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 high 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 study, we present SASSG, a communication-efficient distributed algorithm that enjoys the advantages of sparse communication and adaptive aggregated stochastic gradients. By dynamically determining the workers who need to communicate through an adaptive aggregation rule and sparsifying the transmitted information, the SASSG algorithm reduces both the overhead of communication rounds and the number of communication bits in the distributed system. For the theoretical analysis, we introduce an important auxiliary variable and 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 SASSG scales well with the number of distributed workers. Finally, experiments on training deep neural networks demonstrate that the proposed algorithm can significantly reduce communication overhead compared to previous methods.

Index Terms—Adaptive aggregation, distributed algorithm, efficient communication, 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], [4], [5]. A large number of distributed machine learning tasks can be described as

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

(1)

where $\omega$ represents the parameter vector to be learned, $d$ is the dimension of parameters, and $M := \{1, \ldots, M\}$ denotes the set of distributed workers. $\{f_m\}_{m=1}^M$ are smooth (not necessarily convex) loss functions kept at worker $m$, and $\{\xi_m\}_{m=1}^M$ are independent random data samples associated with probability distribution $\{D_m\}_{m=1}^M$. For simplicity, we define $F_m(\omega) := \mathbb{E}_{\xi_m \sim D_m} \left[ f_m(\omega; \xi_m) \right]$. Let $\omega^\ast$ be the optimal solution and $F(\omega^\ast)$ be the optimal solution. 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). At each iteration $t$, the algorithm updates the parameter vector $\omega$ as

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

(2)

where $\xi_m^t$ is a mini-batch data selected by worker $m$ at the $t$-th iteration, and $\gamma$ is the learning rate. In the context of distributed ML, parameter server (PS) is a common architecture that has been extensively studied in [4], [7]. This architecture involves $M$ workers that parallelly compute the gradients based on their local data while a centralized server updates the global parameter $\omega$. The distributed SGD in PS operates as follows. At iteration $t$, the server broadcasts the current model $\omega^t$ to all workers, and each worker $m \in M$ computes the local gradient $\nabla f_m(\omega^t; \xi_m^t)$ with the mini-batch samples $\xi_m^t \sim D_m$, which is then uploaded to the server. The server aggregates the received gradients and updates $\omega^{t+1}$ 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 accelerate 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 an unaffordable communication overhead in distributed systems. Even worse, when the computation-to-communication ratio is low, e.g., using high-speed computing devices with low-speed interconnects to train a ML model, parallelization across multiple processors can actually result in lower performance than training on a single processor [10]. Consequently, the communication overhead between distributed workers and the server becomes a significant bottleneck.

Various techniques have been proposed to alleviate this issue, including sparse communication [11], [12], [13], [14] and adaptive aggregation methods [15], [16], [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)?

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 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 $\omega$), 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], [19], [20], [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 introduce an auxiliary sequence and define 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 that demonstrate the superiority of the proposed SASG algorithm.\(^1\)

**Notation:** In this paper, $E[\cdot]$ represents the expectation for randomly selected data samples. For a vector $x \in \mathbb{R}^d$ and a scalar $a \in \mathbb{R}$, $\|x\|$ denotes the $\ell_2$-norm and $|a|$ denotes the absolute value. The notation $[d]$ represents the set \{1, 2, \ldots, $d$\}, and $|S|$ indicates the number of elements in the set $S$.

## II. RELATED WORK

Numerous communication-efficient distributed learning approaches have been proposed to fully exploit the computational power of distributed clusters [15], [18], [19], [20], [22], [23], [24], [25], [26], [27]. More information can be found in a comprehensive survey [28]. 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 [29], [30], [31], which reduces each component of the gradient to just its sign (one bit). Adaptive quantization methods are also investigated in [32] to reduce the communication cost.

Sparsification methods aim to reduce the number of transmitted elements at each iteration. These methods can be divided into two main categories: random and deterministic sparsification. Random sparsification selects some entries randomly for communication [26], [33]. 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 preserves only a few coordinates of the stochastic gradient with the largest magnitudes [11], [12], [13], [14]. This ideology is also known as top-$k$. In contrast to the unbiased scheme, it is clear that this approach requires error feedback or accumulation procedures to ensure convergence [14], [34], [35].

### B. Communication Round Reduction

Reducing the number of communication rounds to improve communication efficiency is another research focus. Shamir et al. [36] leveraged higher-order information (newton-type method) instead of traditional gradient information, thereby reducing the number of communication rounds. Hendrikx et al. [37] proposed a distributed preconditioned accelerated gradient method to reduce the number of communication rounds. Additionally, novel aggregation techniques, including periodic aggregation [19], [24], [38] and adaptive aggregation [15], [16], [21], have been explored to skip certain communications. Among these techniques, local SGD allows every worker to perform local model updates independently [24], [38], 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], [39]. Communication-censored distributed stochastic gradient descent (CSGD) algorithm [39] 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 drawn inspiration from LASG. The significant difference is that LASG only reduces communication round overhead, while our SASG method extends this idea to reduce the overhead of both communication rounds and bits in the stochastic setting. Additionally, SASG reduces the memory overhead required for adaptive aggregation technique by storing only the sparsified data on the server side.

---

\(^1\)The code is available at https://github.com/xiaogdeng/SASG

---

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
III. ALGORITHM DEVELOPMENT

This section introduces SASG, a communication-efficient distributed algorithm that reduces both the number of communication rounds and bits. First, we describe the motivation 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 observation is that not all communication rounds between the server and workers contribute equally to the state of the global model [15], [16], [17], [40]. Then, one can design an aggregation rule to skip inefficient communication rounds. LAG has developed an adaptive selection rule that can detect workers with slowly varying gradients and trigger 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.

Two approaches for reducing the number of transmission bits are quantization and sparsification. We employ the latter as quantization methods are limited to a compression rate of up to 32x (1-bit quantization method) in commonly used single-precision floating-point algorithms. When the model dimension $d$ is large, i.e., $d \gg 32$, the sparsification method will far outperform the quantization approach, offering a maximum achievable compression rate of $d \times$.

Although random sparsification methods ensure the unbiasedness of the compression operator and facilitate theoretical analysis, the top-$k$ sparsification method tends to achieve better practical results [14], [33]. We will utilize the top-$k$ sparsification operator, denoted by $T_k(\cdot)$. 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 $1 \leq k < d$, the sparsification operator $T_k(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^d$, is defined for $\mathbf{x} \in \mathbb{R}^d$ as

$$
(T_k(\mathbf{x})) \pi(i) := \begin{cases} 
(x) \pi(i), & \text{if } i \leq k, \\
0, & \text{otherwise},
\end{cases}
$$

(3)

where $\pi$ is a permutation of set $[d]$ such that $|((\mathbf{x}) \pi(i))| \geq |((\mathbf{x}) \pi(i+1))|$ 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 gradient 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 stored on the server. The difference is defined as

$$
\Delta^t_m := \nabla f_m(\omega^t; \xi^t_m) - \nabla f_m(\omega^{t-\tau_m}; \xi^t_m),
$$

(4)

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

$$
\Delta^t_m \leq \frac{1}{M^2} \sum_{d=1}^D \alpha_d \|\omega^{t+1-d} - \omega^{t-d}\|^2,
$$

(5)

where $\{\alpha_d \geq 0\}_{d=1}^D$ 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 $\xi^t_m$ and $\xi^{t-\tau_m}_m$ across two iterations makes the value of $\Delta^t_m$ almost never small, rendering criterion (5) less effective. Our SASG method takes advantage of the adaptive selection rule from LAG [17], formulated as

$$
\left\| \nabla f_m(\omega^t; \xi^t_m) - \nabla f_m(\omega^{t-\tau_m}; \xi^t_m) \right\|^2 \\
\leq \sum_{d=1}^D \frac{\alpha_d}{M^2} \|\omega^{t+1-d} - \omega^{t-d}\|^2.
$$

(6)

This condition is evaluated on the same data $\xi^t_m$ at two different iterations $t$ and $t - \tau_m$. 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 the function $\nabla f_m$, we have

$$
\left\| \nabla f_m(\omega^t; \xi^t_m) - \nabla f_m(\omega^{t-\tau_m}; \xi^t_m) \right\|^2 \leq L_m \|\omega^t - \omega^{t-\tau_m}\|^2,
$$

(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 $\xi^t_m$ and $\xi^{t-\tau_m}_m$. Additionally, this insight contributes to the determination of the threshold hyper-parameters in criterion (6).

At each training iteration $t$, selection rule (6) divides the worker set $\mathcal{M}$ into two disjoint sets, $\mathcal{M}^t$ and $\mathcal{M}^{t'}$. The parameter server only needs to receive the new gradient data from $\mathcal{M}^t$ and reuse outdated gradients (stored on the server side) from $\mathcal{M}^{t'}$, which scale down the per-iteration communication rounds from $M$ to $|\mathcal{M}^t|$. After the adaptive selection procedure, selected workers transmit 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 disrupts algorithmic convergence, intensifying challenge two.
Algorithm 1: SASG Algorithm.

Initialize: Errors $e_m^0 = 0$, delay counters $τ_m^0 = 1$, ∀$m ∈ M$

Input: Learning rate $γ > 0$, maximum delay $D$, Constant weights $\{α_t\}_{t=1}^D$

1: for $t = 0,1, \ldots, T$ do
2: Server broadcasts $ω^t$ to all workers.
3: for worker $m = 1, \ldots, M$ do
4: Compute $g_m^t = γ\nabla f_m(ω^t; ξ_m^t) + e_m^t$.
5: Divide the workers into $M'$ and $M_0'$ according to the selection rule (6).
6: if worker $m ∈ M'$ or $τ_m ≥ D$ then
7: Worker $m$ uploads $T_k(g_m^t)$ to the server.
8: Set $e_m^{t+1} = g_m^t - T_k(g_m^t)$, $τ_m^{t+1} = 1$.
9: else
10: Worker $m$ uploads nothing.
11: Set $e_m^{t+1} = e_m^t$, $τ_m^{t+1} = τ_m^t + 1$.
12: end if
13: end for
14: Server updates parameter $ω$ according to (8).
15: end for

To address this issue, SASG incorporates error feedback techniques, also known as error accumulation or memory [14], [34]. Initially, the algorithm sparsifies the gradient information $g$ to obtain $T_k(g)$. When uploading the sparsified information $T_k(g)$, we calculate the compression error $e = g - T_k(g)$ and retain it for subsequent use. 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 the error feedback mechanism. We introduce an auxiliary sequence $\{ω^t\}_{t=0,1,\ldots}$, which can be regarded as an error approximation of $\{ω^t\}_{t=0,1,\ldots}$. Leveraging 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

$$ω^{t+1} = ω^t - \frac{1}{M} \left[ \sum_{m ∈ M'} T_k(g_m^t) + \sum_{m ∈ M'_0} T_k(g_m^{τ_m}) \right],$$  \hspace{1cm} (8)

where

$$g_m^t := γ\nabla f_m(ω^t; ξ_m^t) + e_m^t, \hspace{0.5cm} e_m^{t+1} := g_m^t - T_k(g_m^t).$$  \hspace{1cm} (9)

Here, $M'$ and $M_0'$ denote the sets of workers that do and do not communicate with the server at the $t$-th iteration, respectively. Staleness $τ_m$ is determined by the selection of the subset $M'$ at the $t$-th iteration. For worker $m ∈ M'$, the server resets $τ_m^{t+1} = 1$, and the worker uploads the local gradient information. Otherwise, the server increases staleness by $τ_m^{t+1} = τ_m^t + 1$, and worker $m$ uploads nothing. The SASG procedure is illustrated in Fig. 1 and outlined in Algorithm 1. Specifically, during each iteration $t = 0, 1, 2, \ldots$

1) server broadcasts the learning parameter $ω^t$ to all workers;
2) each worker $m$ calculates the local gradient $\nabla f_m(ω^t; ξ_m^t)$ and an auxiliary gradient $\nabla f_m(ω^t; τ_m^t; ξ_m^t)$ for computing the selection rule (6);
3) workers in $M'$ identified by condition (6) sparsify the local information $g_m^t = γ\nabla f_m(ω^t; ξ_m^t) + e_m^t$, and upload $T_k(g_m^t)$ to the server;
4) server aggregates the fresh sparsified gradients $T_k(g_m^t)$ from the selected workers $M'$ and the outdated gradient information $T_k(g_m^{τ_m})$ (stored on the server) from $M_0'$ to update the parameter $ω$ via (8).

As shown in Table I, we only need to upload the sparse gradient information from the selected workers, thus significantly saving the communication overhead (note that $M'$ selected in SASG and LASG is not the same due to different update information). Furthermore, SASG stores the sparsified gradient on the server, which reduces the memory overhead of traditional adaptive aggregation methods.

IV. Theoretical Analysis

This section provides a detailed theoretical analysis of the SASG algorithm. As shown in Table I, unlike LASG, the SASG algorithm relies solely on biased gradient information from partial workers for model updates, which complicates its convergence analysis. To overcome these challenges, we first prove that the residuals of the compression operator are bounded (Lemmas 1 and 2) and introduce a critical auxiliary variable that couples the model parameters with the compression error. Subsequently, we define a new Lyapunov function with respect to the auxiliary
variable, which is also compatible with the selection criterion (6). The descent lemmas on the value of the objective function (Lemma 3) and the Lyapunov function (Lemma 4) derive the convergence theorem for the SASG algorithm (Theorem 1), and its convergence rate is consistent with that of the original SGD algorithm. 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. The 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.,\[
\|\nabla f_m(y) - \nabla f_m(x)\| \leq L_m\|y - x\|,
\|\nabla F(y) - \nabla F(x)\| \leq L\|y - x\|.
\] (10)

Assumption 2 (Moment boundedness): The data sampled in each iteration, i.e., $\xi_m, \xi_m^T, \ldots$, are independent, and the stochastic gradient $\nabla f_m(\omega; \xi_m)$ satisfies
\[
\mathbb{E}[\nabla f_m(\omega; \xi_m)] = \nabla F_m(\omega),
\mathbb{E}[\|\nabla f_m(\omega; \xi_m) - \nabla F_m(\omega)\|^2] \leq \sigma_m^2.
\] (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
\[
\mathbb{E}[\|\nabla f_m(\omega; \cdot)\|] \leq B_m.
\] (12)

B. Technical Lemmas

The analysis commences by highlighting a key property of the top-$k$ sparsification operator $T_k(\cdot)$.

Lemma 1: The operator $T_k(\cdot)$ is a $\delta$-approximate compressor, i.e., there exists a constant $\delta = k/d \in (0, 1)$ such that
\[
\|T_k(x) - x\|^2 \leq (1 - \delta)\|x\|^2, \forall x \in \mathbb{R}^d.
\] (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 each iteration $t$ of the SASG algorithm, the residual error $e_{m}^{t}$ in worker $m$ is bounded, i.e., let $\delta = k/d$, we have
\[
\mathbb{E}[\|e_{m}^{t}\|^2] \leq \frac{4(1 - \delta)}{\delta^2} \gamma^2 B_m^2.
\] (14)

Lemma 2 demonstrates that the residual errors maintained in Algorithm 1 do not accumulate too much and are closely related to the sparsification coefficient $k$. Next, we introduce a crucial auxiliary sequence for subsequent analysis.

Definition 2: Let $\{\omega^t\}_{t=0,1,...}$ be the sequence generated by SASG, and the auxiliary variable $\nu^t$ is defined as
\[
\nu^t := \omega^t - \frac{1}{M} \sum_{m \in M} e_{m}^{t}.
\] (15)

The sequence $\{\nu^t\}_{t=0,1,...}$, which can be considered as an error-corrected sequence of $\{\omega^t\}_{t=0,1,...}$, has the following property (Lemma 3 is defined in (4))
\[
\nu^{t+1} - \nu^t = -\gamma M \sum_{m \in M} \nabla f_m(\omega^t; \xi_m) + \frac{\gamma}{M} \sum_{m \in M} \Delta_m^t,
\] (16)

Based on this property, we have the following descent lemma for the loss function concerning the auxiliary variable $\nu^t$.

Lemma 3: Let Assumptions 1 and 2 hold, and the sequence $\{\nu^t\}_{t=0,1,...}$ is defined in (15). Then, the objective function value satisfies (with $\sigma^2 := \sum_{m=1}^{M} \sigma_m^2$)
\[
\mathbb{E}[F(\nu^{t+1})] - \mathbb{E}[F(\nu^t)] \leq -\left(\gamma - \frac{5L + 4}{2} \gamma^2\right) \mathbb{E}[\|\nabla F(\nu^t)\|^2] + \frac{D}{2 M^2 L} \mathbb{E}\left[\left\|\sum_{m \in M} e_{m}^{t}\right\|^2\right] + \frac{L}{M} \mathbb{E}\left[\left\|\sum_{m \in M} \Delta_m^t\right\|^2\right] + \frac{10(L + 1) + 2 DL \gamma^2 \sigma^2}{M}.
\] (17)

It is worth noting that all terms on the right-hand side of inequality (17) exist in SGD analysis except $\|\nu^{t+1} - \nu^t\|^2$ and $\sum_{m \in M} e_{m}^{t}$, 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
\[
L^t := \mathbb{E}[F(\nu^t)] - F^* + \sum_{d=1}^{D} \beta_d \|\nu^{t+1-d} - \nu^{t-d}\|^2,
\] (18)
where \( \{\beta_d \geq 0\}_{d=1} \) 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), \( L_t > 0 \) for any \( t \in \mathbb{N} \). A direct extension of Lemma 3 gives the following descent lemma.

**Lemma 4:** Let Assumptions 1, 2, and 3 hold and denote 
\[
\sigma^2 := \sum_{m=1}^{M} \sigma_m^2, \quad B^2 := \sum_{m=1}^{M} B_m^2,
\]
If the learning rate \( \gamma \) and constant weights \( \{\alpha_d\}_{d=1} \) are chosen properly, the Lyapunov function satisfies
\[
L_t + 1 - L_t \leq -c_f \mathbb{E}[\|\nabla F(\omega^t)\|^2] + a \frac{\gamma^2 \sigma^2}{M} + b \frac{\gamma^2 B^2}{M} - \sum_{d=1}^{D} c_d \|\omega^{t+1-d} - \omega^t\|^2,
\]
where \( c_f, a, b, c_1, \ldots, c_D \geq 0 \) depend on the learning rate \( \gamma \), constants \( D, L \), and \( \{\alpha_d, \beta_d\}_{d=1} \). 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 \( \{\omega^t\}_{t=0,1,\ldots} \) be generated by the SASG algorithm. If constant weights \( \{\alpha_d\}_{d=1} \) are selected properly, and the learning rate is chosen as \( \gamma = \min\{1/(5L + 4 + 16\beta_1), \sqrt{c_d}/\sqrt{T}\} \), where \( c_d > 0 \) is a constant and \( \beta_1 \) is defined in (18), then we have
\[
\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla F(\omega^t)\|^2] \leq \frac{2(5L + 4 + 16\beta_1)(F(\omega^0) - F^*)}{\sqrt{T}} + \frac{2(F(\omega^0) - F^*)}{c_\gamma \sqrt{T}} + \frac{2c_\gamma(\alpha^2 + bB^2)}{M \sqrt{T}}.
\]

Theorem 1 establishes the convergence guarantee of our algorithm with a sublinear convergence rate of \( O(1/\sqrt{T}) \), despite skipping many communication rounds and performing communication compression. In other words, the SASG algorithm, utilizing well-designed adaptive aggregation and sparse communication techniques, can still achieve a convergence rate identical to that of the SGD method.

**Remark 1:** In the analysis of Theorem 1, we also delved into the effect of the number of workers \( M \) on the convergence performance. As indicated by result (20), the convergence is controlled by the average variance of the distributed system and is not affected by the number of workers, making SASG scale well to large-scale distributed systems.

**Remark 2:** It follows from the proof of the theorem that the constant \( b \) is defined as
\[
b := \frac{(2L^2 + 16\beta_1)(1 - \delta)}{\delta^2}, \quad \delta = \frac{k}{d}.
\]
As the sparsification operation retains more components (i.e., with a larger value of \( k \)), the parameter \( b \) diminishes, and thus it is more favorable for the algorithm to converge. This theoretical finding is consistent with empirical intuition. On the other hand, it is important to recognize that the convergence result of the algorithm depends on more than the term \( 2c_\gamma bB^2 / M \sqrt{T} \). Consequently, using different sparse coefficients cannot change the sublinear convergence rate of the algorithm.

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 \( x \in \mathbb{R}^d \), we observe from Definition 1 that only the first \( k \) largest coordinates of the vector are retained. Then, we can derive
\[
\|T_k(x) - x\|^2 \geq \frac{k}{d} \|x\|^2.
\]
Since \( \langle T_k(x) - x, T_k(x) \rangle = 0 \), it follows that
\[
\|T_k(x) - x\|^2 = \|x\|^2 - \|T_k(x)\|^2.
\]
Let \( \delta := k/d \in (0, 1) \), we can deduce that
\[
\|T_k(x) - x\|^2 \leq (1 - \delta) \|x\|^2.
\]

B. **Proof of Lemma 2**

From Algorithm 1 and Lemma 1, it follows that
\[
\mathbb{E}[\|\epsilon_{t+1}^m\|^2] = \mathbb{E}[\|g^m_{t+1} - T_k(g^m_t)\|^2] \leq (1 - \delta) \mathbb{E}[\|g^m_t\|^2]
\]
\[
= (1 - \delta) \sum_{t=0}^{T-1} \mathbb{E}[\|\gamma \nabla f_m(\omega^t, \xi^t_m) + \epsilon^m_t\|^2]
\]
\[
\leq (1 - \delta) \left( (1 + \eta) \mathbb{E}[\|\epsilon^m_t\|^2] + \left( 1 + \frac{1}{\eta} \right) \gamma^2 \mathbb{E}[\|\nabla f_m(\omega^t, \xi^t_m)\|^2] \right).
\]
where we used Young’s inequality \( \|a + b\|^2 \leq (1 + \eta)\|a\|^2 + (1 + 1/\eta)\|b\|^2 \) for any \( \eta > 0 \). By Algorithm 1 and Assumption 3, we have that error \( \epsilon^m_0 = 0 \) and gradient \( \nabla f_m(\omega^t, \xi^t_m) \) is bounded. Simple computation yields
\[
\mathbb{E}[\|\epsilon_{t+1}^m\|^2] \leq \sum_{t=0}^{T-1} \left( (1 - \delta + 1/\eta) \left( 1 + \frac{1}{\eta} \right) \gamma^2 \mathbb{E}[\|\nabla f_m(\omega^t, \xi^t_m)\|^2] \right)
\]
\[
= \frac{(1 - \delta)(1 + 1/\eta)}{1 - (1 - \delta)(1 + \eta)} \gamma^2 B^2_m.
\]
By selecting \( \eta := \frac{\delta}{2(1 - \delta)} \) yielding \( 1 + 1/\eta = (2 - \delta)/\delta \leq 2/\delta \), we arrive at
\[
\mathbb{E}[\|\epsilon_{t+1}^m\|^2] \leq \frac{2(1 - \delta)(1 + 1/\eta)}{\delta} \gamma^2 B^2_m \leq \frac{4(1 - \delta)}{\delta^2} \gamma^2 B^2_m.
\]
C. Proof of (16)
Following Definition 2 and the iterative scheme (8), we have
\[
\begin{align*}
\nu^{t+1} &= \omega^{t+1} - \frac{1}{M} \sum_{m \in M} e_{m}^{t+1} \\
&= \omega^{t} - \frac{1}{M} \sum_{m \in M} \left( \sum_{k \in M} T_k(g_{m}^{t}) + \sum_{m \in M} T_k(g_{m}^{t-\tau_{m}}) + \sum_{m \in M} e_{m}^{t+1} \right) \\
&= \omega^{t} - \frac{1}{M} \sum_{m \in M} \left( \sum_{m \in M} g_{m}^{t} + \sum_{m \in M} g_{m}^{t-\tau_{m}} \right) \\
&= \omega^{t} - \frac{1}{M} \sum_{m \in M} \left( \nabla f_m(\omega^{t}; \xi_{m}^{t}) + e_{m}^{t} \right) \\
&\quad - \frac{1}{M} \sum_{m \in M} \left( \nabla f_m(\omega^{t-\tau_{m}}; \xi_{m}^{t-\tau_{m}}) + e_{m}^{t-\tau_{m}} \right) \\
&= \omega^{t} - \frac{1}{M} \sum_{m \in M} \nabla f_m(\omega^{t}; \xi_{m}^{t}) + \frac{\gamma}{M} \sum_{m \in M} \Delta_{m}^{t}.
\end{align*}
\]
where (\( \ast \)) used \( e_{m}^{t+1} = e_{m}^{t-\tau_{m}}, \forall m \in M_{t} \). Let
\[
\Delta_{m}^{t} := \nabla f_m(\omega^{t}; \xi_{m}^{t}) - \nabla f_m(\omega^{t-\tau_{m}}; \xi_{m}^{t-\tau_{m}}),
\]
it then follows that
\[
\nu^{t+1} - \nu^{t} = -\frac{\gamma}{M} \sum_{m \in M} \nabla f_m(\omega^{t}; \xi_{m}^{t}) + \frac{\gamma}{M} \sum_{m \in M} \Delta_{m}^{t}.
\]
(21)

D. Proof of Lemma 3
We first present the complete derivation and then delve into the corresponding specifics.
\[
\begin{align*}
\mathbb{E} [F(\nu^{t+1})] - \mathbb{E} [F(\nu^{t})] \\
&\leq (a) \mathbb{E} \left( \nabla F(\nu^{t}), \nu^{t+1} - \nu^{t} \right) + \frac{L}{2} \mathbb{E} \| \nu^{t+1} - \nu^{t} \|^{2} \\
&\leq (b) \mathbb{E} \left( \nabla F(\omega^{t}), -\frac{\gamma}{M} \sum_{m \in M_{t}} \nabla f_m(\omega^{t}; \xi_{m}^{t}) + \frac{\gamma}{M} \sum_{m \in M_{t}} \Delta_{m}^{t} \right) \\
&\quad + \mathbb{E} \left( \nabla F(\nu^{t}) - \nabla F(\omega^{t}), \nu^{t+1} - \nu^{t} \right) + \frac{L}{2} \mathbb{E} \| \nu^{t+1} - \nu^{t} \|^{2} \\
&\leq (c) -\gamma \mathbb{E} \| \nabla F(\omega^{t}) \|^{2} + \frac{\gamma}{M} \mathbb{E} \left( \nabla F(\omega^{t}), \sum_{m \in M_{t}} \Delta_{m}^{t} \right) \\
&\quad + \frac{1}{2} \mathbb{E} \| \nabla F(\nu^{t}) - \nabla F(\omega^{t}) \|^{2} + \frac{L + 1}{2} \mathbb{E} \| \nu^{t+1} - \nu^{t} \|^{2} \\
&\leq (d) -\left( \gamma - \frac{L^{2}}{2} \right) \mathbb{E} \| \nabla F(\omega^{t}) \|^{2} + \frac{L + 1}{2} \mathbb{E} \| \nu^{t+1} - \nu^{t} \|^{2} \\
&\quad + \sum_{d=1}^{D} \left[ \frac{\alpha_d}{2 M^{2} L} + \frac{L}{2} \right] \| \omega^{t+1-d} - \omega^{t-d} \|^{2} \\
&\quad + \frac{L^{2}}{2} \mathbb{E} \| \nu^{t} - \omega^{t} \|^{2} + \frac{2 \gamma^{2} D L \sigma^{2}}{M}.
\end{align*}
\]
where (\( a \)) uses the \( L \)-smooth property outlined in Assumption 1. (\( b \)) is obtained by (21). (\( c \)) utilizes Assumption 2 and the inequality \( \| a \|^{2} + \| b \|^{2} \geq 2 \langle a, b \rangle \). To establish (\( d \)), we need to bound \( \langle \nabla F(\omega^{t}), \Delta_{m}^{t} \rangle \) as follows
\[
\mathbb{E} \left( \nabla F(\omega^{t}), \Delta_{m}^{t} \right) \\
= \mathbb{E} \left( \nabla F(\omega^{t}), \nabla f_{m}(\omega^{t}; \xi_{m}^{t}) - \nabla f_{m}(\omega^{t-\tau_{m}}; \xi_{m}^{t-\tau_{m}}) \right) \\
+ \mathbb{E} \left( \nabla F(\omega^{t}), \nabla f_{m}(\omega^{t-\tau_{m}}; \xi_{m}^{t}) - \nabla f_{m}(\omega^{t-\tau_{m}}; \xi_{m}^{t-\tau_{m}}) \right).
\]
By using the inequality \( \langle a, b \rangle \leq \frac{\varepsilon}{2} \| a \|^{2} + \frac{1}{2\varepsilon} \| b \|^{2} \) with \( \varepsilon := L \gamma \), and employing selection rule (6), we obtain
\[
I \leq \frac{L \gamma}{2} \mathbb{E} \| \nabla F(\omega^{t}) \|^{2} + \frac{1}{2 L \gamma} \frac{1}{M} \sum_{d=1}^{D} \alpha_d \| \omega^{t+d} - \omega^{t-d} \|^{2}.
\]
Noticing that
\[
\mathbb{E} \langle \nabla F(\omega^{t-\tau_{m}}), \nabla f_{m}(\omega^{t-\tau_{m}}; \xi_{m}^{t}) - \nabla f_{m}(\omega^{t-\tau_{m}}; \xi_{m}^{t-\tau_{m}}) \rangle = 0,
\]
we then have
\[
\begin{align*}
II &\leq L \mathbb{E} \langle \nabla F(\omega^{t-\tau_{m}}), \nabla f_{m}(\omega^{t-\tau_{m}}; \xi_{m}^{t}) - \nabla f_{m}(\omega^{t-\tau_{m}}; \xi_{m}^{t-\tau_{m}}) \rangle \\
&\leq \frac{\gamma D L}{2 \gamma} \mathbb{E} \| \nabla f_{m}(\omega^{t-\tau_{m}}; \xi_{m}^{t}) - \nabla f_{m}(\omega^{t-\tau_{m}}; \xi_{m}^{t-\tau_{m}}) \|^{2} \\
&\quad + \frac{L}{2 \gamma} \| \omega^{t} - \omega^{t-\tau_{m}} \|^{2} \\
&\leq 2 \gamma D L \sigma_{m}^{2} + \frac{L}{2 \gamma} \sum_{d=1}^{D} \| \omega^{t+d} - \omega^{t-d} \|^{2},
\end{align*}
\]
where we use the inequality \( \langle a, b \rangle \leq \frac{\varepsilon}{2} \| a \|^{2} + \frac{1}{2\varepsilon} \| b \|^{2} \) with \( \varepsilon := D \gamma, \| \sum_{i=1}^{n} \theta_{i} \| \leq n \sum_{i=1}^{n} \| \theta_{i} \|^{2} \), and Assumption 2. Then, we arrive at (with \( \sigma_{m}^{2} := \sum_{m=1}^{M} \sigma_{m}^{2} \))
\[
\begin{align*}
\frac{\gamma}{M} \mathbb{E} \langle \nabla F(\omega^{t}), \sum_{m \in M_{t}} \Delta_{m}^{t} \rangle &\leq \frac{\gamma}{M} \sum_{m \in M_{t}} (I + II) \\
&\leq \frac{L \gamma}{2} \mathbb{E} \| \nabla F(\omega^{t}) \|^{2} + \sum_{d=1}^{D} \left( \frac{\alpha_d}{M^{2} L} + \frac{L}{2} \right) \| \omega^{t+1-d} - \omega^{t-d} \|^{2} \\
&\quad + \frac{2 \gamma^{2} D L \sigma^{2}}{M}.
\end{align*}
\]
By further analyzing $\|\Delta^{t}_m\|^2$ and $\|\nu^{t+1} - \nu^t\|^2$, we derive (e)

$$
\|\sum_{m \in M'_t} \Delta^{t}_m\|^2 \leq |M'_t| \sum_{m \in M'_t} \|\Delta^{t}_m\|^2 \\
\leq |M'_t| \sum_{m \in M'_t} \|\nabla f_m(\omega^{t}; \xi^t_m) - \nabla f_m(\omega^{t-\tau_m}; \xi^{t-\tau_m}_m)\|^2 \\
+ 2M \sum_{m \in M'_t} \|\nabla f_m(\omega^{t-\tau_m}; \xi^{t-\tau_m}_m) - \nabla f_m(\omega^{t-\tau_m}; \xi^{t-\tau_m}_m)\|^2 \\
\leq 2M \sum_{m \in M'_t} \|\nabla f_m(\omega^{t-\tau_m}; \xi^{t-\tau_m}_m) - \nabla f_m(\omega^{t-\tau_m}; \xi^{t-\tau_m}_m)\|^2 \\
\leq 2 \sum_{d=1}^{D} \alpha_d \|\omega^{t+1-d} - \omega^{t-d}\|^2 + 8M\sigma^2, 
$$

(23)

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

$$
\mathbb{E}[\|\nu^{t+1} - \nu^t\|^2] = \mathbb{E}[\|\gamma M \sum_{m \in M} \nabla f_m(\omega^{t}; \xi^t_m) + \gamma M \sum_{m \in M'_t} \Delta^{t}_m\|^2] \\
\leq 2\gamma^2 M^2 \mathbb{E}[\left\|\sum_{m \in M} \nabla f_m(\omega^{t}; \xi^t_m)\right\|^2] + 2\gamma^2 \frac{M^2}{2^\gamma} \mathbb{E}[\sum_{m \in M'_t} \Delta^{t}_m\|^2] \\
\leq 4\gamma^2 \mathbb{E}[\|\nabla F(\omega^t)\|^2] + 4\gamma^2 \frac{M^2}{2^\gamma} \sum_{d=1}^{D} \alpha_d \|\omega^{t+1-d} - \omega^{t-d}\|^2 + 20\gamma^2 \sigma^2 
$$

(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 straightforward calculation yields

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

(25)

By utilizing the properties (21), (24), and also invoking Lemma 2, we can establish the following inequality

$$
\|\omega^{t+1} - \omega^t\|^2 \leq 2\|\nu^{t+1} - \nu^t\|^2 + \frac{2}{M^2} \sum_{m \in M} (\epsilon_m^{t+1} - \epsilon_m^t)\|^2 \\
\leq 2\|\nu^{t+1} - \nu^t\|^2 + \frac{16(1 - \delta) \gamma^2 B^2}{\delta M},
$$

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

$$
\mathcal{L}^{t+1} - \mathcal{L}^t \leq - \left( \gamma - \frac{5L + 4}{2} \gamma \right) \mathbb{E}[\|\nabla F(\omega^t)\|^2] \\
- \sum_{d=1}^{D} c_d \|\omega^{t+1-d} - \omega^{t-d}\|^2 + \frac{\gamma^2 \sigma^2}{M} + \frac{b \gamma^2 B^2}{M^2},
$$

where

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

F. Proof of Theorem 1

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

$$
\beta_d - \beta_{d+1} - \left( \frac{\alpha_d}{2M^2} + \frac{L}{2} + \frac{2\gamma^2 \alpha_d (L+1+4\beta_1)}{M^2} \right) = 0 \\
d = 1, \ldots, D - 1 \\
\beta_D - \left( \frac{\alpha_D}{2M^2} + \frac{L}{2} + \frac{2\gamma^2 \alpha_D (L+1+4\beta_1)}{M^2} \right) = 0.
$$

Solving the linear equations above we can further get

$$
\beta_1 = \frac{\gamma^2 (L+1)}{L^2 - \frac{1}{2M^2}} \sum_{d=1}^{D} \alpha_d + \frac{LDP}{2M^2}.
$$

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
We then have $c_d \geq 0, d = 1, \ldots, D$ and $c_f \geq \gamma / 2$, implying

$$L^{t+1} - L^t \leq -\frac{2}{\gamma} E[\|\nabla F(\omega^t)\|^2] + \frac{aa^2 + bB^2}{M} \gamma^2.$$  

By taking the summation $\sum_{t=0}^{T-1} L^{t+1} - L^t$, we arrive at

$$\sum_{t=0}^{T-1} \frac{\gamma}{2} E[\|\nabla F(\omega^t)\|^2] \leq F(\omega^0) - F^* + \frac{aa^2 + bB^2}{M} T \gamma^2.$$  

If we choose learning rate $\gamma = \min\{1/(5L+4+16\beta_1), c_\gamma / \sqrt{T}\}$, where $c_\gamma > 0$ is a constant. We then have

$$\frac{1}{T} \sum_{t=0}^{T-1} E[\|\nabla F(\omega^t)\|^2] \leq \frac{2M(F(\omega^0) - F^*) + 2T(aa^2 + bB^2)\gamma^2}{MT \gamma} \leq \frac{2(5L + 4 + 16\beta_1)(F(\omega^0) - F^*)}{T} + \frac{2(F(\omega^0) - F^*)}{c_\gamma \sqrt{T}} + \frac{2c_\gamma (aa^2 + bB^2)}{MT \sqrt{T}},$$  

which means

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

### VI. Experiment Results

This section conducts extensive experiments to demonstrate the convergence properties and communication efficiency of our SAG algorithm. The experiments encompass simulated and real distributed training scenarios, examining the improvement of SAG in both communication volume and time. Additionally, we discuss the additional computational and memory overhead of SAG, the effect of sparsity rate, and the algorithm’s performance in heterogeneous data scenarios.

#### A. Setup

We evaluate the performance of the SAG 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 to update the model at each iteration. Hence, we utilize the top-1% (i.e., $k = 0.01 d$) sparsiﬁcation operator in SAG and the sparsiﬁcation method. All our experimental results are based on the PyTorch framework [41].

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 real-world distributed settings. We employed ten Nvidia RTX-3090 GPUs as distributed workers for training, with the ﬁrst GPU 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 [42] 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 SAG and LASG, we set $D = 10, \alpha_d = 1/2 \gamma$ for $d = 1, 2, \ldots, 10$.

- **CIFAR-10**: The CIFAR-10 [43] dataset comprises 60,000 colored images in 10 classes, with 6,000 images per category. We train ResNet18 [44] 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 SAG and LASG, we set $D = 10, \alpha_d = 1/\gamma$ for $d = 1, 2, \ldots, 10$.

- **CIFAR-100**: We also train ResNet18 on the CIFAR-100 [43] 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 SAG and LASG, we set $D = 10, \alpha_d = 1/\gamma$ for $d = 1, 2, \ldots, 10$.

It is important to note that we did not employ training techniques such as warm-up or weight decay in our experiments, and thus state-of-the-art performance was not achieved in certain models. Our goal is to demonstrate, through comparative experiments, that the SAG algorithm can effectively 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. Fig. 2 depicts the variation of test accuracy and training loss against the number of communication rounds, whereas Fig. 2(c) shows the top-5 test accuracy.

The experimental results demonstrate that our SAG algorithm outperforms the previous methods. Notably, as evident in Fig. 2(a), SAG achieves higher test accuracy with the same number of communication rounds, indicating that our algorithm selects more valuable rounds for communication. Furthermore, SAG can signiﬁcantly reduce the number of communication rounds while preserving the same model performance after completing the training process. Fig. 2(d) shows that the SAG algorithm achieves faster and better convergence results with fewer communication rounds. In the experiments on the CIFAR-10 and CIFAR-100 datasets (Fig. 2(e)-(f)), SAG is also able to signiﬁcantly reduce the number of communication rounds required to complete the training while guaranteeing convergence.

According to Fig. 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),
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.

| 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   | 29823    | 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  |

Table II: Number of Communication Rounds and Bits Required for the Four Algorithms to Reach the Same Test Accuracy Baseline (Average of Five Experiments)

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.

C. Communication Time

This part presents the communication time required for each algorithm in real-world 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 Fig. 3(a). The two methods utilizing the sparsification technique are compared in Fig. 3(b), which illustrates that SASG accomplishes the required accuracy faster and completes training in less communication time. Fig. 3(c) 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, Fig. 3(b) 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.

D. More Experimental Discussion

While the aforementioned experimental results show that SASG can significantly improve the communication efficiency
while guaranteeing algorithm performance, we also need to discuss the additional overhead associated with implementing this 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.

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

Fig. 4. Experimental results of training ResNet18 to classify CIFAR-10 using different sparsity rates.

Fig. 5. Experimental results of training the ResNet18 model in heterogeneous data scenarios.

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)

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.

Fig. 2. Experimental results of training the ResNet18 model in heterogeneous data scenarios.

Additionally, we explore the effect of different sparsification levels on SASG, as shown in Fig. 4. The results indicate that varying only the sparsification coefficient $k$ has no noticeable effect on the model accuracy and loss, and SASG still outperforms the sparsification method with error feedback. Besides, the experimental results reveal that SASG skips more communication rounds as the sparsity rate increases. Therefore, varying the sparse coefficient $k$ requires a corresponding adjustment of the parameter $\{\alpha_d\}_{d=1}^D$ within the adaptive aggregation selection criterion to prevent extreme sparsification from compromising the algorithm’s performance. Moreover, the results in Figs. 2 and 4 show that the sparsification method with error feedback works better than the benchmark SGD algorithm, and its performance can be further improved by appropriately increasing the sparsification factor. The potential reason for this phenomenon can be attributed to the fact that the sparsification operation reduces the gradient norm (or the gradient upper bound), thereby improving the convergence [11], [14] and generalization [45], [46], [47] of the algorithm.

For the heterogeneous data experiments, we simulated ten distributed workers and mixed CIFAR-10 and MNIST datasets to construct the non-i.i.d. environment. Specifically, the even-numbered workers distribute only the CIFAR-10 dataset, while the odd-numbered workers contain only MNIST training samples. The experimental results of training the ResNet18 model in this non-i.i.d. setting using the four algorithms are depicted in Fig. 5. This result shows that the SASG algorithm can significantly improve the communication efficiency of the training system and ensure good model accuracy even in heterogeneous data scenarios. In addition, this experiment also demonstrates that judiciously skipping some communication rounds can compensate for the adverse impact on model accuracy caused by...
training with non-i.i.d. data (e.g., federated learning), which is consistent with the findings of CMFL [40].

VII. CONCLUSION

This article proposes a communication-efficient SASG algorithm for distributed learning. SASG 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 a sublinear convergence result for SASG with the help of a new Lyapunov analysis. Extensive experimental results show that SASG can reduce both the number of communication rounds and bits without sacrificing convergence performance, which corroborates our theoretical findings.

REFERENCES

[1] T. B. Brown et al., “Language models are few-shot learners,” in Proc. Adv. Neural Inf. Process. Syst., 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 Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2009, pp. 248–255.
[3] J. Dean et al., “Large scale distributed deep networks,” in Proc. Adv. Neural Inf. Process. Syst., 2012, pp. 1223–1231.
[4] M. Li et al., “Scaling distributed machine learning with the parameter server,” in Proc. 11th USENIX Symp. Operating Syst. Des. Implementation, 2014, pp. 583–598.
[5] E. P. Xing et al., “Petuum: A new platform for distributed machine learning on Big Data,” IEEE Trans. Big Data, vol. 1, no. 2, pp. 49–67, Jun. 2015.
[6] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” IEEE Trans. Autom. Control, vol. 54, no. 1, 2009, Art. no. 48.
[7] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, “Communication efficient distributed machine learning with the parameter server,” in Proc. Adv. Neural Inf. Process. Syst., 2014, pp. 19–27.
[8] M. A. Zinkevich, M. Weimer, A. Smola, and L. Li, “Parallelized stochastic gradient descent,” in Proc. Adv. Neural Inf. Process. Syst., 2010, pp. 2595–2603.
[9] M. I. Jordan, J. D. Lee, and Y. Yang, “Communication-efficient distributed statistical inference,” J. Amer. Stat. Assoc., 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 Proc. Int. Conf. Learn. Representations, 2018.
[11] A. F. Aji and K. Heafield, “Sparse communication for distributed gradient descent,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2014, pp. 440–445.
[12] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and D. Alistarh, “Distributed SGD: Compressed optimisation for non-convex problems,” in Proc. 35th Int. Conf. Mach. Learn., 2018, pp. 559–568.
[13] 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 Proc. 15th Annu. Conf. Int. Speech Commun. Assoc., 2014, pp. 1058–1062.
[14] X. Fan, Y. Wang, Y. Hoo, and Z. Tian, “1-bit compressive sensing for efficient federated learning over the air,” IEEE Trans. Wireless Commun., vol. 22, no. 3, pp. 2139–2155, Mar. 2023.
[15] F. Faghri, I. Tabrizian, I. Markov, D. Alistarh, D. M. Roy, and A. Ramezani-Kebrya, “Adaptive gradient quantization for data-parallel SGD,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 3174–3185.
[16] J. Wangni, J. Wang, J. Liu, and T. Zhang, “Gradient sparsification for communication-efficient distributed optimization,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 1306–1316.
[17] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi, “Error feedback fixes signSGD and other gradient compression schemes,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 3252–3261.
[18] X. Xie, S. Zheng, S. Koyejo, I. Gupta, M. Li, and H. Lin, “CSER: Communication-efficient SGD with error reset,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 12593–12603.
[19] O. Shamir, N. Srebro, and T. Zhang, “Communication-efficient distributed optimization using an approximate newton-type method,” in Proc. Int. Conf. Mach. Learn., 2014, pp. 1000–1008.
[20] H. Hendrikx, L. Xiao, S. Bubeck, F. R. Bach, and L. Massoulié, “Statically preconditioned accelerated gradient method for distributed optimization,” in Proc. Int. Conf. Mach. Learn., 2020, pp. 4203–4227.
[21] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi, “Don’t use large mini-batches, use local SGD,” in Proc. Int. Conf. Learn. Representations, 2019.
[22] W. Li, Z. Wu, T. Chen, L. Li, and Q. Ling, “Communication-censored distributed stochastic gradient descent,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 11, pp. 6831–6843, Nov. 2022.
[23] L. Wang, W. Wang, and B. Li, “CMFL: Mitigating communication overhead for federated learning,” in Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst., 2019, pp. 954–964.
[24] A. Paszke et al., “PyTorch: An imperative style, high-performance deep learning library,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 8024–8035.
[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[26] A. Krizhevsky et al., “Learning multiple layers of features from tiny images,” pp. 32–33, 2009.
[27] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–777.
[45] M. Hardt, B. Recht, and Y. Singer, “Train faster, generalize better: Stability of stochastic gradient descent,” in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1225–1234.

[46] T. Sun, D. Li, and B. Wang, “Stability and generalization of decentralized stochastic gradient descent,” in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 9756–9764.

[47] X. Deng, T. Sun, S. Li, and D. Li, “Stability-based generalization analysis of the asynchronous decentralized SGD,” in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 7340–7348.

Xiaoge Deng received the BS degree in mathematics from the University of Science and Technology of China (USTC), Hefei, China, in 2018, and the MS degree in computer science from the College of Computer Science, National University of Defense Technology, Changsha, China, in 2020. He is currently working toward the PhD degree with 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 BSc degree (with honors) and PhD degree (with honors) in computer science from the 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 with 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 BS, MS and PhD degrees in mathematics from the 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 neural networks.

Xicheng Lu is currently a professor and a PhD 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.