathcal{T}_k(\mathbf{x}) - \mathbf{x}\|^2 = \|\mathbf{x}\|^2 - \|\mathcal{T}_k(\mathbf{x})\|^2.$$

With $\delta := k/d \in (0,1)$, we further have that

$$\|\mathcal{T}_k(\mathbf{x}) - \mathbf{x}\|^2 \le (1-\delta)\|\mathbf{x}\|^2.$$

■

B. Proof of Lemma 2

From Algorithm 1 and Lemma 1, we can derive that

$$\begin{aligned}
&\mathbb{E}[\|e_m^{t+1}\|^2] = \mathbb{E}[\|g_m^t - \mathcal{T}_k(g_m^t)\|^2] \le (1-\delta)\mathbb{E}[\|g_m^t\|^2] \\
&= (1-\delta)\mathbb{E}\left[\|\gamma\nabla f_m(\omega^t,\xi_m^t) + e_m^t\|^2\right] \\
&\le (1-\delta)\Big((1+\eta)\mathbb{E}\|e_m^t\|^2 + (1+\frac{1}{\eta})\gamma^2\mathbb{E}\|\nabla f_m(\omega^t,\xi_m^t)\|^2\Big)
\end{aligned}$$

where we used the Young's inequality $\|a+b\|^2 \le (1+\eta)\|a\|^2 + (1+1/\eta)\|b\|^2$ (for any $\eta > 0$). From Algorithm 1 and Assumption 3, we have that $e_m^0 = 0$ and $\nabla f_m(\omega^t,\xi_m^t)$ is bounded. Simple computation yields

$$\begin{aligned}
&\mathbb{E}[\|e_m^{t+1}\|^2] \\
&\le \sum_{l=0}^{t}\left[(1-\delta)(1+\eta)\right]^{t-l}(1-\delta)(1+\frac{1}{\eta})\gamma^2\mathbb{E}\|\nabla f_m(\omega^l,\xi_m^l)\|^2 \\
&\le \sum_{l=0}^{\infty}\left[(1-\delta)(1+\eta)\right]^{l}(1-\delta)(1+\frac{1}{\eta})\gamma^2 B_m^2 \\
&= \frac{(1-\delta)(1+1/\eta)}{1-(1-\delta)(1+\eta)}\gamma^2 B_m^2.
\end{aligned}$$

Let us pick $\eta := \frac{\delta}{2(1-\delta)}$ such that $1+1/\eta = (2-\delta)/\delta \le 2/\delta$ Then we have

$$\mathbb{E}[\|e_m^{t+1}\|^2] \le \frac{2(1-\delta)(1+1/\eta)}{\delta}\gamma^2 B_m^2 \le \frac{4(1-\delta)}{\delta^2}\gamma^2 B_m^2.$$

■

C. Proof of Equation (16)

Following Definition 2 and the iterative scheme (8), we have

$$\begin{aligned}
\nu^{t+1} &= \omega^{t+1} - \frac{1}{M}\sum_{m\in\mathcal{M}} e_m^{t+1} \\
&= \omega^t - \frac{1}{M}\left[\sum_{m\in\mathcal{M}^t}\mathcal{T}_k(g_m^t) + \sum_{m\in\mathcal{M}_c^t}\mathcal{T}_k(g_m^{t-\tau_m^t}) + \sum_{m\in\mathcal{M}} e_m^{t+1}\right] \\
&\overset{*}{=} \omega^t - \frac{1}{M}\left[\sum_{m\in\mathcal{M}^t} g_m^t + \sum_{m\in\mathcal{M}_c^t} g_m^{t-\tau_m^t}\right] \\
&= \omega^t - \frac{1}{M}\sum_{m\in\mathcal{M}^t}\left(\gamma\nabla f_m(\omega^t;\xi_m^t) + e_m^t\right) \\
&\quad - \frac{1}{M}\sum_{m\in\mathcal{M}_c^t}\left(\gamma\nabla f_m(\omega^{t-\tau_m^t};\xi_m^{t-\tau_m^t}) + e_m^{t-\tau_m^t}\right) \\
&\overset{*}{=} \nu^t - \frac{\gamma}{M}\sum_{m\in\mathcal{M}}\nabla f_m(\omega^t;\xi_m^t) \\
&\quad + \frac{\gamma}{M}\sum_{m\in\mathcal{M}_c^t}\left(\nabla f_m(\omega^t;\xi_m^t) - \nabla f_m(\omega^{t-\tau_m^t};\xi_m^{t-\tau_m^t})\right),
\end{aligned}$$

where $(*)$ used $e_m^{t+1} = e_m^{t-\tau_m^t}, \forall m \in \mathcal{M}_c^t$. Let

$$\Delta_m^t := \nabla f_m(\omega^t;\xi_m^t) - \nabla f_m(\omega^{t-\tau_m^t};\xi_m^{t-\tau_m^t}),$$

then we have that

$$\nu^{t+1} - \nu^t = -\frac{\gamma}{M}\sum_{m\in\mathcal{M}}\nabla f_m(\omega^t;\xi_m^t) + \frac{\gamma}{M}\sum_{m\in\mathcal{M}_c^t}\Delta_m^t. \tag{21}$$

■

D. Proof of Lemma 3

We first give the whole analysis process and then explain the relevant details.

$$\begin{aligned}
&\mathbb{E}\left[F(\nu^{t+1})\right] - \mathbb{E}\left[F(\nu^t)\right] \\
&\overset{(a)}{\le} \mathbb{E}\left\langle\nabla F(\nu^t), \nu^{t+1}-\nu^t\right\rangle + \frac{L}{2}\mathbb{E}\|\nu^{t+1}-\nu^t\|^2
\end{aligned}$$

$$
\begin{aligned}
&\overset{(b)}{\leq} \mathbb{E}\left\langle \nabla F(\omega^t), -\frac{\gamma}{M}\sum_{m\in\mathcal{M}}\nabla f_m(\omega^t;\xi_m^t)+\frac{\gamma}{M}\sum_{m\in\mathcal{M}_c^t}\Delta_m^t\right\rangle \\
&\quad + \mathbb{E}\left\langle \nabla F(\nu^t)-\nabla F(\omega^t), \nu^{t+1}-\nu^t\right\rangle + \frac{L}{2}\mathbb{E}\|\nu^{t+1}-\nu^t\|^2 \\
&\overset{(c)}{\leq} -\gamma\mathbb{E}\|\nabla F(\omega^t)\|^2 + \frac{\gamma}{M}\mathbb{E}\left\langle \nabla F(\omega^t), \sum_{m\in\mathcal{M}_c^t}\Delta_m^t\right\rangle \\
&\quad + \frac{1}{2}\mathbb{E}\|\nabla F(\nu^t)-\nabla F(\omega^t)\|^2 + \frac{L+1}{2}\mathbb{E}\|\nu^{t+1}-\nu^t\|^2 \\
&\overset{(d)}{\leq} -(\gamma-\frac{L\gamma^2}{2})\mathbb{E}\|\nabla F(\omega^t)\|^2 + \frac{L+1}{2}\mathbb{E}\|\nu^{t+1}-\nu^t\|^2 \\
&\quad + \sum_{d=1}^{D}\left[\frac{\alpha_d}{2M^2L}+\frac{L}{2}\right]\|\omega^{t+1-d}-\omega^{t-d}\|^2 \\
&\quad + \frac{L^2}{2}\mathbb{E}\left[\|\nu^t-\omega^t\|^2\right] + \frac{2\gamma^2DL\sigma^2}{M} \\
&\overset{(e)}{\leq} -(\gamma-\frac{5L+4}{2}\gamma^2)\mathbb{E}\left[\|\nabla F(\omega^t)\|^2\right] \\
&\quad + \sum_{d=1}^{D}\left[\frac{\alpha_d}{2M^2L}+\frac{L}{2}+\frac{2\gamma^2(L+1)\alpha_d}{M^2}\right]\|\omega^{t+1-d}-\omega^{t-d}\|^2 \\
&\quad + \frac{L^2}{2M^2}\mathbb{E}\left[\Big\|\sum_{m\in\mathcal{M}}e_m^t\Big\|^2\right] + [10(L+1)+2DL]\frac{\gamma^2\sigma^2}{M},
\end{aligned}
\tag{22}
$$

where (a) uses the L-smooth property in Assumption 1, i.e.,$\forall \mathbf{x},\mathbf{y}\in\mathbb{R}^d$,

$$F(\mathbf{y})-F(\mathbf{x})\leq\langle\nabla F(\mathbf{x}),\mathbf{y}-\mathbf{x}\rangle+\frac{L}{2}\|\mathbf{y}-\mathbf{x}\|^2.$$

(b) is obtained by equation (21). (c) uses Assumption 2 and the inequality $\|a\|^2+\|b\|^2\geq 2\langle a,b\rangle$. In order to get (d), we need to bound $\langle\nabla F(\omega^t),\Delta_m^t\rangle$ as follows

$$
\begin{aligned}
&\mathbb{E}\left\langle\nabla F(\omega^t),\Delta_m^t\right\rangle \\
&=\underbrace{\mathbb{E}\left\langle\nabla F(\omega^t),\nabla f_m(\omega^t;\xi_m^t)-\nabla f_m(\omega^{t-\tau_m^t};\xi_m^t)\right\rangle}_{I} \\
&\quad+\underbrace{\mathbb{E}\left\langle\nabla F(\omega^t),\nabla f_m(\omega^{t-\tau_m^t};\xi_m^t)-\nabla f_m(\omega^{t-\tau_m^t};\xi_m^{t-\tau_m^t})\right\rangle}_{II}.
\end{aligned}
$$

Using the inequality $\langle a,b\rangle\leq\frac{\varepsilon}{2}\|a\|^2+\frac{1}{2\varepsilon}\|b\|^2$ (with $\varepsilon:=L\gamma$) and selection rule (6), we have

$$I\leq\frac{L\gamma}{2}\mathbb{E}[\|\nabla F(\omega^t)\|^2]+\frac{1}{2L\gamma}\frac{1}{M^2}\sum_{d=1}^{D}\alpha_d\|\omega^{t+1-d}-\omega^{t-d}\|^2.$$

Noticed that

$$
\begin{aligned}
&\mathbb{E}\left\langle\nabla F(\omega^{t-\tau_m^t}),\nabla f_m(\omega^{t-\tau_m^t};\xi_m^t)-\nabla f_m(\omega^{t-\tau_m^t};\xi_m^{t-\tau_m^t})\right\rangle \\
&=0,
\end{aligned}
$$

we have

$$
\begin{aligned}
II=\mathbb{E}\Big\langle&\nabla F(\omega^t)-\nabla F(\omega^{t-\tau_m^t}), \\
&\nabla f_m(\omega^{t-\tau_m^t};\xi_m^t)-\nabla f_m(\omega^{t-\tau_m^t};\xi_m^{t-\tau_m^t})\Big\rangle \\
\leq L\mathbb{E}\Big\langle&\omega^t-\omega^{t-\tau_m^t},\nabla f_m(\omega^{t-\tau_m^t};\xi_m^t)-\nabla f_m(\omega^{t-\tau_m^t};\xi_m^{t-\tau_m^t})\Big\rangle \\
\leq\frac{\gamma DL}{2}&\mathbb{E}\left[\|\nabla f_m(\omega^{t-\tau_m^t};\xi_m^t)-\nabla f_m(\omega^{t-\tau_m^t};\xi_m^{t-\tau_m^t})\|^2\right] \\
&+\frac{L}{2\gamma D}\|\omega^t-\omega^{t-\tau_m^t}\|^2 \\
\leq 2\gamma DL\sigma_m^2&+\frac{L}{2\gamma}\sum_{d=1}^{D}\|\omega^{t+1-d}-\omega^{t-d}\|^2,
\end{aligned}
$$

where we uses the inequality $\langle a,b\rangle\leq\frac{\varepsilon}{2}\|a\|^2+\frac{1}{2\varepsilon}\|b\|^2$ (with $\varepsilon:=D\gamma$), $\|\sum_{i=1}^n\theta_i\|^2\leq n\sum_{i=1}^n\|\theta_i\|^2$, and Assumption 2. That is (with $\sigma^2:=\sum_{m=1}^M\sigma_m^2$)

$$
\begin{aligned}
&\frac{\gamma}{M}\mathbb{E}\left\langle\nabla F(\omega^t),\sum_{m\in\mathcal{M}_c^t}\Delta_m^t\right\rangle\leq\frac{\gamma}{M}\sum_{m\in\mathcal{M}_c^t}(I+II) \\
&\leq\frac{L\gamma^2}{2}\mathbb{E}[\|\nabla F(\omega^t)\|^2]+\sum_{d=1}^{D}(\frac{\alpha_d}{2M^2L}+\frac{L}{2})\|\omega^{t+1-d}-\omega^{t-d}\|^2 \\
&\quad+\frac{2\gamma^2DL\sigma^2}{M}.
\end{aligned}
$$

Further analyzing $\|\Delta_m^t\|^2$ and $\|\nu^{t+1}-\nu^t\|^2$, we can get (e)

$$
\begin{aligned}
&\|\sum_{m\in\mathcal{M}_c^t}\Delta_m^t\|^2\leq|\mathcal{M}_c^t|\sum_{m\in\mathcal{M}_c^t}\|\Delta_m^t\|^2 \\
&\leq|\mathcal{M}_c^t|\sum_{m\in\mathcal{M}_c^t}\Big\|\nabla f_m(\omega^t;\xi_m^t)-\nabla f_m(\omega^{t-\tau_m^t};\xi_m^t) \\
&\quad+\nabla f_m(\omega^{t-\tau_m^t};\xi_m^t)-\nabla f_m(\omega^{t-\tau_m^t};\xi_m^{t-\tau_m^t})\Big\|^2 \\
&\leq 2M\sum_{m\in\mathcal{M}_c^t}\|\nabla f_m(\omega^t;\xi_m^t)-\nabla f_m(\omega^{t-\tau_m^t};\xi_m^t)\|^2 \\
&\quad+2M\sum_{m\in\mathcal{M}_c^t}\|\nabla f_m(\omega^{t-\tau_m^t};\xi_m^t)-\nabla f_m(\omega^{t-\tau_m^t};\xi_m^{t-\tau_m^t})\|^2 \\
&\leq 2\sum_{d=1}^{D}\alpha_d\left\|\omega^{t+1-d}-\omega^{t-d}\right\|^2+8M\sigma^2,
\end{aligned}
\tag{23}
$$

where $\sigma^2:=\sum_{m=1}^M\sigma_m^2$. Then, using (21), (23) and Assumption 2, we can get

$$
\begin{aligned}
&\mathbb{E}[\|\nu^{t+1}-\nu^t\|]^2 \\
&=\mathbb{E}[\|-\frac{\gamma}{M}\sum_{m\in\mathcal{M}}\nabla f_m(\omega^t;\xi_m^t)+\frac{\gamma}{M}\sum_{m\in\mathcal{M}_c^t}\Delta_m^t\|^2] \\
&\leq\frac{2\gamma^2}{M^2}\mathbb{E}[\|\sum_{m\in\mathcal{M}}\nabla f_m(\omega^t;\xi_m^t)\|^2]+\frac{2\gamma^2}{M^2}\mathbb{E}[\|\sum_{m\in\mathcal{M}_c^t}\Delta_m^t\|^2] \\
&\leq 4\gamma^2\mathbb{E}[\|\nabla F(\omega^t)\|^2]+\frac{4\gamma^2}{M^2}\sum_{d=1}^{D}\alpha_d\|\omega^{t+1-d}-\omega^{t-d}\|^2 \\
&\quad+\frac{20\gamma^2\sigma^2}{M}.
\end{aligned}
\tag{24}
$$

Organize and summarize these items, and we can get the final result (22). ■

E. Proof of Lemma 4

According to the definition (18) and Lemma 3, a direct calculation gives

$$\begin{aligned}
\mathcal{L}^{t+1}-\mathcal{L}^t &= \mathbb{E}[F(\nu^{t+1})]-\mathbb{E}[F(\nu^t)] \\
&\quad+\sum_{d=1}^{D}\beta_d\|\omega^{t+2-d}-\omega^{t+1-d}\|^2-\sum_{d=1}^{D}\beta_d\|\omega^{t+1-d}-\omega^{t-d}\|^2 \\
&\le -(\gamma-\frac{5L+4}{2}\gamma^2)\mathbb{E}\|\nabla F(\omega^t)\|^2+\frac{L^2}{2M^2}\mathbb{E}\|\sum_{m\in\mathcal{M}}e_m^t\|^2 \\
&\quad+\sum_{d=1}^{D}\left[\frac{\alpha_d}{2M^2L}+\frac{L}{2}+\frac{2\gamma^2(L+1)\alpha_d}{M^2}\right]\|\omega^{t+1-d}-\omega^{t-d}\|^2 \\
&\quad+[10(L+1)+2DL]\frac{\gamma^2\sigma^2}{M}+\beta_1\|\omega^{t+1}-\omega^t\|^2 \\
&\quad+\sum_{d=1}^{D-1}(\beta_{d+1}-\beta_d)\|\omega^{t+1-d}-\omega^{t-d}\|^2 \\
&\quad-\beta_D\|\omega^{t+1-D}-\omega^{t-D}\|^2.
\end{aligned}$$

Using the properties (21), (24), and Lemma 2, we can obtain the following inequality

$$\begin{aligned}
\|\omega^{t+1}-\omega^t\|^2 &\le 2\|\nu^{t+1}-\nu^t\|^2+\frac{2}{M^2}\|\sum_{m\in\mathcal{M}}(e_m^{t+1}-e_m^t)\|^2 \\
&\le 2\|\nu^{t+1}-\nu^t\|^2+\frac{16(1-\delta)\gamma^2B^2}{\delta^2M},
\end{aligned}$$

where $B^2:=\sum_{m=1}^{M}B_m^2$. We then derive that

$$\begin{aligned}
\mathcal{L}^{t+1}-\mathcal{L}^t \le &-\Big[\gamma-(\frac{5L+4}{2}+8\beta_1)\gamma^2\Big]\mathbb{E}[\|\nabla F(\omega^t)\|^2] \\
&-\sum_{d=1}^{D}c_d\|\omega^{t+1-d}-\omega^{t-d}\|^2+a\frac{\gamma^2\sigma^2}{M}+b\frac{\gamma^2B^2}{M},
\end{aligned}$$

where

$$\begin{cases}
c_f := \gamma-(\frac{5L+4}{2}+8\beta_1)\gamma^2 \\
c_d := \beta_d-\beta_{d+1}-\left(\frac{\alpha_d}{2M^2L}+\frac{L}{2}+\frac{2\gamma^2\alpha_d(L+1+4\beta_1)}{M^2}\right) \\
\qquad d=1,\dots,D-1 \\
c_D := \beta_D-\left(\frac{\alpha_D}{2M^2L}+\frac{L}{2}+\frac{2\gamma^2\alpha_D(L+1+4\beta_1)}{M^2}\right) \\
a := 10(L+1)+2DL+40\beta_1 \\
b := \frac{(2L^2+16\beta_1)(1-\delta)}{\delta^2}.
\end{cases} \tag{25}$$

■

F. Proof of Theorem 1

Let $\gamma\le\overline{\gamma}=\frac{1}{5L+4+16\beta_1}$, then choose $\{\beta_d\}_{d=1}^D$ such that

$$\begin{cases}
\beta_d-\beta_{d+1}-\left[\frac{\alpha_d}{2M^2L}+\frac{L}{2}+\frac{2\overline{\gamma}^2\alpha_d(L+1+4\beta_1)}{M^2}\right]=0 \\
\qquad d=1,\dots,D-1 \\
\beta_D-\left[\frac{\alpha_D}{2M^2L}+\frac{L}{2}+\frac{2\overline{\gamma}^2\alpha_D(L+1+4\beta_1)}{M^2}\right]=0.
\end{cases}$$

Solving the linear equations above we can further get

$$\beta_1=\frac{[\frac{2\overline{\gamma}^2(L+1)}{M^2}+\frac{1}{2M^2L}]\sum_{d=1}^{D}\alpha_d+\frac{LD}{2}}{1-\frac{8\overline{\gamma}^2}{M^2}\sum_{d=1}^{D}\alpha_d}.$$

We then have $c_d\ge 0, d=1,\dots,D$ and $c_f\ge\gamma/2$. That is

$$\mathcal{L}^{t+1}-\mathcal{L}^t\le-\frac{\gamma}{2}\mathbb{E}[\|\nabla F(\omega^t)\|^2]+\frac{a\sigma^2+bB^2}{M}\gamma^2.$$

By taking summation $\sum_{t=0}^{T-1}\mathcal{L}^{t+1}-\mathcal{L}^t$, we have

$$\begin{aligned}
\sum_{t=0}^{T-1}\frac{\gamma}{2}\mathbb{E}[\|\nabla F(\omega^t)\|^2] &\le \mathcal{L}^0-\mathcal{L}^T+\frac{a\sigma^2+bB^2}{M}T\gamma^2 \\
&\le F(\omega^0)-F^*+\frac{a\sigma^2+bB^2}{M}T\gamma^2.
\end{aligned}$$

If we choose the learning rate

$$\gamma=\min\left\{\frac{1}{5L+4+16\beta_1},\frac{c_\gamma}{\sqrt{T}}\right\},$$

where $c_\gamma>0$ is a constant. We then have

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla F(\omega^t)\|^2\right] \\
&\le\frac{2M(F(\omega^0)-F^*)+2T(a\sigma^2+bB^2)\gamma^2}{MT\gamma} \\
&\le\frac{2(5L+4+16\beta_1)(F(\omega^0)-F^*)}{T}+\frac{2(F(\omega^0)-F^*)}{c_\gamma\sqrt{T}} \\
&\quad+\frac{2c_\gamma(a\sigma^2+bB^2)}{M\sqrt{T}}.
\end{aligned}$$

That is

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla F(\omega^t)\|^2\right]=\mathcal{O}(1/\sqrt{T}).$$

■

VI. Experiment results

This section conducts extensive experiments to demonstrate the convergence properties and communication efficiency of our SASG algorithm.

A. Setup

We evaluate the performance of the SASG algorithm against LASG [17], the sparsification method with error feedback [11], and distributed SGD [8]. Aji et al. [11] demonstrated that up to 99% of the gradients are not needed for updating the model in each iteration, hence we utilize the top-1% (i.e., $k=0.01d$) sparsification operator in SASG and the sparsification method. All our experimental results are based on the PyTorch framework [39].

We begin by evaluating the communication volume of the four algorithms. We simulated ten workers, and each worker used ten samples per training iteration. Subsequently, we recorded the communication time for each algorithm in the distributed settings. We employed ten Nvidia RTX-3090 GPUs as distributed workers for training, with the first GPU

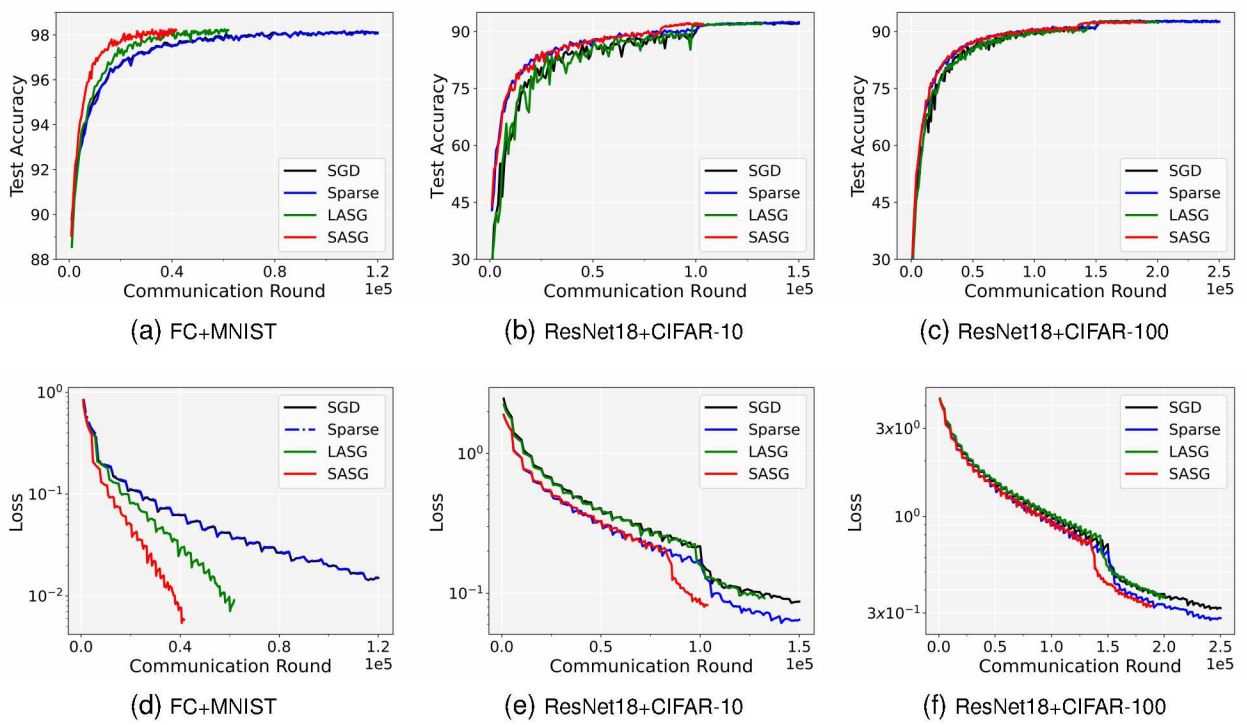


Fig. 2: Experimental results of test accuracy percentage and training loss versus communication round in three settings. All four methods are trained with the same number of epochs. Our algorithm significantly reduces the number of communication rounds required to achieve the same performance and complete the training.

also playing the role of parameter server. During information uploads from the workers to the server, we utilized point-to-point communication with the GLOO backend by invoking the send() and recv() functions in PyTorch. We configured the upload bandwidth of each worker to be 1 Gbps. We carried out evaluations for the following three settings, and repeated every experiment five times.

MNIST: The MNIST [40] dataset contains 70,000 handwritten digits in 10 classes, with 60,000 examples in the training set and 10,000 examples in the test set. We consider a two-layer fully connected (FC) neural network model with 512 neurons in the second layer for 10-category classification on MNIST. For all algorithms, we trained 20 epochs with the learning rate $\gamma = 0.005$. For the adaptive aggregated algorithms SASG and LASG, we set $D = 10, \alpha_d = 1/2\gamma$ for $d = 1, 2, \ldots, 10$.

CIFAR-10: The CIFAR-10 [41] dataset comprises 60,000 colored images in 10 classes, with 6,000 images per category. We test ResNet18 [42] with all the algorithms mentioned above on the CIFAR-10 dataset. Standard data augmentation techniques such as random cropping, flipping, and normalization are performed. We trained 30 epochs for the four algorithms, and the basic learning rate is set to $\gamma = 0.01$, with a learning rate decay of 0.1 at epoch 20. For SASG and LASG, we set $D = 10, \alpha_d = 1/\gamma$ for $d = 1, 2, \ldots, 10$.

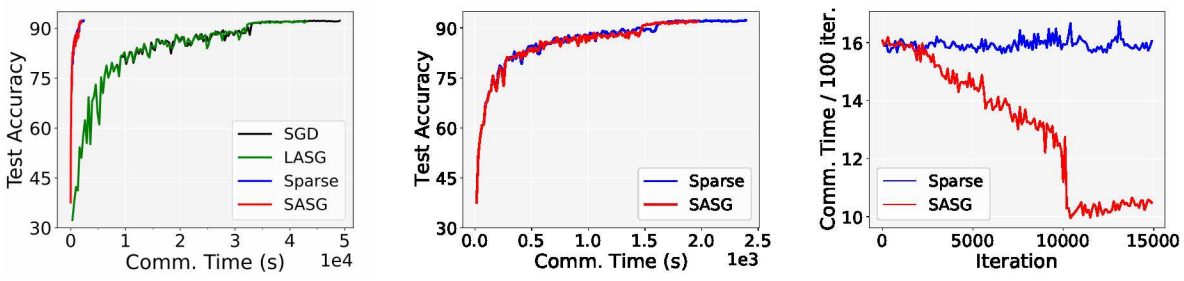
CIFAR-100: We also test ResNet18 on the CIFAR-100 [41] dataset, which consists of 60,000 images in 100 classes. A data augmentation technique similar to CIFAR-10 was performed. The basic learning rate is set to $\gamma = 0.01$, with a learning rate decay of 0.1 at epoch 30, where the total training epoch is 50. In the case of SASG and LASG, we set $D = 10, \alpha_d = 1/\gamma$ for $d = 1, 2, \ldots, 10$.

Note also that we did not use training techniques like warm-up as well as weight decay in our experiments, and thus state-of-the-art performance was not achieved in some models. Our goal is to demonstrate through comparative experiments that the SASG algorithm has the capability to reduce both the number of communication rounds and bits without sacrificing model performance.

B. Communication volume

This part presents the results of our simulation experiments on communication volume, i.e., the number of communication rounds and bits. Figure 2 depicts the variation of test accuracy and training loss against the number of communication rounds, with Figure 2c showing the top-5 test accuracy.

The experimental results demonstrate that our SASG algorithm outperforms the prior methods. For instance, from Figure 2a, SASG achieves a higher test accuracy with an equal number of communication rounds, indicating that our algorithm selects more valuable rounds for communication. Furthermore, SASG can considerably reduce the number of communication rounds while preserving the same model performance after completing the training process. Figure 2d shows that the SASG algorithm achieves faster and better convergence results with fewer communication rounds. In the experiments on the CIFAR-10 and CIFAR-100 datasets (Figure 2e-2f), SASG is



(a) Accuracy v.s. Time (b) Accuracy v.s. Time (c) Time v.s. Iteration

Fig. 3: Experimental results of classifying CIFAR-10 with ResNet18. Figures (a) and (b) show the test accuracy versus communication time. Figure (c) records the communication time every 100 iterations, where the learning rate decay was used at the $10,000$-th iteration.

TABLE II: Number of communication rounds and bits required for the four algorithms to reach the same test accuracy baseline (average of five experiments).

Model & Dataset	Method	# Rounds	# Bits
FC MNIST	SGD	60400	7.87 E+11
	Sparse	67400	8.78 E+09
	LASG	33853	4.41 E+11
	SASG	**22823**	**2.97 E+09**
ResNet18 CIFAR-10	SGD	111200	3.98 E+13
	Sparse	109800	3.93 E+11
	LASG	107389	3.84 E+13
	SASG	**90846**	**3.25 E+11**
ResNet18 CIFAR-100	SGD	152000	5.46 E+13
	Sparse	152800	5.49 E+11
	LASG	145971	5.24 E+13
	SASG	**137937**	**4.95 E+11**

also able to significantly reduce the number of communication rounds required to complete the training while guaranteeing convergence.

According to Figure 2, we choose several baselines (around the best accuracy) to present the specific number of communication rounds and bits required to achieve the same performance for the four algorithms, with the accuracy baseline set as 98% for MNIST, 92% for CIFAR-10 and CIFAR-100 (top-5 accuracy), respectively. As shown in Table II, thanks to the adaptive aggregation technique, both SASG and LASG algorithms reduce the number of communication rounds, while our SASG algorithm is more effective. The number of communication bits required by the different algorithms to reach the same baseline can be obtained by calculating the number of parameters for different models. The last column of Table II shows that the SASG algorithm, blending adaptive aggregation techniques with sparse communication, outperforms the LASG and sparsification methods, significantly reducing the number of communication bits required for the model to achieve the desired performance.

TABLE III: Average communication time of the four algorithms and extra overhead required for the two adaptive aggregated methods when classifying CIFAR-10 with ResNet18 (record every 100 training iterations).

Method	Communication	Computation	Memory
SGD	327.45s	—	—
Sparse	15.94s	—	—
LASG	286.72s	1.25s	426.25MB
SASG	**13.09**s	**1.25**s	**4.26**MB

C. Communication time

This part presents the communication time required for each algorithm to perform distributed training. As mentioned in Section III, the top-k sparsification method can significantly reduce the communication bits. Thus the sparsification method and the SASG algorithm require much less communication time than the LASG and SGD algorithms when completing the same training task, as shown in Figure 3a. The two methods utilizing the sparsification technique are compared in Figure 3b, which illustrates that SASG accomplishes the required accuracy faster and completes training in less communication time. Figure 3c recorded the communication time required by the sparsification method and the SASG algorithm for every 100 training iterations. As training proceeds, SASG skips more communication rounds, leading to a gradual decrease in required communication time, while the sparsification method remains the same. Besides, Figure 3b implies that skipping redundant information does not compromise the final performance of the model, which also indicates that our adaptive selection criterion is effective. The average communication time is shown in Table III.

Finally, we also need to discuss the additional overhead associated with executing the SASG algorithm, i.e., the extra computation time for calculating the selection rule and the memory overhead required for storing the stale gradients. As outlined in Algorithm 1, in a single SASG iteration, each worker needs to store one duplicate of the previous model parameters to compute the auxiliary gradient, and the server needs to store the old gradient (sparse one) from each

worker. Table III presents the additional overhead required for the two adaptive aggregated methods. The computational overhead is approximately the same for both methods, mainly for computing an auxiliary gradient. However, this overhead is negligible when compared to the communication time. On the other hand, the SASG algorithm only needs to store the sparsified gradients on the server side compared to LASG, significantly decreasing the additional memory overhead.

VII. Conclusion

This paper proposes a communication-efficient SASG algorithm for distributed learning. The SASG algorithm adaptively skips several communication rounds with an adaptive selection rule, and further reduces the number of communication bits by sparsifying the transmitted information. For the biased nature of the top-k sparsification operator, we utilize an error feedback framework and provide convergence results for SASG with the help of Lyapunov analysis. Experimental results show that our approach can reduce both the number of communication rounds and bits without sacrificing convergence performance, which corroborates our theoretical findings.

References

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, and Others, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 248–255.

[3] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang *et al.*, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1223–1231.

[4] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, 2014, pp. 583–598.

[5] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu, "Petuum: A new platform for distributed machine learning on big data," *IEEE Transactions on Big Data*, vol. 1, no. 2, pp. 49–67, 2015.

[6] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, p. 48, 2009.

[7] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Advances in Neural Information Processing Systems*, 2014, pp. 19–27.

[8] M. A. Zinkevich, M. Weimer, A. Smola, and L. Li, "Parallelized stochastic gradient descent," in *Advances in Neural Information Processing Systems*, vol. 23, 2010, pp. 2595–2603.

[9] M. I. Jordan, J. D. Lee, and Y. Yang, "Communication-efficient distributed statistical inference," *Journal of the American Statistical Association*, vol. 114, no. 526, pp. 668–681, 2019.

[10] Y. Lin, S. Han, H. Mao, Y. Wang, and B. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *International Conference on Learning Representations*, 2018.

[11] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 440–445.

[12] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Advances in Neural Information Processing Systems*, vol. 31, 2018, pp. 5977–5987.

[13] M. Elibol, L. Lei, and M. I. Jordan, "Variance reduction with sparse gradients," in *International Conference on Learning Representations*, 2020.

[14] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," in *Advances in Neural Information Processing Systems*, vol. 31, 2018, pp. 4447–4458.

[15] T. Chen, G. Giannakis, T. Sun, and W. Yin, "LAG: Lazily aggregated gradient for communication-efficient distributed learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 5050–5060.

[16] T. Chen, Z. Guo, Y. Sun, and W. Yin, "CADA: Communication-adaptive distributed adam," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 613–621.

[17] T. Chen, Y. Sun, and W. Yin, "Communication-adaptive stochastic gradient methods for distributed learning," *IEEE Trans. Signal Process.*, vol. 69, pp. 4637–4651, 2021.

[18] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient sgd via gradient quantization and encoding," in *Advances in Neural Information Processing Systems*, 2017, pp. 1709–1720.

[19] D. Basu, D. Data, C. Karakus, and S. N. Diggavi, "Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 668–14 679.

[20] 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 *Advances in Neural Information Processing Systems*, 2017, pp. 1508–1518.

[21] J. Sun, T. Chen, G. Giannakis, and Z. Yang, "Communication-efficient distributed learning via lazily aggregated quantized gradients," in *Advances in Neural Information Processing Systems*, 2019, pp. 3365–3375.

[22] X. Deng, T. Sun, F. Liu, and D. Li, "SignGD with error feedback meets lazily aggregated technique: Communication-efficient algorithms for distributed learning," *Tsinghua Science and Technology*, vol. 27, no. 1, pp. 174–185, 2022.

[23] S. Horváth and P. Richtárik, "A better alternative to error feedback for communication-efficient distributed learning," in *International Conference on Learning Representations*, 2021.

[24] S. U. Stich, "Local SGD converges fast and communicates little," in *International Conference on Learning Representations*, 2019.

[25] T. Vogels, S. P. Karimireddy, and M. Jaggi, "PowerSGD: Practical low-rank gradient compression for distributed optimization," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019, pp. 14 236–14 245.

[26] H. Wang, S. Sievert, Z. Charles, S. Liu, S. Wright, and D. Papailiopoulos, "ATOMO: Communication-efficient learning via atomic sparsification," in *Advances in Neural Information Processing Systems*, 2018, pp. 9872–9883.

[27] Z. Tang, S. Shi, X. Chu, W. Wang, and B. Li, "Communication-efficient distributed deep learning: A comprehensive survey," *CoRR*, vol. abs/2003.06307, 2020.

[28] J. Bernstein, Y. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed optimisation for non-convex problems," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80. PMLR, 2018, pp. 559–568.

[29] 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 *Fifteenth Annual Conference of the International Speech Communication Association*, 2014, pp. 1058–1062.

[30] F. Faghri, I. Tabrizian, I. Markov, D. Alistarh, D. M. Roy, and A. Ramezani-Kebrya, "Adaptive gradient quantization for data-parallel SGD," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 3174–3185.

[31] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Advances in Neural Information Processing Systems*, 2018, pp. 1306–1316.

[32] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi, "Error feedback fixes signSGD and other gradient compression schemes," in *International Conference on Machine Learning*, 2019, pp. 3252–3261.

[33] C. Xie, S. Zheng, S. Koyejo, I. Gupta, M. Li, and H. Lin, "CSER: Communication-efficient SGD with error reset," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 12 593–12 603.

[34] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *International Conference on Machine Learning*, 2014, pp. 1000–1008.

[35] H. Hendrikx, L. Xiao, S. Bubeck, F. R. Bach, and L. Massoulié, "Statistically preconditioned accelerated gradient method for distributed

optimization," in *International Conference on Machine Learning*, vol. 119. PMLR, 2020, pp. 4203–4227.
[36] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi, "Don't use large mini-batches, use local SGD," in *International Conference on Learning Representations*, 2019.
[37] W. Li, Z. Wu, T. Chen, L. Li, and Q. Ling, "Communication-censored distributed stochastic gradient descent," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2021.
[38] L. Wang, W. Wang, and B. Li, "CMFL: Mitigating communication overhead for federated learning," in *39th IEEE International Conference on Distributed Computing Systems, ICDCS 2019*. IEEE, 2019, pp. 954–964.
[39] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.
[40] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[41] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," pp. 32–33, 2009.
[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

Xiaoge Deng received the B.S. degree in mathematics from University of Science and Technology of China (USTC), Hefei, China, in 2018, and received the M.S. degree in computer science from College of Computer Science, National University of Defense Technology, Changsha, China, in 2020. Currently, he is pursuing a Ph.D. degree from the School of Computer, National University of Defense Technology (NUDT). His research interests include optimization for machine learning and distributed systems.

Dongsheng Li received the B.Sc. degree (with honors) and Ph.D. degree (with honors) in computer science from College of Computer Science, National University of Defense Technology, Changsha, China, in 1999 and 2005, respectively. He was awarded the prize of National Excellent Doctoral Dissertation of PR China by Ministry of Education of China in 2008. He is now a full professor at National Lab for Parallel and Distributed Processing, National University of Defense Technology, China. His research interests include parallel and distributed computing, Cloud computing, and large-scale data management.

Tao Sun received the B.S., M.S. and Ph.D. degree in mathematics from National University of Defense Technology, Changsha, China, in 2012, 2015 and 2018. Currently, he is an Assistant Professor with National Laboratory for Parallel and Distributed Processing, National University of Defense Technology. His research interests include optimization for machine learning, image processing, distributed system and neutral networks.

Xicheng Lu is currently a Professor and a Ph.D. Supervisor with the College of Computer, National University of Defense Technology (NUDT). He has been a member of the Chinese Academy of Engineering, since 1999. His research interests include parallel and distributed processing, and computer networks.