GT-STORM: Taming Sample, Communication, and Memory Complexities in Decentralized Non-Convex Learning

Xin Zhang
Department of Statistics
Iowa State University
Ames, IA 50011, U.S.A
xinzhang@iastate.edu

Jia Liu
Dept. of Electrical and Computer Engineering
The Ohio State University
Columbus, OH 43210, U.S.A
liu@ece.osu.edu

Zhengyuan Zhu
Department of Statistics
Iowa State University
Ames, IA 50011, U.S.A
zhuz@iastate.edu

Elizabeth Serena Bentley
Air Force Research Laboratory
Information Directorate
Rome, NY, 13441, U.S.A
elizabeth.bentley.3@us.af.mil

ABSTRACT

Decentralized nonconvex optimization has received increasing attention in recent years in machine learning due to its advantages in system robustness, data privacy, and implementation simplicity. However, three fundamental challenges in designing decentralized optimization algorithms are how to reduce their sample, communication, and memory complexities. In this paper, we propose a gradient-tracking-based stochastic recursive momentum (GT-STORM) algorithm for efficiently solving nonconvex optimization problems. We show that to reach an ϵ^2-stationary solution, the total number of sample evaluations of our algorithm is $\tilde{O}(m^{1/2}\epsilon^{-3})$ and the number of communication rounds is $\tilde{O}(m^{-1/2}\epsilon^{-3})$, which improve the $O(\epsilon^{-4})$ costs of sample evaluations and communications for the existing decentralized stochastic gradient algorithms. We conduct extensive experiments with a variety of learning models, including non-convex logistical regression and convolutional neural networks, to verify our theoretical findings. Collectively, our results contribute to the state of the art of theories and algorithms for decentralized network optimization.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; **Distributed algorithms**.

KEYWORDS

Network consensus optimization, decentralized learning, communication complexity, sample complexity, memory complexity.

ACM Reference Format:
Xin Zhang, Jia Liu, Zhengyuan Zhu, and Elizabeth Serena Bentley. 2021. GT-STORM: Taming Sample, Communication, and Memory Complexities in Decentralized Non-Convex Learning. In *The Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '21), July 26–29, 2021, Shanghai, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3466772.3467056

MobiHoc '21, July 26–29, 2021, Shanghai, China

ACM ISBN 978-1-4503-8558-9/21/07.
https://doi.org/10.1145/3466772.3467056

1 INTRODUCTION

In recent years, machine learning has witnessed enormous success in many areas, including image processing, natural language processing, online recommender systems, just to name a few. From a mathematical perspective, training machine learning models amounts to solving an optimization problem. However, with the rapidly increasing dataset sizes and the high dimensionality and the non-convex hardness of the training problem (e.g., due to the use of deep neural networks), training large-scale machine learning models by a single centralized machine has become inefficient and unscalable. To address the efficiency and scalability challenges, an effective approach is to leverage *decentralized* computational resources in a computing network, which could follow a parameter server (PS)-worker architecture [7, 28, 47] or fully decentralized peer-to-peer network structure [19, 25]. Also, thanks to the robustness to single-point-of-failure, data privacy, and implementation simplicity, decentralized learning over computing networks has attracted increasing interest recently, and has been applied in various science and engineering areas (including dictionary learning [5], multi-agent systems [3, 44], multi-task learning [35, 42], information retrieval [1], energy allocation [14], etc.).

In the fast growing literature of decentralized learning over networks, a classical approach is the so-called network consensus optimization, which traces its roots to the seminal work by Tsitsiklis in 1984 [34]. Recently, network consensus optimization has gained a lot of renewed interest owing to the elegant decentralized subgradient descent method (DSGD) proposed by Nedic and Ozdaglar [25], which has been applied in decentralized learning due to its simple algorithmic structure and good convergence performance. In network-consensus-based decentralized learning, a set of geographically distributed computing nodes collaborate to train a common learning model. Each node holds a local dataset that may be too large to be sent to a centralized location due to network communication limits, or cannot be shared due to privacy/security

risks. A distinctive feature of network-consensus-baed decentralized learning is that there is a lack of a dedicated PS. As a result, each node has to exchange information with its local neighbors to reach a consensus on a global optimal learning model.

Despite its growing significance in practice, the design of high-performance network-consensus-based decentralized learning faces three fundamental *conflicting* complexities, namely *sample, communication, and memory complexities*. First, due to the high dimensionality of most deep learning models, it is impossible to leverage beyond first-order (stochastic) gradient information to compute the update direction in each iteration. The variability of a stochastic gradient is strongly influenced by the number of training samples in its mini-batch. However, the more training samples in a mini-batch, the higher computational cost of the stochastic gradient. Second, by using fewer training samples in each iteration to trade for a lower computational cost, the resulting stochastic gradient unavoidably has a larger variance, which further leads to more iterations (hence communication rounds) to reach a certain training accuracy (i.e., slower convergence). The low communication efficiency is particularly problematic in many wireless edge networks, where the communication links could be low-speed and highly unreliable. Lastly, in many mobile edge-computing environments, the mobile devices could be severely limited by hardware resources (e.g., CPU/GPU, memory) and they cannot afford reserving a large memory space to run a very sophisticated decentralized learning algorithm that has too many intermediate variables.

Due to the above fundamental trade-off between sample, communication, and computing resource costs, the notions of sample, communication, and memory complexities (to be formally defined in Section 2) become three of the most important measures in assessing the performances of decentralized learning algorithms. However, in the literature, most existing works have achieved low complexities in some of these measures, but not all (see Section 2 for in-depth discussions). The limitations of these existing works motivate to ask the following question: *Could we design a decentralized learning algorithm that strikes a good balance between sample, communication, and memory complexities?* In this paper, we answer the above question positively by proposing a new GT-STORM algorithm (gradient-tracking-based stochastic recursive momentum) that achieves low sample, communication, and memory complexities. Our main results and contributions are summarized as follows:

- Unlike existing approaches, our proposed GT-STORM algorithm adopts a new estimator, which is updated with a consensus mixing of the neighboring estimators of the last iteration, which helps improve the global gradient estimation. Our method achieves the nice features of previous works [6, 9, 20, 33] while avoiding their pitfalls. To some extent, our GT-STORM algorithm can be viewed as an indirect way of integrating the stochastic gradient, variance reduction, and gradient tracking methods.

- We provide a detailed convergence analysis and complexity analysis. Under some mild assumptions and parameter conditions, our algorithm enjoys an $\tilde{O}(T^{-2/3})$ convergence rate. Note that this rate is much faster than the rate of $O(T^{-1/2})$ for the classic decentralized stochastic algorithms, e.g., DSGD [13], PSGD [19] and GNSD [20]. Also, we show that to reach an ϵ^2-stationary solution, the total number of sample evaluations of our algorithm is $\tilde{O}(m^{1/2}\epsilon^{-3})$ and the communication round is $\tilde{O}(m^{-1/2}\epsilon^{-3})$.

- We conduct extensive experiments to examine the performance of our algorithm, including both a non-convex logistic regression model on the LibSVM datasets and convolutional neural network models on MNIST and CIFAR-10 datasets. Our experiments show that the our algorithm outperforms two state-of-the-art decentralized learning algorithms [19, 20]. These experiments corroborate our theoretical results.

The rest of the paper is organized as follows. In Section 2, we first provide the preliminaries of network consensus optimization and discuss related works with a focus on sample, communication, and memory complexities. In Section 3, we present our proposed GT-STORM algorithm, as well as its communication, sample, and memory complexity analysis. We provide numerical results in Section 4 to verify the theoretical results of our GT-STORM algorithm. Lastly in Section 5, we provide concluding remarks.

2 PRELIMINARIES AND RELATED WORK

To facilitate our technical discussions, in Section 2.1, we first provide an overview on network consensus optimization and formally define the notions of sample, communication, and memory complexities of decentralized optimization algorithms for network consensus optimization. Then, in Section 2.2, we first review centralized stochastic first-order optimization algorithms for solving non-convex learning problems from a historical perspective and with a focus on sample, communication, and memory complexities. Here, we introduce several acceleration techniques that motivate our GT-STORM algorithmic design. Lastly, we review the recent developments of optimization algorithms for decentralized learning and compare them with our work.

2.1 Network Consensus Optimization

As mentioned in Section 1, in decentralized learning, there are a set of geographically distributed computing nodes forming a network. In this paper, we represent such a networked by an undirected connected network $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ and $\mathcal{L}$ are the sets of nodes and edges, respectively, with $|\mathcal{N}| = m$. Each node can communicate with their neighbors via the edges in $\mathcal{L}$. The goal of decentralized learning is to use the nodes to *distributively* and *collaboratively* solve a network-wide optimization problem as follows:

$$\min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x}) = \min_{\mathbf{x}\in\mathbb{R}^p} \frac{1}{m}\sum_{i=1}^{m} f_i(\mathbf{x}), \tag{1}$$

where each local objective function $f_i(\mathbf{x}) \triangleq \mathbb{E}_{\zeta\sim\mathcal{D}_i} f_i(\mathbf{x};\zeta)$ is only observable to node i and not necessarily convex. Here, $\mathcal{D}_i$ represents the distribution of the dataset at node i, and $f_i(\mathbf{x};\zeta)$ represents a loss function that evaluates the discrepancy between the learning model's output and the ground truth of a training sample ζ. To solve Problem (1) in a decentralized fashion, a common approach is to rewrite Problem (1) in the following equivalent form:

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{m}\sum_{i=1}^{m} f_i(\mathbf{x}_i) \\ \text{subject to} \quad & \mathbf{x}_i = \mathbf{x}_j, \qquad \forall (i,j) \in \mathcal{L}, \end{aligned} \tag{2}$$

where $\mathbf{x} \triangleq [\mathbf{x}_1^\top, \cdots, \mathbf{x}_m^\top]^\top$ and $\mathbf{x}_i$ is an introduced local copy at node i. In Problem (2), the constraints ensure that the local copies at all nodes are equal to each other, hence the term "consensus." Thus, Problems (1) and (2) share the same solutions. The main goal of network consensus optimization is to design an algorithm to attain an ϵ^2-stationary point $\mathbf{x}$ defined as follows:

$$\underbrace{\left\|\frac{1}{m}\sum_{i=1}^{m}\nabla f_i(\bar{\mathbf{x}})\right\|^2}_{\text{Global gradient magnitude}} + \underbrace{\frac{1}{m}\sum_{i=1}^{m}\|\mathbf{x}_i-\bar{\mathbf{x}}\|^2}_{\text{Consensus error}} \le \epsilon^2, \tag{3}$$

where $\bar{\mathbf{x}} \triangleq \frac{1}{m}\sum_{i=1}^m \mathbf{x}_i$ denotes the global average across all nodes. Different from the traditional ϵ^2-stationary point in centralized optimization problems, the metric in Eq. (3) has two terms: the first term is the gradient magnitude for the (non-convex) global objective and the second term is the average consensus error of all local copies. To date, many decentralized algorithms have been developed to compute the ϵ^2-stationary point (see Section 2.2). However, most of these algorithms suffer limitations in sample, communication, and memory complexities. In what follows, we formally state the definitions of sample, communication, and memory complexities used in the literature (see, e.g., [32]):

Definition 1 (Sample Complexity). *The sample complexity is defined as the total number of the incremental first-order oracle (IFO) calls required across all the nodes to find an ϵ^2-stationary point defined in Eq. (3), where one IFO call evaluates a pair of $(f_i(\mathbf{x};\zeta), \nabla f_i(\mathbf{x};\zeta))$ on a sample $\zeta \sim \mathcal{D}_i$ and parameter $\mathbf{x} \in \mathbb{R}^p$ at node i.*

Definition 2 (Communication Complexity). *The communication complexity is defined as the total rounds of communications required to find an ϵ^2-stationary point defined in Eq. (3), where each node can send and receive a p-dimensional vector with its neighboring nodes in one communication round.*

Definition 3 (Memory Complexity). *The memory complexity is defined as total dimensionality of all intermediate variables in the algorithm run by a node to find an ϵ^2-stationary point in Eq. (3).*

To make sense of these three complexity metrics, consider the standard centralized gradient descent (GD) method as an example. Note that the GD algorithm has an $O(1/T)$ convergence rate for non-convex optimization, which suggests $O(\epsilon^{-2})$ communication complexity. Also, it takes a full gradient evaluation in each iteration, i.e., $O(n)$ per-iteration sample complexity, where n is the total number of samples. This implies $O(n\epsilon^{-2})$ sample complexity to converge to an ϵ^2-stationary point. Hence, the sample complexity of GD is high if the dataset size n is large.

In contrast, consider the classical stochastic gradient descent (SGD) algorithm that is widely used in machine learning. The basic idea of SGD is to lower the gradient evaluation cost by using only a mini-batch of samples in each iteration. However, due to the sample randomness in mini-batches, the convergence rate of SGD for non-convex optimization is reduced to $O(1/\sqrt{T})$ [2, 12, 45]. Thus, to reach an ϵ^2-stationary point $\mathbf{x}$ with $\|\nabla f(\mathbf{x})\|^2 \le \epsilon^2$, SGD has $O(\epsilon^{-4})$ sample complexity, which could be either higher or lower than the $O(n\epsilon^{-2})$ sample complexity of the GD method, depending on the relationship between n and ϵ. Also, for p-dimensional problems, both GD and SGD have memory complexity p, since they only need a p-dimensional vector to store (stochastic) gradients.

2.2 Related Work

1) Centralized First-Order Methods with Low Complexities: Now, we review several state-of-the-art low-complexity centralized stochastic first-order methods that are related to our GT-STORM algorithm. To reduce the overall sample and communication complexities of the standard GD and SGD algorithms, a natural approach is variance reduction. Earlier works following this approach include SVRG [15, 29], SAGA [8] and SCSG [18]. These algorithms has an overall sample complexity of $O(n + n^{2/3}\epsilon^{-2})$. A more recent variance reduction method is the stochastic path-integrated differential estimator (SPIDER) [11], which is based on the SARAH gradient estimator developed by Nguyen *et al.* [27]. SPIDER further lowers the sample complexity to $O(n + \sqrt{n}\epsilon^{-2})$, which attains the $\Omega(\sqrt{n}\epsilon^{-2})$ theoretical lower bound for finding an ϵ^2-stationary point for $n = O(\epsilon^{-4})$. More recently, to improve the small step-size $O(\epsilon L^{-1})$ in SPIDER, a variant called SpiderBoost was proposed in [36], which allows a larger constant step-size $O(L^{-1})$ while keeping the same $O(n + \sqrt{n}\epsilon^{-2})$ sample complexity. It should be noted, however, that the significantly improved sample complexity of SPIDER/SpiderBoost is due to a restrictive assumption that a universal Lipschitz smoothness constant exists for all local objectives $f(\cdot;\zeta_i)$ $\forall i$. This means that the objectives are "similar" and there are no "outliers" in the training samples. Meanwhile, to obtain the optimal communication complexity, SpiderBoost require a (nearly) full gradient every $\sqrt{n}$ iterations and a mini-batch of stochastic gradient evaluation with batch size $\sqrt{n}$ in each iteration.

To overcome the above limitations, a hybrid stochastic gradient descent (Hybrid-SGD) method is recently proposed in [33], where a convex combination of the SARAH estimator [27] and an unbiased stochastic gradient is used as the gradient estimator. The Hybrid-SGD method relaxes the universal Lipschitz constant assumption in SpiderBoost to an average Lipschitz smoothness assumption. Moreover, it only requires two samples to evaluate the gradient per iteration. As a result, Hybrid-SGD has a $O(\epsilon^{-3})$ sample complexity that is *independent* of dataset size. Although Hybrid-SGD is for centralized optimization, the interesting ideas therein motivate our GT-STORM approach for *decentralized learning* following a similar token. Interestingly, we show that in decentralized settings, our GT-STORM method can further improve the gradient evaluation to only *one sample* per iteration, while not degrading the communication complexity order. Lastly, we remark that all algorithms above have memory complexity at least $2p$ for p-dimensional problems. In contrast, GT-STORM enjoys a memory complexity of p.

2) Decentralized Optimization Algorithms: In the literature, many decentralized learning optimization algorithms have been proposed to solve Problem (1), e.g., first-order methods [9, 25, 30, 39], prime-dual methods [23, 31], Newton-type methods [10, 22] (see in [4, 24] for comprehensive surveys). In this paper, we consider decentralized first-order methods for the non-convex network consensus optimization in (2). In the literature, the convergence rate of the well-known decentralized gradient descent (DGD) algorithm [25] was studied in [41], which showed that DGD with a constant step-size converges with an $O(1/T)$ rate to a step-size-dependent

error ball around a stationary point. Later, a gradient tracking (GT) method was proposed in [9] to find an ϵ^2-stationary point with an $O(1/T)$ convergence rate under constant step-sizes. However, these methods require a full gradient evaluation per iteration, which yields $O(n\epsilon^{-2})$ sample complexity. To reduce the per-iteration sample complexity, stochastic gradients are adopted in the decentralized optimization (e.g., DSGD [13], PSGD [19], GNSD [20]). Due to the randomness in stochastic gradients, the convergence rate is reduced to $O(1/\sqrt{T})$. Thus, the sample and communication complexities of these stochastic methods are $O(\epsilon^{-4})$ and $O(m^{-1}\epsilon^{-4})$, two orders of magnitude higher than their deterministic counterparts. To overcome the limitations in stochastic methods, a natural idea is to use variance reduction techniques similar to those for centralized optimization to reduce the sample and communication complexities for non-convex network consensus optimization. So far, existing works on the decentralized stochastic variance reduction methods include DSA [21], diffusion-AVRG [40] and GT-SAGA [38] etc., all of which focus on convex problems. To our knowledge, the decentralized gradient estimation and tracking (D-GET) algorithm in [32] is the only work for non-convex optimization. D-GET integrates the decentralized gradient tracking [20] and the SpiderBoost gradient estimator [36] to obtain $O(mn + m\sqrt{n}\epsilon^{-2})$ *dataset-size-dependent* sample complexity and $O(\epsilon^{-2})$ communication complexity. Recall that the sample and communication complexities of GT-STORM are $O(m^{1/2}\epsilon^{-3})$ and $O(m^{-1/2}\epsilon^{-3})$, respectively. Thus, if dataset size $n = \Omega(\epsilon^{-2})$, D-GET has a higher sample complexity than GT-STORM. As an example, when $\epsilon = 10^{-2}$, n is on the order of 10^4, which is common in modern machine learning datasets. Also, the memory complexity of D-GET is $2p$ as opposed to the p memory complexity of GT-STORM. This implies a huge saving with GT-STORM if p is large (e.g., $p \approx 10^6$ in many deep learning models).

3 A GRADIENT-TRACKING STOCHASTIC RECURSIVE MOMENTUM ALGORITHM

In this section, we introduce our gradient-tracking-based stochastic recursive momentum (GT-STORM) algorithm for solving Problem (2) in Section 3.1. Then, we will state the main theoretical results and their proofs in Sections 3.2 and 3.3, respectively.

3.1 The GT-STORM Algorithm

In the literature, a standard starting point to solve Problem (2) is to reformulate the problem as [25]:

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{m}\sum_{i=1}^{m} f_i(\mathbf{x}_i) \\ \text{subject to} \quad & (\mathbf{W}\otimes \mathbf{I}_p)\mathbf{x} = \mathbf{x}, \end{aligned} \tag{4}$$

where $\mathbf{I}_p$ denotes the p-dimensional identity matrix, the operator $\otimes$ denotes the Kronecker product, and $\mathbf{W} \in \mathbb{R}^{m\times m}$ is often referred to as the consensus matrix. We let $[\mathbf{W}]_{ij}$ represent the element in the i-th row and the j-th column in $\mathbf{W}$. For Problems (4) and (2) to be equivalent, $\mathbf{W}$ should satisfy the following properties:

(a) *Doubly Stochastic:* $\sum_{i=1}^m [\mathbf{W}]_{ij} = \sum_{j=1}^m [\mathbf{W}]_{ij} = 1$.
(b) *Symmetric:* $[\mathbf{W}]_{ij} = [\mathbf{W}]_{ji}, \forall i, j \in \mathcal{N}$.
(c) *Network-Defined Sparsity Pattern:* $[\mathbf{W}]_{ij} > 0$ if $(i, j) \in \mathcal{L}$; otherwise $[\mathbf{W}]_{ij} = 0, \forall i, j \in \mathcal{N}$.

The above properties imply that the eigenvalues of $\mathbf{W}$ are real and can be sorted as $-1 < \lambda_m \leq \cdots \leq \lambda_2 < \lambda_1 = 1$. We define the second-largest eigenvalue in magnitude of $\mathbf{W}$ as $\lambda \triangleq \max\{|\lambda_2|, |\lambda_m|\}$ for the further notation convenience. It can be seen later that λ plays an important role in the step-size selection and the algorithm's convergence rate.

As mentioned in Section 2.1, our GT-STORM algorithm is inspired by the GT method [9, 26] for reducing consensus error and the recursive variance reduction (VR) methods [11, 36] developed for centralized optimization. Specifically, in the centralized GT method, an estimator $\mathbf{y}$ is introduced to track the global gradient:

$$\mathbf{y}_t = \mathbf{W}\mathbf{y}_{t-1} + \mathbf{g}_t - \mathbf{g}_{t-1}, \tag{5}$$

where $\mathbf{g}_t$ is the gradient estimation in the tth iteration. Meanwhile, to reduce the stochastic error, a gradient estimator $\mathbf{v}$ in VR methods is updated recursively based on a *double-loop* structure as follows:

$$\mathbf{v}_t = \mathbf{v}_{t-1} + \nabla f(\mathbf{x}_t;\zeta_t) - \nabla f(\mathbf{x}_{t-1};\zeta_t), \quad \text{if } \mathrm{mod}(t,q) \neq 0, \tag{6}$$

where $\nabla f(\mathbf{x};\zeta)$ is the stochastic gradient dependent on parameter $\mathbf{x}$ and a data sample ζ, and q is the number of the inner loop iterations. On the other hand, if $\mathrm{mod}(t,q) = 0$, $\mathbf{v}_t$ takes a full gradient. Note that these two estimators have a *similar* structure: Both are *recursively* updating the previous estimation based on the difference of the gradient estimations between two consecutive iterations (i.e., momentum). However, they are also quite different due to the double-loop structure in (6) and it is unclear how to integrate them. This motivates us to consider the following question: *Could we somehow "integrate" these two methods to develop a new decentralized gradient estimator to track the global gradient and reduce the stochastic error at the same time?* Unfortunately, the GT and VR estimators can not be combined straightforwardly. The major challenge lies in the structural difference in the outer loop iteration (i.e., $\mathrm{mod}(t,q) = 0$), where the VR estimator requires a full gradient and does not follow the recursive updating structure.

Surprisingly, in this paper, we show that there exists an "indirect" way to achieve the salient features of both GT and VR. Our approach is to abandon the double-loop structure of VR and pursue a *single-loop* structure. Yet, this single-loop structure should still be able to reduce the variance and consistently track the global gradient. Specifically, we introduce a parameter $\beta_t \in [0, 1]$ in the recursive update and integrate it with a consensus step as follows:

$$\mathbf{v}_{i,t} = \beta_t \sum\nolimits_{j\in\mathcal{N}_i} [\mathbf{W}]_{ij}\mathbf{v}_{j,t-1} + \nabla f_i(\mathbf{x}_{i,t};\zeta_{i,t}) - \beta_t \nabla f_i(\mathbf{x}_{i,t-1};\zeta_{i,t}), \tag{7}$$

where $\mathbf{x}_{i,t}$, $\mathbf{v}_{i,t}$ and $\zeta_{i,t}$ are the parameter, gradient estimator, and random sample in the tth iteration at node i, respectively. Note that the estimator reduces to the classical stochastic gradient estimator when $\beta_t = 0$. On the other hand, if we set $\beta_t = 1$, the estimator becomes the (stochastic) gradient tracking estimator based on a single sample (implying low sample complexity). Then, the key to the success of our GT-STORM design lies in meticulously choosing parameter β_t to mimic the gradient estimator technique in centralized optimization [6, 33]. Lastly, the local parameters can be updated by the conventional decentralized stochastic gradient descent step:

$$\mathbf{x}_{i,t+1} = \sum\nolimits_{j\in\mathcal{N}_i} [\mathbf{W}]_{ij}\mathbf{x}_{j,t} - \eta_t \mathbf{v}_{i,t}, \tag{8}$$

where η_t is the step-size in iteration t. To summarize, we state our algorithm in Algorithm 1 as follows.

Algorithm 1: Gradient-Tracking-based Stochastic Recursive Momentum Algorithm (GT-STORM).

Initialization:

1. Choose $T > 0$ and let $t = 1$. Set $\mathbf{x}_{i,0} = \mathbf{x}^0$ at node i. Calculate $\mathbf{v}_{i,0} = \nabla f_i(\mathbf{x}_{i,0}; \zeta_{i,0})$ at node i.

Main Loop:

2. In the t-th iteration, each node sends $\mathbf{x}_{i,t-1}$ and local gradient estimator $\mathbf{v}_{i,t-1}$ to its neighbors. Meanwhile, upon the reception of all neighbors' information, each node performs the following:
   a) Update local parameter: $\mathbf{x}_{i,t} = \sum_{j\in\mathcal{N}_i}[\mathbf{W}]_{ij}\mathbf{x}_{j,t-1} - \eta_{t-1}\mathbf{v}_{i,t-1}$.
   b) Update local gradient estimator: $\mathbf{v}_{i,t} = \beta_t \sum_{j\in\mathcal{N}_i}[\mathbf{W}]_{ij}\mathbf{v}_{j,t-1} + \nabla f_i(\mathbf{x}_{i,t};\zeta_{i,t}) - \beta_t \nabla f_i(\mathbf{x}_{i,t-1};\zeta_{i,t})$.
3. Stop if $t > T$; otherwise, let $t \leftarrow t+1$ and go to Step 2.

Two remarks for Algorithm 1 are in order. First, thanks to the single-loop structure, GT-STORM is easier to implement compared to the low-sample-complexity D-GET [32] method, which has in a double-loop structure. Second, GT-STORM only requires p memory space due to the use of only one intermediate vector $\mathbf{v}$ at each node. In contrast, the memory complexity of D-GET is $2p$ (cf. $\mathbf{y}$ and $\mathbf{v}$ in [32]). This 50% saving is huge particularly for deep learning models, where the number of parameters could be in the range of millions.

3.2 Main Theoretical Results

In this section, we will establish the complexity properties of the proposed GT-STORM algorithm. For better readability, we state the main theorem and its corollary in this section and provide the intermediate lemmas in Section 3.3. We start with the following assumptions on the global and local objectives:

Assumption 1. *The objective function $f(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^m f_i(\mathbf{x})$ with $f_i(\mathbf{x}) = \mathbb{E}_{\zeta\sim\mathcal{D}_i} f_i(\mathbf{x};\zeta)$ satisfies the following assumptions:*

(a) Boundedness from below: *There exists a finite lower bound $f^* = \inf_{\mathbf{x}} f(x) > -\infty$;*

(b) L-average smoothness: *$f_i(\cdot;\zeta_i)$ is L-average smooth on $\mathbb{R}^p$, i.e., there exists a positive constant L, such that $\mathbb{E}_{\zeta\sim\mathcal{D}_i}[\|\nabla f_i(\mathbf{x};\zeta) - \nabla f_i(\mathbf{y};\zeta)\|^2] \le L^2\|\mathbf{x}-\mathbf{y}\|^2, \forall \mathbf{x},\mathbf{y}\in\mathbb{R}^p, i\in[m]$;*

(c) Bounded variance: *There exists a constant $\sigma \ge 0$ such that $\mathbb{E}_{\zeta\sim\mathcal{D}_i}[\|\nabla f_i(\mathbf{x};\zeta) - \nabla f_i(\mathbf{x})\|^2] \le \sigma^2, \forall \mathbf{x}\in\mathbb{R}^p, i\in[m]$;*

(d) Bounded gradient: *There exists a constant $G \ge 0$ such that $\mathbb{E}_{\zeta\sim\mathcal{D}_i}[\|\nabla f_i(\mathbf{x};\zeta)\|^2] \le G^2, \forall \mathbf{x}\in\mathbb{R}^p, i\in[m]$.*

In the above assumptions, (a) and (c) are standard in the stochastic non-convex optimization literature; (b) is an expected Lipschitz smoothness condition over the data distribution, which implies the conventional global Lipschitz smoothness [12] by the Jensen's inequality. Note that (b) is weaker than the individual Lipschitz smoothness in [11, 32, 36]: if there exists an outlier data sample, then the individual objective function might have a very large smoothness parameter while the average smoothness can still be small; (d) is equivalent to the Lipschitz continuity assumption, which is also commonly used for non-convex stochastic algorithms [16, 17, 46] and is essential for analyzing the decentralized gradient descent method [13, 39, 41].[1]

[1]Note that under Assumption (b), as long as the parameter $\mathbf{x}$ is bounded, (d) is satisfied.

For convenience, in the subsequent analysis, we define $\tilde{\mathbf{W}} = \mathbf{W}\otimes\mathbf{I}_m$, $\mathbf{g}_{i,t} = \nabla f_i(\mathbf{x}_{i,t})$, $\mathbf{u}_{i,t} = \nabla f_i(\mathbf{x}_{i,t};\zeta_{i,t})$, $\mathbf{w}_{i,t} = \nabla f_i(\mathbf{x}_{i,t};\zeta_{i,t}) - \nabla f_i(\mathbf{x}_{i,t-1};\zeta_{i,t})$ and $\mathbf{a}_t = [\mathbf{a}_{1,t}^\top, \cdots, \mathbf{a}_{m,t}^\top]^\top$ and $\bar{\mathbf{a}}_t = \frac{1}{m}\sum_{i=1}^m \mathbf{a}_{i,t}$, for $\mathbf{a}\in\{\mathbf{x},\mathbf{u},\mathbf{w},\mathbf{v},\mathbf{g}\}$. Then, the algorithm can be compactly rewritten in the following matrix-vector form:

$$\mathbf{x}_t = \tilde{\mathbf{W}}\mathbf{x}_{t-1} - \eta_{t-1}\mathbf{v}_{t-1}, \tag{9}$$

$$\mathbf{v}_t = \beta_t\tilde{\mathbf{W}}\mathbf{v}_{t-1} + \beta_t\mathbf{w}_t + (1-\beta_t)\mathbf{u}_t. \tag{10}$$

Furthermore, since $\mathbf{1}^\top\mathbf{W} = \mathbf{1}^\top$, we have $\bar{\mathbf{x}}_t = \bar{\mathbf{x}}_{t-1} - \eta_{t-1}\bar{\mathbf{v}}_{t-1}$, $\bar{\mathbf{v}}_t = \beta_t\bar{\mathbf{v}}_{t-1} + \beta_t\bar{\mathbf{w}}_t + (1-\beta_t)\bar{\mathbf{u}}_t$. We first state the convergence result for Algorithm 1 as follows:

Theorem 1. *Under Assumption 1 and with the positive constants c_0 and c_1 satisfying $1-(1+c_1)\lambda^2 - \frac{1}{c_0} > 0$, if we set $\eta_t = \tau/(\omega+t)^{1/3}$ and $\beta_{t+1} = 1-\rho\eta_t^2$, with $\tau > 0$, $\omega \ge \max\{2, \tau^3/\min\{k_1^3, k_2^3, k_3^3\}\}$ and $\rho = 2/(3\tau^3) + 32L^2$, then we have the following result for Algorithm 1:*

$$\begin{aligned}
&\min_{t\in[T]} \mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_t)\|^2] + \frac{1}{m}\mathbb{E}[\|\mathbf{x}_t - \mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2] \\
\le& \frac{2(f(\bar{\mathbf{x}}_0) - f(\bar{\mathbf{x}}^*))}{\tau(T+1)^{2/3}} + \frac{2c_0\mathbb{E}[\|\mathbf{v}_0 - \mathbf{1}\otimes\bar{\mathbf{v}}_0\|^2]}{m\tau(T+1)^{2/3}} \\
&+ \frac{(\omega-1)\sigma^2}{16mL^2\tau^2(T+1)^{2/3}} + \frac{\rho^2\sigma^2\ln(\omega+T-1)}{8mL^2(T+1)^{2/3}} \\
&+ \frac{12(1+\frac{1}{c_1})c_0\tau^{1/3}G^2\rho^2}{(\omega-1)^{1/3}(T+1)^{2/3}} + O\Big(\frac{c_3\omega}{\tau T^{5/3}}\Big),
\end{aligned} \tag{11}$$

where $c_3 = \max\{1, \omega/(m\tau^2), \tau^{4/3}/\omega^{1/3}, \tau\ln(\omega+T)/m\}$, and the constants k_1, k_2 and k_3 are:

$$k_1 = 1/\Big(2L + 32(1+\frac{1}{c_1})c_0L^2\Big), \tag{12}$$

$$k_2 = \Big(1-(1+c_1)\lambda^2\Big)/\Big(1+\frac{1}{c_1}+\frac{1}{c_0}\Big), \tag{13}$$

$$k_3 = \sqrt{\Big(1-(1+c_1)\lambda^2 - \frac{1}{c_0}\Big)/\Big(\frac{2}{3\tau^3} + \frac{2L^2+1}{2c_0}\Big)}. \tag{14}$$

In Theorem 1, c_0 and c_1 are two constants depending on the network topology, which in turn will affect the step-size and convergence: with a sparse network, i.e., λ is close to but not exactly one (recall that $\lambda = \max\{|\lambda_2|, |\lambda_m|\}$). In order for $1-(1+c_1)\lambda^2 - \frac{1}{c_0} > 0$ to hold, c_0 needs to be large and c_1 needs be close to zero, which leads to small k_1, k_2 and k_3. Note that the step-size η_t is of the order $O(t^{-1/3})$, which is larger than the $O(t^{-1/2})$ order for the classical decentralized SGD algorithms. With this larger step-size, the convergence rate is $O(t^{-2/3})$ and faster than the rate $O(t^{-1/2})$ for the decentralized SGD algorithms. Based on Theorem 1, we have the sample and communication complexity results for Algorithm 1:

Corollary 2. *Under the conditions in Theorem 1, if $\tau = O(m^{1/3})$ and $\omega = O(m^{4/3})$, then to achieve an ϵ^2-stationary solution, the total communication rounds are on the order of $\tilde{O}(m^{-1/2}\epsilon^{-3})$ and the total samples evaluated across the network is on the order of $\tilde{O}(m^{1/2}\epsilon^{-3})$.*

3.3 Proofs of the Theoretical Results

Due to space limitation, we provide a proof sketch for Theorem 1 here and relegate the details to the appendices. First, we bound the error of gradient estimator $\mathbb{E}[\|\mathbf{v}_t - \mathbf{g}_t\|^2]$ as follows:

Lemma 1 (Error of Gradient Estimator). *Under Assumption 1 and with $\mathbf{v}_t$ defined in (10), it holds that* $\mathbb{E}[\|\bar{\mathbf{v}}_t-\bar{\mathbf{g}}_t\|^2] \le \beta_t^2\mathbb{E}[\|\bar{\mathbf{v}}_{t-1}-\bar{\mathbf{g}}_{t-1}\|^2] + \frac{2\beta_t^2L^2}{m}\mathbb{E}[\|\mathbf{x}_t-\mathbf{x}_{t-1}\|^2] + \frac{2(1-\beta_t)^2\sigma^2}{m}$.

It can be seen that the upper bound depends on the error in the previous step with a factor β_t^2. This will be helpful when we construct a potential function. Then, according to the algorithm updates (9)–(10), we show the following descent inequality:

Lemma 2 (Descent Lemma). *Under Assumption 1, Algorithm 1 satisfies:* $\mathbb{E}[f(\bar{\mathbf{x}}_{t+1})]-\mathbb{E}[f(\bar{\mathbf{x}}_t)] \le -\frac{\eta_t}{2}\mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_t)\|^2]-(\frac{\eta_t}{2}-\frac{L\eta_t^2}{2})\times \mathbb{E}[\|\bar{\mathbf{v}}_t\|^2] + \eta_t\mathbb{E}[\|\bar{\mathbf{v}}_t-\bar{\mathbf{g}}_t\|^2] + \frac{L^2\eta_t}{m}\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2]$.

We remark that the right-hand-side (RHS) of the above inequality contains the consensus error of local parameters $\sum_{t=0}^{T}\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2]$, which makes the analysis more difficult than that of the centralized optimization. Next, we prove the contraction of iterations in the following lemma, which is useful in analyzing our decentralized gradient tracking algorithm.

Lemma 3 (Iterates Contraction). *The following contraction properties of the iterates produced by Algorithm 1 hold:*

$$\|\mathbf{x}_{t+1}-\mathbf{1}\otimes\bar{\mathbf{x}}_{t+1}\|^2 \le (1+c_1)\lambda^2\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2 + (1+\frac{1}{c_1})\eta_t^2\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2, \quad (15)$$

$$\|\mathbf{v}_{t+1}-\mathbf{1}\otimes\bar{\mathbf{v}}_{t+1}\|^2 \le (1+c_1)\beta_{t+1}^2\lambda^2\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2 + 2(1+\frac{1}{c_1})\big(\beta_{t+1}^2\|\mathbf{w}_{t+1}\|^2+(1-\beta_{t+1})^2\|\mathbf{u}_{t+1}\|^2\big), \quad (16)$$

where c_1 is a positive constant. Additionally, we have

$$\|\mathbf{x}_{t+1}-\mathbf{x}_t\|^2 \le 8\|(\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t)\|^2 + 4\eta_t^2\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2+4\eta_t^2m\|\bar{\mathbf{v}}_t\|^2. \quad (17)$$

Finally, we define a potential function in (18), based on which we prove the convergence bound:

Lemma 4. *(Convergence of Potential Function) Define the following potential function:*

$$H_t = \mathbb{E}[f(\bar{\mathbf{x}}_t)+\frac{1}{32L^2\eta_{t-1}}\|\bar{\mathbf{g}}_t-\bar{\mathbf{v}}_t\|^2+\frac{c_0}{m\eta_{t-1}}\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2 + \frac{c_0}{m}\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2], \quad (18)$$

where c_0 is a positive constant. Under Assumption 1, if we set $\eta_t = \tau/(\omega+t)^{1/3}$ and $\beta_{t+1}=1-\rho\eta_t^2$, where $\tau,\omega\ge 2$, $\rho = 2/(3\tau^3)+32L^2$ are three constants, then it holds that:

$$H_{t+1}-H_t \le -\frac{\eta_t}{2}\mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_t)\|^2]+\frac{\rho^2\sigma^2\eta_t^3}{16mL^2}+2(1+\frac{1}{c_1})c_0G^2\rho^2\eta_t^4 - \frac{c_0C_1}{m\eta_t}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2]-\frac{c_0C_2}{m}\mathbb{E}[\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2] - \frac{C_3\eta_t}{4}\mathbb{E}[\|\bar{\mathbf{v}}_t\|^2], \quad (19)$$

where C_1, C_2, and C_3 are constants: $C_1 = 1-(1+c_1)\lambda^2-\frac{1}{2c_0}-16(1+\frac{1}{c_1})L^2\eta_t-\left(\frac{2}{3\tau^3}+\frac{L^2}{c_0}\right)\eta_t^2$, $C_2 = 1-(1+c_1)\lambda^2-(1+\frac{1}{c_1})\eta_t-\frac{\eta_t}{4c_0}-8(1+\frac{1}{c_1})L^2\eta_t^2$, and $C_3 = 1-2L\eta_t-32(1+\frac{1}{c_1})c_0L^2\eta_t$.

Finally, by properly selecting the parameters, constants C_1, C_2 and C_3 can be made non-negative, which leads to Theorem 1.

4 EXPERIMENTAL RESULTS

In this section, we conduct experiments using several non-convex machine learning problems to evaluate the performance of our method. In particular, we compare our algorithm with the following state-of-art *single-loop* algorithms:

- DSGD [13, 25, 39]: Each node performs: $\mathbf{x}_{i,t+1} = \sum_{j\in\mathcal{N}_i}[\mathbf{W}]_{ij}\mathbf{x}_{j,t} - \eta\nabla f_i(\mathbf{x}_{i,t};\zeta_{i,t})$, where the stochastic gradient $\nabla f_i(\mathbf{x}_{i,t};\zeta_{i,t})$ corresponds to random sample $\zeta_{i,t}$. Then, each node exchanges the local parameter $\mathbf{x}_{i,t}$ with its neighbors.
- GNSD [20]: Each node keeps two variables $\mathbf{x}_{i,t}$ and $\mathbf{y}_{i,t}$. The local parameter $\mathbf{x}_{i,t}$ is updated as $\mathbf{x}_{i,t+1}=\sum_{j\in\mathcal{N}_i}[\mathbf{W}]_{ij}\mathbf{x}_{j,t}-\eta\mathbf{y}_{i,t}$ and the tracked gradient $\mathbf{y}_{i,t}$ is updated as $\mathbf{y}_{i,t+1}=\sum_{j\in\mathcal{N}_i}[\mathbf{W}]_{ij}\mathbf{y}_{j,t}+\nabla f_i(\mathbf{x}_{i,t+1};\zeta_{i,t+1})-\nabla f_i(\mathbf{x}_{i,t};\zeta_{i,t})$.

Here, we compare with the above two classes of stochastic algorithms because they all employ a single-loop structure and do not require full gradient evaluations. We note that it is hard to have a direct and fair comparison with D-GET [32] numerically, since D-GET uses full gradients and has a double-loop structure.

Network Model: The communication graph $\mathcal{G}$ is generated by the Erdös-Rènyi graph with different edge connectivity probability p_c and number of nodes m. We set $m = 10$ and choose the edge connectivity probability as $p_c = 0.5$. The consensus matrix is chosen as $\mathbf{W} = \mathbf{I}-\frac{2}{3\lambda_{\max}(\mathbf{L})}\mathbf{L}$, where $\mathbf{L}$ is the Laplacian matrix of $\mathcal{G}$, and $\lambda_{\max}(\mathbf{L})$ denotes the largest eigenvalue of $\mathbf{L}$.

1) Non-convex logistic regression: In our first experiment, we consider the binary logistic regression problem with a non-convex regularizer [33, 36, 37]:

$$\min_{\mathbf{x}\in\mathbb{R}^d} -\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[y_{ij}\log\big(\frac{1}{1+e^{-\mathbf{x}^\top\zeta_{ij}}}\big)+ (1-y_{ij})\log\big(\frac{e^{-\mathbf{x}^\top\zeta_{ij}}}{1+e^{-\mathbf{x}^\top\zeta_{ij}}}\big)]+\alpha\sum_{i=1}^{d}\frac{\mathbf{x}_i^2}{1+\mathbf{x}_i^2}, \quad (20)$$

where the label $y_{ij}\in\{0,1\}$, the feature $\zeta_{ij}\in\mathbb{R}^d$ and $\alpha = 0.1$.

1-a) Datasets: We consider three commonly used binary classification datasets from LibSVM: *a9a*, *rcv1.binary* and *ijcnn1*. The *a9a* dataset has 32561 samples and 123 features, the *rcv1.binary* dataset has 20242 samples and 47236 features, and the *ijcnn1* dataset has 49990 samples and 22 features. We evenly divide the dataset into m sub-datasets corresponding to the m nodes.

1-b) Parameters: For all algorithms, we set the batch size as one and the initial step-size η_0 is tuned by searching over the grid $\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0\}$. For DSGD and GNSD, the step-size is set to $\eta_t = \eta_0/\sqrt{1+0.1t}$, which is on the order of $O(t^{-1/2})$ following the state-of-the-art theoretical result [20]. For GT-STORM, the step-size is set as $\eta_t = \eta_0/\sqrt[3]{1+0.1t}$, which is on the order of $O(t^{-1/3})$ as specified in our theoretical result. In addition, we choose the parameter ρ for GT-STORM as $1/\eta_0^2$, so that $\beta_1 = 0$ in the first step.

1-c) Results: We first compare the convergence rates of the algorithms. We adopt the consensus loss defined in the left-hand-side (LHS) of (3) as the criterion. After tuning, the best initial step-sizes are 0.1, 0.5 and 0.2 for *a9a*, *ijcnn1* and *rcv1.binary*, respectively. The results are shown in Figs. 1–3. It can be seen that our algorithm

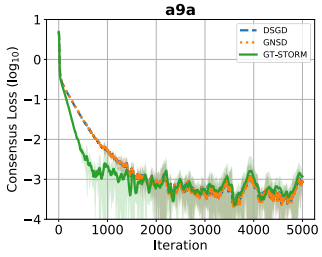

Figure 1: Non-convex logistic regression on LibSVM: a9a.

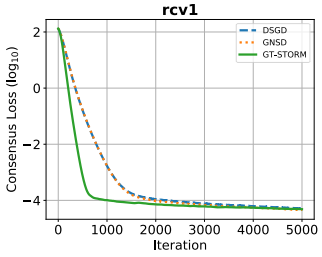

Figure 2: Non-convex logistic regression on LibSVM: ijcnn1.

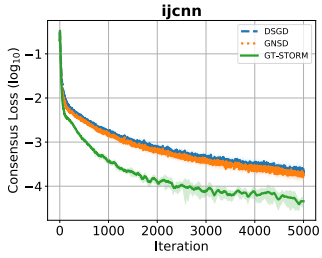

Figure 3: Non-convex logistic regression on LibSVM: rcv1.

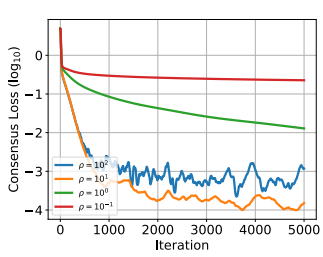

Figure 4: Non-convex logistic regression: The effect of ρ.

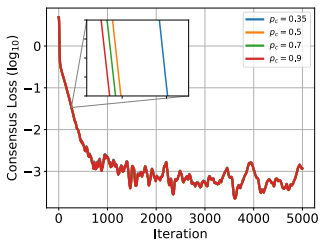

Figure 5: Non-convex logistic regression: The effect of p_c.

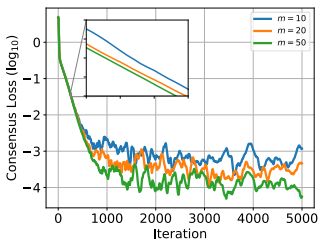

Figure 6: Non-convex logistic regression: The effect of m.

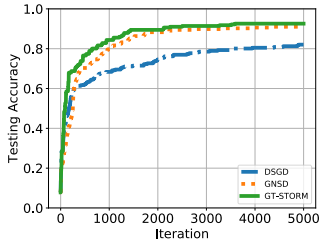

Figure 7: CNN experimental results on MNIST dataset.

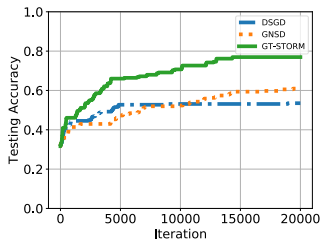

Figure 8: CNN experimental results on CIFAR-10 dataset.

has a better performance: for the *a9a* and *rcv1.binary* datasets, all algorithms reach almost the same accuracy but our algorithm has a faster speed; for the *ijcnn1* dataset, our algorithm outperforms other methods in terms of both speed and accuracy.

Next, we examine the effect of the parameter ρ on our algorithm. We focus on the *a9a* dataset and fix the initial step-size as $\eta_0 = 0.1$. We choose ρ from $\{10^{-1}, 10^0, 10^1, 10^2\}$. Note that $\rho = 10^2$ is corresponding to the case $\rho = 1/\eta_0^2$. The results are shown in Fig. 4. It can be seen that the case $\rho = 10^1$ has the best performance, which is followed by the case $\rho = 10^2$. Also, as ρ decreases, the convergence speed becomes slower (see the cases $\rho = 10^{-1}$ and 10^0).

In addition, we examine the effect of the network topology. We first fix the number of workers as $m = 10$ and change the the edge connectivity probability p_c from 0.35 to 0.9. Note that with a smaller p_c, the network becomes sparser. We set $\eta_0 = 0.1$ and $\rho = 10^2$. The results are shown in Fig. 5. Under different p_c-values, our algorithm has a similar performance in terms of convergence speed and accuracy. But with a larger p_c-values i.e., a denser network, the convergence speed slightly increases (see the zoom-in view in Fig. 6. Then, we fix the the edge connectivity probability $p_c = 0.5$ but change the number of workers m from 10 to 50. We show the results in Fig. 6. It can be seen that with more workers, the algorithm converges faster and reaches a better accuracy.

2) Convolutional neural networks We use all three algorithms to train a convolutional neural network (CNN) model for image classification on MNIST and CIFAR-10 datasets. We adopt the same network topology as in the previous experiment. We use a non-identically distributed data partition strategy: the ith machine can access the data with the ith label. We fix the initial step-size as $\eta_0 = 0.01$ for all three algorithms and the remaining settings are the same as in the previous experiment.

2-a) Learning Models: For MNIST, the adopted CNN model has two convolutional layers (first of size $1 \times 16 \times 5$ and then of size $16 \times 32 \times 5$), each of which is followed by a max-pooling layer with size 2×2, and then a fully connected layer. The ReLU activation is used for the two convolutional layers and the "softmax" activation is applied at the output layer. The batch size is 64 for the CNN training on MNIST. For CIFAR-10, we apply the CNN model with two convolutional layers (first of size $3 \times 6 \times 5$ and then of size $6 \times 16 \times 5$). Each of the convolutional layers is followed by a max-pooling layer of size 2×2, and then three fully connected layers. The ReLU activation is used for the two convolutional layers and the first two fully connected layers, and the "softmax" activation is applied at the output layer. The batch size is chosen as 128 for the CNN training on CIFAR-10.

2-b) Results: Fig. 7 illustrates the testing accuracy of different algorithms versus iterations on MNIST and CIFAR-10 datasets. It can be seen from Fig. 7 that on the MNIST dataset, GNSD and GT-STORM have similar performance, but our GT-STORM maintains a faster speed and a better prediction accuracy. Compared with DSGD, our GT-STORM can gain about 10% more accuracy. On the CIFAR-10 dataset (see Fig. 8), the performances of DSGD and GNSD deteriorate, while GT-STORM can achieve a better accuracy. Specifically, the accuracy of GT-STORM is around 15% higher than that of GNSD and 25% higher than that of DSGD.

5 CONCLUSION

In this paper, we proposed a gradient-tracking-based stochastic recursive momentum (GT-STORM) algorithm for decentralized non-convex optimization, which enjoys low sample, communication, and memory complexities. Our algorithm fuses the gradient tracking estimator and the variance reduction estimator and has a simple single-loop structure. Thus, it is more practical compared to

existing works (e.g. GT-SAGA/SVRG and D-GET) in the literature. We have also conducted extensive numerical studies to verify the performance of our method, including non-convex logistic regression and neural networks. The numerical results show that our method outperforms the state-of-the-art methods when training on the large datasets. Our results in this work contribute to the increasingly important field of decentralized network training.

ACKNOWLEDGMENTS

J. Liu's work is supported in part by NSF grants CAREER CNS-2110259, ECCS-1818791, CCF-2110252, TRIPODS CCF-1934884, ONR grant N00014-17-1-2417, AFRL grant FA8750-18-1-0107, and a Google Faculty Research Award. Z. Zhu's work is supported in part by NSF Grant TRIPODS CCF-1934884. DISTRIBUTION STATEMENT A: Approved for Public Release; distribution unlimited AFRL-2020-0026 on 06 Aug 2020. Other requests shall be referred to AFRL/RIT 525 Brooks Rd Rome, NY 13441.

REFERENCES

[1] Kamal Ali and Wijnand Van Stam. 2004. TiVo: Making show recommendations using a distributed collaborative filtering architecture. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 394–401.

[2] Léon Bottou, Frank E Curtis, and Jorge Nocedal. 2018. Optimization methods for large-scale machine learning. *Siam Review* 60, 2 (2018), 223–311.

[3] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. 2012. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics* 9, 1 (2012), 427–438.

[4] Tsung-Hui Chang, Mingyi Hong, Hoi-To Wai, Xinwei Zhang, and Songtao Lu. 2020. Distributed Learning in the Non-Convex World: From Batch to Streaming Data, and Beyond. *arXiv preprint arXiv:2001.04786* (2020).

[5] Jianshu Chen, Zaid J Towfic, and Ali H Sayed. 2014. Dictionary learning over distributed models. *IEEE Transactions on Signal Processing* 63, 4 (2014), 1001–1016.

[6] Ashok Cutkosky and Francesco Orabona. 2019. Momentum-based variance reduction in non-convex SGD. In *Advances in Neural Information Processing Systems*. 15210–15219.

[7] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. 2012. Large scale distributed deep networks. In *Advances in neural information processing systems*. 1223–1231.

[8] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. 2014. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*. 1646–1654.

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

[10] Mark Eisen, Aryan Mokhtari, and Alejandro Ribeiro. 2017. Decentralized quasi-newton methods. *IEEE Transactions on Signal Processing* 65, 10 (2017), 2613–2628.

[11] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. 2018. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems*. 689–699.

[12] Saeed Ghadimi and Guanghui Lan. 2013. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization* 23, 4 (2013), 2341–2368.

[13] Zhanhong Jiang, Aditya Balu, Chinmay Hegde, and Soumik Sarkar. 2017. Collaborative deep learning in fixed topology networks. In *Advances in Neural Information Processing Systems*. 5904–5914.

[14] Zhanhong Jiang, Kushal Mukherjee, and Soumik Sarkar. 2018. On Consensus-Disagreement Tradeoff in Distributed Optimization. In *2018 Annual American Control Conference (ACC)*. IEEE, 571–576.

[15] Rie Johnson and Tong Zhang. 2013. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*. 315–323.

[16] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian U Stich, and Martin Jaggi. 2019. Error feedback fixes SignSGD and other gradient compression schemes. *arXiv preprint arXiv:1901.09847* (2019).

[17] Anastasia Koloskova, Sebastian U Stich, and Martin Jaggi. 2019. Decentralized stochastic optimization and gossip algorithms with compressed communication. *arXiv preprint arXiv:1902.00340* (2019).

[18] Lihua Lei, Cheng Ju, Jianbo Chen, and Michael I Jordan. 2017. Non-convex finite-sum optimization via scsg methods. In *Advances in Neural Information Processing Systems*. 2348–2358.

[19] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. 2017. Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*. 5330–5340.

[20] Songtao Lu, Xinwei Zhang, Haoran Sun, and Mingyi Hong. 2019. GNSD: A gradient-tracking based nonconvex stochastic algorithm for decentralized optimization. In *2019 IEEE Data Science Workshop, DSW 2019*. Institute of Electrical and Electronics Engineers Inc., 315–321.

[21] Aryan Mokhtari and Alejandro Ribeiro. 2016. DSA: Decentralized double stochastic averaging gradient algorithm. *The Journal of Machine Learning Research* 17, 1 (2016), 2165–2199.

[22] Aryan Mokhtari, Wei Shi, Qing Ling, and Alejandro Ribeiro. 2016. A decentralized second-order method with exact linear convergence rate for consensus optimization. *IEEE Transactions on Signal and Information Processing over Networks* 2, 4 (2016), 507–522.

[23] Joao FC Mota, Joao MF Xavier, Pedro MQ Aguiar, and Markus Püschel. 2013. D-ADMM: A communication-efficient distributed algorithm for separable optimization. *IEEE Transactions on Signal Processing* 61, 10 (2013), 2718–2723.

[24] Angelia Nedić, Alex Olshevsky, and Michael G Rabbat. 2018. Network topology and communication-computation tradeoffs in decentralized optimization. *Proc. IEEE* 106, 5 (2018), 953–976.

[25] Angelia Nedic and Asuman Ozdaglar. 2009. Distributed subgradient methods for multi-agent optimization. *IEEE Trans. Automat. Control* 54, 1 (2009), 48.

[26] Angelia Nedich, Alex Olshevsky, and Wei Shi. 2016. A geometrically convergent method for distributed optimization over time-varying graphs. In *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 1023–1029.

[27] Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takác. 2017. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2613–2621.

[28] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*. 693–701.

[29] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Póczos, and Alex Smola. 2016. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*. 314–323.

[30] Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. 2015. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization* 25, 2 (2015), 944–966.

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

[32] Haoran Sun, Songtao Lu, and Mingyi Hong. 2019. Improving the sample and communication complexity for decentralized non-convex optimization: A joint gradient estimation and tracking approach. *ICML 2020* (2019).

[33] Quoc Tran-Dinh, Nhan H Pham, Dzung T Phan, and Lam M Nguyen. 2019. Hybrid Stochastic Gradient Descent Algorithms for Stochastic Nonconvex Optimization. *arXiv preprint arXiv:1905.05920* (2019).

[34] John Nikolas Tsitsiklis. 1984. *Problems in decentralized decision making and computation*. Technical Report. Massachusetts Inst of Tech Cambridge Lab for Information and Decision Systems.

[35] Weiran Wang, Jialei Wang, Mladen Kolar, and Nathan Srebro. 2018. Distributed stochastic multi-task learning with graph regularization. *arXiv preprint arXiv:1802.03830* (2018).

[36] Zhe Wang, Kaiyi Ji, Yi Zhou, Yingbin Liang, and Vahid Tarokh. 2019. SpiderBoost and Momentum: Faster Variance Reduction Algorithms. In *Advances in Neural Information Processing Systems*. 2403–2413.

[37] Zhe Wang, Yi Zhou, Yingbin Liang, and Guanghui Lan. 2018. Cubic regularization with momentum for nonconvex optimization. *arXiv preprint arXiv:1810.03763* (2018).

[38] Ran Xin, Usman A Khan, and Soummya Kar. 2019. Variance-reduced decentralized stochastic optimization with gradient tracking. *arXiv preprint arXiv:1909.11774* (2019).

[39] Kun Yuan, Qing Ling, and Wotao Yin. 2016. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization* 26, 3 (2016), 1835–1854.

[40] Kun Yuan, Bicheng Ying, Jiageng Liu, and Ali H Sayed. 2018. Variance-reduced stochastic learning by networked agents under random reshuffling. *IEEE Transactions on Signal Processing* 67, 2 (2018), 351–366.

[41] Jinshan Zeng and Wotao Yin. 2018. On nonconvex decentralized gradient descent. *IEEE Transactions on signal processing* 66, 11 (2018), 2834–2848.

[42] Xin Zhang, Jia Liu, and Zhengyuan Zhu. 2019. Distributed Linear Model Clustering over Networks: A Tree-Based Fused-Lasso ADMM Approach. *arXiv preprint arXiv:1905.11549* (2019).

[43] Xin Zhang, Jia Liu, Zhengyuan Zhu, and Elizabeth S. Bentley. 2020. GT-STORM: Taming Sample, Communication, and Memory Complexities in Decentralized

Non-Convex Learning. https://kevinliu-osu-ece.github.io/publications/GT-STORM_TR.pdf
[44] Ke Zhou and Stergios I Roumeliotis. 2011. Multirobot active target tracking with combinations of relative observations. *IEEE Transactions on Robotics* 27, 4 (2011), 678–695.
[45] Pan Zhou, Xiaotong Yuan, and Jiashi Feng. 2018. New insight into hybrid stochastic gradient descent: Beyond with-replacement sampling and convexity. In *Advances in Neural Information Processing Systems*. 1234–1243.
[46] Yi Zhou, Yingbin Liang, and Huishuai Zhang. 2018. Generalization error bounds with probabilistic guarantee for sgd in nonconvex optimization. *arXiv preprint arXiv:1802.06903* (2018).
[47] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. 2010. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*. 2595–2603.

A PROOF OF MAIN RESULTS

Due to space limitation, we provide key proof steps of the key lemmas and theorems in this appendix. We refer readers to [43] for the complete proofs.

A.1 Proof for Lemma 1

Recall that $\bar{\mathbf{v}}_t = \beta_t(\bar{\mathbf{v}}_{t-1}+\bar{\mathbf{w}}_t)+(1-\beta_t)\bar{\mathbf{u}}_t$. Then, with some algebraic derivations, we can show that: $\|\bar{\mathbf{v}}_t-\bar{\mathbf{g}}_t\|^2 = \|\beta_t(\bar{\mathbf{v}}_{t-1}+\bar{\mathbf{w}}_t)+(1-\beta_t)\bar{\mathbf{u}}_t-\bar{\mathbf{g}}_t\|^2+2\langle\bar{\mathbf{v}}_{t-1}-\bar{\mathbf{g}}_{t-1}, \beta_t(\bar{\mathbf{w}}_t-\bar{\mathbf{g}}_t+\bar{\mathbf{g}}_{t-1})+(1-\beta_t)(\bar{\mathbf{u}}_t-\bar{\mathbf{g}}_t)\rangle$. Note that $\mathbb{E}_{\zeta_t}[\bar{\mathbf{w}}_t] = \bar{\mathbf{g}}_t - \bar{\mathbf{g}}_{t-1}$ and $\mathbb{E}_{\zeta_t}[\bar{\mathbf{u}}_t] = \bar{\mathbf{g}}_t$. Taking expectation with respect to ζ_t, we have:

$$\mathbb{E}_{\zeta_t}[\|\bar{\mathbf{v}}_t-\bar{\mathbf{g}}_t\|^2] \le \beta_t^2\|\bar{\mathbf{v}}_{t-1}-\bar{\mathbf{g}}_{t-1}\|^2 + \frac{2\beta_t^2L^2}{m}\mathbb{E}_{\zeta_t}[\|\mathbf{x}_t-\mathbf{x}_{t-1}\|^2] + \frac{2(1-\beta_t)^2\sigma^2}{m}.$$

Then, taking the full expectation leads to the stated result in Lemma 1. This completes the proof.

A.2 Proof for Lemma 2

From the L-smoothness of f, we can show that: $f(\bar{\mathbf{x}}_{t+1}) \le f(\bar{\mathbf{x}}_t) - \frac{\eta_t}{2}\|\nabla f(\bar{\mathbf{x}}_t)\|^2 - (\frac{\eta_t}{2}-\frac{L\eta_t^2}{2})\|\bar{\mathbf{v}}_t\|^2 + \eta_t\|\bar{\mathbf{v}}_t-\bar{\mathbf{g}}_t\|^2 + \frac{L^2\eta_t}{m}\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2$, where $\bar{\mathbf{g}}_t = \frac{1}{m}\sum_{i=1}^m \mathbf{g}_{i,t}$ and $\mathbf{g}_{i,t} = \nabla f_i(\mathbf{x}_{i,t})$. Taking the full expectation on the above inequality yields: $\mathbb{E}[f(\bar{\mathbf{x}}_{t+1})]-\mathbb{E}[f(\bar{\mathbf{x}}_t)] \le -\frac{\eta_t}{2}\mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_t)\|^2]-(\frac{\eta_t}{2}-\frac{L\eta_t^2}{2})\mathbb{E}[\|\bar{\mathbf{v}}_t\|^2]+\eta_t\mathbb{E}[\|\bar{\mathbf{v}}_t-\bar{\mathbf{g}}_t\|^2]+\frac{L^2\eta_t}{m}\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2]$. This completes the proof.

A.3 Proof for Lemma 3

First for the iterate $\mathbf{x}_t$, we have the following contraction:

$$\|\tilde{\mathbf{W}}\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2 = \|\tilde{\mathbf{W}}(\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t)\|^2 \le \lambda^2\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2, \quad (21)$$

This is because $\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t$ is orthogonal to $\mathbf{1}$, which is the eigenvector corresponding to the largest eigenvalue of $\tilde{\mathbf{W}}$, and $\lambda = \max\{|\lambda_2|, |\lambda_m|\}$. Since $\bar{\mathbf{x}}_{t+1} = \bar{\mathbf{x}}_t - \eta_t\bar{\mathbf{v}}_t$, we can then show that:

$$\|\mathbf{x}_{t+1}-\mathbf{1}\otimes\bar{\mathbf{x}}_{t+1}\|^2 = \|\tilde{\mathbf{W}}\mathbf{x}_t-\eta_t\mathbf{v}_t-\mathbf{1}\otimes(\bar{\mathbf{x}}_t-\eta_t\bar{\mathbf{v}}_t)\|^2$$
$$\le (1+c_1)\lambda^2\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2 + (1+\frac{1}{c_1})\eta_t^2\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2. \quad (22)$$

Similarly to (22), we can also show that:

$$\|\mathbf{v}_{t+1}-\mathbf{1}\otimes\bar{\mathbf{v}}_{t+1}\|^2 \le (1+c_1)\beta_{t+1}^2\lambda^2\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2 + 2(1+\frac{1}{c_1})\big(\beta_{t+1}^2\|\mathbf{w}_{t+1}\|^2+(1-\beta_{t+1})^2\|\mathbf{u}_{t+1}\|^2\big).$$

Lastly, according to the update in (8), it follows that

$$\|\mathbf{x}_{t+1}-\mathbf{x}_t\|^2 = \|\tilde{\mathbf{W}}\mathbf{x}_t-\eta_t\mathbf{v}_t-\mathbf{x}_t\|^2 \overset{(a)}{\le} 8\|(\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t)\|^2 + 4\eta_t^2\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2 + 4\eta_t^2m\|\bar{\mathbf{v}}_t\|^2, \quad (23)$$

where (a) is due to $\|\tilde{\mathbf{W}}-\mathbf{I}\| \le 2$ and the proof is complete.

A.4 Proof for Lemma 4

First, with $\eta_t = \tau/(\omega+t)^{1/3}$, we can show that:

$$\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}} = \frac{1}{\tau}\big((\omega+t)^{\frac{1}{3}}-(\omega+t-1)^{\frac{1}{3}}\big) \le \frac{2^{\frac{2}{3}}}{3\tau^3}\frac{\tau^2}{(\omega+t)^{\frac{2}{3}}} \le \frac{2}{3\tau^3}\eta_t.$$

Next, we prove the following three contraction relationships:
i) For $\mathbb{E}[\|\bar{\mathbf{v}}_t-\bar{\mathbf{g}}_t\|^2]$, we have:

$$\frac{1}{\eta_t}\mathbb{E}[\|\bar{\mathbf{v}}_{t+1}-\bar{\mathbf{g}}_{t+1}\|^2]-\frac{1}{\eta_{t-1}}\mathbb{E}[\|\bar{\mathbf{v}}_t-\bar{\mathbf{g}}_t\|^2]$$
$$\le -32\eta_tL^2\mathbb{E}[\|\bar{\mathbf{v}}_t-\bar{\mathbf{g}}_t\|^2] + \frac{2\beta_{t+1}^2L^2}{m\eta_t}\mathbb{E}[\|\mathbf{x}_{t+1}-\mathbf{x}_t\|^2] + \frac{2\rho^2\sigma^2\eta_t^3}{m}. \quad (24)$$

ii) For $\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2]$, we have:

$$\frac{1}{\eta_t}\mathbb{E}[\|\mathbf{x}_{t+1}-\mathbf{1}\otimes\bar{\mathbf{x}}_{t+1}\|^2]-\frac{1}{\eta_{t-1}}\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2] \le \Big(\frac{(1+c_1)\lambda^2-1}{\eta_t} + \frac{2}{3\tau^3}\eta_t\Big)[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2] + (1+\frac{1}{c_1})\eta_t\mathbb{E}[\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2]. \quad (25)$$

iii) For $\mathbb{E}[\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2]$, with Lemma 3, we have:

$$\mathbb{E}[\|\mathbf{v}_{t+1}-\mathbf{1}\otimes\bar{\mathbf{v}}_{t+1}\|^2]-\mathbb{E}[\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2]$$
$$\le \big((1+c_1)\beta_{t+1}^2\lambda^2-1\big)\mathbb{E}[\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2] + 2(1+\frac{1}{c_1})\big(\beta_{t+1}^2L^2\mathbb{E}[\|\mathbf{x}_{t+1}-\mathbf{x}_t\|^2] + mG^2\rho^2\eta_t^4\big) \quad (26)$$

Using the result from Lemma 2, we have

$$\mathbb{E}[f(\bar{\mathbf{x}}_{t+1})]-\mathbb{E}[f(\bar{\mathbf{x}}_t)] \le -\frac{\eta_t}{2}\mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_t)\|^2]-(\frac{\eta_t}{2}-\frac{L\eta_t^2}{2})\mathbb{E}[\|\bar{\mathbf{v}}_t\|^2] + \eta_t\mathbb{E}[\|\bar{\mathbf{v}}_t-\bar{\mathbf{g}}_t\|^2] + \frac{L^2\eta_t}{m}\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2]. \quad (27)$$

Then, with the result in i), we have:

$$\mathbb{E}[f(\bar{\mathbf{x}}_{t+1})+\frac{1}{32L^2\eta_t}\|\bar{\mathbf{g}}_{t+1}-\bar{\mathbf{v}}_{t+1}\|^2]-\mathbb{E}[f(\bar{\mathbf{x}}_t)+\frac{1}{32L^2\eta_{t-1}}\|\bar{\mathbf{g}}_t-\bar{\mathbf{v}}_t\|^2]$$
$$\le -\frac{\eta_t}{2}\mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_t)\|^2]-(\frac{\eta_t}{2}-\frac{L\eta_t^2}{2})\mathbb{E}[\|\bar{\mathbf{v}}_t\|^2] + \frac{\beta_{t+1}^2}{16m\eta_t}\mathbb{E}[\|\mathbf{x}_{t+1}-\mathbf{x}_t\|^2] + \frac{L^2\eta_t}{m}\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2] + \frac{\rho^2\sigma^2\eta_t^3}{16mL^2}. \quad (28)$$

Next, with the results in ii) and iii), we can show that

$$\mathbb{E}[\frac{1}{\eta_t}\|\mathbf{x}_{t+1}-\mathbf{1}\otimes\bar{\mathbf{x}}_{t+1}\|^2+\|\mathbf{v}_{t+1}-\mathbf{1}\otimes\bar{\mathbf{v}}_{t+1}\|^2]$$
$$-\mathbb{E}[\frac{1}{\eta_{t-1}}\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2+\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2]$$
$$\le -\Big(\frac{1-(1+c_1)\lambda^2}{\eta_t}-\frac{2\eta_t}{3\tau^3}\Big)\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2] - \Big(1-(1+c_1)\beta_{t+1}^2\lambda^2-(1+\frac{1}{c_1})\eta_t\Big)\mathbb{E}[\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2] + 2(1+\frac{1}{c_1})\beta_{t+1}^2L^2\mathbb{E}[\|\mathbf{x}_{t+1}-\mathbf{x}_t\|^2] + 2(1+\frac{1}{c_1})mG^2\rho^2\eta_t^4. \quad (29)$$

Thus, for the defined potential function in Lemma 4, its differential can be bounded as:

$$\begin{aligned}
H_{t+1}-H_t &\overset{(a)}{\leq} -\frac{\eta_t}{2}\mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_t)\|^2]+\frac{\rho^2\sigma^2\eta_t^3}{16mL^2}+2(1+\frac{1}{c_1})c_0G^2\rho^2\eta_t^4\\
&-\Big(1-(1+c_1)\lambda^2-\frac{1}{2c_0}-16(1+\frac{1}{c_1})L^2\eta_t-\frac{2\eta_t^2}{3\tau^3}\\
&-\frac{L^2\eta_t^2}{c_0}\Big)\times\frac{c_0}{m\eta_t}\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2]-\Big(1-(1+c_1)\lambda^2-(1+\frac{1}{c_1})\eta_t-\frac{\eta_t}{4c_0}\\
&-8(1+\frac{1}{c_1})L^2\eta_t^2\Big)\times\frac{c_0}{m}\mathbb{E}[\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2]\\
&-\Big(1-2L\eta_t-32(1+\frac{1}{c_1})c_0L^2\eta_t\Big)\times\frac{\eta_t}{4}\mathbb{E}[\|\bar{\mathbf{v}}_t\|^2]. \qquad (30)
\end{aligned}$$

where (a) follows from plugging the result for $\|\mathbf{x}_{t+1}-\mathbf{x}_t\|^2$ from Lemma 3 and $\beta_{t+1}<1$.

A.5 Proof for Theorem 1

From Lemma 4, we have:

$$\begin{aligned}
\sum_{t=0}^{T}\frac{\eta_t}{2}\mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_t)\|^2] &\leq H_0-H_{T+1}+\sum_{t=0}^{T}\frac{\rho^2\sigma^2\eta_t^3}{16mL^2}\\
&+\sum_{t=0}^{T}2(1+\frac{1}{c_1})c_0G^2\rho^2\eta_t^4-\sum_{t=0}^{T}\frac{c_0C_1}{m\eta_t}\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2]\\
&-\sum_{t=0}^{T}\frac{c_0C_2}{m}\mathbb{E}[\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2]-\sum_{t=0}^{T}\frac{C_3\eta_t}{4}\mathbb{E}[\|\bar{\mathbf{v}}_t\|^2]. \qquad (31)
\end{aligned}$$

Noting from $\eta_t=\tau/(\omega+t)^{1/3}$ and $\tau\geq 2$, we have:

$$\sum_{t=0}^{T}\eta_t^3=\sum_{t=0}^{T}\frac{\tau}{\omega+t}\leq\int_{-1}^{T-1}\frac{\tau}{\omega+t}dt\leq\tau\ln(\omega+T-1), \qquad (32)$$

$$\sum_{t=0}^{T}\eta_t^4=\sum_{t=0}^{T}\Big(\frac{\tau}{\omega+t}\Big)^{\frac{4}{3}}\leq\int_{-1}^{T-1}\Big(\frac{\tau}{\omega+t}\Big)^{\frac{4}{3}}dt\leq\frac{3\tau^{4/3}}{(\omega-1)^{1/3}}. \qquad (33)$$

Since η_t is decreasing, we have that:

$$\begin{aligned}
&\frac{\eta_T}{2}\sum_{t=0}^{T}\mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_t)\|^2]+\frac{1}{m}\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2]\\
&\leq H_0-H_{T+1}+\frac{\tau\rho^2\sigma^2\ln(\omega+T-1)}{16mL^2}\\
&+6(1+\frac{1}{c_1})c_0G^2\rho^2\frac{\tau^{4/3}}{(\omega-1)^{1/3}}-\sum_{t=0}^{T}\frac{2c_0C_1-\eta_t^2}{2m\eta_t}\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2]\\
&-\sum_{t=0}^{T}\frac{c_0C_2}{m}\mathbb{E}[\|\mathbf{v}_t-\mathbf{1}\otimes\bar{\mathbf{v}}_t\|^2]-\sum_{t=0}^{T}\frac{C_3\eta_t}{4}\mathbb{E}[\|\bar{\mathbf{v}}_t\|^2]. \qquad (34)
\end{aligned}$$

Now, we show that by properly choosing η_t, c_1, and c_0, $C_1-\eta^2/2c_0$, C_2 and C_3 can be made non-negative. Note that:

$$C_1=1-(1+c_1)\lambda^2-\frac{1}{2c_0}-16(1+\frac{1}{c_1})L^2\eta_t-\Big(\frac{2}{3\tau^3}+\frac{L^2}{c_0}\Big)\eta_t^2, \qquad (35)$$

$$C_2=1-(1+c_1)\lambda^2-(1+\frac{1}{c_1})\eta_t-\frac{\eta_t}{4c_0}-8(1+\frac{1}{c_1})L^2\eta_t^2, \qquad (36)$$

$$C_3=1-2L\eta_t-32(1+\frac{1}{c_1})c_0L^2\eta_t. \qquad (37)$$

In order to have $C_3\geq 0$, we have:

$$\eta_t\leq 1/\Big(2L+32(1+\frac{1}{c_1})c_0L^2\Big):=k_1. \qquad (38)$$

With (38), it follows that $C_2\geq 1-(1+c_1)\lambda^2-(1+\frac{1}{c_1})\eta_t-\frac{\eta_t}{2c_0}$. Thus, $C_2\geq 0$ if we set $\eta_t\leq\Big(1-(1+c_1)\lambda^2\Big)/\Big(1+\frac{1}{c_1}+\frac{1}{2c_0}\Big):=k_2$. For $C_1-\eta^2/2c_0$, it follows from (38) that $C_1-\frac{\eta^2}{2c_0}\geq 1-(1+c_1)\lambda^2-\frac{1}{c_0}-\Big(\frac{2}{3\tau^3}+\frac{2L^2+1}{2c_0}\Big)\eta_t^2$. By choosing $\eta_t\leq\sqrt{(1-(1+c_1)\lambda^2-\frac{1}{c_0})/(\frac{2}{3\tau^3}+\frac{2L^2+1}{2c_0})}:=k_3$, and $0<1-(1+c_1)\lambda^2-\frac{3}{4c_0}$, we have $C_1-\eta^2/2c_0\geq 0$. To summarize, we need to set $\eta_t\leq\min\{k_1,k_2,k_3\}$. Since η_t is decreasing and $\eta_0=\tau/\omega^{1/3}$, it implies that $\omega\geq(\tau/\min\{k_1,k_2,k_3\})^3$.

With the above parameter setting, we have:

$$\begin{aligned}
&\frac{\eta_T}{2}\sum_{t=0}^{T}\mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_t)\|^2]+\frac{1}{m}\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2]\leq H_0-H_{T+1}\\
&+\frac{\tau\rho^2\sigma^2\ln(\omega+T-1)}{16mL^2}+6(1+\frac{1}{c_1})c_0G^2\rho^2\frac{\tau^{4/3}}{(\omega-1)^{1/3}}. \qquad (39)
\end{aligned}$$

Multiplying both sides of the inequality by $2/\eta_T(T+1)$, we have:

$$\begin{aligned}
&\frac{1}{T+1}\sum_{t=0}^{T}\mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_t)\|^2]+\frac{1}{m}\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2]\\
&\leq\frac{2(H_0-H_{T+1})}{\eta_T(T+1)}+\frac{\tau\rho^2\sigma^2\ln(\omega+T-1)}{8mL^2\eta_T(T+1)}+\frac{12(1+\frac{1}{c_1})c_0\tau^{4/3}G^2\rho^2}{(\omega-1)^{1/3}\eta_T(T+1)}. \qquad (40)
\end{aligned}$$

Next, noting that $H_0\leq\mathbb{E}[f(\bar{\mathbf{x}}_0)+\frac{\sigma^2}{32mL^2\eta_{-1}}+\frac{c_0}{m}\|\mathbf{v}_0-\mathbf{1}\otimes\bar{\mathbf{v}}_0\|^2]$, we have $H_{T+1}\geq\mathbb{E}[f(\bar{\mathbf{x}}_{T+1})+\frac{c_0}{m\eta_T}\|\mathbf{x}_{T+1}-\mathbf{1}\bar{\mathbf{x}}_{T+1}\|^2+\frac{c_0}{m}\|\mathbf{v}_{T+1}-\mathbf{1}\otimes\bar{\mathbf{v}}_{T+1}\|^2]\geq f(\bar{\mathbf{x}}^*)$. , it follows that

$$\begin{aligned}
&\frac{1}{T+1}\sum_{t=0}^{T}\mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_t)\|^2]+\frac{1}{m}\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2]\leq\frac{2(f(\bar{\mathbf{x}}_0)-f(\bar{\mathbf{x}}^*))}{\eta_T(T+1)}\\
&+\frac{2c_0\mathbb{E}[\|\mathbf{v}_0-\mathbf{1}\otimes\bar{\mathbf{v}}_0\|^2]}{m\eta_T(T+1)}+\frac{(\omega-1)\sigma^2}{16mL^2\tau\eta_T(T+1)}+\frac{\tau\rho^2\sigma^2\ln(\omega+T-1)}{8mL^2\eta_T(T+1)}\\
&+12(1+\frac{1}{c_1})c_0\frac{\tau^{4/3}G^2\rho^2}{(\omega-1)^{1/3}\eta_T(T+1)}.
\end{aligned}$$

Since $\eta_T=\tau/(\omega+T)^{1/3}$, we have:

$$\begin{aligned}
&\min_{t\in[T]}\mathbb{E}[\|\nabla f(\bar{\mathbf{x}}_t)\|^2]+\frac{1}{m}\mathbb{E}[\|\mathbf{x}_t-\mathbf{1}\otimes\bar{\mathbf{x}}_t\|^2]\\
&\leq\frac{2(f(\bar{\mathbf{x}}_0)-f(\bar{\mathbf{x}}^*))}{\tau(T+1)^{2/3}}+\frac{2c_0\mathbb{E}[\|\mathbf{v}_0-\mathbf{1}\otimes\bar{\mathbf{v}}_0\|^2]}{m\tau(T+1)^{2/3}}\\
&+\frac{(\omega-1)\sigma^2}{16mL^2\tau^2(T+1)^{2/3}}+\frac{\rho^2\sigma^2\ln(\omega+T-1)}{8mL^2(T+1)^{2/3}}\\
&+\frac{12(1+\frac{1}{c_1})c_0\tau^{1/3}G^2\rho^2}{(\omega-1)^{1/3}(T+1)^{2/3}}+O\Big(\frac{c_3\omega}{\tau T^{5/3}}\Big).
\end{aligned}$$

where the Big-O follows from $(\omega+T)^{1/3}-(T+1)^{1/3}\leq(\omega-1)(T+1)^{-2/3}/3$ and $c_3=\max\{1,(\omega-1)/(m\tau^2),\tau^{4/3}/\omega^{1/3},\tau\ln(\omega+T-1)/m\}$.