SASG: Sparsification with Adaptive Stochastic Gradients for Communication-efficient Distributed Learning

Xiaoge Deng, Tao Sun and Dongsheng Li

Abstract—Stochastic optimization algorithms 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 different workers. Sparse communication with memory and the adaptive aggregation methodology are two successful frameworks among the various techniques proposed to address this issue. In this paper, we creatively exploit the advantages of Sparse communication and Adaptive aggregated Stochastic Gradients to design a communication-efficient distributed algorithm named SASG. Specifically, we first determine the workers that need to communicate based on the adaptive aggregation rule and then sparse this transmitted information. Therefore, our algorithm reduces both the overhead of communication rounds and the number of communication bits in the distributed system. We define an auxiliary sequence and give convergence results of the algorithm with the help of Lyapunov function analysis. Experiments on training deep neural networks show that our algorithm can significantly reduce the number of communication rounds and bits compared to the previous methods, with little or no impact on training and testing accuracy.

Index Terms—Distributed algorithm, efficient communication, adaptive aggregation, gradient sparsification

1 INTRODUCTION

O

VER the past few decades, the scale and complexity of machine-learning (ML) models and datasets have increased significantly [1], [2], leading to more computation-intensive and thus time-consuming training processes. What followed is the development of distributed training [3], [4], [5], which uses multiple processors for acceleration. A large number of distributed machine learning tasks can be described as

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

(1)

where $\omega$ is the parameter vector to be learned, $d$ is the dimension of parameters, and $M := \{1, \ldots, M\}$ denote the set of distributed workers. $\{f_m\}_{m=1}^M$ are smooth (unecessary to be 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 $F_m(\omega) := \mathbb{E}_{\xi_m \sim D_m} [f_m(\omega; \xi_m)]$; let $\omega^*$ be the optimal solution and $F^* := F(\omega^*)$. Problem (1) arises in a large number of distributed machine learning tasks, ranging from linear models to deep neural networks [5], [6].

Stochastic Gradient Descent (SGD) is the workhorse for problem (1), performing 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 $\gamma$ is the learning rate, and $\xi_m^t$ is the mini-batch data selected by worker $m$ at the $t$-th iteration. The commonly-used parameter-server (PS) architecture is studied in [4], [6], where $M$ workers compute the gradients in parallel and a centralized server updates the variable $\omega$. The implementation details of distributed SGD method are as follows: at iteration $t$, the server broadcasts $\omega^t$ to all workers; every worker $m \in M$ computes the local gradient $\nabla f_m(\omega^t; \xi_m^t)$ with a randomly selected mini-batch of samples $\xi_m^t \sim D_m$ and uploads it to the server; the server can simply use the aggregated gradient, i.e., $\sum_{m \in M} \nabla f_m(\omega^t; \xi_m^t)$ to update the parameters via (2). The convergence of this distributed implementation of SGD has been proved under several mild assumptions [8].

Intuitively, multi-processor collaborative training for one task can accelerate the training process and reduce training time. However, the communication cost between processors usually hampers the scalability of distributed systems [9], [10]. Even worse, when the computation-to-communication ratio is low, e.g., using the high-speed computing devices with low-speed interconnect to train a ML model, multiple processors might suffer lower performance than that of a single processor [11].

More specifically, at each iteration of PS training (2), the server needs to communicate with all workers to obtain fresh gradients $\{\nabla f_m(\omega^t; \xi_m^t)\}_{m=1}^M$. In several settings, however, communication is much slower than computation [12]. Therefore, the communication between workers and the server has become a bottleneck, especially in the scenarios where the number of workers increases or deep learning-based models are trained with high-dimensional parameters. Thus, communication-efficient solvers are crucial for distributed ML training, and many popular techniques have been proposed, including sparse communication [13], [14], [15], [16] and adaptive aggregation methods [17], [18], [19].

• This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.
E-mail: {mutsxgdeng, nudtsuntao}@163.com, dsl@nudt.edu.cn
A natural question is then can we combine these two approaches to yield a better mechanism for problem (1)?

1.1 Related Work

To take full advantage of the computational power of distributed clusters, a tremendous amount of communication-efficient distributed learning approaches are dedicated to addressing this scalability problem [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31]. More information can be found in a comprehensive survey [32].

We briefly review two kinds of related work in this paper: 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.

Reduce communication bits. This category of research is mainly around the idea of quantization and sparsification.

The quantization approach compresses the 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) [31] utilizes an adjustable quantization level, which provides additional flexibility to control the trade-off between the per-iteration communication cost and the convergence rate. TernGrad [33] uses the ternary gradients to reduce the communication data size. 1-bit quantization method was developed in [34], [35], which reduces each component of the gradient to just its sign (one bit). Adaptive quantization methods are also investigated in [22], [36] to reduce the communication cost.

The sparsification method aims to reduce the number of elements transmitted at each iteration. Such methods can be divided into two main categories: random and deterministic sparsification. Random sparsification [30], [37] is to select some entries to communicate randomly. This ideology is named random-k, where k denotes the number of selected elements. This random choice method is usually an unbiased estimate of the original gradient, making it quite friendly to theoretical analysis. Unlike random sparsification, deterministic sparsification [13], [14], [15], [16] preserves only a very few coordinates of the stochastic gradient by considering the coordinates with the largest magnitudes. This ideology is also known as top-k. In contrast to the unbiased scheme, it is clear that this approach should work with some error feedback or accumulation procedure [14], [20], [23], [27], [38], [39], which ensures that all gradients are eventually added up into the model with full accuracy, albeit with a delay.

Reduce communication rounds. Reducing the number of communication rounds to improve communication efficiency is another research focus. Shamir et al. [40] leveraged higher-order information (Newton-type method) to replace traditional gradient information, thereby reducing the number of communication rounds. Hendrikx et al. [41] proposed a distributed preconditioned accelerated gradient method to reduce the number of communication rounds. Novel aggregation techniques such as periodic aggregation [28], [29], [42] and adaptive aggregation [17], [19], [26] are also investigated and used to skip certain communications. Among them, local SGD [28], [42] allows every worker to perform local model updates independently, and the resultant models are averaged periodically. Lazily aggregated gradient (LAG) [19] approach updates the model at 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 [18]. Recent efforts have been made towards adaptive uploading in stochastic settings [18], [25]. Communication-Censored distributed Stochastic Gradient Descent (C SGD) [25] algorithm increases the batch size to alleviate the effect of stochastic gradient noise. The Lazily Aggregated Stochastic Gradient (LASG) [18] method designed a set of new adaptive communication rules tailored for stochastic gradients, achieving impressive empirical performance. We borrowed some ideas from LASG in this line of research. The main difference is that our approach creatively exploits the advantages of sparse communication and adaptive aggregation techniques, which can reduce the number of communication rounds and bits simultaneously.

1.2 Our Contributions

This paper proposes a communication-efficient distributed algorithm combining Sparse communication with Adaptive aggregated Stochastic Gradients, namely SASG. Our focus is on reducing the overhead of worker-to-server uplink communication in the PS architecture, which we also refer to as uploading. As opposed to the server-to-worker downlink communication (i.e., broadcast the same parameter w) that can be performed simultaneously (or implemented in a tree-structured manner as in many MPI implementations), the server must receive gradients of workers sequentially to avoid interference from other workers, which leads to extra latency. Uploading is therefore the major communication overhead in PS and is the topic of many related work [19], [26], [29], [33]. Throughout this paper, one communication round means one upload from a worker. Our SASG method can simultaneously save communication bits and rounds without sacrificing the desired convergence properties.

Considering that not all communication rounds between the server and workers are equally important in the distributed learning system, we can adjust the frequency of communication between a worker and the server according to the importance of the information transmitted in this worker. More specifically, in terms of reducing the number of communication rounds, we have formulated an adaptive selection rule to divide the set of workers \( \mathcal{M} \) into two disjoint sets \( \mathcal{M}^t \) and \( \mathcal{M}^t_w \) at the t-th iteration. We will use only the new gradient information from the selected workers in \( \mathcal{M}^t \) while reusing the outdated compressed gradients from the rest of the workers in \( \mathcal{M}^t_w \) which scale down the per-iteration communication rounds from \( \mathcal{M} \) to \( |\mathcal{M}^t| \).

On the other hand, since the quantization methods can only achieve a maximum compression rate of 32x over the commonly used SGD with single-precision floating-point arithmetic, we adopt a more aggressive sparsification method in our algorithm. Specifically, we will select the top-k gradient coordinates (in terms of absolute values) at each
iteration and zero the rest of the gradient terms, making the zero-valued elements free from communication and thus significantly reducing the communication bits.

In addition, note that top-$k$ is a biased estimate operator, so the error feedback technique, i.e., incorporating the error made by the sparsification operator into the next step, was employed for the convergence guarantee. We defined an auxiliary sequence $\nu^t = \omega^t - \frac{1}{\gamma} \sum_{m \in M} e^t_m$, where $e^t_m$ is the error of the $t$-th iteration on the $m$-th worker. This sequence has some fascinating properties, which gifts convergence guarantees of the SASG method.

The contributions of this paper can be summarized as follows:

- We propose a communication-efficient algorithm for distributed learning which can reduce both communication rounds and bits.
- We define an auxiliary sequence and develop a new Lyapunov function to establish the convergence analysis of SASG. The convergence rate matches that of the original SGD.
- We conduct extensive experiments to demonstrate the superiority of the proposed SASG algorithm.

2 Algorithm Development

This section describes the motivation for developing the SASG algorithm. We first briefly review two very relevant works: adaptive aggregated method and top-$k$ sparsification technique. Then, we present the scheme of our proposed algorithm.

2.1 Adaptive Aggregated Method

Observing that not all communication rounds between the server and workers contribute equally in the distributed learning system, we employ an adaptive aggregated approach to develop aggregation rules that can skip inefficient communication rounds. This adaptive aggregated approach originates from the lazily aggregated gradient (LAG) method, which designs an adaptive selection to detect workers with slowly-varying gradients and trigger the reuse of outdated gradients. With this adaptive aggregation rule, the following iterative scheme can be obtained

$$
\omega^{t+1} = \omega^t - \frac{1}{M} \left( \sum_{m \in M} \mathcal{T}_k(g_m^t) + \sum_{m \in M_c} \mathcal{T}_k(g_{\tau_m}^{t-\tau_m}) \right)
$$

where $M$ and $M_c$ are the sets of workers that do and do not communicate with the server at the $t$-th iteration, respectively. Staleness $\tau_m$ is determined by the selection of the subset $M_t$: at the $t$-th iteration, if worker $m \in M_t$, the server resets $\tau_m^{t+1} = 1$ and worker $m$ uploads local gradient information; otherwise worker $m$ uploads nothing and the server increases staleness by $\tau_m^{t+1} = \tau_m^t + 1$.

Clearly, it is crucial to devise a principled criterion for selecting the subset of workers $M_t$ that do not communicate with the server at each iteration. The selection rule in LAG method is inspired by the elegant “larger descent per upload” rationale and has been proved to be effective in the deterministic setting \cite{Xin2017}. In addition, a stochastic selection criterion is established in \cite{Xin2017}

$$
\left\| \nabla f_m(\omega^t; \xi^t_m) - \nabla f_m(\omega^{t-\tau_m}; \xi^t_m) \right\|_2^2 \\
\leq \frac{1}{M^2} \sum_{d=1}^D \alpha_d \left\| \omega^{t+1-d} - \omega^{t-d} \right\|_2^2,
$$

where $\{\alpha_d \geq 0\}_{d=1}^D$ are constant weights. This condition is evaluated on the same data $\xi^t_m$ at two different iterations $\omega^t$ and $\omega^{t-\tau_m}$. That is when the difference between two consecutive local gradients on worker $m$ is less than an adaptive threshold, the worker will be included in $M_t$ and the server will skip this communication round.

2.2 Top-$k$ Sparsification

Nevertheless, LAG is far from enough in limited bandwidth scenarios. We can do more beyond LAG, e.g., reducing the
Algorithm 1 SASG algorithm.

Input: Learning rate $\gamma > 0$, maximum delay $D$.

Input: Constant weights $\{\alpha_d\}_{d=1}^{D}$.

Initialize: Set errors $e^0_m = 0$, delay counters $\tau^0_m = 1$, $\forall m \in \mathcal{M}$.

1: for $t = 0, 1, \ldots, T$ do
2: \hspace{0.5cm} Server broadcasts $\omega^t$ to all workers.
3: \hspace{0.5cm} for Worker $m = 1, \ldots, M$ do
4: \hspace{1cm} Compute $\nabla f_m(\omega^t; \xi^t_m)$ and $\nabla f_m(\omega^{t-\tau^t_m}; \xi^t_m)$.
5: \hspace{1cm} $g^t_m = \gamma \nabla f_m(\omega^t; \xi^t_m) + e^t_m$.
6: \hspace{1cm} Check condition (3).
7: \hspace{1cm} if Condition (3) is violated, or $\tau^t_m \geq D$ then
8: \hspace{1.5cm} Worker $m$ uploads $T_k(g^t_m)$.
9: \hspace{1.5cm} $e^{t+1}_m = g^t_m - T_k(g^t_m)$, $\tau^{t+1}_m = 1$.
10: \hspace{1cm} else
11: \hspace{1.5cm} Worker $m$ uploads nothing.
12: \hspace{1.5cm} $e^{t+1}_m = e^t_m$, $\tau^{t+1}_m = \tau^t_m + 1$.
13: \hspace{0.5cm} end if
14: end for
15: Server updates parameter $\omega$ according to (6).
16: end for

transmitted bits of uploaded information rather than just the rounds. There are always two lines of transmission bits reduction: quantization and sparsification. The latter is used in this paper because quantization methods can only reach a maximum compression rate of 32x (1-bit quantization method) in the commonly used single-precision floating-point algorithms. When the dimension $d$ of the task is large, i.e., $d \gg 32$, the sparsification method will far outperform the quantization approach (the maximum compression rate of $d \times$ may be reached).

Although random sparsification methods guarantee the unbiasedness of the compression operator and facilitate theoretical analysis, top-$k$ sparsification tends to achieve better practical results \[14, 37\]. Our algorithm applies the top-$k$ sparsification operator, namely $T_k(\cdot)$. 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 $x \in \mathbb{R}^d$ as

$$(T_k(x)_{\pi(i)} := \begin{cases} (x)_{\pi(i)}, & \text{if } i \leq k, \\ 0, & \text{otherwise,} \end{cases} \quad (4)$$

where $\pi$ is a permutation of set $[d]$ such that $|x|_{\pi(i)} \geq |x|_{\pi(i+1)}$ for $i = 1, \ldots, d$, and $| \cdot |$ denotes the absolute value.

2.3 SASG Algorithm

We propose the SASG algorithm for distributed learning by combining the advantages of the two techniques mentioned above, i.e., sparse communication with adaptive aggregated method. In other words, after the adaptive selection procedure, workers send the sparse information derived by the top-$k$ operator to the parameter server.

However, this combination encounters many challenges. On the one hand, due to non-diminishing variance of the stochastic gradients (i.e., variance in $\|\nabla f_m(\omega^t; \xi^t_m) - \nabla f_m(\omega^{t-\tau^t_m}; \xi^t_m)\|_2^2$), direct extension of LAG does not work well with SGD \[18\]. The SASG method takes advantage of the adaptive selection rule (3) (i.e., computing $\|\nabla f_m(\omega^t; \xi^t_m) - \nabla f_m(\omega^{t-\tau^t_m}; \xi^t_m)\|_2^2$), and we will give a new insight that this condition reduces the variance appeared in the stochastic gradients. According to the Lipschitz continuous property of $\nabla f_m$, we have

$$\|\nabla f_m(\omega^t; \xi^t_m) - \nabla f_m(\omega^{t-\tau^t_m}; \xi^t_m)\|_2^2 \leq L_m \|\omega^t - \omega^{t-\tau^t_m}\|_2^2,$$

which eliminates the inherent variance caused by different data $\xi^t_m$ and $\xi^{t-\tau^t_m}$. We can see that as the iterative sequence \{$\omega^t$\}$_{t=0,1, \ldots}$ converges, the right-hand side of (5) diminishes, and thus the left-hand side of (5) diminishes, which indicates that the selection criterion (3) is effective.

On the other hand, the biased nature of the top-$k$ operator breaks the algorithmic convergence, which needs to be fixed by an error feedback format (a.k.a., error accumulation or memory). The “memory” mechanism has been studied in a series of researches \[14, 16, 21, 38\]. With the error feedback format, the SASG algorithm first sparse the gradient information $\xi^t_m$ and upload the compressed information $T_k(g^t_m)$. When uploading the compressed information $T_k(g^t_m)$, we calculate the compression error $e = g - T_k(g^t_m)$ and store it. The error $e$ will be added to the gradient information and compressed together at the next iteration. Eventually, all the gradient information will be transmitted, despite the delay caused by error feedback.

In a nutshell, we can formulate the iterative scheme of the SASG algorithm as

$$\omega^{t+1} = \omega^t - \frac{1}{M} \left( \sum_{m \in \mathcal{M}^t} T_k(g^t_m) + \sum_{m \in \mathcal{M}^t} T_k(g^{t-\tau^t_m}_m) \right), \quad (6)$$

where

$$g^t_m = \gamma \nabla f_m(\omega^t; \xi^t_m) + e^t_m, \quad e^{t+1}_m = g^t_m - T_k(g^t_m).$$

Our SASG algorithm exploits the advantages of both sparse communication and adaptive aggregated stochastic gradients, aiming at saving the number of communication rounds and bits simultaneously without sacrificing the desired convergence properties. The algorithm is illustrated in Figure I and summarized in Algorithm II. Specifically, during each iteration $t = 0, 1, 2, \ldots$

i. server broadcasts the learning parameter $\omega^t$ to all workers;
ii. worker $m$ calculates the local gradient $\nabla f_m(\omega^t; \xi^t_m)$ and an auxiliary gradient $\nabla f_m(\omega^{t-\tau^t_m}; \xi^t_m)$ for computing the selection rule (3);
iii. worker in $\mathcal{M}^t$ selected by condition (3) will sparse the local information $g^t_m = \gamma \nabla f_m(\omega^t; \xi^t_m) + e^t_m$ and upload $T_k(g^t_m)$ to the server;
iv. server aggregates the fresh sparse gradients $T_k(g^t_m)$ from the selected workers $\mathcal{M}^t$ and the outdated gra-
Lemma 2 shows that the residual errors maintained in Algorithm 1 do not accumulate too much. In the following, we define a crucial auxiliary sequence for our analysis.

**Definition 2.** Let \( \{\nu^t\}_{t=0,1,...} \) be the sequence generated by the SASG algorithm, and the auxiliary variable \( \nu^t \) is defined as

\[
\nu^t = \nu^0 - \frac{1}{M} \sum_{m \in M} e^t_m. 
\]

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

\[
\nu^{t+1} - \nu^t = -\frac{\gamma}{M} \sum_{m \in M} \nabla f_m(\omega^t; \xi^t_m) + \frac{\gamma}{M} \sum_{m \in M} \Delta^t_m, 
\]

where

\[
\Delta^t_m := \nabla f_m(\omega^t; \xi^t_m) - \nabla f_m(\omega^{t-1}; \xi^{t-1}_m).
\]

Following this property, we will give a descent lemma of the loss function.

**Lemma 3.** Under the Assumptions 1, 2, the sequence \( \{\nu^t\}_{t=0,1,...} \) is defined in (7), then the objective function value satisfies (with \( \sigma^2 := \sum_{m=1}^M \sigma^2_m \))

\[
E[F(\nu^{t+1})] - E[F(\nu^t)] 
\leq -(\gamma - \frac{5L + 4}{2} \gamma^2) \cdot \frac{1}{M} E[\|\nabla F(\omega^t)\|^2] 
+ \frac{D}{2ML} + \frac{L + 1}{2} \gamma^4 \frac{\alpha_d}{M^2} \|\nu^{t+1-d} - \omega^{t-d}\|^2 
+ \frac{L^2}{2M^2} E[\sum_{m \in M} e^t_m]^2 + [10(L + 1) + DL] \frac{\gamma^2 \sigma^2}{M}. 
\]

Note that all the terms on the right-hand side of the inequality (9) show up in SGD analysis except \( \|\omega^{t+1-d} - \omega^{t-d}\|^2 \) and \( \sum_{m \in M} e^t_m \)^2, which exist due to stale information and compression errors. To deal with these terms, we will introduce an associated Lyapunov function. With \( F^* \) denotes the optimal value of the problem (1), the Lyapunov function is defined as:

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

where \( \{\beta_d \geq 0\}_{d=1}^D \) are constants that will be determined later. The Lyapunov function is coupled with the selection rule (3) that contains the parameter difference terms. We also highlight that in the definition (10), \( L^t > 0 \) for any \( t \in N \). A direct extension of Lemma 1 gives the following descent lemma.

**Lemma 4.** Under the Assumptions 1, 2 and 3 let \( \sigma^2 := \sum_{m=1}^M \sigma^2_m, B_2 := \sum_{m=1}^M B_2^2 \), if the learning rate \( \gamma \) and constant weights \( \{\alpha_d\}_{d=1}^D \) are chosen properly, the Lyapunov function satisfies

\[
L^{t+1} - L^t \leq -\frac{\gamma}{M} E[\|\nabla F(\omega^t)\|^2] + a_0 \frac{\gamma^2 \sigma^2}{M} + b_0 \frac{\gamma^2 B_2^2}{M} 
- \sum_{d=1}^D c_d \|\omega^{t+1-d} - \omega^{t-d}\|^2. 
\]
where \(c_f, a, b, c_1, \ldots, c_D \geq 0\) depend on the learning rate \(\gamma\), constants \(D, L\) and \(\{\alpha_d, \beta_d\}_{d=1}^D\). More specific explanatory notes can be found in Section 5.

### 3.3 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 the Assumption 1 and 2 let the sequence \(\{\omega_t\}_{t=0,1,\ldots}\) be generated by SASG algorithm, if constant weights \(\{\alpha_d\}_{d=1}^D\) are selected properly and the learning rate is chosen as

\[
\gamma = \min \left\{ \frac{1}{5L + 4 + 16\beta_1} \frac{c_f}{\sqrt{T}} \right\},
\]

where \(c_f > 0\) is a constant, we then have

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

Theorem 1 shows that our algorithm is guaranteed to converge and achieve a sublinear convergence rate 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.

**Remark 1.** In the analysis of Theorem 1 we also delve into the effect of the number of workers \(M\) on the convergence result. From the inequality [16], we have

\[
\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} \left[ \|\nabla F(\omega_t)\|^2 \right] \leq \zeta_1 \frac{M}{T} + \zeta_2 \frac{1}{\sqrt{T}}
\]

where \(\zeta_1, \zeta_2\) are constants independent of \(M\). If the number of iterations satisfies \(T \geq (\frac{\zeta_1}{\zeta_2} M)^2\), then the number of workers does not affect the convergence rate.

## 4 Experiment results

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

### 4.1 Setup

We benchmark the SASG algorithm with lazily aggregated stochastic gradient (LASG) method [18], sparsification method with error feedback [16], and distributed SGD [8]. As empirically demonstrated by Aji et al. [16] that up to 99% of the gradients are not needed to update the model in each iteration, we use the top-1% sparsification operator in SASG and the sparsification method. The training data are distributed across \(M = 10\) workers during all experiments, and each worker uses 10 samples for one training iteration. We manually select the hyperparameter \(D\) and perform a grid search for the values of \(\alpha_d\) to roughly optimize the performance of each algorithm while ensuring convergence. We completed the evaluation for the following three settings, and each experiment was repeated five times.

**MNIST:** The MNIST [43] 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 choose a learning rate \(\gamma = 0.005\). For the adaptive aggregated algorithms SASG and LASG, we set \(D = 10\), \(\alpha = 1/2\gamma\) for \(d = 1, 2, \ldots, 10\).

**CIFAR-10:** The CIFAR-10 [44] dataset consists of 60,000 color images in 10 classes, with 6,000 images per category. We test ResNet18 [45] with all the above mentioned algorithms on CIFAR-10 dataset. Common data augmentation techniques such as random cropping, random flipping and standardization are performed. 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 \(D = 10\), \(\alpha = 1/\gamma\) for \(d = 1, 2, \ldots, 10\).

**CIFAR-100:** The CIFAR-100 [44] dataset consists of 60,000 color images in 100 classes, with 600 images per category. We test VGG16 [46] on CIFAR-100 dataset. Similar data augmentation techniques are performed. The basic learning rate is set to \(\gamma = 0.01\), with a learning rate decay of 0.1 at epoch 30. For SASG and LASG, we set \(D = 10\), \(\alpha = 4/D/\gamma^2\) for \(d = 1, 2, \ldots, 10\).

Our experimental results are based on a PyTorch [47] implementation of all the methods running on an Ubuntu 20.04 machine with an Nvidia RTX-2080Ti GPU.
Table 1: 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        | SASG   | 22721    | 2.96 E+09 |
|                 | LASG   | 37129    | 4.84 E+11 |
|                 | Sparse | 66600    | 8.68 E+09 |
|                 | SGD    | 63200    | 8.23 E+11 |
| ResNet18 CIFAR-10 | SASG  | 83808    | 3.00 E+11 |
|                 | LASG   | 98771    | 3.53 E+13 |
|                 | Sparse | 101800   | 3.64 E+11 |
|                 | SGD    | 101400   | 3.63 E+13 |
| VGG16 CIFAR-10  | SASG   | 140168   | 1.53 E+12 |
|                 | LASG   | 149366   | 1.63 E+14 |
|                 | Sparse | 152600   | 1.66 E+12 |
|                 | SGD    | 152000   | 1.65 E+14 |

4.2 Results

The experimental results of test accuracy and training loss against the number of communication rounds are reported in Figures 2-3, where Figure 2(c) plots the top-5 test accuracy. We set the total number of epochs for training MNIST, CIFAR-10, and CIFAR-100 datasets to 20, 30, and 50, respectively, and trained the same number of epochs for all four algorithms in each setting.

The results in Figure 2 show that our SASG algorithm is more efficient than the previous methods. For example, in Figure 2(a), SASG can obtain higher test accuracy using the same number of communication rounds. Moreover, after completing the training process, SASG can significantly reduce the number of communication rounds while achieving the same performance.

Figure 3 gives the experimental results of the training loss with respect to the number of communication rounds. In Figure 3(a) (training MNIST dataset), we can see 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 3(b) and 3(c)), SASG is also able to significantly reduce the number of communication rounds required to complete the training while guaranteeing convergence.

We choose several baselines 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, 91% for CIFAR-10, and 89% (top-5 accuracy) for CIFAR-100 dataset, respectively. As can be seen from Table 1, 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 1 shows that the SASG algorithm combining adaptive aggregation techniques and sparse communication outperforms the LASG and sparse algorithms by significantly reducing the number of communication bits required for the model to achieve the same performance.
From Algorithm 1 and Lemma 1, we can see that our SASG algorithm is superior to the sparsification method. For the sake of experimental completeness, we also tested different levels of sparsification (top-10%, top-5%, and top-1%). The results in Figure 5 show that our algorithm can significantly reduce the number of communication bits. The results on training epochs (Figures 4(a), 4(c)) show that the SASG algorithm is convergent with the same rate as SGD, which is consistent with our theoretical results. The remaining experimental results (Figures 4(b), 4(d)) show that the SASG algorithm is convergent with the same rate as SGD, which is consistent with our theoretical results.
then we have that
\[ \nu^{t+1} - \nu^t = -\frac{\gamma}{M} \sum_{m \in \mathcal{M}} \nabla f_m(\omega^t; \xi_m^t) + \frac{\gamma}{M} \sum_{m \in \mathcal{M}_c^t} \Delta_m^t. \tag{12} \]

### 5.4 Proof of Lemma \[3\]

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

\[ \begin{align*}
E \left[ F(\nu^{t+1}) \right] &- E \left[ F(\nu^t) \right] \\
&\leq E \left[ \nabla F(\nu^t), \nu^{t+1} - \nu^t \right] + \frac{L}{2} E \|\nu^{t+1} - \nu^t\|^2 \\
&\leq E \left\{ \nabla F(\omega^t), \frac{\gamma}{M} \sum_{m \in \mathcal{M}} \nabla f_m(\omega^t; \xi_m^t) + \frac{\gamma}{M} \sum_{m \in \mathcal{M}_c^t} \Delta_m^t \right\} \\
&\quad + E \left\{ \nabla F(\nu^t) - \nabla F(\omega^t), \nu^{t+1} - \nu^t \right\} + \frac{L}{2} E \|\nu^{t+1} - \nu^t\|^2 \\
&\quad + \frac{1}{2} E \|\nabla F(\nu^t) - \nabla F(\omega^t)\|^2 + \frac{L + 1}{2} E \|\nu^{t+1} - \nu^t\|^2 \\
&\quad + \sum_{d=1}^D \left\{ \frac{\alpha_d}{2ML} + \frac{L}{2} \right\} \|\omega^{t+1-d} - \omega^{t-d}\|^2 \\
&\quad + \frac{L^2}{2} E \|\nu^{t+1-d} - \nu^{t-d}\|^2 + \frac{\gamma^2 D L \sigma^2}{M} \\
&\quad \leq -\left( \gamma - \frac{\gamma^2}{2} \right) \cdot \frac{\gamma}{M} E \|\nabla F(\omega^t)\|^2 + \frac{L + 1}{2} E \|\nu^{t+1} - \nu^t\|^2 \\
&\quad + \sum_{d=1}^D \left\{ \frac{\alpha_d}{2ML} + \frac{L}{2} + \frac{2 \gamma^2 (L + 1) \alpha_d}{M^2} \right\} \|\omega^{t+1-d} - \omega^{t-d}\|^2 \\
&\quad + \frac{L^2}{2M^2} E \left\{ \sum_{m \in \mathcal{M}} \epsilon_m^t \right\}^2 + [10(L + 1) + DL] \frac{\gamma^2 \sigma^2}{M}, \tag{13} \end{align*} \]

where (a) uses the L-smooth property in Assumption \[4\] i.e., \( \forall x, y \in \mathbb{R}^d, \)
\[ F(y) - F(x) \leq \langle \nabla F(x), y - x \rangle + \frac{L}{2} \|y - x\|^2. \]

(b) is obtained by equation \[12\]. (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 analyze \( \langle \nabla F(\omega^t), \Delta_m^t \rangle \) as follows

\[ \begin{align*}
E \langle \nabla F(\omega^t), \Delta_m^t \rangle &= E \langle \nabla F(\omega^t), \nabla f_m(\omega^t; \xi_m^t) - \nabla f_m(\omega^{t-\tau_m}; \xi_m^{t-\tau_m}) \rangle \\
&= E \langle \nabla F(\omega^t), \nabla f_m(\omega^t; \xi_m^t) - \nabla f_m(\omega^{t-\tau_m}; \xi_m^{t-\tau_m}) \rangle \\
&\quad + E \langle \nabla F(\omega^t), \nabla f_m(\omega^{t-\tau_m}; \xi_m^{t-\tau_m}) - \nabla f_m(\omega^{t-\tau_m}; \xi_m^{t-\tau_m}) \rangle. \tag{11} \end{align*} \]

Using the inequality \( \langle a, b \rangle \leq \frac{\varepsilon}{2} \|a\|^2 + \frac{1}{2\varepsilon} \|b\|^2 \) (with \( \varepsilon := L\gamma/M \)) and selection rule \[3\] we have

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

 Noticed that
\[ E \langle \nabla F(\omega^{t-\tau_m}), \nabla f_m(\omega^{t-\tau_m}; \xi_m^t) - \nabla f_m(\omega^{t-\tau_m}; \xi_m^{t-\tau_m}) \rangle = 0, \]
we have that
\[ II = E \langle \nabla F(\omega^t) - \nabla F(\omega^{t-\tau_m}), \nabla f_m(\omega^{t-\tau_m}; \xi_m^t) - \nabla f_m(\omega^{t-\tau_m}; \xi_m^{t-\tau_m}) \rangle \leq \frac{L\gamma\Delta}{2M} \|\nabla f_m(\omega^{t-\tau_m}; \xi_m^t) - \nabla f_m(\omega^{t-\tau_m}; \xi_m^{t-\tau_m})\|^2 \\
&\quad + \frac{L\gamma\Delta}{2M} \|\omega^{t-\tau_m}\|^2 \\
&\leq \frac{\gamma D L \sigma^2}{M} + \frac{L\gamma\Delta}{2M} \|\omega^{t+1-d} - \omega^{t-d}\|^2, \]
where we use the inequality \( \langle a + b \rangle \leq \frac{\varepsilon}{2} \|a\|^2 + \frac{1}{2\varepsilon} \|b\|^2 \) (with \( \varepsilon := \gamma D \)), \( \|\sum_{i=1}^n \theta_i\|^2 \leq n \sum_{i=1}^n \|\theta_i\|^2 \), and Assumption \[2\] That is (with \( \sigma^2 := \sum_{m=1}^M \sigma_m^2 \))
\[ \frac{\gamma}{M} E \langle \nabla F(\omega^t), \sum_{m \in \mathcal{M}_c^t} \Delta_m^t \rangle \leq \frac{\gamma}{M} \sum_{m \in \mathcal{M}_c^t} (I + II) \leq \frac{L^2}{2M} E \|\nabla F(\omega^t)\|^2 + \sum_{d=1}^D \left( \frac{\alpha_d}{2ML} + \frac{L}{2} \right) \|\omega^{t+1-d} - \omega^{t-d}\|^2 \\
&\quad + \frac{\gamma^2 D L \sigma^2}{M}. \]

Further analyzing \( \|\Delta_m^t\|^2 \) and \( \|\nu^{t+1} - \nu^t\|^2 \), we can get (e)
\[ \| \sum_{m \in \mathcal{M}_c^t} \Delta_m^t \|^2 \leq |\mathcal{M}_c^t| \sum_{m \in \mathcal{M}_c^t} \|\Delta_m^t\|^2 \leq |\mathcal{M}_c^t| \sum_{m \in \mathcal{M}_c^t} \|\nabla f_m(\omega^t; \xi_m^t) - \nabla f_m(\omega^{t-\tau_m}; \xi_m^{t-\tau_m})\|^2 \\
&\quad + \|\nabla f_m(\omega^{t-\tau_m}; \xi_m^t) - \nabla f_m(\omega^{t-\tau_m}; \xi_m^{t-\tau_m})\|^2 \\
&\leq 2|\mathcal{M}_c^t| \|\nabla f_m(\omega^t; \xi_m^t) - \nabla f_m(\omega^{t-\tau_m}; \xi_m^{t-\tau_m})\|^2 \\
&\quad + 2|\mathcal{M}_c^t| \sum_{m \in \mathcal{M}_c^t} \|\nabla f_m(\omega^{t-\tau_m}; \xi_m^t) - \nabla f_m(\omega^{t-\tau_m}; \xi_m^{t-\tau_m})\|^2 \leq 2 \sum_{d=1}^D \|\omega^{t+1-d} - \omega^{t-d}\|^2 + 8M \sigma^2, \tag{14} \]
where \( \sigma^2 := \sum_{m=1}^M \sigma_{m}^2 \). Then, using (12), (14) and Assumption 2 we can get

\[
\mathbb{E}[||\nu^{t+1} - \nu^t||^2] = \mathbb{E}[|| - \frac{\gamma}{M} \sum_{m \in M} \nabla f_m(\omega^t; \xi_m^t) + \frac{\gamma}{M} \sum_{m \in M} \Delta_m^t ||^2]
\]

\[
\leq \frac{2\gamma^2}{M^2} \mathbb{E}[|| \sum_{m \in M} \nabla f_m(\omega^t; \xi_m^t) ||^2] + \frac{2\gamma^2}{M^2} \mathbb{E}[|| \sum_{m \in M} \Delta_m^t ||^2]
\]

\[
\leq 4\gamma^2 \mathbb{E}[||\nabla F(\omega^t)||^2] + \frac{4\gamma^2}{M^2} \sum_{d=1}^D \alpha_d ||\omega^{t+1-d} - \omega^{t-d}||^2 + \frac{20\gamma^2 \sigma^2}{M}.
\]

The final result (13).

Organize and summarize these items, and we can get the

\[
\mathbb{E}[||\nu^{t+1} - \nu^t||^2] \leq \frac{[\gamma - (\frac{5L+4}{2} + 8\beta_1)\gamma^2]}{M^2} \mathbb{E}[||\nabla F(\omega^t)||^2] + \frac{\gamma^2}{M^2} \mathbb{E}[|| \sum_{m \in M} \Delta_m^t ||^2]
\]

\[
\leq \frac{[\gamma - (\frac{5L+4}{2} + 8\beta_1)\gamma^2]}{M^2} \mathbb{E}[||\nabla F(\omega^t)||^2] + \frac{\gamma^2}{M^2} \sum_{d=1}^D \alpha_d ||\omega^{t+1-d} - \omega^{t-d}||^2 + \frac{20\gamma^2 \sigma^2}{M}.
\]

(15)

5.5 Proof of Lemma 4

According to the definition (10) and Lemma 3 a direct calculation gives

\[
\mathcal{L}^{t+1} - \mathcal{L}^t = \mathbb{E}[F(\nu^{t+1})] - \mathbb{E}[F(\nu^t)]
\]

\[
+ \sum_{d=1}^D \beta_d ||\omega^{t+2-d} - \omega^{t+1-d}||^2 - \sum_{d=1}^D \beta_d ||\omega^{t+1-d} - \omega^{t-d}||^2
\]

\[
\leq -(\gamma - \frac{5L+4}{2}\gamma^2) \cdot \frac{1}{M} \mathbb{E}[||\nabla F(\omega^t)||^2] + \frac{L^2}{2M^2} \mathbb{E}[|| \sum_{m \in M} \xi_m^t ||^2]
\]

\[
+ \sum_{d=1}^D \left[ \alpha_d \frac{ML}{2} + \frac{2\gamma^2(L+1)\alpha_d}{M^2} \right] ||\omega^{t+1-d} - \omega^{t-d}||^2
\]

\[
+ \frac{[10(L+1) + DL] \gamma^2 \sigma^2}{M} + \beta_1 ||\omega^{t+1} - \omega^{t}||^2
\]

\[
+ \sum_{d=1}^{D-1} (\beta_{d+1} - \beta_d) ||\omega^{t+1-d} - \omega^{t-d}||^2
\]

\[
- \beta_D ||\omega^{t+1-D} - \omega^{t-D}||^2.
\]

Using the properties (12), (15), and Lemma 2 we can obtain the following inequality

\[
||\omega^{t+1} - \omega^{t}||^2 = ||\nu^{t+1} - \nu^t||^2 + \frac{1}{M} \sum_{m \in M} (\xi_m^t - \xi_m^t) ||^2
\]

\[
\leq 2||\nu^{t+1} - \nu^t||^2 + \frac{2}{M^2} \sum_{m \in M} (\epsilon_m^t - \epsilon_m^t) ||^2
\]

\[
\leq 2||\nu^{t+1} - \nu^t||^2 + \frac{16(1-\delta) \gamma^2 B^2}{\delta^2 M}.
\]

where \( B^2 := \sum_{m=1}^M B_m^2 \). Then we can further get

\[
\mathcal{L}^{t+1} - \mathcal{L}^t \leq - \left[ \gamma - (\frac{5L+4}{2} + 8\beta_1)\gamma^2 \right] \cdot \frac{1}{M} \mathbb{E}[||\nabla F(\omega^t)||^2]
\]

\[
- \sum_{d=1}^D \left[ \beta_d - \beta_{d+1} - \frac{\alpha_d}{2M} + \frac{2\gamma^2(L+1)\alpha_d}{M^2} \right] \cdot \mathbb{E}[||\omega^{t+1-d} - \omega^{t-d}||^2]
\]

\[
- \left[ \beta_D - \frac{\alpha_D}{2M} + \frac{2\gamma^2(L+1)\alpha_D}{M^2} + \frac{8\beta_1 \gamma^2 \sigma^2}{M^2} \right]
\]

\[
+ \frac{[10(L+1) + DL + 40\beta_1 \gamma^2 B^2}{M^2} + \frac{(2L^2 + 16\beta_1)(1-\delta) \gamma^2 B^2}{\delta^2 M}.
\]

For the sake of brevity, we make the following notation

\[
\begin{align*}
    c_f &:= \gamma - \left( \frac{5L+4}{2} + 8\beta_1 \right) \gamma^2 \\
    c_d &:= \beta_d - \beta_{d+1} - \left( \frac{\alpha_d}{2L} + \frac{ML}{2} \right) + 8\beta_1 \gamma^2 \sigma^2 \\
    a &:= 10L + 1 + 4\beta_1 M + DL \\
    b &:= (2L^2 + 16\beta_1)(1-\delta) M
\end{align*}
\]

Eventually we have that

\[
\mathcal{L}^{t+1} - \mathcal{L}^t \leq - \left[ \gamma - (\frac{5L+4}{2} + 8\beta_1)\gamma^2 \right] \cdot \frac{1}{M} \mathbb{E}[||\nabla F(\omega^t)||^2]
\]

\[
- \sum_{d=1}^D \left[ \beta_d - \beta_{d+1} - \frac{\alpha_d}{2ML} + \frac{2\gamma^2 \alpha_d(L+1+4\beta_1)}{M^2} \right] \cdot \mathbb{E}[||\omega^{t+1-d} - \omega^{t-d}||^2]
\]

\[
- \left[ \beta_D - \frac{\alpha_D}{2ML} + \frac{2\gamma^2 \alpha_D(L+1+4\beta_1)}{M^2} + 8\beta_1 \gamma^2 \sigma^2 \right]
\]

\[
+ \frac{[10(L+1) + DL + 40\beta_1 \gamma^2 B^2}{M^2} + \frac{(2L^2 + 16\beta_1)(1-\delta) \gamma^2 B^2}{\delta^2 M}.
\]

5.6 Proof of Theorem 1

From Lemma 4 we have

\[
\mathcal{L}^{t+1} - \mathcal{L}^t \leq - \left[ \frac{c_f}{M} \mathbb{E}[||\nabla F(\omega^t)||^2] + a \gamma^2 \sigma^2 + b \gamma^2 \sigma^2 \right]
\]

\[
- \sum_{d=1}^D \left[ \beta_d - \beta_{d+1} - \frac{\alpha_d}{2ML} + \frac{2\gamma^2 \alpha_d(L+1+4\beta_1)}{M^2} \right] \cdot \mathbb{E}[||\omega^{t+1-d} - \omega^{t-d}||^2]
\]

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

\[
\begin{align*}
    \beta_d - \beta_{d+1} - \left[ \frac{\alpha_d}{2ML} + \frac{L}{2} + \frac{2\gamma^2 \alpha_d(L+1+4\beta_1)}{M^2} \right] &= 0 \\
    &d = 1, \ldots, D - 1 \\
    \beta_D - \left[ \frac{\alpha_D}{2ML} + \frac{L}{2} + \frac{2\gamma^2 \alpha_D(L+1+4\beta_1)}{M^2} \right] &= 0
\end{align*}
\]

Solving the linear equations above we can further get

\[
\beta_1 = \frac{(2L+1+4\beta_1)(1-\delta) M}{1 - \frac{5L+4}{2M} \sum_{d=1}^D \alpha_d}
\]

\[
\beta_d = \frac{(2L+1+4\beta_1)(1-\delta) M}{1 - \frac{5L+4}{2M} \sum_{d=1}^D \alpha_d}
\]

\[
\beta_D = \frac{(2L+1+4\beta_1)(1-\delta) M}{1 - \frac{5L+4}{2M} \sum_{d=1}^D \alpha_d}
\]

\[
\beta_1 = \frac{(2L+1+4\beta_1)(1-\delta) M}{1 - \frac{5L+4}{2M} \sum_{d=1}^D \alpha_d}
\]
We then have $c_d \geq 0$, $d = 1, \ldots, D$ and $c_f \geq \gamma/2$. That is
\[
\mathcal{L}^{t+1} - \mathcal{L}^t \leq -\frac{\gamma}{2M} \mathbb{E}[\|\nabla F(\omega^t)\|^2] + (a\sigma^2 + bB^2)\frac{\gamma^2}{M}.
\]
By taking sum $\sum_{t=0}^{T-1} \mathcal{L}^{t+1} - \mathcal{L}^t$, we have
\[
\sum_{t=0}^{T-1} \frac{\gamma}{2} \mathbb{E}[\|\nabla F(\omega^t)\|^2] \leq M(C^0 - \mathcal{T})^2 + (a\sigma^2 + bB^2)T \gamma^2
\]
\[
\leq M(F(\omega^0) - F^* + (a\sigma^2 + bB^2)T \gamma^2.
\]
If we choose the learning rate
\[
\gamma = \min \left\{ \frac{1}{5L + 4 + 16\beta_1}, \frac{c_0}{\sqrt{T}} \right\},
\]
where $c_0 > 0$ is a constant. We then have
\[
\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla F(\omega^t)\|^2] \leq \frac{2M(F(\omega^0) - F^*) + 2T(a\sigma^2 + bB^2)\gamma^2}{T \gamma}
\]
\[
\leq \frac{2(5L + 4 + 16\beta_1)M(F(\omega^0) - F^*)}{T} + \frac{2(F(\omega^0) - F^*)}{c_0 \sqrt{T}}
\]
\[
+ \frac{2c_r(a\sigma^2 + bB^2)}{\sqrt{T}}.
\]
That is
\[
\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla F(\omega^t)\|^2] = O(1/\sqrt{T}).
\]

6 Conclusion

This paper proposes a distributed algorithm with the advantages of sparse communication and adaptive aggregation stochastic gradients, named SASG. 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, our algorithm uses an error feedback format, and we give 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, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “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.-I. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in 2009 IEEE conference on computer vision and pattern recognition. IEEE, 2009, pp. 248–255.

[3] 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.

[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] 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.

[6] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. A. Tucker, K. Yang, and A. Y. Ng, “Large scale distributed deep networks,” in Advances in Neural Information Processing Systems, 2012, pp. 1232–1240.

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

[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] S. Shi, Q. Wang, and X. Chu, “Performance modeling and evaluation of distributed deep learning frameworks on gpus,” in 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PICOM/DataCom/CyberSciTech). IEEE, 2018, pp. 949–957.

[11] 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.

[12] 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.

[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] D. Alistarh, T. Hoeffer, M. Johansson, N. Konstantinov, S. Khiri-Rat, and C. Renggli, “The convergence of sparsified gradient methods,” in Advances in Neural Information Processing Systems, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, 2018, pp. 5977–5987.

[16] A. F. Aji and K. Heafside, “Sparse communication for distributed gradient descent,” in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 440–445.

[17] 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.

[18] T. Chen, Y. Sun, and W. Yin, “LASG: lazily aggregated stochastic gradients for communication-efficient distributed learning,” CoRR, vol. abs/2002.11360, 2020.

[19] 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.

[20] H. Wang, S. Guo, Z. Qu, R. Li, and Z. Liu, “Error-compensated sparsification for communication-efficient decentralized training in edge environment,” IEEE Trans. Parallel Distributed Syst., vol. 33, no. 1, pp. 14–25, 2022.

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

[22] Y. Mao, Z. Zhao, G. Yan, Y. Liu, T. Lan, L. Song, and W. Ding, “Communication efficient federated learning with adaptive quantization,” CoRR, abs/2104.06023, 2021.

[23] 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, H. Larochelle, M. Ranzato,
