BEER: Fast $O(1/T)$ Rate for Decentralized Nonconvex Optimization with Communication Compression

Haoyu Zhao
Princeton University
haoyu@princeton.edu

Boyue Li
Carnegie Mellon University
boyuel@andrew.cmu.edu

Zhize Li*
Carnegie Mellon University
zhizel@andrew.cmu.edu

Peter Richtárik
King Abdullah University of Science and Technology
peter.richtarik@kaust.edu.sa

Yuejie Chi
Carnegie Mellon University
yuejiec@andrew.cmu.edu

Abstract

Communication efficiency has been widely recognized as the bottleneck for large-scale decentralized machine learning applications in multi-agent or federated environments. To tackle the communication bottleneck, there have been many efforts to design communication-compressed algorithms for decentralized nonconvex optimization, where the clients are only allowed to communicate a small amount of quantized information (aka bits) with their neighbors over a predefined graph topology. Despite significant efforts, the state-of-the-art algorithm in the nonconvex setting still suffers from a slower rate of convergence $O((G/T)^{2/3})$ compared with their uncompressed counterpart, where G measures the data heterogeneity across different clients, and T is the number of communication rounds. This paper proposes BEER, which adopts communication compression with gradient tracking, and shows it converges at a *faster rate* of $O(1/T)$. This significantly improves over the state-of-the-art rate, by matching the rate without compression even under arbitrary data heterogeneity. Numerical experiments are also provided to corroborate our theory and confirm the practical superiority of BEER in the data heterogeneous regime.

1 Introduction

Decentralized machine learning is gaining attention in both academia and industry because of its emerging applications in multi-agent systems such as the internet-of-things (IoT) and networked autonomous systems [31, 43]. One of the key problems in decentralized machine learning is on-device training, which aims to optimize a machine learning model using the datasets stored on (geographically) different clients, and can be formulated as a *decentralized optimization* problem.

Decentralized optimization aims to solve the following optimization problem without sharing the local datasets with other clients:

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \left\{ f(\boldsymbol{x};\mathcal{D}) := \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{x};\mathcal{D}_i) \right\}, \tag{1}$$

where $f(\boldsymbol{x};\mathcal{D}_i) := \mathbb{E}_{\xi_i\sim\mathcal{D}_i} f(\boldsymbol{x};\xi_i)$ for $i\in[n]$, and n is the total number of clients. Here, $\boldsymbol{x}\in\mathbb{R}^d$ is the machine learning model, $f(\boldsymbol{x};\mathcal{D})$, $f(\boldsymbol{x};\mathcal{D}_i)$, and $f(\boldsymbol{x};\xi_i)$ denote the loss functions of the model $\boldsymbol{x}$ on the entire dataset $\mathcal{D}$, the local dataset $\mathcal{D}_i$, and a random data sample ξ_i, respectively. Different

*Corresponding author.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

from the widely studied distributed or federated learning setting where there is a central server to coordinate the parameter sharing across all clients, in the decentralized setting, each client can only communication with its neighbors over a communication network determined by a predefined network topology.

The main bottleneck of decentralized optimization—when it comes to large-scale machine learning applications—is communication efficiency, due to the large number of clients involved in the network [43] and the enormous size of machine learning models [4], exacerbated by resource constraints such as limited bandwidth availability and stringent delay requirements. One way to reduce the communication cost is communication compression, which only transmits compressed messages (with fewer bits) between the clients using *compression operators*. The compression operators come with many design choices and offer great flexibility in different trade-offs of communication and computation in practice. Even though communication compression has been extensively applied to distributed or federated optimization with a central server [47, 12, 6, 26, 11, 25, 40, 41, 27], its use in the decentralized setting has been relatively sparse. Most of the existing approaches only apply to the strongly convex setting [39, 17, 30, 19, 29, 23], and only a few consider the general nonconvex setting [16, 51, 45].

Our contributions This paper considers decentralized optimization with communication compression, focusing on the *nonconvex* setting due to its critical importance in modern machine learning, such as training deep neural networks [21], word embeddings, and other unsupervised learning models [42]. Unfortunately, existing algorithms [16, 51, 45] suffer from several important drawbacks in the nonconvex setting: they need strong bounded gradient or bounded dissimilarity assumptions to guarantee convergence, and the convergence rate is order-wise slower than their uncompressed counterpart in terms of the communication rounds (see Table 1).

In this paper, we introduce BEER, which is a decentralized optimization algorithm with communication compression using gradient tracking. BEER not only removes the strong assumptions required in all prior works, but enjoys a faster convergence rate in the nonconvex setting. Concretely, we have the following main contributions (see Tables 1 and 2).

1. We show that BEER converges at a fast rate of $O(1/T)$ in the nonconvex setting, which improves over the state-of-the-art rate $O(1/T^{2/3})$ of CHOCO-SGD [16] and Deepsqueeze [51], where T is the number of communication rounds. This matches the rate without compression even under arbitrary data heterogeneity across the clients.
2. We also provide the analysis of BEER under the Polyak- Łojasiewicz (PL) condition (Assumption 4), and show that BEER converges at a linear rate (see Table 2). Note that strong convexity implies the PL condition, and thus BEER also achieves linear convergence in the strongly convex setting.
3. We run numerical experiments on real-world datasets and show BEER achieves superior or competitive performance when the data are heterogeneous compared with state-of-the-art baselines with and without communication compression.

To the best of our knowledge, BEER is the *first* algorithm that achieves $O(1/T)$ rate without the bounded gradient or bounded dissimilarity assumptions, supported by a strong empirical performance in the data heterogeneous setting.

Notation Throughout this paper, we use boldface letters to denote vectors, e.g., $\boldsymbol{x} \in \mathbb{R}^d$. Let $[n]$ denote the set $\{1, 2, \cdots, n\}$, $\mathbf{1}$ be the all-one vector, $\boldsymbol{I}$ be the identity matrix, $\|\cdot\|$ denote the Euclidean norm of a vector, and $\|\cdot\|_{\mathrm{F}}$ denote the Frobenius norm of a matrix. Let $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ denote the standard Euclidean inner product of two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. In addition, we use the standard order notation $O(\cdot)$ to hide absolute constants.

Due to space constraints, additional discussions of related work, further experiments, and the proof details can be found in the appendix.

2 Problem Setup

In this section, we formally define the decentralized optimization problem with communication compression, and introduce a few important quantities and assumptions that will be used in developing our algorithm and theory.

Algorithm	Convergence rate	Strong assumption
SQuARM-SGD [45]	$O\left(\frac{1}{\sqrt{nT}}+\frac{nG^2}{T}\right)$	Bounded Gradient
DeepSqueeze [51]	$O\left(\left(\frac{G}{T}\right)^{2/3}\right)$	Bounded Dissimilarity
CHOCO-SGD [16]	$O\left(\left(\frac{G}{T}\right)^{2/3}\right)$	Bounded Gradient
BEER (Algorithm 1)	$O\left(\frac{1}{T}\right)$	—

Table 1: Comparison of convergence rates for existing decentralized methods with communication compression in the nonconvex setting. Here, the parameter G refers the quantity either in the bounded gradient assumption $\mathbb{E}_{\xi_i\sim\mathcal{D}_i}\left\|\nabla f(\boldsymbol{x},\xi_i)\right\|^2 \leq G^2$ or the bounded dissimilarity assumption $\mathbb{E}_i\left\|\nabla f(\boldsymbol{x},\mathcal{D}_i)-\nabla f(\boldsymbol{x},\mathcal{D})\right\|^2 \leq G^2$, both of which are very strong assumptions (the bounded dissimilarity assumption is slightly weaker) that BEER does *not* require. All algorithms support the use of stochastic gradients with bounded local variance at local clients.

Assumptions	Convergence rate	Theorem
f_i is L-smooth	$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla f(\bar{\boldsymbol{x}}^t)\right\|^2 \leq \frac{2(\Phi_0-\Phi_T)}{\eta T}$	Theorem 1
f_i is L-smooth f satisfies PL condition	$\Phi_T \leq (1-\mu\eta)^T\Phi_0$	Theorem 3

Table 2: Summary of the established convergence rates for the proposed BEER algorithm in the nonconvex setting. Here, $\bar{\boldsymbol{x}}^t$ is the average model of all clients, η is the step size, Φ_t is the Lyapunov function (cf. (4)), and μ is the PL-condition parameter (cf. Assumption 4). We do not assume the bounded gradient or bounded dissimilarity assumption.

Decentralized optimization The goal of decentralized optimization is to solve

$$\min_{\boldsymbol{x}\in\mathbb{R}^d}\left\{f(\boldsymbol{x}):=\frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{x})\right\},$$

where n is the number of clients, $f(\boldsymbol{x})$ is the global objective function, and $f_i(\boldsymbol{x}):=f(\boldsymbol{x};\mathcal{D}_i):=\mathbb{E}_{\xi_i\sim\mathcal{D}_i}f(\boldsymbol{x};\xi_i)$ is the local objective function, with $\boldsymbol{x}$ the parameter of interest, ξ_i a random data sample drawn from the local dataset $\mathcal{D}_i$.

In the decentralized setting, the clients can only communicate with their local neighbors over a prescribed network topology, which is specified by an undirected weighted graph $\mathcal{G}([n],E)$. Here, each node in $[n]$ represents a client, and E is the set of possible communication links between different clients. Information sharing across the clients is implemented mathematically by the use of a mixing matrix $\boldsymbol{W}=[w_{ij}]\in[0,1]^{n\times n}$, which is defined in accordance with the network topology: we assign a positive weight w_{ij} for any $(i,j)\in E$ and $w_{ij}=0$ for all $(i,j)\notin E$. We make the following standard assumption on the mixing matrix [36].

Assumption 1 (Mixing matrix) *The mixing matrix $\boldsymbol{W}=[w_{ij}]\in[0,1]^{n\times n}$ is symmetric ($\boldsymbol{W}^\top=\boldsymbol{W}$) and doubly stochastic ($\boldsymbol{W}\mathbf{1}=\mathbf{1},\mathbf{1}^\top\boldsymbol{W}=\mathbf{1}^\top$). Let its eigenvalues be $1=|\lambda_1(\boldsymbol{W})|>|\lambda_2(\boldsymbol{W})|\geq\cdots\geq|\lambda_n(\boldsymbol{W})|$. The spectral gap is denoted by*

$$\rho:=1-|\lambda_2(\boldsymbol{W})|\in(0,1]. \tag{2}$$

The spectral gap of a mixing matrix is closely related to the network topology, see Nedić et al. [36] for its scaling with respect to the network size (i.e. the number of clients n) for representative network topologies.

Compression operators Compression, in the forms of quantization or sparsification, can be used to reduce the total communication cost. We now introduce the notion of a randomized *compression operator*, which is widely used in the decentralized/federated optimization literature, e.g. Tang et al. [49], Stich et al. [47], Koloskova et al. [16], Richtárik et al. [40], Fatkhullin et al. [8].

Definition 1 (Compression operator) *A randomized map $\mathcal{C} : \mathbb{R}^d \mapsto \mathbb{R}^d$ is an α-compression operator if for all $\boldsymbol{x} \in \mathbb{R}^d$, it satisfies*

$$\mathbb{E}\left[\left\|\mathcal{C}(\boldsymbol{x}) - \boldsymbol{x}\right\|^2\right] \leq (1-\alpha)\left\|\boldsymbol{x}\right\|^2. \tag{3}$$

In particular, no compression ($\mathcal{C}(\boldsymbol{x}) \equiv \boldsymbol{x}$) implies $\alpha = 1$.

Compared with the *unbiased compression operator* used in, e.g., Alistarh et al. [1], Khirirat et al. [15], Mishchenko et al. [33], Li and Richtárik [24], the compression operator in Definition 1 does not impose the additional constraint on the expectation such that $\mathbb{E}[\mathcal{C}(\boldsymbol{x})] = \boldsymbol{x}$. Besides, it is always possible to convert an *unbiased compression operator* into a biased one satisfying Definition 1, and thus Definition 1 is a generalization of the unbiased compression operator that allows *biased* compression. Practical examples of the compression operators are provided in Appendix B.

Assumptions on functions We now state the assumptions on the functions $\{f_i\}$ and f. Throughout this paper, we assume that $f^* = \min_{\boldsymbol{x}} f(\boldsymbol{x})$ exists and $f^* > -\infty$.

In the nonconvex setting, we assume that the functions $\{f_i\}_{i\in[n]}$ are arbitrary functions that satisfy the following standard smoothness assumption.

Assumption 2 (Smoothness) *The function f is L-smooth if there exists $L \geq 0$ such that*

$$\left\|\nabla f(\boldsymbol{x}_1) - \nabla f(\boldsymbol{x}_2)\right\| \leq L\left\|\boldsymbol{x}_1 - \boldsymbol{x}_2\right\|, \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^d.$$

In addition, we allow local computation to be performed via stochastic gradient updates, where $\tilde{\nabla} f_i(\boldsymbol{x}) := \nabla f_i(\boldsymbol{x}; \xi_i)$ denotes a local stochastic gradient computed via a sample ξ_i drawn i.i.d. from $\mathcal{D}_i$, and $\tilde{\nabla}_b f_i(\boldsymbol{x}) := \frac{1}{b}\sum_{j=1}^b \nabla f_i(\boldsymbol{x}; \xi_{i,j})$ denotes the stochastic gradient computed by a minibatch with size b drawn i.i.d. from $\mathcal{D}_i$. We assume $\tilde{\nabla} f_i(\boldsymbol{x})$ and $\tilde{\nabla}_b f_i(\boldsymbol{x})$ have bounded variance, which is again standard in the decentralized/federated optimization literature [32, 13, 16].

Assumption 3 (Bounded variance) *There exists a constant $\sigma \geq 0$ such that for all $i \in [n]$ and $\boldsymbol{x} \in \mathbb{R}^d$,*

$$\mathbb{E}\left\|\tilde{\nabla} f_i(\boldsymbol{x}) - \nabla f_i(\boldsymbol{x})\right\|^2 \leq \sigma^2.$$

For a stochastic gradient with minibatch size b, we have

$$\mathbb{E}\left\|\tilde{\nabla}_b f_i(\boldsymbol{x}) - \nabla f_i(\boldsymbol{x})\right\|^2 \leq \frac{\sigma^2}{b}.$$

In addition, we consider the setting when the function f additionally satisfies the following Polyak-Łojasiewicz (PL) condition [37], which can lead to fast linear convergence even when the function is nonconvex.

Assumption 4 (PL condition) *There exists some constant $\mu > 0$ such that for any $\boldsymbol{x} \in \mathbb{R}^d$,*

$$\left\|\nabla f(\boldsymbol{x})\right\|^2 \geq 2\mu(f(\boldsymbol{x}) - f^*).$$

Note that the PL condition is a weaker assumption than strong convexity, which means that if the objective function f is μ-strongly convex, then the PL condition also holds with the parameter μ.

3 Proposed Algorithm

In this section, we introduce our proposed algorithm BEER for decentralized nonconvex optimization with compressed communication. Before embarking on the description of BEER, we introduce some convenient matrix notation. Since in a decentralized setting, the parameter estimates at different clients are typically different, we use $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$ to denote the collection of parameter estimates from all clients, where $\boldsymbol{x}_i$ is from client i. The average of $\{\boldsymbol{x}_i\}_{i\in[n]}$ is denoted by $\bar{\boldsymbol{x}} := \frac{1}{n}\boldsymbol{X}\boldsymbol{1}$. Other quantities are defined similarly. With slight abuse of notation, we define

$$\nabla F(\boldsymbol{X}) := [\nabla f_1(\boldsymbol{x}_1), \nabla f_2(\boldsymbol{x}_2), \ldots, \nabla f_n(\boldsymbol{x}_n)] \in \mathbb{R}^{d\times n},$$

Algorithm 1 BEER: BEtter comprEssion for decentRalized optimization

1: **Input:** Initial point $\boldsymbol{X}^0 = \boldsymbol{x}_0\mathbf{1}^\top$, $\boldsymbol{G}^0 = \mathbf{0}$, $\boldsymbol{H}^0 = \mathbf{0}$, $\boldsymbol{V}^0 = \nabla F(\boldsymbol{X}_0)$, step size η, mixing step size γ, minibatch size b.
2: **for** $t = 0, 1, \ldots$ **do**
3: $\quad \boldsymbol{X}^{t+1} = \boldsymbol{X}^t + \gamma \boldsymbol{H}^t(\boldsymbol{W} - \boldsymbol{I}) - \eta \boldsymbol{V}^t$
4: $\quad \boldsymbol{Q}_h^{t+1} = \mathcal{C}(\boldsymbol{X}^{t+1} - \boldsymbol{H}^t)$ // client i sends $\boldsymbol{q}_{h,i}^{t+1}$ to all its neighbors
5: $\quad \boldsymbol{H}^{t+1} = \boldsymbol{H}^t + \boldsymbol{Q}_h^{t+1}$
6: $\quad \boldsymbol{V}^{t+1} = \boldsymbol{V}^t + \gamma \boldsymbol{G}^t(\boldsymbol{W} - \boldsymbol{I}) + \tilde{\nabla}_b F(\boldsymbol{X}^{t+1}) - \tilde{\nabla}_b F(\boldsymbol{X}^t)$
7: $\quad \boldsymbol{Q}_g^{t+1} = \mathcal{C}(\boldsymbol{V}^{t+1} - \boldsymbol{G}^t)$ // client i sends $\boldsymbol{q}_{g,i}^{t+1}$ to all its neighbors
8: $\quad \boldsymbol{G}^{t+1} = \boldsymbol{G}^t + \boldsymbol{Q}_g^{t+1}$
9: **end for**

which collects the local gradients computed at the local parameters. Similarly, the stochastic variant is defined as $\tilde{\nabla}_b F(\boldsymbol{X}) := [\tilde{\nabla}_b f_1(\boldsymbol{x}_1), \tilde{\nabla}_b f_2(\boldsymbol{x}_2), \ldots, \tilde{\nabla}_b f_n(\boldsymbol{x}_n)]$. We also allow the compression operator to take vector values, which are applied in a column-wise fashion, i.e., $\mathcal{C}(\boldsymbol{X}) := [\mathcal{C}(\boldsymbol{x}_1), \ldots, \mathcal{C}(\boldsymbol{x}_n)] \in \mathbb{R}^{d\times n}$.

We now proceed to describe BEER, which is detailed in Algorithm 1 using the matrix notation introduced above. At the t-th iteration, BEER maintains the current model estimates $\boldsymbol{X}^t$ and the global gradient estimates $\boldsymbol{V}^t$ across the clients. At the crux of its design, BEER also tracks and maintains two control sequences $\boldsymbol{H}^t$ and $\boldsymbol{G}^t$ that serve as compressed surrogates of $\boldsymbol{X}^t$ and $\boldsymbol{V}^t$, respectively. In particular, these two control sequences are updated by aggregating the received compressed messages alone (cf. Line 5 and Line 8).

It then boils down to how to carefully update these quantities in each iteration with communication compression, which we now explain in details. To begin, note that for each client i, BEER not only maintains its own parameters $\{\boldsymbol{x}_i^t, \boldsymbol{v}_i^t, \boldsymbol{h}_i^t, \boldsymbol{g}_i^t\}$, but also the control variables from its neighbors, namely, $\{\boldsymbol{h}_j^t\}_{j\in\mathcal{N}(i)}$ and $\{\boldsymbol{g}_j^t\}_{j\in\mathcal{N}(i)}$.

Update the model estimate Each client i first updates its model $\boldsymbol{x}_i^{t+1}$ according to Line 3. By thinking of $\{\boldsymbol{h}_j^t\}_{j\in\mathcal{N}(i)}$ as a surrogate of $\{\boldsymbol{x}_j^t\}_{j\in\mathcal{N}(i)}$, the second term aims to achieve better consensus among the clients through mixing, while the last term performs a gradient descent update.

Update the global gradient estimate Each client i updates the global gradient estimate $\boldsymbol{v}_i^{t+1}$ according to Line 6, where the last correction term—based on the difference of the gradients at consecutive models—is known as a trick called *gradient tracking* [38, 7, 35]. The use of gradient tracking is critical: as shall be seen momentarily, it contributes to the key difference from CHOCO-SGD that enables the fast rate of $O(1/T)$ without any bounded dissimilarity or bounded gradient assumptions. Indeed, if we remove the control sequence $\boldsymbol{G}^t$ and substitute Lines 6-8 by $\boldsymbol{V}^{t+1} = \tilde{\nabla}_b F(\boldsymbol{X}^{t+1})$, we recover CHOCO-SGD from BEER.

Update the compressed surrogates with communication To update $\{\boldsymbol{h}_j^t\}_{j\in\mathcal{N}(i)}$, each client i first computes a compressed message $\boldsymbol{q}_{h,i}^{t+1}$ that encodes the difference $\boldsymbol{x}_i^{t+1} - \boldsymbol{h}_i^t$, and broadcasts to its neighbors (cf. Line 4). Then, each client i updates $\{\boldsymbol{h}_j^t\}_{j\in\mathcal{N}(i)}$ by aggregating the received compressed messages $\{\boldsymbol{q}_{h,j}^{t+1}\}_{j\in\mathcal{N}(i)}$ following Line 5. The updates of $\{\boldsymbol{g}_j^t\}_{j\in\mathcal{N}(i)}$ can be performed similarly. Moreover, all the compressed messages can be sent in a single communication round at one iteration, i.e., the communications in Lines 4 and 7 can be performed at once. This leverages EF21 [40] for communication compression, which is a *better and simpler* algorithm that deals with biased compression operators compared with the error feedback (or error compensation, EF/EC) framework [12, 46]. Using the control sequence $\boldsymbol{G}^t$, BEER does not need to apply EF/EC explicitly and can deal with the error implicitly.

4 Convergence Guarantees

In this section, we show the convergence guarantees of BEER under different settings: the $O(1/T)$ rate in the nonconvex setting in Section 4.1, and the improved linear rate under the PL condition (Assumption 4) in Section 4.2. In Section 4.3, we briefly sketch the proof.

Our convergence guarantees are based on an appropriately designed Lyapunov function, given by

$$\Phi_t = \mathbb{E} f(\bar{\boldsymbol{x}}^t) - f^* + \frac{c_1 L}{n}\Omega_1^t + \frac{c_2\rho^2}{nL}\Omega_2^t + \frac{c_3 L}{n}\Omega_3^t + \frac{c_4\rho^4}{nL}\Omega_4^t, \tag{4}$$

where the choice of constants $\{c_i\}_{i=1}^4$ might be different from theorem to theorem, $\mathbb{E} f(\bar{\boldsymbol{x}}^t) - f^*$ represents the sub-optimality gap, and the errors $\{\Omega_i^t\}_{i=1}^4$ are defined by

$$\begin{aligned} \Omega_1^t &:= \mathbb{E}\left\|\boldsymbol{H}^t - \boldsymbol{X}^t\right\|_{\mathrm{F}}^2, \quad \Omega_2^t := \mathbb{E}\left\|\boldsymbol{G}^t - \boldsymbol{V}^t\right\|_{\mathrm{F}}^2, \\ \Omega_3^t &:= \mathbb{E}\left\|\boldsymbol{X}^t - \bar{\boldsymbol{x}}^t\mathbf{1}^\top\right\|_{\mathrm{F}}^2, \quad \Omega_4^t := \mathbb{E}\left\|\boldsymbol{V}^t - \bar{\boldsymbol{v}}^t\mathbf{1}^\top\right\|_{\mathrm{F}}^2. \end{aligned} \tag{5}$$

Here, Ω_1^t and Ω_2^t denote the compression errors for $\boldsymbol{X}^t$ and $\boldsymbol{V}^t$ when approximated using the compressed surrogates $\boldsymbol{H}^t$ and $\boldsymbol{G}^t$ respectively, and Ω_3^t and Ω_4^t denote the consensus errors of $\boldsymbol{X}^t$ and $\boldsymbol{V}^t$.

4.1 Convergence in the nonconvex setting

First, we present the following convergence result of BEER in the nonconvex setting when there is no local variance ($\sigma^2 = 0$), i.e., we can use the local full gradient $\nabla F(\boldsymbol{X}^t)$ instead of $\tilde{\nabla}_b F(\boldsymbol{X}^t)$ in Line 6 of Algorithm 1.

Theorem 1 (Convergence in the nonconvex setting without local variance) *Suppose Assumptions 1, and 2 hold, and we can compute the local full gradient $\nabla f_i(\boldsymbol{x})$ for any $\boldsymbol{x}$. Then there exist absolute constants $c_1, c_2, c_3, c_4, c_\gamma, c_\eta > 0$, such that if we set $\gamma = c_\gamma \alpha\rho$, $\eta = c_\eta\gamma\rho^2/L$, then for the Lyapunov function Φ_t in (4), it holds*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla f(\bar{\boldsymbol{x}}^t)\right\|^2 \le \frac{2(\Phi_0 - \Phi_T)}{\eta T}.$$

Theorem 1 shows that BEER converges at a rate of $O(1/T)$ when there is no local variance ($\sigma^2 = 0$), which is faster than the $O(1/T^{2/3})$ rate by CHOCO-SGD [16] and DeepSqueeze [51], and the $O(1/\sqrt{T})$ rate by SQuARM-SGD [45]; see also Table 1.

More specifically, to achieve $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla f(\bar{\boldsymbol{x}}^t)\right\|^2 \le \epsilon^2$, BEER needs $O\left(\frac{1}{\rho^3\alpha\epsilon^2}\right)$ iterations or communication rounds, where ρ and α are the spectral gap (cf. (2)) and the compression parameter (cf. (3)), respectively. In comparison, the state-of-the-art algorithm CHOCO-SGD [16] converges at a rate of $O((G/\rho^2\alpha T)^{2/3})$, which translates to an iteration complexity of $O\left(\frac{G}{\rho^2\alpha\epsilon^3}\right)$, with G being the bounded gradient parameter, namely, $\mathbb{E}_{\xi_i\sim\mathcal{D}_i}\left\|\nabla f(\boldsymbol{x},\xi_i)\right\|^2 \le G^2$. Therefore, BEER improves over CHOCO-SGD not only in terms of a better dependency on ϵ, but also removing the bounded gradient assumption, which is significant since in practice, G can be excessively large due to data heterogeneity across the clients.

The dependency on α of BEER is consistent with other compression schemes, such as CHOCO-SGD, DeepSqueeze and SQuARM-SGD for the nonconvex setting, as well as LEAD [30] and EF-C-GT [29] for the strongly convex setting.

As for the dependency on ρ, BEER is slightly worse than CHOCO-SGD, where CHOCO-SGD has a dependency of $O(1/\rho^2)$ whereas BEER has a dependency of $O(1/\rho^3)$. This degeneration is also seen in the analysis of uncompressed decentralized algorithms using gradient tracking [48, 54], where the rate $O(1/\rho^2)$ is worse than the rate of $O(1/\rho)$ for basic decentralized SGD algorithms [14, 28] by a factor of ρ. In addition, both BEER and CHOCO-SGD use small mixing step size γ to guarantee convergence, which makes the dependency on ρ worse than their uncompressed counterparts.

Stochastic gradient oracles BEER also supports the use of stochastic gradient oracles with bounded local variance (Assumption 3). More specifically, we have the following theorem, which generalizes Theorem 1.

Theorem 2 (Convergence in the nonconvex setting) *Suppose Assumptions 1, 2 and 3 hold. Then there exist absolute constants $c_1, c_2, c_3, c_4, c_\gamma, c_\eta > 0$, such that if we set $\gamma = c_\gamma\alpha\rho$, $\eta = c_\eta\gamma\rho^2/L$,*

Algorithm	Communication rounds	Gradient complexity
SQuARM-SGD [45]	$O\left(\frac{nG^2}{\epsilon^2}+\frac{\sigma^2}{bn\epsilon^4}\right)$	$O\left(\frac{\sigma^2}{n\epsilon^4}+\frac{nG^2}{\epsilon^2}\right)$
DeepSqueeze [51]	$O\left(\frac{G}{\epsilon^3}+\frac{\sigma^2}{bn\epsilon^4}\right)$	$O\left(\frac{\sigma^2}{n\epsilon^4}+\frac{G}{\epsilon^3}\right)$
CHOCO-SGD [16]	$O\left(\frac{G}{\epsilon^3}+\frac{\sigma^2}{bn\epsilon^4}\right)$	$O\left(\frac{\sigma^2}{n\epsilon^4}+\frac{G}{\epsilon^3}\right)$
BEER (Algorithm 1)	$O\left(\frac{1}{\epsilon^2}\right)$	$O\left(\frac{\sigma^2}{\epsilon^4}+\frac{1}{\epsilon^2}\right)$

Table 3: A more detailed comparison of the communication complexity and the gradient complexity with existing decentralized stochastic methods in the nonconvex setting to reach ϵ-accuracy. Here, G again measures the bounded gradient or bounded dissimilarity assumption, σ^2 and b denote the gradient variance and batch size respectively. We omit the dependency on the compression ratio and the network topology parameter for brevity.

then for the Lyapunov function Φ_t in (4), it holds

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla f(\bar{\boldsymbol{x}}^t)\right\|^2 \leq \frac{2(\Phi_0-\Phi_T)}{\eta T}+\frac{6c_4\sigma^2}{c_\gamma b\alpha L}.$$

In the presence of local variance, the squared gradient norm of BEER has an additional term that scales on the order of $O\left(\frac{\sigma^2}{\alpha b}\right)$ (ignoring other parameters). By choosing a sufficiently large minibatch size b, i.e. $b \geq O\left(\frac{\sigma^2}{\alpha\epsilon^2}\right)$, BEER maintains the iteration complexity $O\left(\frac{1}{\rho^3\alpha\epsilon^2}\right)$ to reach $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla f(\bar{\boldsymbol{x}}^t)\right\|^2 \leq \epsilon^2$, without the bounded gradient assumption, thus inheriting similar advantages over CHOCO-SGD as discussed earlier.

While our focus is on communication efficiency, to gain more insights, Table 3 summarizes both the communication rounds and the gradient complexity for different decentralized stochastic methods. While BEER does not require the bounded gradient assumption, it may lead to a worse gradient complexity in the data homogeneous setting due to the use of large minibatch size. Fortunately, this only impacts the local computation cost, and does not exacerbate the communication complexity, which is often the bottleneck. It is of great interest to further refine the design and analysis of BEER in terms of the gradient complexity.

4.2 Linear convergence with PL condition

Now, we show that the convergence of BEER can be strengthened to a linear rate with the addition of the PL condition (Assumption 4). Similar to the nonconvex setting, we first show the convergence result without local gradient variance ($\sigma^2 = 0$).

Theorem 3 (Convergence under the PL condition without local variance) *Suppose Assumptions 1, 2, and 4 hold, and we can compute the local full gradient $\nabla f_i(\boldsymbol{x})$ for any $\boldsymbol{x}$. Then there exist constants $c_1, c_2, c_3, c_4, c_\gamma, c_\eta > 0$, such that if we set $\gamma = c_\gamma\alpha\rho$, $\eta = c_\eta\gamma\rho^2/L$, then for the Lyapunov function Φ_t in (4), it holds*

$$\Phi_T \leq (1-\mu\eta)^T\Phi_0.$$

Theorem 3 demonstrates that under the PL condition, BEER converges linearly to the global optimum f^*, where it finds an ϵ-optimal solution in $O\left(\frac{\kappa}{\rho^3\alpha}\log\left(\frac{1}{\epsilon}\right)\right)$ iterations, with $\kappa := L/\mu$ the condition number. Note that in the strongly convex case, Liao et al. [29] proposed an algorithm that also uses error feedback compression and gradient tracking simultaneously, which achieves a linear rate of convergence with unclear dependencies with salient problem parameters. In comparison, BEER achieves an explicit linear rate of convergence in the strongly convex case as well, given strong convexity implies the PL condition. In fact, our analysis for the nonconvex setting—as will be made evident in our proof—almost implies immediately the linear convergence under the PL condition, thus provides a major step forward compared with prior analyses that only considered the strongly convex setting.

Stochastic gradient oracles Under the PL condition, BEER also supports the use of stochastic gradient oracles with bounded local variance (Assumption 3). The following theorem shows that BEER linearly converges to a neighborhood of size $O\left(\frac{\sigma^2}{\alpha b}\right)$ around the global optimum.

Theorem 4 (Convergence under PL condition) *Suppose Assumptions 1, 2, 3, and 4 hold. Then there exist absolute constants $c_1, c_2, c_3, c_4, c_\gamma, c_\eta > 0$, such that if we set $\gamma = c_\gamma \alpha \rho$, $\eta = c_\eta \gamma \rho^2 / L$, then for the Lyapunov function Φ_t in (4), it holds*

$$\Phi_T \leq (1 - \mu\eta)^T \Phi_0 + \frac{6 c_4 \sigma^2}{c_\gamma L b \alpha}.$$

4.3 Proof sketch

We now provide a proof sketch of Theorem 1, which establishes the $O(1/T)$ rate of BEER in the nonconvex setting using full gradient, highlighting the technical reason of the rate improvement of BEER over CHOCO-SGD. The full proofs of our theorems are delegated to Appendix D.

Recalling the quantities Ω_1^t to Ω_4^t from (5), which capture the approximation errors using compression and the consensus errors of $\boldsymbol{X}^t$ and $\boldsymbol{V}^t$, we would like to control these errors by obtaining inequalities of the form:

$$\Omega_i^{t+1} \leq (1 - a_i)\Omega_i^t + b_i, \quad \forall i \in \{1, 2, 3, 4\},$$

where $0 < a_i < 1$ denotes the size of the contraction, and $b_i > 0$ wraps together other terms which may be dependent on Ω_j^t for $j \neq i$ as well as the expected squared gradient norm of $\bar{\boldsymbol{v}}^t$, i.e.,

$$\Omega_5^t = \mathbb{E} \left\| \bar{\boldsymbol{v}}^t \right\|^2. \tag{6}$$

Then, by choosing the Lyapunov function properly (cf. (4)), we can show that the Lyapunov function actually descends, and small manipulations lead to the claimed convergence results in Theorem 1.

We now explain briefly how gradient tracking helps in BEER. Note that CHOCO-SGD also has the control variable $\boldsymbol{H}^t$ for the model $\boldsymbol{X}^t$, therefore in its analysis, it also deals with the quantities Ω_1^t and Ω_3^t. However, CHOCO-SGD also needs to bound the term $\|\boldsymbol{V}^t\|_{\mathrm{F}}^2$, where $\boldsymbol{V}^t = \nabla F(\boldsymbol{X}^t)$ for CHOCO-SGD when using full gradients. Thus, CHOCO-SGD needs to assume the bounded gradient assumption and only obtain a slower $O(1/T^{2/3})$ convergence rate. In contrast, BEER deals with the term $\|\boldsymbol{V}^t\|_{\mathrm{F}}^2$ by decomposing it using Young's inequality, leading to

$$\left\|\boldsymbol{V}^t\right\|_{\mathrm{F}}^2 \leq (1 + \beta)(\Omega_4^t)^2 + (1 + 1/\beta)(\Omega_5^t)^2$$

for some $\beta > 0$. Here, Ω_4^t can be controlled via the *gradient tracking* technique (see Line 6 in Algorithm 1) *without* the bounded gradient assumption, and Ω_5^t can be handled using the smoothness assumption (Assumption 2).

5 Numerical Experiments

This section presents numerical experiments on real-world datasets to showcase BEER's superior ability to handle data heterogeneity across the clients, by running each experiment on unshuffled datasets and comparing the performances with the state-of-the-art baseline algorithms both with and without communication compression. The code can be accessed at:

`https://github.com/liboyue/beer`.

Experiment setup We run experiments on two nonconvex problems to compare with the baseline algorithms both with and without communication compression: logistic regression with a nonconvex regularizer [52] on the `a9a` dataset [5], and training a 1-hidden layer neural network on the `MNIST` dataset [20].

For logistic regression with a nonconvex regularizer, following Wang et al. [52], the objective function over a datum $(\boldsymbol{a}, b)$ is defined as

$$f(\boldsymbol{x}; (\boldsymbol{a}, b)) = \log\left(1 + \exp(-b\boldsymbol{a}^\top \boldsymbol{x})\right) + \alpha \sum_{j=1}^{d} \frac{x_j^2}{1 + x_j^2},$$

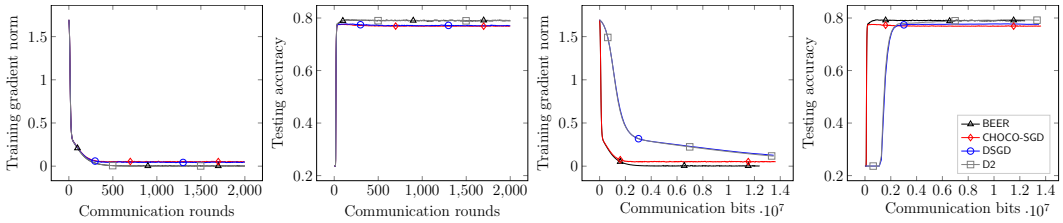

Figure 1: The training gradient norm and testing accuracy against communication rounds (left two panels) and communication bits (right two panels) for logistic regression with nonconvex regularization on unshuffled a9a dataset. Both BEER and CHOCO-SGD employ the biased gsgd_b compression [1] with $b = 5$.

where the last term is the nonconvex regularizer and the regularization parameter is set to $\alpha = 0.05$.

For 1-hidden layer neural network training, we use 32 hidden neurons, sigmoid activation functions and cross-entropy loss. The objective function over a datum $(\boldsymbol{a}, b)$ is defined as

$$f(\boldsymbol{x}; (\boldsymbol{a}, b)) = \ell(\mathsf{softmax}(\boldsymbol{W}_2\ \mathsf{sigmoid}(\boldsymbol{W}_1\boldsymbol{a} + \boldsymbol{c}_1) + \boldsymbol{c}_2), b),$$

where $\ell(\cdot, \cdot)$ denotes the cross-entropy loss, the optimization variable is collectively denoted by $\boldsymbol{x} = \text{vec}(\boldsymbol{W}_1, \boldsymbol{c}_1, \boldsymbol{W}_2, \boldsymbol{c}_2)$, where the dimensions of the network parameters $\boldsymbol{W}_1$, $\boldsymbol{c}_1$, $\boldsymbol{W}_2$, $\boldsymbol{c}_2$ are 64×784, 64×1, 10×64, and 10×1, respectively.

For both experiments, we split the *unshuffled* datasets evenly to 10 clients that are connected by a ring topology. By using unshuffled data, we can simulate the scenario with high data heterogeneity across clients. Approximately, for the a9a dataset, 5 clients receive data with label 1 and others receive data with label 0; for the MNIST dataset, client i receives data with label i. We use the FDLA matrix [53] as the mixing matrix to perform weighted information aggregation to accelerate convergence.

Results We compare BEER with 1) CHOCO-SGD [17], which is the state-of-the-art nonconvex decentralized optimizing algorithm using communication compression, and 2) DSGD [28] and D^2 [50], which are decentralized optimization algorithms without compression. All algorithms are initialized in the same experiment by the same initial point. Moreover, we use the same best-tuned learning rate $\eta = 0.1$, batch size $b = 100$, and biased compression operator (gsgd_b) [1] for BEER and CHOCO-SGD on both experiments.

Figure 1 and Figure 2 plot the training gradient norm and testing accuracy against communication rounds and communication bits for logistic regression with nonconvex regularization and 1-hidden-layer neural network training, respectively.

In the nonconvex logistic regression experiment (cf. Figure 1), the algorithms with communication compression (BEER and CHOCO-SGD) converge faster than the uncompressed algorithms (DSGD and D^2) in terms of the communication bits. However, CHOCO-SGD fails to converge to a small gradient norm solution since it cannot tolerate a high level of data dissimilarity across different clients, and its performance is not comparable to D^2. In contrast, BEER can converge to a point with a relatively smaller gradient norm, which is comparable to D^2. The performance of testing accuracy is similar to that of the training gradient norm, where BEER achieves the best testing accuracy and also learns faster than the uncompressed algorithms.

Turning to the neural network experiment (cf. Figure 2), in terms of the final training gradient norm, BEER converges to a solution comparable to D^2, but at a lower communication cost, while CHOCO-SGD and DSGD cannot converge due to the data heterogeneity. In terms of testing accuracy, BEER and D^2 have very similar performance, and outperform CHOCO-SGD and DSGD.

Convolutional neural network training We further compare the performance of BEER and CHOCO-SGD on training a convolutional neural network using the unshuffled MNIST dataset. Specifically, the network is consist of three modules: the first module is a 2-d convolution layer (1 input channel, 16 output channels, kernel size 5, stride 1 and padding 2) followed by 2-d batch normalization, ReLU activation and 2-d max pooling (kernel size 2 and stride 2); the second module is the same as the first module, except the convolution layer has 16 input channels and 32 output channels; the last module is a fully-connected layer with 1568 inputs and 10 outputs. We adopt the standard

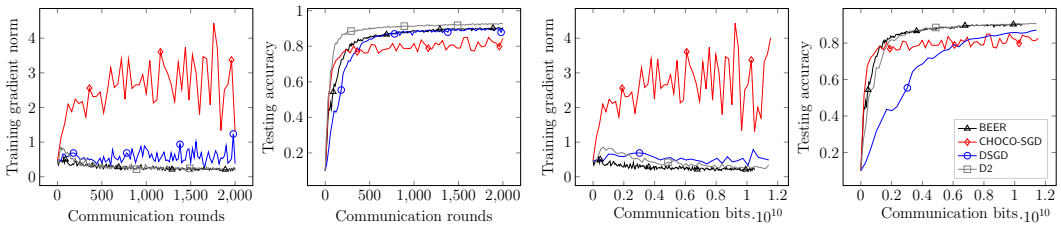

Figure 2: The training gradient norm and testing accuracy against communication rounds (left two panels) and communication bits (right two panels) for classification on unshuffled `MNIST` dataset using a 1-hidden-layer neural network. Both BEER and CHOCO-SGD employ the biased gsgd_b compression [1] with $b = 20$.

cross-entropy loss, and simply average each agent's model with its neighbors. Figure 3 shows the testing gradient norm and accuracy against the communication bits. It can be seen that BEER outperforms CHOCO-SGD in terms of both testing gradient norm and testing accuracy. Both algorithms converge fast initially, however, due to to the extreme data heterogeneity, their convergence speeds significantly degenerate after a short time. BEER keeps improving the objective when CHOCO-SGD hits its error floor, which highlights BEER's advantage to deal with data heterogeneity.

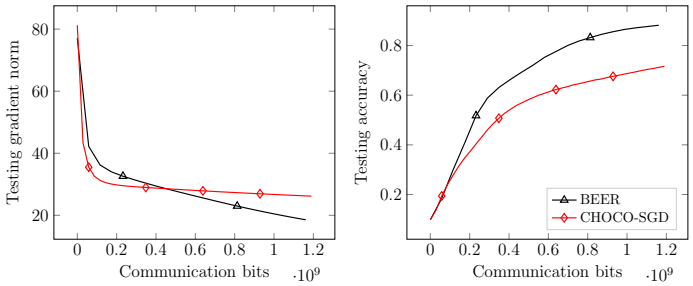

Figure 3: The testing gradient norm and testing accuracy against communication bits on unshuffled `MNIST` dataset using a 3-layer convolutional neural network. Both BEER and CHOCO-SGD employ the biased gsgd_b compression [1] with $b = 5$.

In summary, BEER has much better performance in terms of communication efficiency than CHOCO-SGD in heterogeneous data scenario, which corroborates our theory. BEER also performs similarly or even better than the uncompressed baseline algorithm D^2, and much better than DSGD. In addition, by leveraging different communication compression schemes, BEER allows more flexible trade-offs between communication and computation, making it an appealing choice in practice.

6 Conclusion

This paper presents BEER, which achieves a faster $O(1/T)$ convergence rate for decentralized nonconvex optimization with communication compression, *without* imposing the bounded dissimilarity or bounded gradient assumptions. In addition, a faster linear rate of convergence is established for BEER under the PL condition. Numerical experiments are provided to corroborate our theory on the advantage of BEER in the data heterogeneous scenario. An interesting direction of future work is to investigate the lower bounds for decentralized (nonconvex) optimization with communication compression. In addition, improving the dependency of BEER with the network topology parameter ρ, possibly leveraging the analysis in Koloskova et al. [18], is of interest.

Acknowledgement

The work of H. Zhao is supported in part by NSF, ONR, Simons Foundation, DARPA and SRC through awards to S. Arora. The work of B. Li, Z. Li and Y. Chi is supported in part by ONR N00014-19-1-2404, by AFRL under FA8750-20-2-0504, and by NSF under CCF-1901199, CCF-2007911 and CNS-2148212. B. Li is also gratefully supported by Wei Shen and Xuehong Zhang Presidential Fellowship at Carnegie Mellon University. The work of P. Richtárik is supported by KAUST Baseline Research Fund.

References

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

[2] 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*, pages 5977–5987, 2018.

[3] D. Bertsekas and J. Tsitsiklis. *Parallel and distributed computation: numerical methods*. Athena Scientific, 2015.

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[5] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[6] R. Das, A. Hashemi, S. Sanghavi, and I. S. Dhillon. Improved convergence rates for non-convex federated learning with compression. *arXiv preprint arXiv:2012.04061*, 2020.

[7] P. Di Lorenzo and G. Scutari. Next: In-network nonconvex optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(2):120–136, 2016.

[8] I. Fatkhullin, I. Sokolov, E. Gorbunov, Z. Li, and P. Richtárik. EF21 with bells & whistles: Practical algorithmic extensions of modern error feedback. *arXiv preprint arXiv:2110.03294*, 2021.

[9] R. Glowinski and A. Marrocco. On the solution of a class of non linear dirichlet problems by a penalty-duality method and finite elements of order one. In *Optimization Techniques IFIP Technical Conference*, pages 327–333. Springer, 1975.

[10] E. Gorbunov, F. Hanzely, and P. Richtárik. Local SGD: Unified theory and new efficient methods. *arXiv preprint arXiv:2011.02828*, 2020.

[11] E. Gorbunov, K. Burlachenko, Z. Li, and P. Richtárik. MARINA: Faster non-convex distributed learning with compression. In *International Conference on Machine Learning*, pages 3788–3798. PMLR, 2021.

[12] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pages 3252–3261. PMLR, 2019.

[13] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[14] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 482–491. IEEE, 2003.

[15] S. Khirirat, H. R. Feyzmahdavian, and M. Johansson. Distributed learning with compressed gradients. *arXiv preprint arXiv:1806.06573*, 2018.

[16] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi. Decentralized deep learning with arbitrary communication compression. In *International Conference on Learning Representations*, 2019.

[17] A. Koloskova, S. Stich, and M. Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *International Conference on Machine Learning*, pages 3478–3487. PMLR, 2019.

[18] A. Koloskova, T. Lin, and S. U. Stich. An improved analysis of gradient tracking for decentralized machine learning. *Advances in Neural Information Processing Systems*, 34:11422–11435, 2021.

[19] D. Kovalev, A. Koloskova, M. Jaggi, P. Richtarik, and S. Stich. A linearly convergent algorithm for decentralized optimization: Sending less bits for free! In *International Conference on Artificial Intelligence and Statistics*, pages 4087–4095. PMLR, 2021.

[20] Y. LeCun, L. D. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, and P. Simard. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261(276): 2, 1995.

[21] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[22] B. Li, Z. Li, and Y. Chi. DESTRESS: Computation-optimal and communication-efficient decentralized nonconvex finite-sum optimization. *SIAM Journal on Mathematics of Data Science*, 4(3):1031–1051, 2022.

[23] Y. Li, X. Liu, J. Tang, M. Yan, and K. Yuan. Decentralized composite optimization with compression. *arXiv preprint arXiv:2108.04448*, 2021.

[24] Z. Li and P. Richtárik. A unified analysis of stochastic gradient methods for nonconvex federated optimization. *arXiv preprint arXiv:2006.07013*, 2020.

[25] Z. Li and P. Richtárik. CANITA: Faster rates for distributed convex optimization with communication compression. In *Advances in Neural Information Processing Systems*, 2021.

[26] Z. Li, D. Kovalev, X. Qian, and P. Richtárik. Acceleration for compressed gradient descent in distributed and federated optimization. In *International Conference on Machine Learning*, pages 5895–5904. PMLR, 2020.

[27] Z. Li, H. Zhao, B. Li, and Y. Chi. SoteriaFL: A unified framework for private federated learning with communication compression. *arXiv preprint arXiv:2206.09888*, 2022.

[28] 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. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5336–5346, 2017.

[29] Y. Liao, Z. Li, K. Huang, and S. Pu. Compressed gradient tracking methods for decentralized optimization with linear convergence. *arXiv preprint arXiv:2103.13748*, 2021.

[30] X. Liu, Y. Li, R. Wang, J. Tang, and M. Yan. Linear convergent decentralized optimization with compression. In *International Conference on Learning Representations*, 2020.

[31] A. K. Marvasti, Y. Fu, S. DorMohammadi, and M. Rais-Rohani. Optimal operation of active distribution grids: A system of systems framework. *IEEE Transactions on Smart Grid*, 5(3): 1228–1237, 2014.

[32] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

[33] K. Mishchenko, E. Gorbunov, M. Takáč, and P. Richtárik. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*, 2019.

[34] A. Nedic and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[35] A. Nedic, A. Olshevsky, and W. Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

[36] A. Nedić, A. Olshevsky, and M. G. Rabbat. Network topology and communication-computation tradeoffs in decentralized optimization. *Proceedings of the IEEE*, 106(5):953–976, 2018.

[37] B. T. Polyak. Gradient methods for the minimisation of functionals. *USSR Computational Mathematics and Mathematical Physics*, 3(4):864–878, 1963.

[38] G. Qu and N. Li. Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control of Network Systems*, 5(3):1245–1260, 2017.

[39] A. Reisizadeh, A. Mokhtari, H. Hassani, and R. Pedarsani. An exact quantized decentralized gradient descent algorithm. *IEEE Transactions on Signal Processing*, 67(19):4934–4947, 2019.

[40] P. Richtárik, I. Sokolov, and I. Fatkhullin. EF21: A new, simpler, theoretically better, and practically faster error feedback. *arXiv preprint arXiv:2106.05203*, 2021.

[41] P. Richtárik, I. Sokolov, E. Gasanov, I. Fatkhullin, Z. Li, and E. Gorbunov. 3PC: Three point compressors for communication-efficient distributed training and a better theory for lazy aggregation. In *International Conference on Machine Learning*, pages 18596–18648. PMLR, 2022.

[42] N. Saunshi, O. Plevrakis, S. Arora, M. Khodak, and H. Khandeparkar. A theoretical analysis of contrastive unsupervised representation learning. In *International Conference on Machine Learning*, pages 5628–5637. PMLR, 2019.

[43] S. Savazzi, M. Nicoli, and V. Rampa. Federated learning with cooperating devices: A consensus approach for massive IoT networks. *IEEE Internet of Things Journal*, 7(5):4641–4654, 2020.

[44] W. Shi, Q. Ling, G. Wu, and W. Yin. EXTRA: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

[45] N. Singh, D. Data, J. George, and S. Diggavi. Squarm-sgd: Communication-efficient momentum sgd for decentralized optimization. *IEEE Journal on Selected Areas in Information Theory*, 2 (3):954–969, 2021.

[46] S. U. Stich and S. P. Karimireddy. The error-feedback framework: Better rates for sgd with delayed gradients and compressed updates. *Journal of Machine Learning Research*, 21:1–36, 2020.

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

[48] H. Sun, S. Lu, and M. Hong. Improving the sample and communication complexity for decentralized non-convex optimization: Joint gradient estimation and tracking. In *International Conference on Machine Learning*, pages 9217–9228. PMLR, 2020.

[49] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu. Communication compression for decentralized training. *Advances in Neural Information Processing Systems*, 31:7652–7662, 2018.

[50] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu. D^2: Decentralized training over decentralized data. In *International Conference on Machine Learning*, pages 4848–4856. PMLR, 2018.

[51] H. Tang, X. Lian, S. Qiu, L. Yuan, C. Zhang, T. Zhang, and J. Liu. Deepsqueeze: Decentralization meets error-compensated compression. *arXiv preprint arXiv:1907.07346*, 2019.

[52] Z. Wang, K. Ji, Y. Zhou, Y. Liang, and V. Tarokh. SpiderBoost and momentum: Faster stochastic variance reduction algorithms. *arXiv preprint arXiv:1810.10690*, 2018.

[53] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004.

[54] R. Xin, U. Khan, and S. Kar. Fast decentralized non-convex finite-sum optimization with recursive variance reduction. *arXiv preprint arXiv:2008.07428*, 2020.

[55] R. Xin, U. A. Khan, and S. Kar. A fast randomized incremental gradient method for decentralized non-convex optimization. *IEEE Transactions on Automatic Control*, 2021.

[56] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed. Exact diffusion for distributed optimization and learning—part I: Algorithm development. *IEEE Transactions on Signal Processing*, 67(3): 708–723, 2018.

[57] H. Zhao, K. Burlachenko, Z. Li, and P. Richtárik. Faster rates for compressed federated learning with client-variance reduction. *arXiv preprint arXiv:2112.13097*, 2021.

[58] H. Zhao, Z. Li, and P. Richtárik. FedPAGE: A fast local stochastic gradient method for communication-efficient federated learning. *arXiv preprint arXiv:2108.04755*, 2021.

[59] M. Zhu and S. Martínez. Discrete-time dynamic average consensus. *Automatica*, 46(2):322–329, 2010.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the discussions on the dependency with ρ in Section 4.1.
   (c) Did you discuss any potential negative societal impacts of your work? [No] This is a theoretical work that we don't foresee any negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 2.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix D.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We provided the code for our experiments.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]