Research Article

Decentralized Distributed Deep Learning with Low-Bandwidth Consumption for Smart Constellations

Qingliang Meng, Meiyu Huang , Yao Xu, Naijin Liu, and Xueshuang Xiang

Qian Xuesen Laboratory of Space Technology, China Academy of Space Technology, Beijing, China

Correspondence should be addressed to Naijin Liu; liunaijin@qxslab.cn and Xueshuang Xiang; xiangxueshuang@qxslab.cn

Received 2 August 2021; Accepted 9 October 2021; Published 31 October 2021

Copyright © 2021 Qingliang Meng et al. Exclusive Licensee Beijing Institute of Technology Press. Distributed under a Creative Commons Attribution License (CC BY 4.0).

For the space-based remote sensing system, onboard intelligent processing based on deep learning has become an inevitable trend. To adapt to the dynamic changes of the observation scenes, there is an urgent need to perform distributed deep learning onboard to fully utilize the plentiful real-time sensing data of multiple satellites from a smart constellation. However, the network bandwidth of the smart constellation is very limited. Therefore, it is of great significance to carry out distributed training research in a low-bandwidth environment. This paper proposes a Randomized Decentralized Parallel Stochastic Gradient Descent (RD-PSGD) method for distributed training in a low-bandwidth network. To reduce the communication cost, each node in RD-PSGD just randomly transfers part of the information of the local intelligent model to its neighborhood. We further speed up the algorithm by optimizing the programming of random index generation and parameter extraction. For the first time, we theoretically analyze the convergence property of the proposed RD-PSGD and validate the advantage of this method by simulation experiments on various distributed training tasks for image classification on different benchmark datasets and deep learning network architectures. The results show that RD-PSGD can effectively save the time and bandwidth cost of distributed training and reduce the complexity of parameter selection compared with the TopK-based method. The method proposed in this paper provides a new perspective for the study of onboard intelligent processing, especially for online learning on a smart satellite constellation.

1. Introduction

With the breakthrough development of artificial intelligence and the rapid improvement of onboard computing and storage capabilities, it is an inevitable trend for remote sensing satellite systems to directly generate information required by users through intelligent processing onboard [1, 2]. As earth observation scenes usually present high dynamic characteristics, the traditional *training on the ground-prediction onboard* working mode cannot satisfy the real-time and accurate perception requirement of users. There is an urgent need to learn and update the intelligent model onboard to adapt to the dynamic changes of the scenes.

Affected by factors such as satellite orbits, payloads, physical characteristics of target objects, and imaging methods, more and more intelligent tasks, such as emergency observations in disaster areas and searching for missing Malaysia Airlines, require the cooperation of multiple satellites. Therefore, relying only on the observation data of a single satellite makes it difficult to achieve precise learn-

ing of the global intelligent interpretation model for these cooperation tasks.

Benefited from the development of satellite technology and the reduction of satellite development cost, the number of satellites in orbit has increased sharply and the intersatellite networks have been gradually established, which build the foundation for multisatellite collaboration or smart constellation. Based on the collaborative working mode, it is available to integrate the real-time sensing data and computing capabilities of multiple satellites through distributed deep learning technology. Compared with learning the intelligent model on only one satellite, distributed deep learning can achieve the overall optimization and global convergence of the intelligent model without global information or human intervention and thus improve the collaborative perception and cognitive capabilities of the space-based remote sensing system. Depending on how the tasks are parallelized across satellites, the distributed training can be divided into two categories: model parallelism and data parallelism [3]. Model parallelism means training

different parts of networks with multiple workers, which is mainly used for training very large models [4, 5]. In contrast, data parallelism refers to the strategy of partitioning the dataset into smaller splits [6] or collecting data on different devices independently, which is the scenario we study here.

However, due to the particularity of the operating environment of the satellites, which is different from the cluster system on the ground, the network bandwidth of the smart constellation is often very limited. Therefore, it is of great significance and practical urgency to develop distributed deep learning research under a low-bandwidth environment. To deal with this problem, the traditional distributed training methods can be improved from two aspects.

The first aspect is to use decentralized network structures [7–9]. In the traditional centralized network structure, all nodes need to transmit their trained parameters or gradients of the intelligent model to the central server, waiting for the parameter or gradient fusion, and then receive the fused parameters or gradients from the central server. Instead, the decentralized network structure removes the central parameter server and allows all nodes to exchange parameters or gradients with adjacent nodes. In this way, the pressure of communication can be shared with each node to avoid congestion and improve the real-time capability of distributed training.

The second aspect is to reduce data transmission and save bandwidth usage. This can be achieved by communication delay, quantization, and sparsification. These techniques can be used independently or in combination to develop a comprehensive distributed training framework, such as sparse binary compression [10]. Communication delay means communicating after training several batches locally instead of one batch, which reduces the frequency of communication. This technique is used in Local SGD (Stochastic Gradient Descent) [11, 12], federated averaging [13], and federated learning [14]. Quantization means using a low-precision value to replace the original precise parameters. For example, QSGD [15] (Quantized Stochastic Gradient Descent) adjust the number of bits sent per iteration to smoothly trade off the communication bandwidth against convergence time. The TernGrad approach [16] requires only three numerical levels $\{-1, 0, 1\}$, which aggressively reduce communication time. DoReFa-Net [17] stochastically quantized gradients to low bitwidth numbers.

This paper mainly focuses on using the sparsification technique to overcome communication bottleneck in a low-bandwidth environment. In the sparsification method, only part of network parameters or gradients is sent. For example, Alistarh et al. [18] proposed sorting the gradients in decreasing order of magnitude and truncating the gradient to its top K components. They prove the convergence of this TopK-based method analytically. Deep gradient compression [19] also uses the gradient magnitude as a simple heuristic for importance and employs momentum correction, local gradient clipping, momentum factor masking, and warm-up training to preserve accuracy. Tsuzuku et al. [20] used the variance of gradients as a signal for compression. Adacom [21] adaptively tunes the compression rate based on local gradient activity. Amiri and Gündüz [22] considered the physical layer aspects of wireless com-

munication and proposed an analog computation scheme A-DSGD (Analog Distributed Stochastic Gradient Descent).

We notice that all these methods choose the magnitude or variance as the indicator of importance, sort the gradients by importance, and then truncate the gradients to top K components. For a deep neural network with millions to billions of parameters, this process could be time-consuming due to its high complexity. In this paper, a novel method named RD-PSGD (Randomized Decentralized Parallel Stochastic Gradient Descent) for reducing communication bandwidth by parameter sparsification is proposed. Unlike existing methods utilizing TopK sparsification, in each iteration, we select the parameter to be transferred in a random way, which greatly reduces the complexity of parameter screening. We prove that this strategy can also guarantee the convergence by both theoretical and experimental analysis and optimize the programming to fully leverage the advantage of random parameter sparsification.

The remainder of this paper is organized as follows: Section 2 proposes the RD-PSGD method for smart satellite constellations, and Section 3 presents the programming optimization for the proposed method. Section 4 validates our method by experiments. Conclusions and future works are presented in Section 5.

2. Methodology

In this section, we first introduce the distributed training framework for smart satellite constellations. Then, based on the framework, we briefly review a classic distributed deep learning training method, namely, the D-PSGD (Decentralized Parallel Stochastic Gradient Descent) method [7]. Lastly, motivated by the analysis of the communication complexity of D-PSGD, we propose our RD-PSGD method more suitable for a low-bandwidth satellite constellation environment.

2.1. Distributed Training Framework for Smart Satellite Constellations. The distributed training framework for smart satellite constellations is shown in Figure 1. In the framework, each satellite collects remote sensing images in real time and stores them locally. Besides, each satellite is equipped with an intelligent model to perform a certain perception or cognitive task, such as object detection and scene classification of the collected remote sensing images. If every satellite keeps its intelligent model unchanged, it cannot deal with remote sensing images of dynamic scenes or objects, and thus, it is necessary to learn and update the intelligent model onboard, whereas only training the intelligent model with its own local dataset is hard to achieve overall optimization and global convergence. Instead, for distributed training on a satellite constellation, multiple satellites can be connected by intersatellite links to form a communication network. A satellite is called a worker node in this network. A node not only trains the model using its own dataset but also exchanges and averages model parameters with adjacent nodes. In this way, the perception and cognitive capabilities of the satellite constellation can be fully utilized.

In this paper, we assume that the network is of a fixed ring structure, and there is no centralized parameter server

FIGURE 1: Illustration of the distributed training framework for smart satellite constellations.

in the system. As we mentioned earlier, this design can effectively avoid congestion of communication. However, the RD-PSGD method proposed later can be easily applied to other network structures.

2.2. A Review of D-PSGD. The distributed training method proposed in this paper is built upon D-PSGD [7], which is a very popular decentralized distributed deep learning technique. Lian et al. [7] proved the convergence of D-PSGD and showed that D-PSGD outperforms centralized algorithms. The D-PSGD [7] considers the following stochastic optimization problem:

$$\min_{x \in \mathbb{R}^N} f(x) := \mathbb{E}_{\xi \in \mathcal{D}} F(x; \xi), \quad (1)$$

where $\mathcal{D}$ is the training dataset and ξ is a data sample. $x \in \mathbb{R}^N$ denotes the serialized parameter vector of an intelligent model with a specified deep learning network architecture, which is usually a convolutional neural network (such as ResNets [23]), and $F(x; \xi)$ is a predefined loss function. The above optimization problem can be efficiently and effectively solved by the SGD algorithm [24].

To design parallel SGD algorithms on a decentralized network, the data are distributed onto all nodes such that the original objective defined in (1) can be rewritten as

$$\min_{x \in \mathbb{R}^N} f(x) = \frac{1}{n} \sum_{i=1}^n \mathbb{E}_{\xi \in \mathcal{D}_i} F_i(x; \xi). \quad (2)$$

Define $f_i(x) := \mathbb{E}_{\xi \in \mathcal{D}_i} F_i(x; \xi)$ with $F_i(\cdot; \cdot) = F(\cdot; \cdot)$. There are two ways to distribute $\mathcal{D}$: shared data, $\mathcal{D}_i = \mathcal{D}$; local data with the same distribution, i.e., $\mathcal{D}_i$ has the same distribution of $\mathcal{D}$ on local data, which is used in this paper.

2.2.1. Definitions and Notations. Throughout this paper, the following definitions and notations are used:

- (i) $\|\cdot\|$ denotes the L_2 norm of a vector or the spectral norm of a matrix
- (ii) $\|\cdot\|_F$ denotes the Frobenius norm of a matrix

- (iii) 1_n denotes the column vector of $\mathbb{R}^n$ with 1 for all elements
- (iv) $|S|$ denotes the size of a set S
- (v) $[N] := \{1, 2, \dots, N\}$
- (vi) $\mathcal{T}(a)$ denotes the set of locations of nonzero entries in vector a
- (vii) $\nabla g(\cdot)$ denotes the gradient of a function g
- (viii) $x_{k,i}$ denotes the parameter vector at iteration k on the i th node
- (ix) $X_k := [x_{k,1}, x_{k,2}, \dots, x_{k,n}] \in \mathbb{R}^{N \times n}$ denotes the concatenation of the local parameter vectors at iteration k
- (x) $W \in \mathbb{R}^{n \times n}$ denotes the weight matrix, i.e., the network topology, satisfying (i) $W_{ij} \in [0, 1]$ and (ii) $\sum_j W_{ji} = 1$. We use $\text{Deg}(W)$ to denote the degree of network W . For a ring-structured network, W is denoted by

$$= \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & 0 & \cdots & 0 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \cdots & 0 & 0 \\ & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{1}{3} & 0 & 0 & \cdots & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \in \mathbb{R}^{n \times n} \quad (3)$$

- (xi) $P_\alpha \in \mathbb{R}^{N \times N}$ denotes the matrix $\text{Diag}(a)$, where $a \in \mathbb{R}^N$ is a vector of independent Bernoulli random variables, each of which has probability α of 1
- (xii) $B(n, \alpha)$ denotes the binomial distribution
- (xiii) $\nabla f(X_k) := [\nabla f_1(x_{k,1}), \dots, \nabla f_n(x_{k,n})]$

Input: Initial parameter guess $x_{0,i} = x_0$, learning rate γ , weight matrix W and maximum iteration K .
Output: $x_{K,i}$.
1: **for** $k = 0, 1, 2, \dots, K$ **do**
2: Randomly sample $\xi_{k,i}$ from local dataset $\mathcal{D}_i$;
3: Compute gradient at current optimization parameters: $\nabla F_i(x_{k,i}; \xi_{k,i})$;
4: Compute neighborhood weighted average by fetching optimization parameters from neighbors: $x_{k+1/2,i} = \sum_{j=1}^n x_{k,j} W_{ji}$;
5: Update $x_{k+1,i} = x_{k+1/2,i} - \gamma \nabla F_i(x_{k,i}; \xi_{k,i})$;
6: **end for**

ALGORITHM 1: D-PSGD on the i th node.

Input: Initial parameter guess $x_{0,i} = x_0$, learning rate γ , weight matrix W and maximum iteration K .
Output: $x_{K,i}$.
1: **for** $k = 0, 1, 2, \dots, K$
2: **At the 1-th node**, construct a vector a_k , where each entry has probability α of 1, and then transfer this vector to other nodes;
3: Randomly sample $\xi_{k,i}$ from local dataset $\mathcal{D}_i$;
4: Compute gradient at current optimization parameters: $\nabla F_i(x_{k,i}; \xi_{k,i})$;
5: Compute neighborhood weighted average by fetching optimization parameters indicated in a_k from neighbors:
 $x_{k+1/2,i}(s) = \sum_{j=1}^n x_{k,j}(s) W_{ji}, s \in \mathcal{T}(a_k)$;
6: Let $x_{k+1/2,i}(s) = x_{k,i}(s), s \in [N] \setminus \mathcal{T}(a_k)$;
7: Update $x_{k+1,i} = x_{k+1/2,i} - \gamma \nabla F_i(x_{k,i}; \xi_{k,i})$;
8: **end for**

ALGORITHM 2: RD-PSGD on the i th node.

$$(xiv) \nabla F(X_k, \xi_k) := [\nabla F_1(x_{k,1}; \xi_{k,1}), \dots, \nabla F_n(x_{k,n}; \xi_{k,n})]$$

By the definitions and notations, we have the D-PSGD in Algorithm 1.

2.3. RD-PSGD. It is easy to check that the communication complexity of D-PSGD is $O(N \cdot \text{Deg}(\text{network}))$. Whereas in a network with low communication capacity, D-PSGD may suffer from latency. Here, we introduce a random transferring technique to reduce communication, named RD-PSGD (Randomized Decentralized Parallel Stochastic Gradient Descent). Specifically, in the process of model synchronization with adjacent nodes, only a part of the parameters of the intelligent model need to be randomly selected for transmission, which reduces the bandwidth cost and the complexity of parameter filtering compared with the TopK-based methods [18, 19]. The details are stated in Algorithm 2.

Now, we prove the convergence of RD-PSGD. Firstly, in D-PSGD, it makes a commonly used assumption on the weight matrix.

Assumption 1. W is a symmetric stochastic matrix with $\rho_W := (\max \{|\lambda_2(W)|, \lambda_n(W)|\})^2 < 1$.

From a global view, at the k th iteration, Algorithm 1 can be viewed as

$$X_{k+1} = X_k W - \gamma \nabla F(X_k, \xi_k). \quad (4)$$

To prove the convergence of D-PSGD, it needs a critical property on the weight matrix W , i.e., Lemma 5 in the orig-

inal publication of D-PSGD [7], which we reformulate as follows:

Lemma 2. Under Assumption 1, for $\forall X \in \mathbb{R}^{N \times n}, i \in \{1, 2, \dots, n\}, k \in \mathbb{N}$, we have $\|X 1_n/n - X W^k e_i\|^2 \leq \rho_W^k \|X\|^2$.

Similarly, at the k th iteration, Algorithm 2 can be viewed as

$$X_{k+1} = P_\alpha X_k W + (I - P_\alpha) X_k - \gamma \nabla F(X_k, \xi_k). \quad (5)$$

Denote the neighborhood weighted average as

$$\mathcal{G}_{\alpha, W}(X_k) := P_\alpha X_k W + (I - P_\alpha) X_k, \quad (6)$$

with α the sparsity ratio ($\alpha \in (0, 1]$). Then, for each node, we just need to transfer the corresponding information specified in P_α . Then, the communication complexity is almost $O(\alpha N \cdot \text{Deg}(\text{network}))$. When $\alpha < 1$, the communication complexity is reduced. Denote $\mathcal{G}_{\alpha, W}^k$ the operator with k times composite of $\mathcal{G}_{\alpha, W}$. Compared with D-PSGD, we need to prove a similar property of $\mathcal{G}_{\alpha, W}$ like W in Lemma 2, to complete the proof of the convergence of RD-PSGD:

Lemma 3. Under Assumption 1, for $\forall X \in \mathbb{R}^{N \times n}, i \in \{1, 2, \dots, n\}, k \in \mathbb{N}$, we have $\|X 1_n/n - \mathcal{G}_{\alpha, W}^k(X) e_i\|^2 \leq \rho_W^\alpha \|X\|_F^2$ with probability at least

$$\left(\sum_{j=s}^k \mathcal{P}(Y=j) \right)^N, \quad (7)$$

where $Y \sim B(k, \alpha)$.

Proof. Denote a_k the random vector used to construct P_α in the k th operator $\mathcal{G}_{\alpha, W}^k$ in $\mathcal{G}_{\alpha, W}^k$. Let $t_i := [a_1(i), a_2(i), \dots, a_k(i)] \in \mathbb{R}^k$, each entry of which indicates whether the i th entry of local optimization parameters will be averaged at the k th iteration in Algorithm 2. Thus, t_i is a vector of independent Bernoulli random variables. Denote $\tilde{X}_k = \mathcal{G}_{\alpha, W}^k(X)$ and $\tilde{X}_{k,i}$ the i th row of matrix $\tilde{X}_k$. Denote X_i the i th row of matrix X . By (6) and the definition of P_α , a_k and t_i , we have

$$\tilde{X}_{k,i} = X_i W^{t_i(1)} W^{t_i(2)} \dots W^{t_i(k)}. \quad (8)$$

Let $W^\infty := \lim_{k \rightarrow \infty} W^k$. Since W is doubly stochastic and $\rho_W < 1$, we have $1_n/n = W^\infty e_i$, $\forall i$. Then, we have

$$\begin{aligned} \left(\frac{X_i 1_n}{n} - X \wedge_{k,i} e_i \right)^2 &= \left(\frac{X_i 1_n}{n} - X_i W^{t_i(1)} W^{t_i(2)} \dots W^{t_i(k)} e_i \right)^2 \\ &= \left(X_i W^\infty e_i - X_i W^{t_i(1)} W^{t_i(2)} \dots W^{t_i(k)} e_i \right)^2 \\ &\leq \|X_i\|^2 \|W^\infty - W^{t_i(1)} W^{t_i(2)} \dots W^{t_i(k)}\|^2 \|e_i\|^2 \\ &\leq \|X_i\|^2 \|W^\infty - W^{Y_i}\|^2 \|e_i\|^2 \leq \|X_i\|^2 \rho_W^{Y_i}, \end{aligned} \quad (9)$$

where $Y_i \sim B(k, \alpha)$ denotes the times of 1 in t_i . Since $\rho_W < 1$, we have

$$\begin{aligned} \mathcal{P} \left(\left(\frac{X_{k,i} 1_n}{n} - X \wedge_{k,i} e_i \right)^2 \leq \|X_i\|^2 \rho_W^s \right) \\ = \sum_{j=s}^k \mathcal{P} \left(\left(\frac{X_{k,i} 1_n}{n} - X \wedge_{k,i} e_i \right)^2 \leq \|X_i\|^2 \rho_W^j \right) \\ = \sum_{j=s}^k \mathcal{P}(Y_i = j). \end{aligned} \quad (10)$$

If $\forall i$, we have

$$\left(\frac{X_{k,i} 1_n}{n} - X \wedge_{k,i} e_i \right)^2 \leq \|X_i\|^2 \rho_W^s, \quad (11)$$

then it is easy to check

$$\begin{aligned} \left\| \frac{X 1_n}{n} - \mathcal{G}_{\alpha, W}^k(X) e_i \right\|^2 &= \sum_{i=1}^N \left(\frac{X_{k,i} 1_n}{n} - X \wedge_{k,i} e_i \right)^2 \\ &\leq \sum_{i=1}^N \|X_i\|^2 \rho_W^s = \rho_W^s \|X\|_F^2. \end{aligned} \quad (12)$$

Thus,

$$\begin{aligned} \mathcal{P} \left(\left\| \frac{X 1_n}{n} - \mathcal{G}_{\alpha, W}^k(X) e_i \right\|^2 \leq \rho_W^s \|X\|_F^2 \right) \\ \geq \prod_{i=1}^N \mathcal{P} \left(\left(\frac{X_{k,i} 1_n}{n} - X \wedge_{k,i} e_i \right)^2 \leq \|X_{k,i}\|^2 \rho_W^s \right) \\ = \prod_{i=1}^N \left(\sum_{j=s}^k \mathcal{P}(Y_i = j) \right), \end{aligned} \quad (13)$$

combined with that $\{Y_i\}$ are independent, yielding

$$\mathcal{P} \left(\left\| \frac{X 1_n}{n} - \mathcal{G}_{\alpha, W}^k(X) e_i \right\|^2 \leq \rho_W^s \|X\|_F^2 \right) \geq \left(\sum_{j=s}^k \mathcal{P}(Y=j) \right)^N. \quad (14)$$

This ends the proof. $\square$

Denote

$$T_\beta(\alpha) = \frac{1}{k} \min_s \left\{ s \in \mathbb{Z}^+, \sum_{j=s}^k \mathcal{P}(Y=j) > \beta \right\}, \quad (15)$$

which represents the decaying rate under probability than β . Suppose empirical, we have

$$\left\| \frac{X 1_n}{n} - \mathcal{G}_{\alpha, W}^k(X) e_i \right\|^2 \leq \rho_\alpha^k \|X\|_F^2. \quad (16)$$

Figure 2 shows the numerical ρ_α and theoretical $\rho_\alpha := \rho_W^{T_\beta(\alpha)}$ in a ring network of 8 nodes. The value of α varies from 0.1 to 1 with step 0.05 under different minimal probability $\beta = 0.4, 0.6, 0.8, 0.9$. The theoretical and numerical values of the convergence rate are in good agreement, indicating that RD-PSGD can meet the convergence requirements.

Assuming that the average speed of the intersatellite link is 2 MB/s, the effect of utilizing RD-PSGD is shown in Table 1. It can be seen that RD-PSGD can reduce bandwidth and time cost linearly and thus make distributed training in low-bandwidth environment practical.

3. Programming Optimization

The refinement of the proposed RD-PSGD algorithm over D-PSGD also introduces additional programming problems. For the D-PSGD method, a full cycle of parameter transmission for model synchronization, i.e., step 4 in Algorithm 1, can be divided into three parts: serialization of the parameters of the intelligent model (t_{serial}), communication of the parameters (t_{comm}), and deserialization of the parameters to recover the deep learning network structure of the intelligent model (t_{deserial}); namely, the time cost of

FIGURE 2: Numerical estimation and theoretical estimation of ρ_α when varying α from 0.1 to 1 with step 0.05 under different minimal probability $\beta = 0.4, 0.6, 0.8, 0.9$ of ring network topology.

TABLE 1: Comparison of theoretical bandwidth and time cost of a complete exchange of the widely used deep learning models, namely, ResNets, using D-PSGD and RD-PSGD ($\alpha = 0.1$).

		Bandwidth (MB)	Time (s)
ResNet-20 [23]	D-PSGD	45.7	22.8
	RD-PSGD	4.6	2.3
ResNet-50 [23]	D-PSGD	97.7	48.8
	RD-PSGD	9.8	4.9

each cycle of parameter transmission for D-PSGD is

$$t_{\text{D-PSGD}} = t_{\text{serial}} + t_{\text{comm}} + t_{\text{deserial}}. \quad (17)$$

For RD-PSGD, aiming at low-bandwidth communication of the parameters (t'_{comm}), extra steps are needed, including generation of the random index that indicates which parameters need to be transmitted and transmission of the random index (t_{rand}), i.e., step 2 in Algorithm 2, and extraction of parameters to be transmitted according to the random index (t_{extract}) and expansion of the extracted sparse parameters into dense network parameters (t_{expand}) in steps 5 and 6 in Algorithm 2; namely, the time cost of each cycle of parameter transmission for RD-PSGD is

$$t_{\text{RD-PSGD}} = t_{\text{rand}} + t_{\text{serial}} + t_{\text{extract}} + t'_{\text{comm}} + t_{\text{expand}} + t_{\text{deserial}}. \quad (18)$$

Then, the difference between $t_{\text{D-PSGD}}$ and $t_{\text{RD-PSGD}}$ is

$$t_{\text{D-PSGD}} - t_{\text{RD-PSGD}} = (t_{\text{comm}} - t'_{\text{comm}}) - (t_{\text{rand}} + t_{\text{expand}} + t_{\text{extract}}). \quad (19)$$

Therefore, RD-PSGD has lower time complexity only if

$$t_{\text{comm}} - t'_{\text{comm}} > t_{\text{rand}} + t_{\text{expand}} + t_{\text{extract}}. \quad (20)$$

Equation (19) shows that

- (i) the generation and transmission of the random index and the extraction and expansion of the parameters of the intelligent model should be optimized as far as possible to give full play to the acceleration effect of RD-PSGD
- (ii) the lower the network bandwidth, the higher the value of $t_{\text{comm}} - t'_{\text{comm}}$, thus the more obvious the acceleration

In order to improve the acceleration effect of RD-PSGD, we optimize the programming from two aspects.

Random index generation. We first observe step 2 of Algorithm 2, in which the random index vector a_k is generated from Bernoulli distribution with probability α . Suppose the total number of the parameters of the intelligent model is N , this direct approach has to perform N times of random number generation and thresholding regardless of α . Consider a_k is a binary vector, the indices of elements with value 1 are denoted by

$$A_{1k} = \{i | a_k[i] = 1\}, \quad (21)$$

where the size of A_{1k} is αN . The difference of adjacent elements of A_{1k} is

$$A_{1k, \text{diff}} = \{A_{1k}[j+1] - A_{1k}[j]\}, \quad j = 0, 1, \dots, \alpha N - 1. \quad (22)$$

We can infer that the elements of $A_{1k, \text{diff}}$ obey geometric distribution. Therefore, if we transform the random index vector a_k into $A_{1k, \text{diff}}$, we need only perform αN times of random number generation. Another advantage of transforming a_k into $A_{1k, \text{diff}}$ is reducing bandwidth cost when the sparsity ratio is high. For example, when $\alpha = 0.05$, an 8-bit integer is enough to represent the value of each element in $A_{1k, \text{diff}}$. In total, $A_{1k, \text{diff}}$ employ $0.05 \times 8N = 0.4N$ bit of data, which is less than N bit using 0-1 Boolean representation.

Parameter extraction. Regarding serialization, deserialization, extraction, and expansion operations in steps 5 and 6 of Algorithm 2, the general implementation refers to a series of time-consuming for-loop operations. Aiming at speeding up RD-PSGD, the built-in functions in Numpy and PyTorch is used as much as possible to take advantage of dedicated CPU and GPU acceleration for vector and tensor operations.

FIGURE 3: Training loss and accuracy of ResNet-20 on CIFAR10 with different sparsity ratios using RD-PSGD.

4. Experiments

We evaluate our RD-PSGD methods on several distributed training tasks for image classification on different benchmark datasets and deep learning network architectures by ground simulation. Specifically, we studied the performance of ResNet-20 [23] and VGG-16 [25] on CIFAR10 [26] and ResNet-50 [23] on ImageNet-1k [27]. In our experiments, we test RD-PSGD on a ring-structured network which consists of 8 worker nodes, each of which is simulated by a workstation with a RTX 3070 GPU. The dataset is randomly split into 8 subsets to simulate the different data collected by each satellite. The algorithm is implemented using PyTorch with Gloo as a communication backend.

Models are trained using SGD with momentum and weight decay on every single node. The setup of hyperparameter is as follows:

- (1) Batch size: 32 for ResNet-20, 128 for VGG-16, and 64 for ResNet-50
- (2) Learning rate γ : For ResNet-20, starting from 0.1, and divided by 10 at the 80th and 120th epoch. For VGG-16, starting from 0.5 and divided by 2 every 25 epochs. For ResNet-50, starting from 0.1 and divided by 10 every 30 epochs
- (3) Number of epochs and iterations: For ResNet-20, the maximum number of epochs is 200, and there are 196 iterations per epoch on a single node. For VGG-16, there are 200 epochs and 98×200 iterations in total. For ResNet-50, there are 90 epochs and 2500×90 iterations
- (4) Momentum: 0.9
- (5) Weight decay: 0.0001
- (6) Synchronization delay: For ResNet-20 and VGG-16, the parameter synchronization is performed after every batch, while for ResNet-50, the parameter synchronization is performed after every 100 batches

The convergence and bandwidth saving effect of RD-PSGD are analyzed. The acceleration effect of programming optimization is evaluated, and the time cost is compared with the TopK-based methods [18, 19].

4.1. Convergence of RD-PSGD. Figures 3–5 show the convergence of the loss function and prediction accuracy of different models with different sparsity ratios using RG-PSGD. As shown in Figure 3, training ResNet-20 on CIFAR-10 can achieve convergence under different sparsity rates with no accuracy loss. When the sparsity ratio is 0.1, i.e., the transmitted model parameters are reduced by 10 times, the training accuracy can still reach more than 90%. A similar phenomenon also presents when training VGG-16 on CIFAR-10 and training ResNet-50 on ImageNet-1k, as shown in Figures 4 and 5. These results demonstrate that the proposed RD-PSGD method can converge on different distributed training tasks under different sparsity rates.

4.2. Bandwidth Cost. The bandwidth cost of one full cycle of transmitting ResNet-50 from one node to another is shown in Figure 6. When the sparsity ratio is close to 1, the bandwidth cost of RD-PSGD is higher than D-PSGD, because an extra vector containing the indices of parameters is transmitted. After the critical value around 0.8, as the sparsity ratio continues to descend, the bandwidth cost decreases approximately linearly.

4.3. Programming Optimization. We evaluate the time cost of training ResNet-50 on ImageNet for one epoch using different methods. Table 2 shows that, without programming optimization, the time cost of RD-PSGD is even higher than D-PSGD. After the programming optimization is applied, the average time cost of generation of random index reduces from 0.432 s to 0.056 s, which is speeded up by $\sim 8x$. Meanwhile, the average time cost of extraction and expansion of parameters reduces from 12.855 s to 0.431 s, which is speeded up by $\sim 30x$. And the whole speed up effect of programming optimization is shown in Table 2, where the time

FIGURE 4: Training loss and accuracy of VGG-16 on CIFAR10 with different sparsity ratios using RD-PSGD.

FIGURE 5: Training loss and accuracy of ResNet-50 on ImageNet-1k with different sparsity ratios using RD-PSGD.

FIGURE 6: The bandwidth cost of one full parameter transmission cycle of ResNet-50 on a single node using RD-PSGD.

cost of RD-PSGD is reduced to 781.15s, less than that of D-PSGD.

As we mentioned earlier, the lower the bandwidth, the more obvious the acceleration. To prove this, we use the Trickle tool to limit the bandwidth to no more than 200 kb/s. We define the time cost of parameter synchronization in one epoch as the whole time cost deducting the time needed for gradient calculation and backpropagation. The result in Table 3 shows that the speed-up ratio is indeed higher when the bandwidth is lower, which is more relevant to the sparsity rate.

TABLE 2: The time cost of training ResNet-50 on ImageNet for one epoch with and without programming optimization using different methods.

	1 node	8-node D-PSGD	8-node RD-PSGD ($\alpha = 0.1$)
Time cost(s) (no opt.)	6069.36	838.03	1078.5
Time cost(s) (with opt.)	6069.36	838.03	781.15

TABLE 3: The time cost of parameter synchronization of training ResNet-50 on ImageNet for one epoch using different distributed training methods with 8 nodes.

	D-PSGD	RD-PSGD ($\alpha = 0.1$)
High bandwidth	87.7	33.9
Low bandwidth (~ 200 kb/s)	285.6	58.3

4.4. Comparison with the TopK-Based Methods. Table 4 shows the time cost of parameter extraction of the TopK-based methods [18, 19] and RD-PSGD at sparsity ratios 0.1 and 0.5. The result indicates that RD-PSGD can accelerate the parameter extraction by $4 \times \sim 7 \times$ compared with the TopK-based methods, through selecting the parameter to be

TABLE 4: Comparison of time cost of parameter extraction between the TopK-based methods [18, 19] and RD-PSGD ($\alpha = 0.1, 0.5$).

		ResNet-20	VGG-16	ResNet-50
$\alpha = 0.1$	TopK	0.0060	0.369	0.659
	RD-PSGD	0.0015	0.059	0.099
$\alpha = 0.5$	TopK	0.0134	0.932	1.619
	RD-PSGD	0.0033	0.199	0.419

transferred in a random way instead of screening the parameters according to their magnitudes, which requires sorting operation of high time complexity.

5. Conclusion and Future Work

This paper proposed RD-PSGD, a decentralized distributed training algorithm with low-bandwidth consumption for a smart constellation that randomly selects a part of model parameters to transmit. We prove the convergence of this algorithm theoretically and optimize the programming to further speed up the practical application. The experiment results show that the convergence and acceleration requirements in a low-bandwidth environment can be met, and the algorithm can outperform the TopK-based method in parameter extraction, which shows that this is a promising method for future distributed deep learning on a space-based remote sensing system.

The work in this paper can be improved in the future. Firstly, the algorithm is tested on distributed training tasks on a labeled dataset, while the data used for onboard training are usually unlabeled. The algorithm can be extended for semisupervised or unsupervised training. Secondly, our current experiment is conducted in a cluster environment with fixed network topology and homogeneous nodes on the ground, using software to simulate the low-bandwidth intersatellite network. We will study the performance of the algorithm in a dynamic heterogeneous network and carry out the onboard verification and corresponding engineering optimization research in the future.

Data Availability

The datasets used in this paper include CIFAR10 (<http://www.cs.toronto.edu/~kriz/cifar.html>) and ImageNet-1k (<https://image-net.org/challenges/LSVRC/2012/>).

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this article.

Authors' Contributions

X. Xiang and N. Liu conceived the idea of this study and supervised the study. X. Xiang performed theoretical proof and numerical analysis. Q. Meng, M. Huang, and X. Yao conducted the experiments. Q. Meng and M. Huang per-

formed data analysis. X. Xiang, Q. Meng, and M. Huang wrote the manuscript. All authors discussed the results and contributed to the final version of the manuscript. Qingliang Meng and Meiyu Huang contributed equally to this work.

Acknowledgments

This is supported by the Beijing Nova Program of Science and Technology under Grant Z191100001119129 and the National Natural Science Foundation of China 61702520.

References

- [1] G. Giuffrida, L. Diana, F. de Gioia et al., "Cloudscout: a deep neural network for on-board cloud detection on hyperspectral images," *Remote Sensing*, vol. 12, no. 14, p. 2205, 2020.
- [2] H. Li, H. Zheng, C. Han, H. Wang, and M. Miao, "Onboard spectral and spatial cloud detection for hyperspectral remote sensing images," *Remote Sensing*, vol. 10, no. 1, p. 152, 2018.
- [3] H. Zhang, *Machine Learning Parallelism Could Be Adaptive, Composable and Automated*, [Ph.D. thesis], Carnegie Mellon University, 2020.
- [4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding*, NAACL, 2019.
- [5] T. Brown, B. Mann, N. Ryder et al., "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.
- [6] X. Jia, S. Song, W. He et al., "Highly scalable deep learning training system with mixed-precision: training imagenet in four minutes," 2018, <https://arxiv.org/abs/1807.11205>.
- [7] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," *Advances in Neural Information Processing Systems*, pp. 5330–5340, 2017.
- [8] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu, "D2: decentralized training over decentralized data," in *International Conference on Machine Learning*, PMLR, pp. 4848–4856, 2018.
- [9] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *International Conference on Machine Learning*, PMLR, pp. 3043–3052, Stockholm, Sweden, 2018.
- [10] F. Sattler and S. Wiedemann, "Sparse binary compression: towards distributed deep learning with minimal communication," in *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, Budapest, Hungary, 2019.
- [11] S. U. Stich, "Local sgd converges fast and communicates little," in *7th International Conference on Learning Representations*, pp. 1–17, New Orleans, USA, 2019.
- [12] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi, "Don't use large mini-batches, use local sgd," in *8th International Conference on Learning Representations*, pp. 1–13, Addis Ababa, Ethiopia, 2020.
- [13] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, pp. 1273–1282, Seattle, Washington, USA, 2017.
- [14] J. Konečný, M. M. HB, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: strategies for improving

- communication efficiency,” 2016, <https://arxiv.org/abs/1610.05492>.
- [15] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, “Qsgd: communication-efficient sgd via gradient quantization and encoding,” *Advances in Neural Information Processing Systems*, vol. 30, pp. 1709–1720, 2017.
 - [16] W. Wen, C. Xu, F. Yan et al., “Terngrad: ternary gradients to reduce communication in distributed deep learning,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1508–1518, Long Beach, CA, USA, 2017.
 - [17] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, “Dorefa-net: training low bitwidth convolutional neural networks with low bitwidth gradients,” 2016, <https://arxiv.org/abs/1606.06160>.
 - [18] D. Alistarh, T. Hoefler, M. Johansson, S. Khirirat, N. Konstantinov, and C. Renggli, “The convergence of sparsified gradient methods,” in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 5977–5987, Montreal Convention Center, Montreal, Canada, 2018.
 - [19] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, “Deep gradient compression: reducing the communication bandwidth for distributed training,” in *6th International Conference on Learning Representations*, pp. 1–14, Vancouver, BC, Canada, 2018.
 - [20] Y. Tsuzuku, H. Imachi, and T. Akiba, “Variance-based gradient compression for efficient distributed deep learning,” in *6th International Conference on Learning Representations*, pp. 1–12, Vancouver, BC, Canada, 2018.
 - [21] C.-Y. Chen, J. Choi, D. Brand, A. Agrawal, W. Zhang, and K. Gopalakrishnan, “Adacomp: adaptive residual gradient compression for data-parallel distributed training,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 2827–2827, New Orleans, Louisiana, USA, 2018.
 - [22] M. Mohammadi Amiri and D. Gunduz, “Machine learning at the wireless edge: distributed stochastic gradient descent over-the-air,” *IEEE Transactions on Signal Processing*, vol. 68, pp. 2155–2169, 2020.
 - [23] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, Las Vegas, NV, USA, 2016.
 - [24] O. Dekel, G. B. Ran, O. Shamir, and L. Xiao, “Optimal distributed online prediction using minibatches,” *JMLR*, vol. 13, no. 1, pp. 165–202, 2012.
 - [25] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *3rd International Conference on Learning Representations*, pp. 1–14, San Diego, CA, USA, 2015.
 - [26] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” *Handbook of Systemic Autoimmune Diseases*, vol. 1, no. 4, 2009.
 - [27] O. Russakovsky, J. Deng, H. Su et al., “ImageNet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.