Optimal Complexity in Decentralized Training

Yucheng Lu [1] **Christopher De Sa** [1]

Abstract

Decentralization is a promising method of scaling up parallel machine learning systems. In this paper, we provide a tight lower bound on the iteration complexity for such methods in a stochastic non-convex setting. Our lower bound reveals a theoretical gap in known convergence rates of many existing decentralized training algorithms, such as D-PSGD. We prove by construction this lower bound is tight and achievable. Motivated by our insights, we further propose DeTAG, a practical gossip-style decentralized algorithm that achieves the lower bound with only a logarithm gap. Empirically, we compare DeTAG with other decentralized algorithms on image classification tasks, and we show DeTAG enjoys faster convergence compared to baselines, especially on unshuffled data and in sparse networks.

1. Introduction

Parallelism is a ubiquitous method to accelerate model training (Abadi et al., 2016; Alistarh, 2018; Alistarh et al., 2020; Lu et al., 2020). A parallel learning system usually consists of three layers (Table 1): an application to solve, a communication protocol deciding how parallel workers coordinate, and a network topology determining how workers are connected. Traditional design for these layers usually follows a centralized setup: in the application layer, training data is required to be shuffled and shared among parallel workers; while in the protocol and network layers, workers either communicate via a fault-tolerant single central node (e.g. Parameter Server) (Li et al., 2014a;b; Ho et al., 2013) or a fully-connected topology (e.g. AllReduce) (Gropp et al., 1999; Patarasuk & Yuan, 2009). This centralized design limits the scalability of learning systems in two aspects. First, in many scenarios, such as Federated Learning (Koloskova et al., 2019a; McMahan et al., 2016) and Internet of Things (IOT) (Kanawaday & Sane, 2017), a shuffled dataset or a complete (bipartite) communication graph is not possible or affordable to obtain. Second, a centralized communication protocol can significantly slow down the training, especially with a low-bandwidth or high-latency network (Lian et al., 2017b; Tang et al., 2019b; Yu et al., 2018).

Table 1. Design choice of centralization and decentralization in different layers of a parallel machine learning system. The protocol specifies how workers communicate. The topology refers to the overlay network that logically connects all the workers.

Layer	Centralized	Decentralized
Application	Shuffled Data	Unshuffled Data (Federated Learning)
Protocol	AllReduce/AllGather Parameter Server	Gossip
Network Topology	Complete-(Bipartite) Graph	Arbitrary Graph

The rise of decentralization. To mitigate these limitations, decentralization comes to the rescue. Decentralizing the application and network allows workers to learn with unshuffled local datasets (Li et al., 2019) and arbitrary topologies (Seaman et al., 2017; Shanthamallu et al., 2017). Furthermore, the decentralized protocol, i.e. Gossip, helps to balance load, and has been shown to outperform centralized protocols in many cases (Lian et al., 2017a; Yu et al., 2019; Nazari et al., 2019; Lu & De Sa, 2020).

Understanding decentralization with layers. Many decentralized training designs have been proposed, which can lead to confusion as the term "decentralization" is used inconsistently in the literature. Some works use "decentralized" to refer to approaches that can tolerate non-iid or unshuffled datasets (Li et al., 2019), while others use it to mean gossip communication (Lian et al., 2017a), and still others use it to mean a sparse topology graph (Wan et al., 2020). To eliminate this ambiguity, we formulate Table 1, which summarizes the different "ways" a system can be decentralized. Note that the choices to decentralize different layers are *independent*, e.g., the centralized protocol AllReduce can still be implemented on a decentralized topology like the Ring graph (Wan et al., 2020).

[1] Department of Computer Science, Cornell University, Ithaca, New York, United States. Correspondence to: Yucheng Lu <yl2967@cornell.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

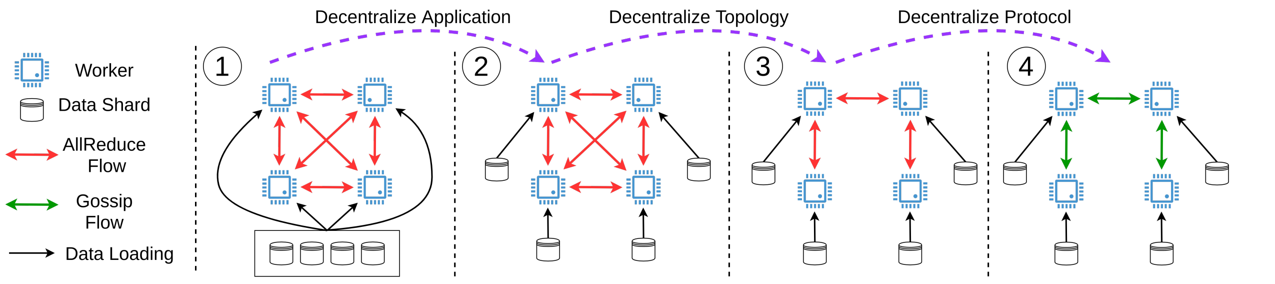


Figure 1. Figure illustrating how decentralization in different layers lead to different learning systems. From left to right: ①: A fully centralized system where workers sample from shared and shuffled data; ②: Based on ①, workers maintain their own data sources, making it decentralized in the application layer; ③: Based on ②, workers are decentralized in the topology layer; ④: A fully decentralized system in all three layers where the workers communicate via Gossip. Our framework and theory are applicable to all kinds of decentralized learning systems.

The theoretical limits of decentralization. Despite the empirical success, the best convergence rates achievable by decentralized training—and how they interact with different notions of decentralization—remains an open question. Previous works often show complexity of a given decentralized algorithm with respect to the number of iterations T or the number of workers n, ignoring other factors including network topologies, function parameters or data distribution. Although a series of decentralized algorithms have been proposed showing theoretical improvements—such as using variance reduction (Tang et al., 2018b), acceleration (Seaman et al., 2017), or matching (Wang et al., 2019)—we do not know how close they are to an "optimal" rate or whether further improvement is possible.

In light of this, a natural question is: *What is the optimal complexity in decentralized training? Has it been achieved by any algorithm yet?* Previous works have made initial attempts on this question, by analyzing this theoretical limit in a non-stochastic or (strongly) convex setting (Seaman et al., 2017; Scaman et al., 2018; Koloskova et al., 2020; Woodworth et al., 2018; Dvinskikh & Gasnikov, 2019; Sun & Hong, 2019). These results provide great heuristics but still leave the central question open, since stochastic methods are usually used in practice and many real-world problems of interest are non-convex (e.g. deep learning). In this paper we give the first full answer to this question: our contributions are as follows.

- In Section 4, we prove the first (to our knowledge) tight lower bound for decentralized training in a stochastic non-convex setting. Our results reveal an asymptotic gap between our lower bound and known convergence rates of existing algorithms.
- In Section 5, we prove our lower bound is tight by exhibiting an algorithm called DeFacto that achieves it—albeit while only being decentralized in the sense of the application and network layers.
- In Section 6, we propose DeTAG, a practical algorithm that achieves the lower bound with only a logarithm gap and that is decentralized in all three layers.
- In Section 7, we experimentally evaluate DeTAG on the CIFAR benchmark and show it converges faster compared to decentralized learning baselines.

2. Related Work

Decentralized Training. In the application layer, decentralized training usually denotes federated learning (Zhao et al., 2018). Research on decentralization in this sense investigates convergence where each worker samples only from a local dataset which is not independent and identically distributed to other workers' datasets (Bonawitz et al., 2019; Tran et al., 2019; Yang et al., 2019; Konečnỳ et al., 2016). Another line of research on decentralization focuses on the protocol layer—with average gossip (Boyd et al., 2005; 2006), workers communicate by averaging their parameters with neighbors on a graph. D-PSGD (Lian et al., 2017a) is one of the most basic algorithms that scales SGD with this protocol, achieving a linear parallel speed up. Additional works extend D-PSGD to asynchronous and variance-reduced cases (Lian et al., 2017b; Tang et al., 2018b; Tian et al., 2020; Zhang & You, 2019b; Hendrikx et al., 2019; Xin et al., 2021a). After those, Zhang & You (2019c); Xin et al. (2019; 2021b) propose adding gradient trackers to D-PSGD. Other works discuss the application of decentralization on specific tasks such as linear models or deep learning (He et al., 2018; Assran et al., 2018). Zhang & You (2019a) treats the case where only directed communication can be performed. Wang et al. (2019) proposes using matching algorithms to optimize the gossip protocol. Multiple works discuss using compression to decrease communication costs in decentralized training (Koloskova et al., 2019b;a; Lu & De Sa, 2020; Tang et al., 2019a; 2018a), and other papers connect decentralized training to other parallel methods and present a unified theory (Lu et al., 2020; Koloskova et al., 2020; Wang & Joshi, 2018). In some even earlier works like (Nedic & Ozdaglar, 2009; Duchi et al., 2010), full local gradients on a convex setting is investigated.

Lower Bounds in Stochastic Optimization. Lower bounds are a well studied topic in non-stochastic optimization, especially in convex optimization (Agarwal & Bottou, 2014; Arjevani & Shamir, 2015; Lan & Zhou, 2018; Fang et al., 2018; Arjevani & Shamir, 2017). In the stochastic setting, Allen-Zhu (2018) and Foster et al. (2019) discuss the complexity lower bound to find stationary points on convex problems. Other works study the lower bound in a convex, data-parallel setting (Diakonikolas & Guzmán, 2018; Balkanski & Singer, 2018; Tran-Dinh et al., 2019), and Colin et al. (2019) extends the result to a model-parallel setting. In the domain of non-convex optimization, Carmon et al. (2017; 2019) propose a zero-chain model that obtains tight bound for a first order method to obtain stationary points. Zhou & Gu (2019) extends this lower bound to a finite sum setting, and Arjevani et al. (2019) proposes a probabilistic zero-chain model that obtains tight lower bounds for first-order methods on stochastic and non-convex problems.

3. Setting

In this section, we introduce the notation and assumptions we will use. Throughout the paper, we consider the standard *data-parallel* training setup with n parallel workers. Each worker i stores a copy of the model $\boldsymbol{x} \in \mathbb{R}^d$ and a local dataset $\mathcal{D}_i$. The model copy and local dataset define a local loss function (or empirical risk) f_i. The ultimate goal of the parallel workers is to output a target model $\hat{\boldsymbol{x}}$ that minimizes the average over all the local loss functions, that is,

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in\mathbb{R}^d} \left[f(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\mathbb{E}_{\xi_i\sim\mathcal{D}_i} f_i(\boldsymbol{x};\xi_i)}_{f_i(\boldsymbol{x})} \right]. \tag{1}$$

Here, ξ_i is a data sample from $\mathcal{D}_i$ and is used to compute a stochastic gradient via some oracle, e.g. back-propagation on a mini-batch of samples. The loss functions can (potentially) be non-convex so finding a global minimum is NP-Hard; instead, we expect the workers to output a point $\hat{\boldsymbol{x}}$ at which $f(\hat{\boldsymbol{x}})$ has a small gradient magnitude in expectation: $\mathbb{E}\|\nabla f(\hat{\boldsymbol{x}})\| \leq \epsilon$, for some small ϵ.[1] The assumptions our theoretical analysis requires can be categorized by the layers from Table 1: in each layer, "being decentralized" corresponds to certain assumptions (or lack of assumptions). We now describe these assumptions for each layer separately.

[1]There are many valid stopping criteria. We adopt ϵ-stationary point as the success signal. $\mathbb{E}\|\nabla f(\hat{\boldsymbol{x}})\|^2 \leq \epsilon^2$ is another commonly used criterion; we adopt the non-squared one following (Carmon et al., 2019). Other criterions regarding stationary points can be converted to hold in our theory.

3.1. Application Layer

Application-layer assumptions comprise constraints on the losses f_i from (1) and the gradient oracle via which they are accessed by the learning algorithm, as these are constraints on the learning task itself.

Function class (Δ and L). As is usual in this space, we assume the local loss functions $f_i : \mathbb{R}^d \to \mathbb{R}$ are L-smooth,

$$\|\nabla f_i(\boldsymbol{x}) - \nabla f_i(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|, \ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d, \tag{2}$$

for some constant $L > 0$, and that the total loss f is range-bounded by Δ in the sense that $f(\mathbf{0}) - \inf_{\boldsymbol{x}} f(\boldsymbol{x}) \leq \Delta$. We let the function class $\mathcal{F}_{\Delta,L}$ denote the set of all functions that satisfy these conditions (for any dimension $d \in \mathbb{N}^+$).

Oracle class (σ^2). We assume each worker interacts with its local function f_i only via a stochastic gradient oracle $\tilde{g}_i$, and that when we query this oracle with model $\boldsymbol{x}$, it returns an independent unbiased estimator to $\nabla f_i(\boldsymbol{x})$ based on some random variable z with distribution Z (e.g. the index of a mini-batch randomly chosen for backprop). Formally,

$$\mathbb{E}_{z\sim Z}[\tilde{g}_i(\boldsymbol{x}, z)] = \nabla f_i(\boldsymbol{x}), \ \forall \boldsymbol{x} \in \mathbb{R}^d. \tag{3}$$

As per the usual setup, we additionally assume the local estimator has bounded variance: for some constant $\sigma > 0$,

$$\mathbb{E}_{z\sim Z}\|\tilde{g}_i(\boldsymbol{x}, z) - \nabla f_i(\boldsymbol{x})\|^2 \leq \sigma^2, \ \forall \boldsymbol{x} \in \mathbb{R}^d. \tag{4}$$

We let O denote a set of these oracles $\{\tilde{g}_i\}_{i\in[n]}$, and let the oracle class $\mathcal{O}_{\sigma^2}$ denote the class of all such oracle sets that satisfy these two assumptions.

Data shuffling (ς^2 and ς_0^2). At this point, an analysis with a centralized application layer would make the additional assumption that all the f_i are equal and the $\tilde{g}_i$ are identically distributed: this roughly corresponds to the assumption that the data all comes independently from a single centralized source. *We do not make this assumption*, and lacking such an assumption is what makes an analysis decentralized in the application layer. Still, some assumption that bounds the f_i relative to each other somehow is needed: we now discuss two such assumptions used in the literature, from which we use the weaker (and more decentralized) one.

One commonly made assumption (Lian et al., 2017a; Koloskova et al., 2019b;a; Lu & De Sa, 2020; Tang et al., 2018a) in decentralized training is

$$\frac{1}{n}\sum_{i=1}^{n}\|\nabla f_i(\boldsymbol{x}) - \nabla f(\boldsymbol{x})\|^2 \leq \varsigma^2, \ \forall \boldsymbol{x} \in \mathbb{R}^d, \tag{5}$$

for some constant ς, which is said to bound the "outer variance" among workers. This is often unreasonable, as it suggests the local datasets on workers must have close distribution: in practice, ensuring this often requires some sort

Table 2. Complexity comparison among different algorithms in the stochastic non-convex setting on arbitrary graphs. The blue text are the results from this paper. Definitions to all the parameters can be found in Section 3. Other algorithms like EXTRA (Shi et al., 2015) or MSDA (Scaman et al., 2017) are not comparable since they are designed for (strongly) convex problems. Additionally, Liu & Zhang (2021) provides alternative complexity bound for algorithms like D-PSGD which improves upon the spectral gap. However, the new bound would compromise the dependency on ϵ, which does not conflict with our comparison here.

	Source	Protocol	Sample Complexity	Comm. Complexity	Gap to Lower Bound
Lower Bound	Theorem 1	Central	$\Omega\left(\frac{\Delta L\sigma^2}{nB\epsilon^4}\right)$	$\Omega\left(\frac{\Delta LD}{\epsilon^2}\right)$	/
	Corollary 1	Decentral	$\Omega\left(\frac{\Delta L\sigma^2}{nB\epsilon^4}\right)$	$\Omega\left(\frac{\Delta L}{\epsilon^2\sqrt{1-\lambda}}\right)$	/
Upper Bound	DeFacto (Theorem 2)	Central	$O\left(\frac{\Delta L\sigma^2}{nB\epsilon^4}\right)$	$O\left(\frac{\Delta LD}{\epsilon^2}\right)$	$O(1)$
	DeTAG (Theorem 3)	Decentral	$O\left(\frac{\Delta L\sigma^2}{nB\epsilon^4}\right)$	$O\left(\frac{\Delta L\log\left(\frac{\varsigma_0 n}{\epsilon\sqrt{\Delta L}}\right)}{\epsilon^2\sqrt{1-\lambda}}\right)$	$O\left(\log\left(\frac{\varsigma_0 n}{\epsilon\sqrt{\Delta L}}\right)\right)$
	D-PSGD (Lian et al., 2017a)	Decentral	$O\left(\frac{\Delta L\sigma^2}{nB\epsilon^4}\right)$	$O\left(\frac{\Delta Ln\varsigma}{\epsilon^2(1-\lambda)^2}\right)$	$O\left(\frac{n\varsigma}{(1-\lambda)^{\frac{3}{2}}}\right)$
	SGP (Assran et al., 2019)	Decentral	$O\left(\frac{\Delta L\sigma^2}{nB\epsilon^4}\right)$	$O\left(\frac{\Delta Ln\varsigma}{\epsilon^2(1-\lambda)^2}\right)$	$O\left(\frac{n\varsigma}{(1-\lambda)^{\frac{3}{2}}}\right)$
	D^2 (Tang et al., 2018b)	Decentral	$O\left(\frac{\Delta L\sigma^2}{nB\epsilon^4}\right)$	$O\left(\frac{\lambda^2\Delta Ln\varsigma_0}{\epsilon^2(1-\lambda)^3}\right)$	$O\left(\frac{\lambda^2 n\varsigma_0}{(1-\lambda)^{\frac{5}{2}}}\right)$
	DSGT (Zhang & You, 2019c)	Decentral	$O\left(\frac{\Delta L\sigma^2}{nB\epsilon^4}\right)$	$O\left(\frac{\lambda^2\Delta Ln\varsigma_0}{\epsilon^2(1-\lambda)^3}\right)$	$O\left(\frac{\lambda^2 n\varsigma_0}{(1-\lambda)^{\frac{5}{2}}}\right)$
	GT-DSGD (Xin et al., 2021b)	Decentral	$O\left(\frac{\Delta L\sigma^2}{nB\epsilon^4}\right)$	$O\left(\frac{\lambda^2\Delta Ln\varsigma_0}{\epsilon^2(1-\lambda)^3}\right)$	$O\left(\frac{\lambda^2 n\varsigma_0}{(1-\lambda)^{\frac{5}{2}}}\right)$

of shuffling or common centralized data source. We **do not assume** (5) but instead adopt the much weaker assumption

$$\frac{1}{n}\sum_{i=1}^{n}\|\nabla f_i(\mathbf{0})-\nabla f(\mathbf{0})\|^2 \le \varsigma_0^2, \tag{6}$$

for constant $\varsigma_0 > 0$.[2] This assumption only requires a bound at point $\mathbf{0}$, which is, to the best of our knowledge, the weakest assumption of this type used in the literature (Tang et al., 2018b; Zhang & You, 2019c). Requiring such a weak assumption allows workers to (potentially) sample from different distributions or vary largely in their loss functions (e.g. in a federated learning environment).

3.2. Protocol Layer

Protocol-layer assumptions comprise constraints on the parallel learning algorithm itself, and especially on the way that the several workers communicate to approach consensus.

Algorithm class (B). We consider algorithms A that divide training into multiple iterations, and between two adjacent iterations, there must be a synchronization process among workers (e.g. a barrier) such that they start each iteration simultaneously.[3] Each worker running A has a local copy of the model, and we let $\boldsymbol{x}_{t,i} \in \mathbb{R}^d$ denote this model on worker i at iteration t. We assume without loss of generality that A initializes each local model at zero: $\boldsymbol{x}_{0,i} = \mathbf{0}$ for all i. At each iteration, each worker makes *at most* B queries to its gradient oracle $\tilde{g}_i$, for some constant $B \in \mathbb{N}^+$, and then uses the resulting gradients to update its model. We do not make any explicit rules for output and allow the output of the algorithm $\hat{\boldsymbol{x}}_t$ at the end of iteration t (the model that A would output if it were stopped at iteration t) to be any linear combination of all the local models, i.e.

$$\hat{\boldsymbol{x}}_t \in \operatorname{span}(\{\boldsymbol{x}_{t,j}\}_{j\in[n]}) = \{\textstyle\sum_{j=1}^{n} c_j\boldsymbol{x}_{t,j} \mid c_j \in \mathbb{R}\}. \tag{7}$$

Beyond these basic properties, we further require A to satisfy the following "zero-respecting" property from Carmon et al. (2017). Specifically, if $\boldsymbol{z}$ is any vector worker i queries its gradient oracle with at iteration t, then for any $k \in [d]$, if $\boldsymbol{e}_k^\top\boldsymbol{z} \neq 0$, then there exists a $s \le t$ and a $j \in [n]$ such that either $j = i$ or j is a neighbor of i in the network connectivity graph G (i.e. $(i,j) \in \{(i,i)\} \cup G$) and $(\boldsymbol{e}_k^\top\boldsymbol{x}_{s,j}) \neq 0$. More informally, the worker will not query its gradient oracle with a nonzero value for some weight unless that weight was already nonzero in the model state of the worker or one of its neighbors at some point in the past. Similarly, for any $k \in [d]$, if $(\boldsymbol{e}_k^\top\boldsymbol{x}_{t+1,i}) \neq 0$, then either there exists an $s \le t$ and j such that $(i,j) \in \{(i,i)\} \cup G$ and $(\boldsymbol{e}_k^\top\boldsymbol{x}_{s,j}) \neq 0$, or one of the gradient oracle's outputs $\boldsymbol{v}$ on worker i at iteration t has $\boldsymbol{e}_k^\top\boldsymbol{v} \neq 0$. Informally, a worker's model will not have a nonzero weight unless either (1) that weight was nonzero on that worker or one of its neighbors at a previous iteration,

[2] As we only use ς_0 for upper bounds, not lower bounds, we do not define a "class" that depends on this parameter.

[3] We consider synchronous algorithms only here for simplicity of presentation; further discussion of extension to asynchronous algorithms is included in the supplementary material.

or (2) the corresponding entry in one of the gradients the worker sampled at that iteration was nonzero.

Intuitively, we are requiring that algorithm A will not modify those coordinates that remain zero in all previous oracle outputs and neighboring models.[4] This lets A use a wide space of accessible information in communication and allows our class to cover first-order methods including SGD (Ghadimi & Lan, 2013), Momentum SGD (Nesterov, 1983), Adam (Kingma & Ba, 2014), RMSProp (Tieleman & Hinton, 2012), Adagrad (Ward et al., 2018), and AdaDelta (Zeiler, 2012). We let algorithm class $\mathcal{A}_B$ denote the set of all algorithms A that satisfy these assumptions.

So far our assumptions in this layer cover both centralized and decentralized protocols. Decentralized protocols, however, must satisfy the additional assumption that they communicate via *gossip* (see Section 2) (Boyd et al., 2005; 2006). A single step of gossip protocol can be expressed as

$$\boldsymbol{z}_{t,i} \leftarrow \textstyle\sum_{j\in\mathcal{N}_i} \boldsymbol{y}_{t,j}\boldsymbol{W}_{ji}, \ \forall i \in [n] \tag{8}$$

for some constant doubly stochastic matrix $\boldsymbol{W} \in \mathbb{R}^{n\times n}$ called the *communication matrix* and $\boldsymbol{y}$ and $\boldsymbol{z}$ are the input and output of the gossip communication step, respectively. The essence of a single Gossip step is to take weighted average over the neighborhood specified by a fixed matrix. To simplify later discussion, we further define the gossip matrix class $\mathcal{W}_n$ as the set of all matrices $\boldsymbol{W} \in \mathbb{R}^{n\times n}$, where $\boldsymbol{W}$ is doubly stochastic and $\boldsymbol{W}_{ij} \neq 0$ only if $(i,j) \in G$. We call every $\boldsymbol{W} \in \mathcal{W}_n$ a gossip matrix and we use $\lambda = \max\{|\lambda_2|, |\lambda_n|\} \in [0,1)$ to denote its general second-largest eigenvalue, where λ_i denotes the i-th largest eigenvalue of $\boldsymbol{W}$. We let gossip algorithm class $\mathcal{A}_{B,\boldsymbol{W}}$ denote the set of all algorithms $A \in \mathcal{A}_B$ that only communicate via gossip using a single matrix $\boldsymbol{W} \in \mathcal{W}_n$. It trivially holds that $\mathcal{A}_{B,\boldsymbol{W}} \subset \mathcal{A}_B$.

3.3. Topology Layer

Topology-layer assumptions comprise constraints on how workers are connected topologically. We let the graph class $\mathcal{G}_{n,D}$ denote the class of graphs G connecting n workers (vertices) with diameter D, where diameter of a graph measures the maximum distance between two arbitrary vertices (so $1 \le D \le n-1$). A centralized analysis here typically will also require that G be either complete or complete-bipartite (with parameter servers and workers as the two parts): lacking this requirement and allowing arbitrary graphs is what makes an analysis decentralized in the topology layer.

3.4. Complexity Measures

Now that we have defined the classes we are interested in, we can use them to define the complexity measures we will bound in our theoretical results. Given a loss function $f \in \mathcal{F}_{\Delta,L}$, a set of underlying oracles $O \in \mathcal{O}_{\sigma^2}$, a graph $G \in \mathcal{G}_{n,D}$, and an algorithm $A \in \mathcal{A}_B$, let $\hat{\boldsymbol{x}}_t^{A,f,O,G}$ denote the output of algorithm A at the end of iteration t under this setting. Then the *iteration complexity* of A solving f under O and G is defined as

$$T_\epsilon(A,f,O,G) = \min\left\{ t \in \mathbb{N} \,\middle|\, \mathbb{E}\left\|\nabla f(\hat{\boldsymbol{x}}_t^{A,f,O,G})\right\| \le \epsilon \right\},$$

that is, the least number of iterations required by A to find a ϵ-stationary-in-expectation point of f.

4. Lower Bound

Given the setup in Section 3, we can now present and discuss our lower bound on the iteration complexity. Note that in the formulation of protocol layer, the algorithm class $\mathcal{A}_B$ only specifies the information available for each worker, and thus $\mathcal{A}_B$ covers both centralization and decentralization in the protocol layer. Here, we show our lower bound in two parts: first a general bound where an arbitrary protocol that follows $\mathcal{A}_B$ is allowed, and then a corollary bound for the case where only decentralized protocol is allowed.

4.1. Lower Bound for Arbitrary Protocol

We start from the general bound. We expect this lower bound to show given arbitrary setting (functions, oracles and graph), the smallest iteration complexity we could obtain from $\mathcal{A}_B$, i.e.

$$\inf_{A\in\mathcal{A}_B} \sup_{f\in\mathcal{F}_{\Delta,L}} \sup_{O\in\mathcal{O}_{\sigma^2}} \sup_{G\in\mathcal{G}_{n,D}} T_\epsilon(A,f,O,G), \tag{9}$$

it suffices to construct a hard instance containing a loss function $\hat{f} \in \mathcal{F}_{\Delta,L}$, a graph $\hat{G} \in \mathcal{G}_{n,D}$ and a set of oracles $\hat{O} \in \mathcal{O}_{\sigma^2}$ and obtain a valid lower bound on $\inf_{A\in\mathcal{A}_B} T_\epsilon(A,\hat{f},\hat{O},\hat{G})$ since Equation (9) is always lower bounded by $\inf_{A\in\mathcal{A}_B} T_\epsilon(A,\hat{f},\hat{O},\hat{G})$.

For the construction, we follow the idea of probabilistic zero-chain model (Carmon et al., 2017; 2019; Arjevani et al., 2019; Zhou & Gu, 2019), which is a special loss function where adjacent coordinates are closely dependent on each other like a "chain." Our main idea is to use this function as f and split this chain onto different workers. Then the workers must conduct a sufficient number of optimization steps and rounds of communication to make progress.[5] From this, we obtain the following lower bound.

[4]On the other hand, it is possible to even drop the zero-respecting requirement and extend A to all the deterministic (not in the sense of sampling but the actual executions) algorithms. At a cost, we would need the function class to follow an "orthogonal invariant" property, and the model dimension needs to be large enough. We leave this discussion to the appendix.

[5]For brevity, we leave details in the supplementary material.

Theorem 1. *For function class $\mathcal{F}_{\Delta,L}$, oracle class $\mathcal{O}_{\sigma^2}$ and graph class $\mathcal{G}_{n,D}$ defined with any $\Delta > 0$, $L > 0$, $n \in \mathbb{N}^+$, $D \in \{1, 2, \dots, n-1\}$, $\sigma > 0$, and $B \in \mathbb{N}^+$, there exists $f \in \mathcal{F}_{\Delta,L}$, $O \in \mathcal{O}_{\sigma^2}$, and $G \in \mathcal{G}_{n,D}$, such that no matter what $A \in \mathcal{A}_B$ is used, $T_\epsilon(A, f, O, G)$ will always be lower bounded by*

$$\Omega\left(\frac{\Delta L\sigma^2}{nB\epsilon^4} + \frac{\Delta LD}{\epsilon^2}\right). \tag{10}$$

Dependency on the parameters. The bound in Theorem 1 consists of a sample complexity term, which is the dominant one for small ϵ, and a communication complexity term. We can see the increase of query budget B will only reduce the sample complexity. On the other hand, as the diameter D of a graph will generally increase as the number of vertices n increases, we can observe a trade-off between two terms when the system scales up: when more workers join the system, the communication complexity will gradually become the dominant term.

Consistency with the literature. Theorem 1 is tightly aligned with the state-of-the-art bounds in many settings. With $n = B = D = 1$, we recover the tight bound for sequential stochastic non-convex optimization $\Theta(\Delta L\sigma^2\epsilon^{-4})$ as shown in Arjevani et al. (2019). With $\sigma = 0, D = 1$, we recover the tight bound for sequential non-stochastic non-convex optimization $\Theta(\Delta L\epsilon^{-2})$ as shown in Carmon et al. (2019). With $B = 1, D = 1$, we recover the tight bound for centralized training $\Theta(\Delta L\sigma^2(n\epsilon^4)^{-1})$ given in Li et al. (2014b).

Improvement upon previous results. Previous works like Seaman et al. (2017); Scaman et al. (2018) provide similar lower bounds in a convex setting which relates to the diameter. However, these results treat D as a fixed value, i.e., $D = n - 1$, and thus makes the bound to be only tight on linear graph. By comparison, Theorem 1 allows D to be chosen independently to n.

4.2. Lower Bound for Decentralized Protocol

The bound in Theorem 1 holds for both centralized and decentralized protocols. A natural question is: *How would the lower bound adapt if the protocol is restricted to be decentralized?* i.e., the quantity of

$$\inf_{A\in\mathcal{A}_{B,\boldsymbol{W}}} \sup_{f\in\mathcal{F}_{\Delta,L}} \sup_{O\in\mathcal{O}_{\sigma^2}} \sup_{G\in\mathcal{G}_{n,D}} T_\epsilon(A, f, O, G),$$

we can extend the lower bound to Gossip in the following corollary.

Corollary 1. *For every $\Delta > 0$, $L > 0$, $n \in \{2, 3, 4, \cdots\}$, $\lambda \in [0, \cos(\pi/n)]$, $\sigma > 0$, and $B \in \mathbb{N}^+$, there exists a loss function $f \in \mathcal{F}_{\Delta,L}$, a set of underlying oracles $O \in \mathcal{O}_{\sigma^2}$, a gossip matrix $\boldsymbol{W} \in \mathcal{W}_n$ with second largest eigenvalue being λ, and a graph $G \in \mathcal{G}_{n,D}$, such that no matter what $A \in \mathcal{A}_{B,\boldsymbol{W}}$ is used, $T_\epsilon(A, f, O, G)$ will always be lower bounded by*

$$\Omega\left(\frac{\Delta L\sigma^2}{nB\epsilon^4} + \frac{\Delta L}{\epsilon^2\sqrt{1-\lambda}}\right). \tag{11}$$

Algorithm 1 Decentralized Stochastic Gradient Descent with Factorized Consensus Matrices (DeFacto) on worker i

```
Require: initialized model $\boldsymbol{x}_{0,i}$, a copy of model $\tilde{\boldsymbol{x}}_{0,i} \leftarrow \boldsymbol{x}_{0,i}$, gradient buffer $\boldsymbol{g} = \boldsymbol{0}$, step size $\alpha$, a sequence of communication matrices $\{\boldsymbol{W}_r\}_{1\le r\le R}$ of size $R$, number of iterations $T$, neighbor list $\mathcal{N}_i$
1:  for $t = 0, 1, \cdots, T-1$ do
2:    $k \leftarrow \lfloor t/2R \rfloor$.
3:    $r \leftarrow t \bmod 2R$.
4:    if $0 \le r < R$ then
5:      Spend all $B$ oracle budgets to compute stochastic gradient $\tilde{\boldsymbol{g}}$ at point $\boldsymbol{x}_{k,i}$ and accumulate it to gradient buffer: $\boldsymbol{g} \leftarrow \boldsymbol{g} + \tilde{\boldsymbol{g}}$.
6:    else
7:      Update model copy with the $r$-th matrix in $\{\boldsymbol{W}_r\}_{1\le r\le R}$:
          $\tilde{\boldsymbol{x}}_{t+1,i} \leftarrow \sum_{j\in\mathcal{N}_i\cup\{i\}} \tilde{\boldsymbol{x}}_{t,j}[\boldsymbol{W}_r]_{ji}$     (12)
8:    end if
9:    if $r = 2R-1$ then
10:     Update Model: $\boldsymbol{x}_{t+1,i} \leftarrow \tilde{\boldsymbol{x}}_{t+1,i} - \alpha\frac{\boldsymbol{g}}{R}$.
11:     Reinitialize gradient buffer: $\boldsymbol{g} \leftarrow \boldsymbol{0}$.
12:     Copy the current model: $\tilde{\boldsymbol{x}}_{t+1,i} \leftarrow \boldsymbol{x}_{t+1,i}$.
13:   end if
14: end for
15: return $\hat{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_{T,i}$
```

Gap in the existing algorithms. Comparing this lower bound with many state-of-the-art decentralized algorithms (Table 2), we can see they match on the sample complexity but leave a gap on the communication complexity. In many cases, the spectral gap significantly depends on the number of workers n and thus can be arbitrarily large. For example, when the graph G is a cycle graph or a linear graph, the gap of those baselines can increase by up to $O(n^6)$ (Brooks et al., 2011; Gerencsér, 2011)!

5. DeFacto: Optimal Complexity in Theory

In the previous section we show the existing algorithms have a gap compared to the lower bound. This gap could indicate the algorithms are suboptimal, but it could also be explained by our lower bound being loose. In this section we address this issue by proposing DeFacto, an example algorithm showing the lower bound is achievable, which verifies the tightness of our lower bound—showing that (10) would hold with equality and $\Theta(\cdot)$, not just $\Omega(\cdot)$.

We start with the following insight on the theoretical gap: the goal of communication is to let all the workers obtain information from neighbors. Ideally, the workers would, at each iteration, perform (8) with $\boldsymbol{W}^* = \boldsymbol{1}_n\boldsymbol{1}_n^\top/n$, where $\boldsymbol{1}_n$

Algorithm 2 Decentralized Stochastic Gradient Tracking with By-Phase Accelerated Gossip (DeTAG) on worker i

```
Require: initialized model $x_{0,i}$, a copy of model $\tilde{x}_{0,i} \leftarrow x_{0,i}$, gradient tracker $y_{0,i}$, gradient buffer $g_{(0)} = g_{(-1)} = 0$, step size $\alpha$, a gossip matrix $W$, number of iterations $T$, neighbor list $\mathcal{N}_i$
1:  for $t = 0, 1, \cdots, T-1$ do
2:      $k \leftarrow \lfloor t/R \rfloor$.
3:      $r \leftarrow t \bmod R$.
4:      Perform the $r$-th step in Accelerate Gossip:
            $\tilde{x}_{t+1,i} \leftarrow AG(\tilde{x}_{t,i}, W, \mathcal{N}_i, i)$    (13)
            $y_{t+1,i} \leftarrow AG(y_{t,i}, W, \mathcal{N}_i, i)$    (14)
5:      Spend all $B$ oracle budgets to compute stochastic gradient $\tilde{g}$ at point $x_{k,i}$ and accumulate it to gradient buffer: $g_{(k)} \leftarrow g_{(k)} + \tilde{g}$.
6:      if $r = R-1$ then
7:          Update gradient tracker and model:
                $x_{t+1,i} \leftarrow \tilde{x}_{t+1,i} - \alpha y_i$    (15)
                $y_{t+1,i} \leftarrow y_{t+1,i} + g_{(k)} - g_{(k-1)}$    (16)
8:          Reinitialize gradient buffer: $g_{(k-1)} \leftarrow g_{(k)}$ and then $g_{(k)} \leftarrow 0$.
9:          Copy the current model: $\tilde{x}_{t+1,i} \leftarrow x_{t+1,i}$.
10:     end if
11: end for
12: return $\hat{x} = \frac{1}{n}\sum_{i=1}^{n} x_{T,i}$
```

Algorithm 3 Accelerated Gossip (AG) with R steps

```
Require: $z_{0,i}, W, \mathcal{N}_i, i$
1: $z_{-1,i} \leftarrow z_{0,i}$
2: $\eta \leftarrow \frac{1-\sqrt{1-\lambda^2}}{1+\sqrt{1-\lambda^2}}$
3: for $r = 0, 1, 2, \cdots, R-1$ do
4:     $z_{r+1,i} \leftarrow (1+\eta)\sum_{j \in \mathcal{N}_i \cup \{i\}} z_{r,j} W_{ji} - \eta z_{r-1,i}$
5: end for
6: return $z_{R,i}$
```

is the n-dimensional all-one vector. We call this matrix the *Average Consensus* matrix. The Average Consensus is statistically equivalent to centralized communication (All-Reduce operation). However, due to the graph constraints, we can not use this W^* unless workers are fully connected; instead, a general method is to repeatedly apply a sequence communication matrices in consecutive iterations and let workers achieve or approach the Average Consensus. Previous work uses Gossip matrix W and expect $\prod_{r=1}^{R} W \approx \mathbf{1}_n\mathbf{1}_n^\top/n$ for some R. This R is known to be proportional to the mixing time of the Markov Chain W defines (Lu et al., 2020; Lu & De Sa, 2020), which is related to the inverse of its spectral gap (Levin & Peres, 2017). This limits convergence depending on the spectrum of the W chosen. The natural question to ask here is: can we do better? What are the limits of how fast we can reach average consensus on a connectivity graph G? This question is answered by the following lemma.

Lemma 1. *For any $G \in \mathcal{G}_{n,D}$, let $\mathcal{W}_G$ denote the set of $n \times n$ matrices such that for all $W \in \mathcal{W}_G$, $W_{ij} = 0$ if edge (i,j) does not appear in G. There exists a sequence of R matrices $\{W_r\}_{r\in[R]}$ that belongs to $\mathcal{W}_G$ such that $R \in [D, 2D]$ and*

$$W_{R-1}W_{R-2}\cdots W_0 = \frac{\mathbf{1}_n\mathbf{1}_n^\top}{n} = W^*.$$

Lemma 1 is a classic result in the literature of graph theory. The formal proof and detailed methods to identify these matrices can be found in many previous works (Georgopoulos, 2011; Ko, 2010; Hendrickx et al., 2014). Here we treat this as a black box procedure.[6]

Lemma 1 shows that we can achieve the exact average consensus by factorizing the matrix $\mathbf{1}_n\mathbf{1}_n^\top/n$, and we can obtain the factors from a preprocessing step. From here, the path to obtain an optimal rate becomes clear: starting from $t = 0$, workers first spend R iterations only computing stochastic gradients and then another R iterations to reach consensus communicating via factors from Lemma 1; they then repeat this process until a stationary point is found. We call this algorithm DeFacto (Algorithm 1).

DeFacto is statistically equivalent to centralized SGD operating $T/2R$ iterations with a mini-batch size of BR. It can be easily verified that DeFacto holds membership in $\mathcal{A}_B$. A straightforward analysis gives the convergence rate of DeFacto shown in the following Theorem.

Theorem 2. *Let A_1 denote Algorithm 1. For $\mathcal{F}_{\Delta,L}$, $\mathcal{O}_{\sigma^2}$ and $\mathcal{G}_{n,D}$ defined with any $\Delta > 0$, $L > 0$, $n \in \mathbb{N}^+$, $D \in \{1, 2, \ldots, n-1\}$, $\sigma > 0$, and $B \in \mathbb{N}^+$, the convergence rate of A_1 running on any loss function $f \in \mathcal{F}_{\Delta,L}$, any graph $G \in \mathcal{G}_{n,D}$, and any oracles $O \in \mathcal{O}_{\sigma^2}$ is bounded by*

$$T_\epsilon(A_1, f, O, G) \le O\left(\frac{\Delta L\sigma^2}{nB\epsilon^4} + \frac{\Delta LD}{\epsilon^2}\right). \tag{17}$$

Comparing Theorem 1 and Theorem 2, DeFacto achieves the optimal rate asymptotically. *This shows that our lower bound in Theorem 1 is tight.*

Despite its optimality, the design of DeFacto is unsatisfying in three aspects: (1) It compromises the throughput[7] by a factor of two because in each iteration, a worker either communicates with neighbors or computes gradients but not both. This fails to overlap communication and computation and creates extra idle time for the workers. (2) It needs to iterate over all the factor matrices before it can query the gradient oracle at subsequent parameters. When diameter D increases, the total time to finish such round will increase proportionally. (3) DeFacto works with decentralized data and arbitrary graph, achieving decentralization in both application and topology layers. However, the matrices used

[6] We cover specific algorithms and details in the supplementary.
[7] The number of stochastic gradients computed per iteration.

in Lemma 1 are not Gossip matrices as defined in $\mathcal{W}_n$, and thus it fails to be decentralized in the protocol-layer sense.

6. DeTAG: Optimal Complexity in Practice

To address the limitations of DeFacto, a natural idea is to replace all the factor matrices in Lemma 1 with a gossip matrix $\boldsymbol{W}$. The new algorithm after this mild modification is statistically equivalent to a D-PSGD variant: every R iterations, it updates the model the same as one iteration in D-PSGD with a mini-batch size of BR and communicate with a matrix $\boldsymbol{W}'$ whose second largest eigenvalue $\lambda' = \lambda^R$, with T/R iterations in total. However, even with arbitrarily large R, the communication complexity in this "updated D-PSGD" is still $O(\Delta L n \varsigma \epsilon^{-2})$ (Table 2), leaving an $O(n\varsigma)$ gap compared to our lower bound.

To close this gap, we adopt two additional techniques:[8] one is a gradient tracker $\boldsymbol{y}$ that is used as reference capturing gradient difference in the neighborhood; the other is using acceleration in gossip as specified in Algorithm 3. Modifying DeFacto results in Algorithm 2, which we call DeTAG. DeTAG works as follows: it divides the total number of iterations T into several phases where each phase contains R iterations. In each iteration, the communication process calls Accelerated Gossip to update a model replica $\tilde{\boldsymbol{x}}$ and the gradient tracker (line 4) while the computation process constantly computes gradients at the same point (line 5). At the end of each phase, model $\boldsymbol{x}$, its replica $\tilde{\boldsymbol{x}}$ and gradient tracker $\boldsymbol{y}$ are updated in line 7-10 and then DeTAG steps into the next phase. Aside from the two additional techniques, the main difference between DeTAG and DeFacto is that the communication matrix in DeTAG is a fixed gossip matrix $\boldsymbol{W}$, which allows DeTAG to benefit from decentralization in the protocol layer as well as to adopt arbitrary $R \geq 1$ in practice (allowing R to be tuned independently of G).

Improvement on design compared to baselines. Comparing with other baselines in Table 2, the design of DeTAG improves in the sense that (1) It removes the dependency on the outer variance ς. (2) It drops the requirement[9] on the gossip matrix assumed in Tang et al. (2018b). (3) The baseline DSGT (Zhang & You, 2019c) and GT-DSGD (Xin et al., 2021b) can be seen as special cases of taking $R = 1$ and $\eta = 0$ in DeTAG. That implies in practice, a well tuned DeTAG can never perform worse than the baseline DSGT or GT-DSGD.

The convergence rate of DeTAG is given in the following theorem.

[8]Note that neither of these techniques is our original design, and we do not take credit for them. Our main contribution here is to prove their combination leads to optimal complexity.

[9]Tang et al. (2018b) requires the gossip matrix to be symmetric and its smallest eigenvalue is lower bounded by $-\frac{1}{3}$.

Theorem 3. *Let A_2 denote Algorithm 2. For $\mathcal{F}_{\Delta,L}$, $\mathcal{O}_{\sigma^2}$ and $\mathcal{G}_{n,D}$ defined with any $\Delta > 0$, $L > 0$, $n \in \mathbb{N}^+$, $\lambda \in [0, 1)$, $\sigma > 0$, and $B \in \mathbb{N}^+$, under the assumption of Equation (6), if we set the phase length R to be*

$$R = \frac{\max\left(\frac{1}{2}\log(n), \frac{1}{2}\log\left(\frac{\varsigma_0^2 T}{\Delta L}\right)\right)}{\sqrt{1-\lambda}},$$

the convergence rate of A_2 running on any loss function $f \in \mathcal{F}_{\Delta,L}$, any graph $G \in \mathcal{G}_{n,D}$, and any oracles $O \in \mathcal{O}_{\sigma^2}$ is bounded by

$$T_\epsilon(A_2, f, O, G) \leq O\left(\frac{\Delta L \sigma^2}{nB\epsilon^4} + \frac{\Delta L \log\left(n + \frac{\varsigma_0 n}{\epsilon\sqrt{\Delta L}}\right)}{\epsilon^2\sqrt{1-\lambda}}\right).$$

Comparing Theorem 1 and Theorem 3, DeTAG achieves the optimal complexity with only a logarithm gap.

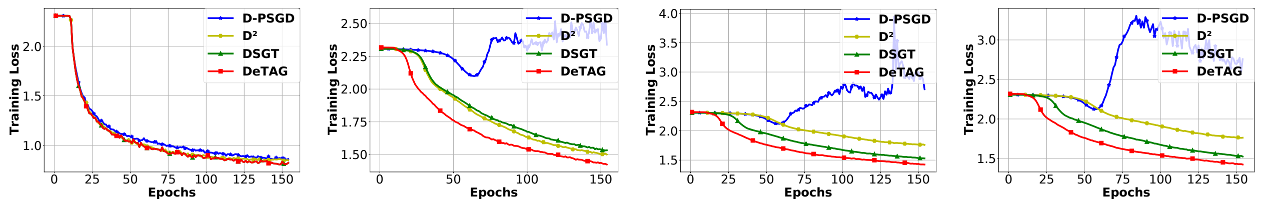
Improvement on complexity. Revisiting Table 2, we can see the main improvement of DeTAG's complexity is in the two terms on communication complexity: (1) DeTAG only depends on the outer variance term ς_0 inside a log, and (2) It reduces the dependency on the spectral gap $1 - \lambda$ to the lower bound of square root, as shown in Corollary 1.

Understanding the phase length R. In DeTAG, the phase length R is a tunable parameter. Theorem 3 provides a suggested value for R. Intuitively, the value of R captures the level of consensus of workers should reach before they step into the next phase. Theoretically, we observe R is closely correlated to the mixing time of $\boldsymbol{W}$: if we do not use acceleration in Gossip, then R will become $\tilde{O}\left(\frac{1}{1-\lambda}\right)$, which is exactly the upper bound on the mixing time of the Markov Chain $\boldsymbol{W}$ defines (Levin & Peres, 2017).

7. Experiments

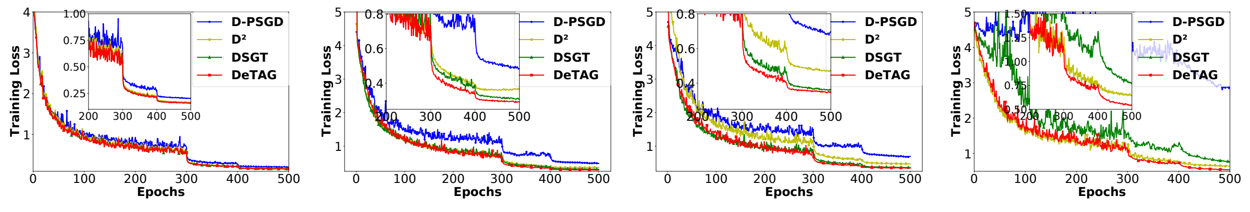
In this section we empirically compare the performance among different algorithms. All the models and training scripts in this section are implemented in PyTorch and run on an Ubuntu 16.04 LTS cluster using a SLURM workload manager running CUDA 9.2, configured with 8 NVIDIA GTX 2080Ti GPUs. We launch one process from the host as one worker and let them use `gloo` as the communication backend. In each experiment, we compare the following algorithms[10]: D-PSGD (Lian et al., 2017a), D^2 (Tang et al., 2018b), DSGT (Zhang & You, 2019c) and DeTAG. Note that GT-DSGD (Xin et al., 2021b) and DSGT (Zhang & You, 2019c) are essentially the same algorithm so we omit the comparison to GT-DSGD. Also note that SGP (Assran et al.,

[10]Since DeFacto is a only a "motivation" algorithm and in practice we observe it performs bad, we do not include the discussion of that.



(a) 100% Shuffled CIFAR10 (b) 50% Shuffled CIFAR10 (c) 25% Shuffled CIFAR10 (d) 0% Shuffled CIFAR10

Figure 2. Fine tuned results of training LeNet on CIFAR10 with different shuffling strategies.



(a) $\kappa = 1$ ($1-\lambda \approx$ 4e-2) (b) $\kappa = 0.1$ ($1-\lambda \approx$ 4e-3) (c) $\kappa = 0.05$ ($1-\lambda \approx$ 2e-3) (d) $\kappa = 0.01$ ($1-\lambda \approx$ 4e-4)

Figure 3. Fine tuned results of training Resnet20 on CIFAR100 with different spectral gaps.

2019) reduces to D-PSGD for symmetric mixing matrices in undirected graphs. Throughout the experiment we use Ring graph. Hyperparameters can be found in the supplementary material.

Convergence over different outer variance. In the first experiments, we investigate the correlation between convergence speed and the outer variance $\varsigma(\varsigma_0)$. We train LeNet on CIFAR10 using 8 workers, which is a standard benchmark experiment in the decentralized data environment (Tang et al., 2018b; Zhang & You, 2019c). To create the decentralized data, we first sort all the data points based on its labels, shuffle the first $X\%$ data points and then evenly split to different workers. The X controls the degree of decentralization, we test $X = 0, 25, 50, 100$ and plot the results in Figure 2.

We can see in Figure 2(a) when the dataset is fully shuffled, all the algorithms converge at similar speed while D-PSGD converges a little slower than other variance reduced algorithms. From Figure 2(b) to Figure 2(d) we can see when we shuffle less portion of the dataset, i.e., the dataset becomes more decentralized, D-PSGD fails to converge even with fine-tuned hyperparameter. Meanwhile, among D^2, DSGT and DeTAG, we can see DeTAG converges the fastest. When dataset becomes more decentralized, DSGT seems to receive more stable performance than D^2.

Convergence over different spectral gaps. In the second experiments, we proceed to explore the relation between convergence speed and spectral gap $1-\lambda$ of the gossip matrix $\boldsymbol{W}$. We use 16 workers connected with a Ring graph to train Resnet20 on CIFAR100, and we generate a $\boldsymbol{W}_0$ on such graph using Metropolis method. Then we adopt the slack matrix method to modify the spectral gap (Lu et al., 2020): $\boldsymbol{W}_\kappa = \kappa \boldsymbol{W}_0 + (1-\kappa)\boldsymbol{I}$, where κ is a control parameter. We test $\kappa = 1, 0.1, 0.05, 0.01$ and plot the results in Figure 3. We can see with different κ, DeTAG is able to achieve faster convergence compared to baselines. When the network becomes sparse, i.e., κ decreases, DeTAG enjoys more robust convergence.

8. Conclusion

In this paper, we investigate the tight lower bound on the iteration complexity of decentralized training. We propose two algorithms, DeFacto and DeTAG, that achieve the lower bound in terms of different decentralization in a learning system. DeTAG uses Gossip protocol, and is shown to be empirically competitive to many baseline algorithms, such as D-PSGD. In the future, we plan to investigate the variants of the complexity bound with respect to communication that are compressed, asynchronous, etc.

Acknowledgement

This work is supported by NSF IIS-2046760. The authors would like to thank A. Feder Cooper, Jerry Chee, Zheng Li, Ran Xin, Jiaqi Zhang and anonymous reviewers from ICML 2021 for providing valuable feedbacks on earlier versions of this paper.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

Agarwal, A. and Bottou, L. A lower bound for the optimization of finite sums. *arXiv preprint arXiv:1410.0723*, 2014.

Alistarh, D. A brief tutorial on distributed and concurrent machine learning. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pp. 487–488, 2018.

Alistarh, D., Chatterjee, B., and Kungurtsev, V. Elastic consistency: A general consistency model for distributed stochastic gradient descent. *arXiv preprint arXiv:2001.05918*, 2020.

Allen-Zhu, Z. How to make the gradients small stochastically: Even faster convex and nonconvex sgd. In *Advances in Neural Information Processing Systems*, pp. 1157–1167, 2018.

Arjevani, Y. and Shamir, O. Communication complexity of distributed convex learning and optimization. In *Advances in neural information processing systems*, pp. 1756–1764, 2015.

Arjevani, Y. and Shamir, O. Oracle complexity of second-order methods for finite-sum problems. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 205–213. JMLR. org, 2017.

Arjevani, Y., Carmon, Y., Duchi, J. C., Foster, D. J., Srebro, N., and Woodworth, B. Lower bounds for non-convex stochastic optimization. *arXiv preprint arXiv:1912.02365*, 2019.

Assran, M., Loizou, N., Ballas, N., and Rabbat, M. Stochastic gradient push for distributed deep learning. *arXiv preprint arXiv:1811.10792*, 2018.

Assran, M., Loizou, N., Ballas, N., and Rabbat, M. Stochastic gradient push for distributed deep learning. In *International Conference on Machine Learning*, pp. 344–353. PMLR, 2019.

Balkanski, E. and Singer, Y. Parallelization does not accelerate convex optimization: Adaptivity lower bounds for non-smooth convex minimization. *arXiv preprint arXiv:1808.03880*, 2018.

Berthier, R., Bach, F., and Gaillard, P. Accelerated gossip in networks of given dimension using jacobi polynomial iterations. *SIAM Journal on Mathematics of Data Science*, 2(1):24–47, 2020.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, H. B., et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. Gossip algorithms: Design, analysis and applications. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3, pp. 1653–1664. IEEE, 2005.

Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. Randomized gossip algorithms. *IEEE transactions on information theory*, 52(6):2508–2530, 2006.

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. *Handbook of markov chain monte carlo*. CRC press, 2011.

Carmon, Y., Duchi, J. C., Hinder, O., and Sidford, A. Lower bounds for finding stationary points ii: First-order methods. *arXiv preprint arXiv:1711.00841*, 2017.

Carmon, Y., Duchi, J. C., Hinder, O., and Sidford, A. Lower bounds for finding stationary points i. *Mathematical Programming*, pp. 1–50, 2019.

Colin, I., Dos Santos, L., and Scaman, K. Theoretical limits of pipeline parallel optimization and application to distributed deep learning. In *Advances in Neural Information Processing Systems*, pp. 12350–12359, 2019.

Diakonikolas, J. and Guzmán, C. Lower bounds for parallel and randomized convex optimization. *arXiv preprint arXiv:1811.01903*, 2018.

Duchi, J. C., Agarwal, A., and Wainwright, M. J. Distributed dual averaging in networks. In *NIPS*, pp. 550–558. Citeseer, 2010.

Dvinskikh, D. and Gasnikov, A. Decentralized and parallelized primal and dual accelerated methods for stochastic convex programming problems. *arXiv preprint arXiv:1904.09015*, 2019.

Fang, C., Li, C. J., Lin, Z., and Zhang, T. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems*, pp. 689–699, 2018.

Foster, D., Sekhari, A., Shamir, O., Srebro, N., Sridharan, K., and Woodworth, B. The complexity of making the gradient small in stochastic convex optimization. *arXiv preprint arXiv:1902.04686*, 2019.

Georgopoulos, L. Definitive consensus for distributed data inference. Technical report, EPFL, 2011.

Gerencsér, B. Markov chain mixing time on cycles. *Stochastic processes and their applications*, 121(11):2553–2570, 2011.

Ghadimi, S. and Lan, G. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Gropp, W., Thakur, R., and Lusk, E. *Using MPI-2: Advanced features of the message passing interface*. MIT press, 1999.

He, L., Bian, A., and Jaggi, M. Cola: Decentralized linear learning. In *Advances in Neural Information Processing Systems*, pp. 4536–4546, 2018.

Hendrickx, J. M., Jungers, R. M., Olshevsky, A., and Vankeerberghen, G. Graph diameter, eigenvalues, and minimum-time consensus. *Automatica*, 50(2):635–640, 2014.

Hendrikx, H., Bach, F., and Massoulié, L. Asynchronous accelerated proximal stochastic gradient for strongly convex distributed finite sums. *arXiv preprint arXiv:1901.09865*, 2019.

Ho, Q., Cipar, J., Cui, H., Lee, S., Kim, J. K., Gibbons, P. B., Gibson, G. A., Ganger, G., and Xing, E. P. More effective distributed ml via a stale synchronous parallel parameter server. In *Advances in neural information processing systems*, pp. 1223–1231, 2013.

Kanawaday, A. and Sane, A. Machine learning for predictive maintenance of industrial machines using iot sensor data. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 87–90. IEEE, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ko, C.-K. *On matrix factorization and scheduling for finite-time average-consensus*. PhD thesis, California Institute of Technology, 2010.

Koloskova, A., Lin, T., Stich, S. U., and Jaggi, M. Decentralized deep learning with arbitrary communication compression. *arXiv preprint arXiv:1907.09356*, 2019a.

Koloskova, A., Stich, S. U., and Jaggi, M. Decentralized stochastic optimization and gossip algorithms with compressed communication. *arXiv preprint arXiv:1902.00340*, 2019b.

Koloskova, A., Loizou, N., Boreiri, S., Jaggi, M., and Stich, S. U. A unified theory of decentralized sgd with changing topology and local updates. *arXiv preprint arXiv:2003.10422*, 2020.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Lan, G. and Zhou, Y. An optimal randomized incremental gradient method. *Mathematical programming*, 171(1-2): 167–215, 2018.

Levin, D. A. and Peres, Y. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

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

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

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 5330–5340, 2017a.

Lian, X., Zhang, W., Zhang, C., and Liu, J. Asynchronous decentralized parallel stochastic gradient descent. *arXiv preprint arXiv:1710.06952*, 2017b.

Lin, T., Stich, S. U., Patel, K. K., and Jaggi, M. Don't use large mini-batches, use local sgd. *arXiv preprint arXiv:1808.07217*, 2018.

Liu, J. and Morse, A. S. Accelerated linear iterations for distributed averaging. *Annual Reviews in Control*, 35(2): 160–165, 2011.

Liu, J. and Zhang, C. Distributed learning systems with first-order methods. *arXiv preprint arXiv:2104.05245*, 2021.

Lu, Y. and De Sa, C. Moniqua: Modulo quantized communication in decentralized sgd. *arXiv preprint arXiv:2002.11787*, 2020.

Lu, Y., Nash, J., and De Sa, C. Mixml: A unified analysis of weakly consistent parallel learning. *arXiv preprint arXiv:2005.06706*, 2020.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

Nazari, P., Tarzanagh, D. A., and Michailidis, G. Dadam: A consensus-based distributed adaptive gradient method for online optimization. *arXiv preprint arXiv:1901.09109*, 2019.

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

Nesterov, Y. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady an ussr*, volume 269, pp. 543–547, 1983.

Patarasuk, P. and Yuan, X. Bandwidth optimal all-reduce algorithms for clusters of workstations. *Journal of Parallel and Distributed Computing*, 69(2):117–124, 2009.

Scaman, K., Bach, F., Bubeck, S., Lee, Y. T., and Massoulié, L. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *international conference on machine learning*, pp. 3027–3036. PMLR, 2017.

Scaman, K., Bach, F., Bubeck, S., Massoulié, L., and Lee, Y. T. Optimal algorithms for non-smooth distributed optimization in networks. In *Advances in Neural Information Processing Systems*, pp. 2740–2749, 2018.

Seaman, K., Bach, F., Bubeck, S., Lee, Y. T., and Massoulié, L. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3027–3036. JMLR. org, 2017.

Shanthamallu, U. S., Spanias, A., Tepedelenlioglu, C., and Stanley, M. A brief survey of machine learning methods and their sensor and iot applications. In *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, pp. 1–8. IEEE, 2017.

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

Sun, H. and Hong, M. Distributed non-convex first-order optimization and information processing: Lower complexity bounds and rate optimal algorithms. *IEEE Transactions on Signal processing*, 67(22):5912–5928, 2019.

Tang, H., Gan, S., Zhang, C., Zhang, T., and Liu, J. Communication compression for decentralized training. In *Advances in Neural Information Processing Systems*, pp. 7652–7662, 2018a.

Tang, H., Lian, X., Yan, M., Zhang, C., and Liu, J. D2: Decentralized training over decentralized data. *arXiv preprint arXiv:1803.07068*, 2018b.

Tang, H., Lian, X., Qiu, S., Yuan, L., Zhang, C., Zhang, T., and Liu, J. Deepsqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. *arXiv preprint arXiv:1907.07346*, 2019a.

Tang, H., Lian, X., Yu, C., Zhang, T., and Liu, J. Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. *arXiv preprint arXiv:1905.05957*, 2019b.

Tian, Y., Sun, Y., and Scutari, G. Achieving linear convergence in distributed asynchronous multiagent optimization. *IEEE Transactions on Automatic Control*, 65(12): 5264–5279, 2020.

Tieleman, T. and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4 (2):26–31, 2012.

Tran, N. H., Bao, W., Zomaya, A., Nguyen, M. N., and Hong, C. S. Federated learning over wireless networks: Optimization model design and analysis. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 1387–1395. IEEE, 2019.

Tran-Dinh, Q., Alacaoglu, A., Fercoq, O., and Cevher, V. An adaptive primal-dual framework for nonsmooth convex minimization. *Mathematical Programming Computation*, pp. 1–41, 2019.

Wan, X., Zhang, H., Wang, H., Hu, S., Zhang, J., and Chen, K. Rat-resilient allreduce tree for distributed machine learning. In *4th Asia-Pacific Workshop on Networking*, pp. 52–57, 2020.

Wang, J. and Joshi, G. Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms. *arXiv preprint arXiv:1808.07576*, 2018.

Wang, J., Sahu, A. K., Yang, Z., Joshi, G., and Kar, S. Matcha: Speeding up decentralized sgd via matching decomposition sampling. *arXiv preprint arXiv:1905.09435*, 2019.

Ward, R., Wu, X., and Bottou, L. Adagrad stepsizes: Sharp convergence over nonconvex landscapes, from any initialization. *arXiv preprint arXiv:1806.01811*, 2018.

Woodworth, B. E., Wang, J., Smith, A., McMahan, B., and Srebro, N. Graph oracle models, lower bounds, and gaps for parallel stochastic optimization. In *Advances in neural information processing systems*, pp. 8496–8506, 2018.

Xin, R., Khan, U. A., and Kar, S. Variance-reduced decentralized stochastic optimization with gradient tracking. *arXiv preprint arXiv:1909.11774*, 2019.

Xin, R., Khan, U. A., and Kar, S. A hybrid variance-reduced method for decentralized stochastic non-convex optimization. *arXiv preprint arXiv:2102.06752*, 2021a.

Xin, R., Khan, U. A., and Kar, S. An improved convergence analysis for decentralized online stochastic non-convex optimization. *IEEE Transactions on Signal Processing*, 69:1842–1858, 2021b.

Yang, Q., Liu, Y., Cheng, Y., Kang, Y., Chen, T., and Yu, H. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(3):1–207, 2019.

Ye, H., Luo, L., Zhou, Z., and Zhang, T. Multi-consensus decentralized accelerated gradient descent. *arXiv preprint arXiv:2005.00797*, 2020.

Yu, C., Tang, H., Renggli, C., Kassing, S., Singla, A., Alistarh, D., Zhang, C., and Liu, J. Distributed learning over unreliable networks. *arXiv preprint arXiv:1810.07766*, 2018.

Yu, H., Jin, R., and Yang, S. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. *arXiv preprint arXiv:1905.03817*, 2019.

Zeiler, M. D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Zhang, J. and You, K. Asynchronous decentralized optimization in directed networks. *arXiv preprint arXiv:1901.08215*, 2019a.

Zhang, J. and You, K. Asyspa: An exact asynchronous algorithm for convex optimization over digraphs. *IEEE Transactions on Automatic Control*, 65(6):2494–2509, 2019b.

Zhang, J. and You, K. Decentralized stochastic gradient tracking for empirical risk minimization. *arXiv preprint arXiv:1909.02712*, 2019c.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

Zhou, D. and Gu, Q. Lower bounds for smooth nonconvex finite-sum optimization. *arXiv preprint arXiv:1901.11224*, 2019.