Variance Reduction Methods Do Not Need to Compute Full Gradients: Improved Efficiency Through Shuffling

Daniil Medyakov[1,2,3] **Gleb Molodtsov**[1,2,3] **Savelii Chezhegov**[1,2,3,4]
Alexey Rebrikov[1,2] **Aleksandr Beznosikov**[1,2,3,5]
[1] Moscow Independent Research Institute of Artificial Intelligence
[2] Basic Research of Artificial Intelligence Laboratory (BRAIn Lab)
[3] Federated Learning Problems Laboratory
[4] Sber AI Lab
[5] Innopolis University

Abstract

Stochastic optimization algorithms are widely used for machine learning with large-scale data. However, their convergence often suffers from non-vanishing variance. Variance Reduction (VR) methods, such as SVRG and SARAH, address this issue but introduce a bottleneck by requiring periodic full gradient computations. In this paper, we explore popular VR techniques and propose an approach that eliminates the necessity for expensive full gradient calculations. To avoid these computations and make our approach memory-efficient, we employ two key techniques: the shuffling heuristic and the concept of SAG/SAGA methods. For non-convex objectives, our convergence rates match those of standard shuffling methods, while under strong convexity, they demonstrate an improvement. We empirically validate the efficiency of our approach and demonstrate its scalability on large-scale machine learning tasks including image classification problem on CIFAR-10 and CIFAR-100 datasets.

1 INTRODUCTION

The pursuit of enhanced performance in machine learning has resulted in increasingly large training datasets. The standard approach for scalable training in this context is to formulate the task as a finite-sum minimization problem:

$$\min_{x\in\mathbb{R}^d}\Big[f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)\Big], \tag{1}$$

where $f_i : \mathbb{R}^d \to \mathbb{R}$ and the number of functions n is large. When training machine learning models, n represents the size of the training set, and $f_i(x)$ denotes the loss of the model on the i-th data point, where $x \in \mathbb{R}^d$ is the vector of model parameters.

Stochastic methods are widely used to solve this problem as they do not calculate full gradients at each iteration. This necessity is prohibitively expensive in real-world problems due to the large n values. Most well-known stochastic methods for solving the problem are SGD [Robbins and Monro, 1951, Moulines and Bach, 2011] and its various modifications [Ghadimi and Lan, 2013, Ghadimi et al., 2016, Lan, 2020]. At iteration t, the algorithm selects a single index i_t from the set $\{1, \dots, n\}$ and performs the following step with the stepsize γ.

$$x^{t+1} = x^t - \gamma \nabla f_{i_t}(x^t).$$

Variance reduction technique. Despite the simplicity of the classic SGD method and the extensive study of its properties, it suffers from one significant drawback: the variance of its stochastic gradient estimators remains high throughout the learning process. Consequently, SGD with a constant learning rate achieves linear convergence only to a neighborhood of the optimal solution, whose size is proportional to the stepsize and the variance [Gower et al., 2020]. The variance reduction (VR) technique [Johnson and Zhang, 2013] was proposed to address this issue. Some of the most popular methods based on this technique are SVRG [Johnson and Zhang, 2013], SARAH [Nguyen

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

et al., 2017, Hu et al., 2019], SAG [Roux et al., 2012], SAGA [Defazio et al., 2014a], FINITO [Defazio et al., 2014b], SPIDER [Fang et al., 2018]. In this paper, we analyze two of these algorithms: SVRG and SARAH.

We first outline the SVRG method, formalized as

$$\begin{aligned} v^t &= \nabla f_{i_t}(x^t) - \nabla f_{i_t}(\omega^t) + \nabla f(\omega^t), \\ x^{t+1} &= x^t - \gamma v^t, \end{aligned} \tag{2}$$

where index i_t is selected at the t-th iteration. Regarding the reference point ω^t, it should be updated periodically, either after a fixed number of iterations (e.g., once per epoch) or probabilistically (as in loopless versions – see [Kovalev et al., 2020]). The goal of update mechanisms as (2) is to move beyond the limitations of naive gradient estimators. It employs an iterative process to construct and apply a gradient estimator with progressively diminished variance. This approach allows for the safe use of larger learning rates, thereby accelerating the training process. Now, we present another method, SARAH, which employs the recursive estimator update:

$$\begin{aligned} v^t &= \nabla f_{i_t}(x^t) - \nabla f_{i_t}(x^{t-1}) + v^{t-1}, \\ x^{t+1} &= x^t - \gamma v^t. \end{aligned} \tag{3}$$

To converge to the optimal solution x^*, this methods requires periodical updates of v^t using the full gradient [Nguyen et al., 2017]. As in SVRG, this process restarts after either a fixed number of iterations or probabilistically [Li et al., 2020]. The practical variant, SARAH+ [Nguyen et al., 2017], outperforms SVRG by automating the full-gradient update schedule through a heuristic based on the ratio $|v^t|/|v^0|$.

Shuffling heuristic. The choice of sample indices (i_t) at each iteration critically impacts the convergence and stability of stochastic optimization, yet is often overlooked. Below, we describe the heuristic used for this selection in our algorithms. In classic stochastic methods, the index i_t is selected randomly and independently at each iteration. In turn, we take a more practical approach, utilizing the shuffling heuristic [Bottou, 2009, Mishchenko et al., 2020, Safran and Shamir, 2020]. We first permute the sequence of indices $1, \ldots, n$. Then, we select an index based on its position in the permutation during an epoch. There are several popular data shuffling methods. One of them is Random Reshuffle (RR), which involves shuffling the data before each epoch. Another approach is Shuffle Once (SO), where the data is shuffled only once at the beginning of training. There is also Cyclic Permutation, which accesses the data in a fixed, cyclic order without any randomness. In our study, we do not explore the differences between these approaches. Instead, we highlight one important common property among them: during one epoch, we calculate the stochastic gradient for each data sample exactly once.

Let us formalize this setting. At each epoch s, we have a set of indices $\{\pi_s^0, \pi_s^1, \ldots, \pi_s^{n-1}\}$ that is a random permutation of the set $\{0, 1, \ldots, n-1\}$. Then, for example, the SVRG update (2) at the t-th iteration of this epoch transforms into

$$v^t = \nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(\omega^t) + \nabla f(\omega^t).$$

Nevertheless, the analysis of shuffling methods has some specific details. The key difference between shuffling and independent choice is that shuffling methods lack one essential property: the unbiasedness of stochastic gradients derived from the i.i.d. sampling.

$$\mathbb{E}_{\pi_s^t}\left[\nabla f_{\pi_s^t}(x_s^t)\right] \neq \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x_s^t) = \nabla f(x_s^t).$$

This restriction leads to a more complex analysis and non-standard proof techniques.

2 BRIEF LITERATURE REVIEW

No full gradient methods. SVRG (2) and SARAH (3) are now the standard choices for solving finite-sum problems. Nevertheless, they necessitate the full gradient computation of the target function. There is a considerable interest in VR methods that avoid calculating full gradients. While SAG [Roux et al., 2012] and SAGA [Defazio et al., 2014a] address this issue, they require additional memory usage, with a complexity of $\mathcal{O}(nd)$. There also were some attempts to eliminate the computation of the full gradient in SARAH. The first approach was proposed in [Nguyen et al., 2021]. The authors introduced an inexact SARAH algorithm, where they replaced the computation of the full gradient by a mini-batch gradient estimate $\frac{1}{|S|}\sum_{i\in S} f_i(x), S \subset \{1, \ldots, n\}$. To converge to the solution with ε-accuracy, this algorithm requires a batch size $|S| \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$ and a stepsize $\gamma \sim \mathcal{O}\left(\frac{\varepsilon}{L}\right)$. Another approach employs the recursive SARAH update of the full gradient estimator. In a set of works, the authors proposed a hybrid scheme without restarts:

$$v^t = \beta_t \nabla f_{i_t}(x^t) + (1-\beta_t)(\nabla f_{i_t}(x^t) - \nabla f_{i_t}(x^{t-1}) + v^{t-1}).$$

In the work [Liu et al., 2020], this scheme was used with a constant parameter $\beta_t = \beta$. The STORM method [Cutkosky and Orabona, 2019] considers β_t decreasing to zero and ZEROSARAH [Li et al., 2021] combines it with SAG/SAGA.

We now examine methods that eliminate full-gradient computations and leverage shuffling strategies, as this combination aligns with the core objective of our work.

Table 1: Comparison of the Convergence Results.

Algorithm	No Full Grad.?	Memory	Non-Convex	Strongly Convex
SAGA [Park and Ryu, 2020]	✓	$\mathcal{O}(nd)$	\	$\mathcal{O}\left(n\frac{L^2}{\mu^2}\log\left(\frac{1}{\varepsilon}\right)\right)$
IAG [Gurbuzbalaban et al., 2017]	✓	$\mathcal{O}(nd)$	\	$\mathcal{O}\left(n^2\frac{L^2}{\mu^2}\log\left(\frac{1}{\varepsilon}\right)\right)$
PIAG [Vanli et al., 2016]	✓	$\mathcal{O}(nd)$	\	$\mathcal{O}\left(n\frac{L}{\mu}\log\left(\frac{1}{\varepsilon}\right)\right)$
DIAG [Mokhtari et al., 2018]	✓	$\mathcal{O}(nd)$	\	$\mathcal{O}\left(n\frac{L}{\mu}\log\left(\frac{1}{\varepsilon}\right)\right)$
Prox-DFinito [Huang et al., 2021]	✓	$\mathcal{O}(nd)$	\	$\mathcal{O}\left(n\frac{L}{\mu}\log\left(\frac{1}{\varepsilon}\right)\right)$
AVRG [Ying et al., 2020]	✓	$\mathcal{O}(d)$	\	$\mathcal{O}\left(n\frac{L^2}{\mu^2}\log\left(\frac{1}{\varepsilon}\right)\right)$
SVRG [Sun et al., 2019]	✗	$\mathcal{O}(d)$	\	$\mathcal{O}\left(n^3\frac{L^2}{\mu^2}\log\left(\frac{1}{\varepsilon}\right)\right)$
SVRG [Malinovsky et al., 2023]	✗	$\mathcal{O}(d)$	$\mathcal{O}\left(\frac{nL}{\varepsilon^2}\right)$	$\mathcal{O}\left(n\frac{L^{3/2}}{\mu^{3/2}}\log\left(\frac{1}{\varepsilon}\right)\right)^{(1)}$
SARAH [Beznosikov and Takáč, 2023]	✓	$\mathcal{O}(d)$	\	$\mathcal{O}\left(n^2\frac{L}{\mu}\log\left(\frac{1}{\varepsilon}\right)\right)$
SVRG (Algorithm 1 in this paper)	✓	$\mathcal{O}(d)$	$\mathcal{O}\left(\frac{nL}{\varepsilon^2}\right)$	$\mathcal{O}\left(n\frac{L}{\mu}\log\left(\frac{1}{\varepsilon}\right)\right)$
SARAH (Algorithm 1 in this paper)	✓	$\mathcal{O}(d)$	$\mathcal{O}\left(\frac{nL}{\varepsilon^2}\right)$	$\mathcal{O}\left(n\frac{L}{\mu}\log\left(\frac{1}{\varepsilon}\right)\right)$

Columns: No Full Grad.?= whether the method computes full gradients, Memory = amount of additional memory.
Notation: μ = constant of strong convexity, L = smoothness constant, n = size of the dataset, d = dimension of the problem, ε = accuracy of the solution.
(1): In this work, there are also improved results that hold in the big data regime: $n \gg \mathcal{O}\left(\frac{L}{\mu}\right)$, but it is out of the scope of this work.

Several studies have pursued this direction, including IAG [Gurbuzbalaban et al., 2017], PIAG [Vanli et al., 2016], DIAG [Mokhtari et al., 2018], SAGA [Park and Ryu, 2020, Ying et al., 2020], and Prox-DFinito [Huang et al., 2021], which naively store stochastic gradients. Among them, PIAG, DIAG, Prox-DFinito provide the best oracle complexity for methods with the shuffling heuristic as of now [1]. Nevertheless, they still require $\mathcal{O}(nd)$ of extra memory, which is not optimal for large-scale problems. Methods AVRG [Ying et al., 2020] and the modification of SARAH [Beznosikov and Takáč, 2023] eliminate this issue. However, they deteriorate the oracle complexity. Thus, there are still no VR methods that obviate the need for computing full gradients while being optimal in terms of additional memory. We, in turn, address this gap.

Shuffling methods. Given that our approach relies on a shuffling heuristic, we provide a review of existing shuffling techniques alongside a complexity comparison with our method. A key property of shuffling in this context is that it guarantees each component function's gradient is computed precisely once per epoch. It has been demonstrated that Random Reshuffle converges more rapidly than SGD on multiple practical tasks [Bottou, 2009, Recht and Ré, 2013].

Nevertheless, the theoretical estimates of shuffling methods remained significantly worse than those of SGD-like methods [Rakhlin et al., 2012, Drori and Shamir, 2020, Nguyen et al., 2019]. A breakthrough was the work [Mishchenko et al., 2020] which presented new proof techniques and approaches to interpret shuffling. In particular, the results for strongly convex problems coincided with those of SGD which utilized an independent choice of indices [Moulines and Bach, 2011, Stich, 2019]. However, in the non-convex case, the results remained inferior to those of classic SGD [Ghadimi and Lan, 2013]. Furthermore, a major issue was the requirement for a large number of epochs to obtain convergence estimates in the non-convex case. Since modern neural networks are trained on a relatively small number of epochs, this requirement is unnatural. The solution to this problem was presented in the work [Koloskova et al., 2024]. The authors based their analysis on convergence over a shorter period, termed the correlation period, rather than over entire epoch. This technique helped to improve rate.

In these works, shuffling was analyzed primarily for vanilla SGD. However, SGD is suboptimal for finite-sum problems due to its variance [Zhang et al., 2020, Allen-Zhu, 2018]. Shuffling was later extended to variational inequalities [Beznosikov et al., 2023], specifically,

[1] In [Malinovsky et al., 2023], the authors derived estimates that outperform those in the discussed works. However, these results were obtained under the big-data regime: $n \gg L/\mu$, which is a specific assumption. For this reason, we do not consider them in the current work.

to EXTRAGRADIENT [Medyakov et al., 2024], maintaining classic convergence rates. Consequently, researchers focused on the shuffling heuristic in conjunction with variance reduction methods, achieving linear convergence for these methods. In finite-sum minimization, we pay special attention to the works [Malinovsky et al., 2023, Sun et al., 2019]. However, the theoretical estimates presented in these papers remain significantly below those of methods with an independent choice of indices [Allen-Zhu and Hazan, 2016]. This paper improves the estimates for the strongly convex objective.

The convergence results from the aforementioned works are presented in Table 1. Our results are compared with studies employing a shuffling setting that achieves linear convergence.

3 CONTRIBUTIONS

Our main contributions are summarized as follows.

• **Full gradient approximation.** We explore an approach whose key feature is approximating the full gradient of the objective function using a shuffling heuristic. Moreover, this method *does not require* additional $\mathcal{O}(nd)$ memory. Instead, we iteratively construct the approximation during an epoch.
• **No need to compute full gradient in variance reduction algorithms.** We construct an analysis for the full gradient approximation and enable its application to two variance reduction methods:

(a) Classic SVRG [Johnson and Zhang, 2013],

(b) SARAH in the closest form to one in [Beznosikov and Takáč, 2023]. We transform their algorithm to improve convergence rates.

• **Convergence results.** We obtain convergence results under various assumptions on the objective function. We consider both the non-convex case, which is of greatest interest in contemporary machine learning problems, and the strongly convex case. To the best of our knowledge, our convergence guarantees are superior to existing variance reduction methods that do not compute the full gradient. Furthermore, we enhance upper bounds of shuffling methods for the strongly convex case.
• **Experiments.** We empirically validate our methods on CIFAR-10 and CIFAR-100 using the ResNet-18 model, demonstrating that our methods achieve faster and more stable convergence than prior baselines.

4 ASSUMPTIONS

We present a list of assumptions below.

Assumption 1. *Each function f_i is L-smooth, i.e., it satisfies $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|$ for any $x, y \in \mathbb{R}^d$.*

Assumption 2.

(a) Strong Convexity: Each function f_i is μ-strongly convex, i.e., for any $x, y \in \mathbb{R}^d$, it satisfies

$$f_i(y) \geqslant f_i(x) + \langle \nabla f_i(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2.$$

(b) Non-Convexity: The function f has a (may be not unique) finite minimum, i.e. $f^* = \inf\limits_{x\in\mathbb{R}^d} f(x) > -\infty$.

5 ALGORITHMS AND CONVERGENCE ANALYSIS

5.1 Full gradient approximation

In this section, we introduce an approach based on the shuffling heuristic and SAG/SAGA ideas. It approximates the full gradient while optimizing memory usage by avoiding the storage of past gradient values. The SAG algorithm is one of the early methods designed to improve the convergence speed of stochastic gradient methods by reducing the variance of gradient updates. In SAG, the update step can be written as

$$x^{t+1} = x^t - \tfrac{\gamma}{n}\Big(\nabla f_{i_t}(x^t) - \nabla f_{i_t}(\phi^t_{i_t}) + \sum_{j=1}^{n} \nabla f_j(\phi^t_j)\Big), \quad (4)$$

where ϕ^t_j represents the past iteration at which the gradients for the j-th function are considered. The core idea is to store old gradients for each function (essentially storing gradients $\nabla f_j(\phi_j)$ rather than points ϕ_j), and update one component of this sum with a newly computed gradient at each iteration. As i_t are sampled randomly, it is unclear when the gradient for a specific index was last computed. However, in the case of shuffling, we know that during an epoch, the gradient for each ∇f_j is computed. Thus, at the beginning of an epoch, we reliably approximate the gradient in the same way as in SAG:

$$v_{s+1} = \tfrac{1}{n}\sum_{t=1}^{n} \nabla f_{\pi^t_s}(x^t_s), \quad (5)$$

where π^t_s is the sequence of data points after shuffling at the beginning of epoch s. This computation is performed without additional memory, using a simple moving average during the previous epoch:

$$\widetilde{v}^{t+1}_s = \tfrac{t}{t+1}\widetilde{v}^t_s + \tfrac{1}{t+1}\nabla f_{\pi^t_s}(x^t_s); \quad v_{s+1} = \widetilde{v}^n_s. \quad (6)$$

It is straightforward to prove (6) matches (5), and we demonstrate this in the proof of Lemma 2.

5.2 SVRG without full gradients

As previously mentioned, variance reduction methods face a significant obstacle: they require the computation of the full gradient once per epoch or necessitate additional memory of a larger size. To address this limitation, in this section, we propose a novel algorithm based on the classical SVRG that eliminates the need for full gradient computation and optimizes memory usage by avoiding the storage of past gradient values.

Algorithm 1 No Full Grad SVRG

1: **Input:** Initial points $x_0^0 \in \mathbb{R}^d, \omega_0 = x_0^0$; Initial gradients $\widetilde{v}_0^0 = 0^d, v_0 = 0^d$
2: **Parameter:** Stepsize $\gamma > 0$
3: **for** epochs $s = 0, 1, 2, \ldots, S$ **do**
4: Sample a permutation $\pi_s^0, \ldots, \pi_s^{n-1}$ of $\overline{0, n-1}$ *// Sampling depends on shuffling heuristic*
5: **for** $t = 0, 1, 2, \ldots, n-1$ **do**
6: $\widetilde{v}_s^{t+1} = \frac{t}{t+1}\widetilde{v}_s^t + \frac{1}{t+1}\nabla f_{\pi_s^t}(x_s^t)$
7: $v_s^t = \nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(\omega_s) + v_s$
8: $x_s^{t+1} = x_s^t - \gamma v_s^t$
9: **end for**
10: $x_{s+1}^0 = x_s^n$
11: $\omega_{s+1} = x_s^n$
12: $\widetilde{v}_{s+1}^0 = 0^d$
13: $v_{s+1} = \widetilde{v}_s^n$
14: **end for**

In Section 5.1, we defined the technique to approximate the full gradient: (5), (6). However, this approach introduces a question: how should we alter and use the gradient approximation v_s throughout an epoch? In SAG (4), we added a new $\nabla f_{i_t}(x^t)$ and removed its previous version $\nabla f_{i_t}(\phi_{i_t}^t)$, but this requires memory for all $\nabla f_{i_t}\ (\phi_{i_t}^t)$. Building on the concept from SVRG (2), rather than computing $\nabla f_{i_t}\ (\phi_{i_t}^t)$, our update uses the gradient at a reference point ω_s. While SVRG uses the full gradient at this reference point as shown in (2), we utilize an approximation provided by (5):

$$v_s^t = \nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(\omega_s) + v_s.$$

Finally, we perform the step of the algorithm in Line 8, where the approximation v_s is calculated in the previous epoch as (6) in Line 6. We now present the formal description of No Full Grad SVRG (Algorithm 1). The outstanding issue is selecting the point ω_s (Line 11). The approximation v_s, representing the full gradient at ω_s, is derived from gradients evaluated at points between x_{s-1}^1 and x_{s-1}^n. A reasonable choice for ω_s appears to be the average of these points. However, during this epoch, we are continuously moving away from this point, computing $\nabla f_{\pi_s^t}(x_s^t)$ and adding to the reduced gradient. Consequently, the average point changes over time. By the end of the current epoch, it evolves into an average calculated from points ranging from x_s^1 to x_s^n. Thus, choosing ω_s as the last point from the previous epoch is a logical compromise, since we estimate not only how far we can move during the past epoch but also how far we have moved during the current one (see Lemma 1). An intriguing question for future research is whether more frequent or adaptive updates of ω_s could further improve convergence rates [Allen-Zhu and Hazan, 2016].

With this dynamic strategy, our approach improves the efficiency of gradient updates and optimizes memory usage, making it suitable for large-scale optimization problems. Now, we proceed to the theoretical analysis. The problem is examined in both non-convex and strongly convex settings.

5.2.1 Non-convex setting

For a more detailed analysis of this method, we examine the interim results. Our analysis is structured as follows. First, we dissect convergence over a single epoch. Next, we extend this recursively across all epochs. The crucial aspect here is understanding how gradients change within an epoch. To begin, we need to show that these changes depend on two critical factors. First, how well we approximate the true full gradient at the start of each epoch. Second, how far our updates deviate from this initial reference point as we progress through it. To obtain this, we present a lemma.

Lemma 1. *Suppose Assumptions 1, 2 hold. Then for Algorithm 1 a valid estimate is*

$$\left\|\nabla f(\omega_s) - \frac{1}{n}\sum_{t=0}^{n-1} v_s^t\right\|^2 \leqslant 2\|\nabla f(\omega_s) - v_s\|^2 + \frac{2L^2}{n}\sum_{t=0}^{n-1}\|x_s^t - \omega_s\|^2.$$

In contrast to classical SVRG, where only one term matters due to the exact computation of full gradients ($v_s = \nabla f(\omega_s)$), our algorithm avoids such computations. Instead, it introduces an additional term representing the errors in approximating these gradients. This error fundamentally reflects discrepancies between our approximation and the actual gradients at ω_s. We note that v_s averages stochastic gradients from previous epochs (as per Equation 5) with ω_s set as their final point. Thus, this error quantifies shifts among those points relative to further reference points. Beginning each epoch at ω_s, we gauge both potential movements within upcoming epochs and the progress made during past ones. This perspective underscores a strategic balance involved when selecting ω_s, aligning with discussions in Section 5.2. We present estimates for these deviations through a subsequent lemma.

Lemma 2. *Suppose Assumptions 1, 2 hold. Let the stepsize $\gamma \leqslant \frac{1}{2Ln}$. Then for Algorithm 1 a valid estimate is*

$$\left\|\nabla f(\omega_s) - \frac{1}{n}\sum_{t=0}^{n-1} v_s^t\right\|^2 \leqslant 8\gamma^2 L^2 n^2 \|v_s\|^2 + 32\gamma^2 L^2 n^2 \|v_{s-1}\|^2.$$

Now we are ready to present the final result of this section.

Theorem 1. *Suppose Assumptions 1, 2(b) hold. Then Algorithm 1 with $\gamma \leqslant \frac{1}{20Ln}$ to reach ε-accuracy, where $\varepsilon^2 = \frac{1}{S}\sum_{s=1}^{S}\|\nabla f(\omega_s)\|^2$, needs $\mathcal{O}\left({}^{nL}/_{\varepsilon^2}\right)$ iterations and oracle calls.*

Detailed proofs of the results obtained are in Appendix, Section C. We present the first variance reduction method that does not require the calculation of the full gradient and is optimal concerning additional memory in the non-convex setting. Our results demonstrate that the score is, by an order of magnitude, inferior compared to the classical SVRG using independent sampling: $\mathcal{O}(nL)$ versus $\mathcal{O}(n^{2/3}L)$ [Allen-Zhu and Hazan, 2016]. This outcome reflects that we approximate the full gradient at the reference point, rather than considering it at the current state. Regarding Shuffle SVRG, we replicate the current optimal estimate [Malinovsky et al., 2023]. Considering that our method does not necessitate the calculation of full gradients, it is valid and merits further investigation. Finally, the development of the no-full-grad option for non-convex problems is a significant contribution, as this area has not been extensively explored previously (Table 1).

5.2.2 Strongly convex setting

Let us analyze this algorithm for the strongly convex case. Based on the proof of Theorem 1, we construct an analysis that employs the Polyak-Lojasiewicz condition (see Appendix B).

Theorem 2. *Suppose Assumptions 1, 2(a) hold. Then Algorithm 1 with $\gamma \leqslant \frac{1}{20Ln}$ to reach ε-accuracy, where $\varepsilon = f(\omega_{S+1}) - f(x^*)$, needs $\mathcal{O}\left({}^{nL}/_{\mu}\log {}^{1}/_{\varepsilon}\right)$ iterations and oracle calls.*

Our results for the No Full Grad SVRG algorithm under strong convexity conditions are similar to those observed in the non-convex setting. Moreover, it significantly outperforms existing estimates of no-full-grad methods. When comparing our results to those of other shuffling methods (see Table 1), our algorithm improves convergence rates while maintaining optimal extra memory. Thus, it contributes to the entire class of shuffling algorithms.

5.3 SARAH without full gradients

We have previously discussed that SARAH was designed as a variance reduction method that outperforms SVRG in practice and has numerous interesting applications [Nguyen et al., 2017]. This section discusses how to modify the SARAH method to avoid restarts. We consider two approaches: taking steps based on an accurate full gradient or developing a version of SARAH that does not require full gradient computations.

Algorithm 2 No Full Grad SARAH

Input: Initial points $x_0^0 \in \mathbb{R}^d$; Initial gradients $\widetilde{v}_0^0 = 0^d, v_0 = 0^d$
Parameter: Stepsize $\gamma > 0$
for epochs $s = 0, 1, 2, \ldots, S$ **do**
  Sample a permutation $\pi_s^1, \ldots, \pi_s^n$ of $\overline{1, n}$ // *Sampling depends on shuffling heuristic*
  $v_s^0 = v_s$
  $x_s^1 = x_s^0 - \gamma v_s^0$
  for $t = 1, 2, 3, \ldots, n$ **do**
    $\widetilde{v}_s^{t+1} = \frac{t-1}{t}\widetilde{v}_s^t + \frac{1}{t}\nabla f_{\pi_s^t}(x_s^t)$
    $v_s^t = \frac{1}{n}\left(\nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(x_s^{t-1})\right) + v_s^{t-1}$
    $x_s^{t+1} = x_s^t - \gamma v_s^t$
  end for
  $x_{s+1}^0 = x_s^{n+1}$
  $\widetilde{v}_{s+1}^1 = 0$
  $v_{s+1} = \widetilde{v}_s^{n+1}$
end for

There exists a version of SARAH that obviates the need for the full gradient computation [Beznosikov and Takáč, 2023], however its upper estimate, $\mathcal{O}\left({}^{n^2L}/_{\mu}\log {}^{1}/_{\varepsilon}\right)$, significantly deviates from the one obtained for SVRG in the previous section. Let us demonstrate what can be modified in their method to improve this estimate. The authors in the mentioned work employed the same technique to approximate the full gradient as (5). The algorithm applied the standard SARAH update formula (3). To initiate a new recursive cycle at the start of each epoch, an extra step was taken with an approximated full gradient. In this way, they also used the SAG/SAGA idea but provided a recursive reduced gradient update to avoid storing all stochastic gradients during the epoch.

To continue the analysis, we aim to shed light on the differences between SAG and SAGA algorithms, as this is crucial for our modifications. We discussed the SAG update (4). The SAGA update is almost the same, except for the absence of the factor ${}^{1}/_{n}$. Thus, maintaining the notation used for SAG, we can express

the SAGA step as

$$x^{t+1} = x^t - \gamma\Big(\nabla f_{i_t}(x^t) - \nabla f_{i_t}(\phi^t_{i_t}) + \tfrac{1}{n}\sum_{j=1}^{n}\nabla f_j(\phi^t_j)\Big). \quad (7)$$

The key difference hides in the reduction of the variance of the SAG update in n^2 times compared to SAGA with the same ϕ's, however, the payback for such a gain is a non-zero bias in SAG. The choice of unbiasedness was made in SAGA primarily to develop a simple and tight theory for variance reduction methods and to provide theoretical estimates for proximal operators [Defazio et al., 2014a].

Now we can specify and state that the idea behind SAGA was applied in [Beznosikov and Takáč, 2023]. Nevertheless, attempting to increase variance for the sake of zero bias appears illogical here because the shuffling heuristic remains in use, thereby inherently introducing bias. As a result, achieving convergence requires very small step sizes, leading to significantly worse estimates. In contrast, we propose leveraging the concept of SAG, similar to what we did in No Full Grad SVRG, and modifying the SARAH update during each epoch, as

$$v_s^{t+1} = \tfrac{1}{n}\left(\nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(x_s^{t-1})\right) + v_s^{t-1}.$$

This approach enables the use of larger steps and improves convergence rates. We provide the formal description of the No Full Grad SARAH method (Algorithm 2). One can observe that we slightly modify the coefficients in the full gradient approximation scheme (Line 8 of Algorithm 2) compared to Algorithm 1. The discrepancy stems solely from differences in indexing. In this case, the full gradient approximation begins at iteration $t = 1$ rather than $t = 0$, as we incorporate an extra restart step not present in SVRG. Therefore, we make this adjustment to prevent the factor from becoming $1/(n+1)$ instead of $1/n$ in (5). Now we proceed to the theoretical analysis of Algorithm 2 under both non-convex and strongly convex assumptions on the objective function.

5.3.1 Non-convex setting

During the proof of the convergence estimates of SARAH, we proceed similarly to our approach for SVRG. Initially, we focus on a single epoch and demonstrate convergence within it. To achieve this, we estimate the difference between the gradient at the start of the epoch and the average of the reduced gradients used for updates throughout the epoch.

Lemma 3. *Suppose that Assumptions 1, 2 hold. Then for Algorithm 2 a valid estimate is*

$$\left\|\nabla f(x_s^0) - \tfrac{1}{n+1}\sum_{t=0}^{n} v_s^t\right\|^2 \leqslant 2\|\nabla f(x_s^0) - v_s\|^2 + \tfrac{2L^2}{n+1}\sum_{t=1}^{n}\|x_s^t - x_s^{t-1}\|^2.$$

The first term is identical to that previously encountered in the analysis of Algorithm 1 – the difference between the true full gradient and its approximation. Additionally, the second term conveys a similar meaning to its counterpart in Lemma 1. It represents the difference between the current and reference points during an epoch, with the reference point consistently set as the previous one. Thus, we follow a similar approach to SVRG and proceed to the next lemma.

Lemma 4. *Suppose that Assumptions 1, 2 hold. Let the stepsize $\gamma \leqslant \frac{1}{3L}$. Then for Algorithm 2 a valid estimate is*

$$\left\|\nabla f(x_s^0) - \tfrac{1}{n+1}\sum_{t=0}^{n} v_s^t\right\|^2 \leqslant 9\gamma^2L^2\|v_s\|^2 + 36\gamma^2L^2n^2\|v_{s-1}\|^2.$$

Obtaining this crucial lemma, we can now present the final result of this section.

Theorem 3. *Suppose Assumptions 1, 2(b) hold. Then Algorithm 2 with $\gamma \leqslant \frac{1}{20L(n+1)}$ to reach ε-accuracy, where $\varepsilon^2 = \frac{1}{S}\sum_{s=1}^{S}\|\nabla f(x_s^0)\|^2$, needs $\mathcal{O}\left(nL/\varepsilon^2\right)$ iterations and oracle calls.*

We obtain the expected result. Notably, the upper bound for the convergence of No Full Grad SARAH aligns with that of No Full Grad SVRG, as stated in Theorem 1. Our comparison with previous estimates is consistent and detailed in Section 5.2.1.

5.3.2 Strongly convex setting

We extend our analysis on the strongly convex objective function using (PL) (see Appendix B).

Theorem 4. *Suppose Assumptions 1, 2(a) hold. Then Algorithm 2 with $\gamma \leqslant \frac{1}{20L(n+1)}$ to reach ε-accuracy, where $\varepsilon = f(x_{S+1}^0) - f(x^*)$, needs $\mathcal{O}\left(nL/\mu \log 1/\varepsilon\right)$ iterations and oracle calls.*

6 LOWER BOUNDS

A natural question arises: is Algorithm 1 optimal? We address its optimality in the non-convex case (in fact, we can also consider the strongly convex case, but the concept remains the same). A comprehensive explanation requires understanding the essence of *smoothness*

assumptions, which are used to study variance-reduced schemes.

As a result, we present the lower bound for the non-convex finite-sum problem (1) under Assumption 1. Furthermore, we provide an explanation of why it is impossible to construct a lower bound that matches the result of Theorem 1.

Theorem 5 (**Lower bound**). *For any $L > 0$, there exists a problem (1) which satisfies Assumption 1, such that for any output of a first-order algorithm, number of oracle calls N_c required to reach ε-accuracy is lower bounded as $N_c = \Omega\left(L\Delta/\varepsilon^2\right)$.*

In fact, this result has already been stated (see Theorem 4.7 in [Zhou and Gu, 2019]). However, to enhance the clarity of the lower bound, we construct it in a different form. Furthermore, the interpretation of the obtained result (see Remark 4.8 in [Zhou and Gu, 2019]) is not entirely correct. For example, the comparison with the upper bound from [Fang et al., 2018] is inconsistent, as the smoothness parameters are considered different. Consequently, the problem classes *do not coincide*.

Theorem 6 (**Non-optimality**). *For any $L > 0$, there is **no** problem (1) which satisfies Assumption 1, such that for any output of first-order algorithm, number of oracle calls N_c required to reach ε-accuracy is lower bounded with $p > \frac{1}{2}$: $N_c = \Omega\left(n^p L\Delta/\varepsilon^2\right)$.*

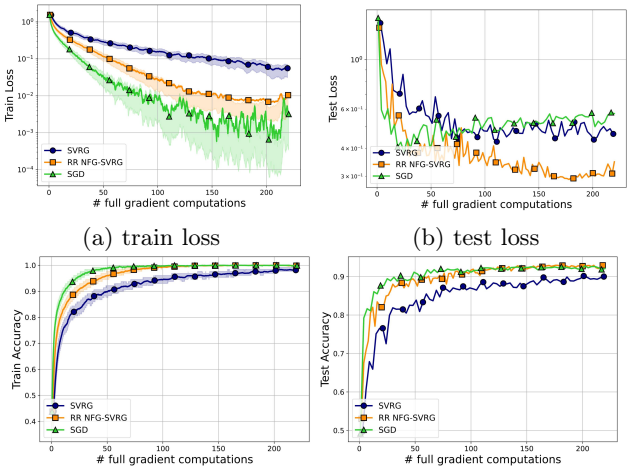
The theorem shows that the best result that can potentially be obtained in terms of lower bounds is $\Omega\left(\sqrt{n}L\Delta/\varepsilon^2\right)$. Therefore, the results for the upper bound (Theorems 1 and 3) are non-optimal, and the lower bound (Theorem 5) could be non-optimal in the class of problems induced by Assumption 1. Theorem 6 signifies that despite superior performance compared to existing results (Table 1), a gap remains between the upper and lower bounds. For details, see Appendix E.

7 EXPERIMENTS

To assess the efficiency of our No Full Gradient (NFG) modifications to SVRG and SARAH, we perform experiments on image-classification benchmarks: CIFAR-10 and CIFAR-100 datasets [Krizhevsky et al., 2009] using the ResNet18 model [He et al., 2016]. In all experiments, the model is trained for 200 epochs with a batch size of 128. We apply weight decay $\lambda_1 = 5 \times 10^{-4}$ and a cosine learning-rate schedule, initializing at 0.1 and decaying to 10^{-3}. For each algorithm, we record training and test curves for (a) cross-entropy loss and (b) accuracy. Curves are plotted against cumulative full-gradient computations, highlighting efficiency gains.

7.1 Results on CIFAR-10

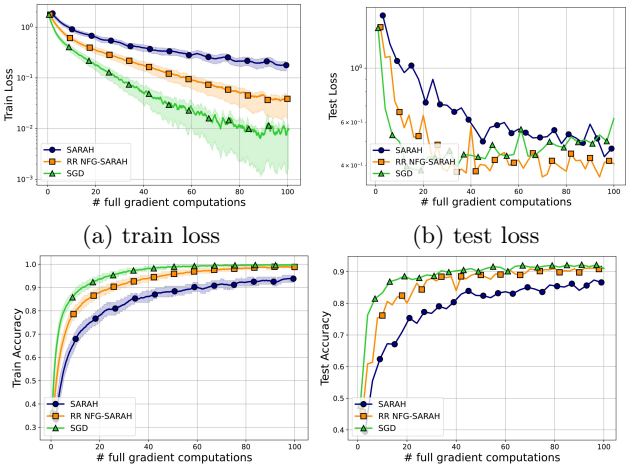
Figures 1, 2 present training and test metrics on CIFAR-10.



(a) train loss (b) test loss

(c) train accuracy (d) test accuracy

Figure 1: No Full Grad SVRG and SVRG.



(a) train loss (b) test loss

(c) train accuracy (d) test accuracy

Figure 2: No Full Grad SARAH and SARAH.

NFG SVRG reduces training loss oscillations compared to SGD, particularly in low-diversity datasets. Despite batch fluctuations, convergence remains smooth. On the test set, NFG SVRG shows better loss reduction, and test accuracy exceeds that of SGD, stabilizing from epoch 150.

NFG SARAH ensures stable training loss convergence, surpassing the standard SARAH in speed concerning full gradient computations. On the test set, the loss reaches a comparable minimum to that of SGD but continues to decrease, while SGD begins to fluctuate. The

final test loss is lower, and test accuracy progressively improves, with enhancement in the later stages.

We also extend the experimental validation of our methods. We present experiments on CIFAR-100 and fine-tune a Swin Transformer [Liu et al., 2021] on Tiny ImageNet [Le and Yang, 2015] in Appendix A.

8 CONCLUSION

This paper introduces an approach that eliminates the necessity of full gradient computations in variance-reduced stochastic methods. Our technique approximates the full gradient via a moving average of stochastic gradients throughout an epoch, theoretically enabled by a shuffling heuristic. We establish upper convergence bounds by integrating this technique into both SVRG and SARAH. Furthermore, we provide lower bounds for the class of stochastic first-order methods employing shuffling.

While this work establishes both upper and lower complexity bounds, a complete picture requires closing the gap between them. Future research could aim to achieve this, for instance, through a refined analysis incorporating mini-batching strategies.

Acknowledgments

The work was done in the Laboratory of Federated Learning Problems (Supported by Grant App. No. 2 to Agreement No. 075-03-2024-214).

References

Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *Journal of Machine Learning Research*, 18(221):1–51, 2018.

Zeyuan Allen-Zhu and Elad Hazan. Variance reduction for faster non-convex optimization. In *International conference on machine learning*, pages 699–707. PMLR, 2016.

Yossi Arjevani, Yair Carmon, John C Duchi, Dylan J Foster, Nathan Srebro, and Blake Woodworth. Lower bounds for non-convex stochastic optimization. *Mathematical Programming*, 199(1):165–214, 2023.

Aleksandr Beznosikov and Martin Takáč. Random-reshuffled sarah does not need full gradient computations. *Optimization Letters*, pages 1–23, 2023.

Aleksandr Beznosikov, Boris Polyak, Eduard Gorbunov, Dmitry Kovalev, and Alexander Gasnikov. Smooth monotone stochastic variational inequalities and saddle point problems: A survey. *European Mathematical Society Magazine*, (127):15–28, 2023.

Léon Bottou. Curiously fast convergence of some stochastic gradient descent algorithms. In *Proceedings of the symposium on learning and data science, Paris*, volume 8, pages 2624–2633. Citeseer, 2009.

Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex sgd. *Advances in neural information processing systems*, 32, 2019.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014a.

Aaron Defazio, Justin Domke, et al. Finito: A faster, permutable incremental gradient method for big data problems. In *International Conference on Machine Learning*, pages 1125–1133. PMLR, 2014b.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Yoel Drori and Ohad Shamir. The complexity of finding stationary points with stochastic gradient descent. In *International Conference on Machine Learning*, pages 2658–2667. PMLR, 2020.

Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. *Advances in neural information processing systems*, 31, 2018.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM journal on optimization*, 23(4): 2341–2368, 2013.

Saeed Ghadimi, Guanghui Lan, and Hongchao Zhang. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155(1):267–305, 2016.

Robert M Gower, Mark Schmidt, Francis Bach, and Peter Richtárik. Variance-reduced methods for machine learning. *Proceedings of the IEEE*, 108(11): 1968–1983, 2020.

Mert Gurbuzbalaban, Asuman Ozdaglar, and Pablo A Parrilo. On the convergence rate of incremental aggregated gradient algorithms. *SIAM Journal on Optimization*, 27(2):1035–1048, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Wenqing Hu, Chris Junchi Li, Xiangru Lian, Ji Liu, and Huizhuo Yuan. Efficient smooth non-convex stochastic compositional optimization via stochastic recursive gradient descent. *Advances in Neural Information Processing Systems*, 32, 2019.

Xinmeng Huang, Kun Yuan, Xianghui Mao, and Wotao Yin. An improved analysis and rates for variance reduction under without-replacement sampling orders. *Advances in Neural Information Processing Systems*, 34:3232–3243, 2021.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013.

Anastasia Koloskova, Nikita Doikov, Sebastian U. Stich, and Martin Jaggi. On convergence of incremental gradient for non-convex smooth functions, 2024.

Dmitry Kovalev, Samuel Horváth, and Peter Richtárik. Don't jump through hoops and remove those loops: Svrg and katyusha are better without the outer loop. In *Algorithmic Learning Theory*, pages 451–467. PMLR, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Guanghui Lan. *First-order and stochastic optimization methods for machine learning*, volume 1. Springer, 2020.

Yann Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Bingcong Li, Meng Ma, and Georgios B Giannakis. On the convergence of sarah and beyond. In *International Conference on Artificial Intelligence and Statistics*, pages 223–233. PMLR, 2020.

Zhize Li, Slavomír Hanzely, and Peter Richtárik. Zerosarah: Efficient nonconvex finite-sum optimization with zero full gradient computation. *arXiv preprint arXiv:2103.01447*, 2021.

Deyi Liu, Lam M Nguyen, and Quoc Tran-Dinh. An optimal hybrid variance-reduced algorithm for stochastic composite nonconvex optimization. *arXiv preprint arXiv:2008.09055*, 2020.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

Grigory Malinovsky, Alibek Sailanbayev, and Peter Richtárik. Random reshuffling with variance reduction: New analysis and better rates. In *Uncertainty in Artificial Intelligence*, pages 1347–1357. PMLR, 2023.

Daniil Medyakov, Gleb Molodtsov, Evseev Grigoriy, Egor Petrov, and Aleksandr Beznosikov. Shuffling heuristic in variational inequalities: Establishing new convergence guarantees. In *International Conference on Computational Optimization*, 2024.

Dmitry Metelev, Savelii Chezhegov, Alexander Rogozin, Aleksandr Beznosikov, Alexander Sholokhov, Alexander Gasnikov, and Dmitry Kovalev. Decentralized finite-sum optimization over time-varying networks. *arXiv preprint arXiv:2402.02490*, 2024.

Konstantin Mishchenko, Ahmed Khaled, and Peter Richtárik. Random reshuffling: Simple analysis with vast improvements. *Advances in Neural Information Processing Systems*, 33:17309–17320, 2020.

Aryan Mokhtari, Mert Gurbuzbalaban, and Alejandro Ribeiro. Surpassing gradient descent provably: A cyclic incremental method with linear convergence rate. *SIAM Journal on Optimization*, 28(2):1420–1447, 2018.

Eric Moulines and Francis Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. *Advances in neural information processing systems*, 24, 2011.

Yurii Nesterov et al. *Lectures on convex optimization*, volume 137. Springer, 2018.

Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *International conference on machine learning*, pages 2613–2621. PMLR, 2017.

Lam M Nguyen, Katya Scheinberg, and Martin Takáč. Inexact sarah algorithm for stochastic optimization. *Optimization Methods and Software*, 36(1):237–258, 2021.

PH Nguyen, LM Nguyen, and M van Dijk. Tight dimension independent lower bound on the expected convergence rate for diminishing step sizes in sgd. In *33rd Annual Conference on Neural Information Processing Systems, NeurIPS 2019*. Neural information processing systems foundation, 2019.

Youngsuk Park and Ernest K Ryu. Linear convergence of cyclic saga. *Optimization Letters*, 14(6):1583–1598, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. 2012.

Benjamin Recht and Christopher Ré. Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation*, 5(2):201–226, 2013.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

Nicolas Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. *Advances in neural information processing systems*, 25, 2012.

Itay Safran and Ohad Shamir. How good is sgd with random shuffling? In *Conference on Learning Theory*, pages 3250–3284. PMLR, 2020.

Sebastian U Stich. Unified optimal analysis of the (stochastic) gradient method. *arXiv preprint arXiv:1907.04232*, 2019.

Tao Sun, Yuejiao Sun, Dongsheng Li, and Qing Liao. General proximal incremental aggregated gradient algorithms: Better and novel results under general scheme. *Advances in Neural Information Processing Systems*, 32, 2019.

Nuri Denizcan Vanli, Mert Gurbuzbalaban, and Asu Ozdaglar. A stronger convergence result on the proximal incremental aggregated gradient method. *arXiv preprint arXiv:1611.08022*, 2016.

Bicheng Ying, Kun Yuan, and Ali H Sayed. Variance-reduced stochastic learning under random reshuffling. *IEEE Transactions on Signal Processing*, 68:1390–1408, 2020.

Min Zhang, Yao Shu, and Kun He. Tight lower complexity bounds for strongly convex finite-sum optimization. *arXiv preprint arXiv:2010.08766*, 2020.

Dongruo Zhou and Quanquan Gu. Lower bounds for smooth nonconvex finite-sum optimization. In *International Conference on Machine Learning*, pages 7574–7583. PMLR, 2019.

CHECKLIST

The checklist follows the references. For each question, choose your answer from the three possible options: Yes, No, Not Applicable. You are encouraged to include a justification to your answer, either by referencing the appropriate section of your paper or providing a brief inline description (1-2 sentences). Please do not modify the questions. Note that the Checklist section does not count towards the page limit. Not including the checklist in the first submission won't result in desk rejection, although in such case we will ask you to upload it during the author response period and include it in camera ready (if accepted).

In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes, Section 4]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes, Sections 7, A]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes, Section 4]
   (b) Complete proofs of all theoretical results. [Yes, Sections C, D, E]
   (c) Clear explanations of any assumptions. [Yes, Sections 4, 6]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes, Sections 7, A]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes, Sections 7, A]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes, Sections 7, A]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes, Sections 7, A]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

   (a) The full text of instructions given to participants and screenshots. [Not Applicable]

   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Variance Reduction Methods Do Not Need to Compute Full Gradients: Improved Efficiency through Shuffling

A ADDITIONAL EXPERIMENTS

A.1 Least squares regression.

We consider the non-linear least squares loss problem:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n}(y_i - h_i)^2, \tag{8}$$

where n is the number of samples, y_i is the true value for sample i, h_i is value for sample i, calculated as $h_i = \frac{1}{1+\exp(-z_i)}$, with $z_i = A_i \cdot x$, addressing problem (8). Based on our theoretical estimates, which suggest inferior performance compared to standard SVRG and SARAH, we expect less favorable convergence. To address this limitation, we expand our investigation to examine the convergence of this method by tuning the stepsize, a topic that falls outside the scope of our current theoretical framework. The plots are shown in Figures 3-4.

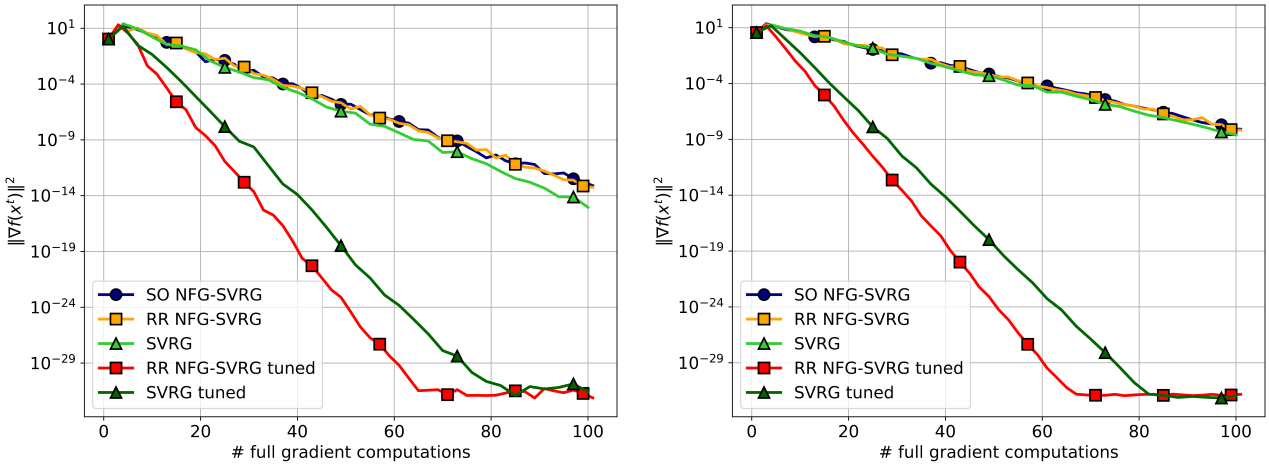


Figure 3: No Full Grad SVRG and SVRG convergence with theoretical and tuned step sizes on problem (8) on the `ijcnn1` (left) and `a9a` (right) datasets.

Upon examining the plots, we notice that although No Full Grad versions may converge slightly slower although comparable than its regular counterpart when using the theoretical step size, it significantly outperforms SVRG and SARAH, respectively, when the step size is optimally tuned. This highlights the potential of our method to achieve superior convergence rates with proper parameter adjustments, providing a robust alternative for large-scale optimization.

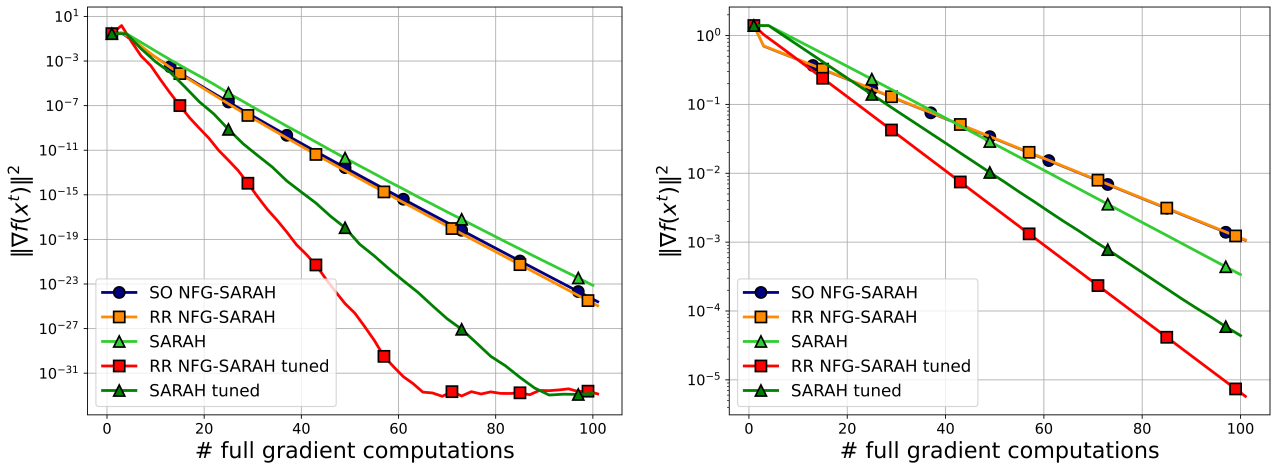


Figure 4: No Full Grad SARAH and SARAH convergence with theoretical and tuned step sizes on problem (8) on the `ijcnn1` (left) and `a9a` (right) datasets.

A.2 ResNet-18 on CIFAR-10/CIFAR-100 classification.

Experiments on CIFAR-100

We provide the results for image classification on the CIFAR-100 dataset. We keep the same experimental setup as for classification on the CIFAR-10 dataset (see Section 7). The plots are provided in Figures 5-6.

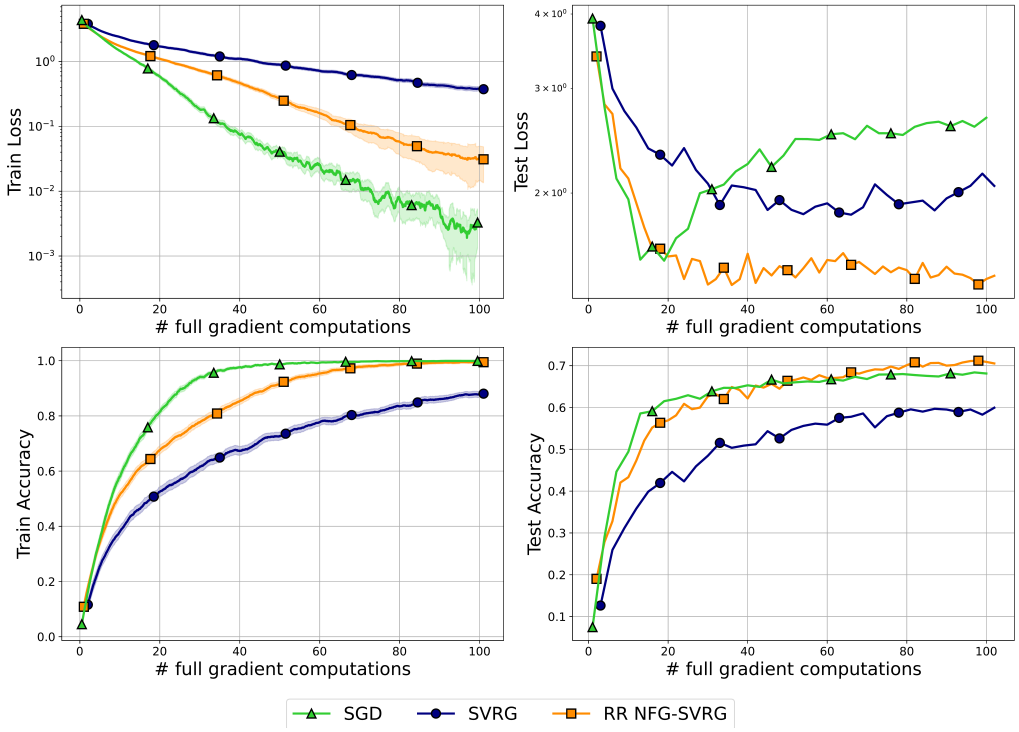


Figure 5: No Full Grad SVRG and SVRG on CIFAR-100 convergence.

The SVRG algorithm follows a similar trend, with test loss stabilizing instead of increasing, unlike SGD. While SGD rebounds, SVRG maintains a plateau before further improvement. Test accuracy surpasses SGD from epoch

50 onward.

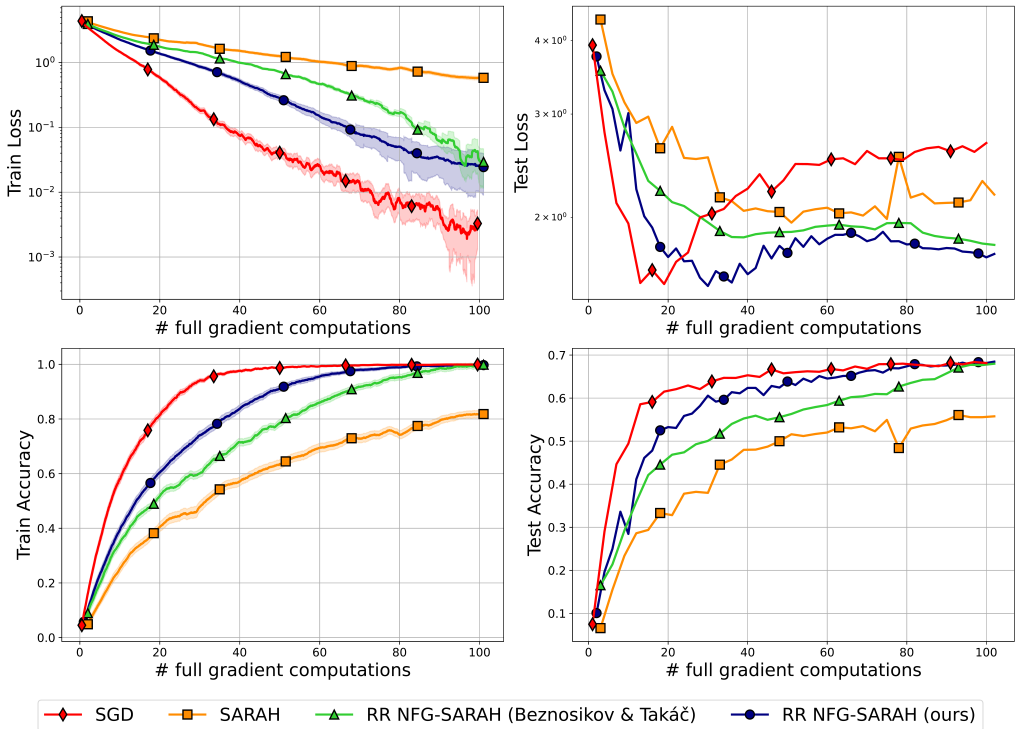


Figure 6: No Full Grad SARAH and SARAH on CIFAR-100 convergence.

The SARAH algorithm stabilizes training loss convergence and outpaces standard SARAH. Test loss decreases beyond SGD's minimum, leading to a lower final loss. Test accuracy initially rises slowly but later accelerates, reducing overfitting.

Experimental Design

The experiments were implemented in Python using the PyTorch library [Paszke et al., 2019], leveraging both a single CPU (Intel Xeon 2.20 GHz) and a single GPU (NVIDIA Tesla P100) for computation. To emulate a distributed environment, we split batches across multiple workers, simulating a decentralized optimization setting.

Our algorithms are evaluated in terms of accuracy and the number of full gradient computations. The experiments are conducted with the following setup:

- number of workers $M = 5$;
- learning rate $\gamma = 0.1$ for both optimizers decaying to 10^{-3};
- regularization parameter $\lambda_1 = 0.0005$.

A.3 Tiny ImageNet Classification with Swin Transformer fine-tuning

Experimental Protocol

Our image classification experiments on the Tiny ImageNet dataset [Le and Yang, 2015] employed the Tiny Swin Transformer architecture [Liu et al., 2021]. This lightweight variant of the Swin Transformer is characterized by its hierarchical design and the use of shifted windows for efficient self-attention computation. The specific configuration utilized involved non-overlapping 4×4 input patches and a 7×7 window size for local self-attention.

We initialized the model using pretrained weights from ImageNet-1K [Deng et al., 2009], specifically the `swin_T_patch4_window7_224` checkpoint provided in the official Swin Transformer repository[2]. The model was then fine-tuned on Tiny ImageNet.

The Tiny ImageNet dataset comprises 200 classes with images of 64×64 resolution. To meet the model's input requirements, all images were upsampled to 224×224. A standard ImageNet-style data augmentation pipeline was implemented, including random resized cropping and horizontal flipping.

Training spanned approximately 30 full gradient computations, with a batch size of 256. A cosine learning rate schedule was adopted, featuring a linear warm-up phase for the initial 10% of total training steps, followed by decay to 10% of the peak learning rate. Weight decay was selected from $\{0, 0.01, 0.1\}$ based on validation performance. All optimization methods incorporated gradient clipping with a threshold of 1.0.

Performance on Image Classification

Further results and training curves for the Tiny Swin Transformer on the Tiny ImageNet classification task are presented in Figure 7.

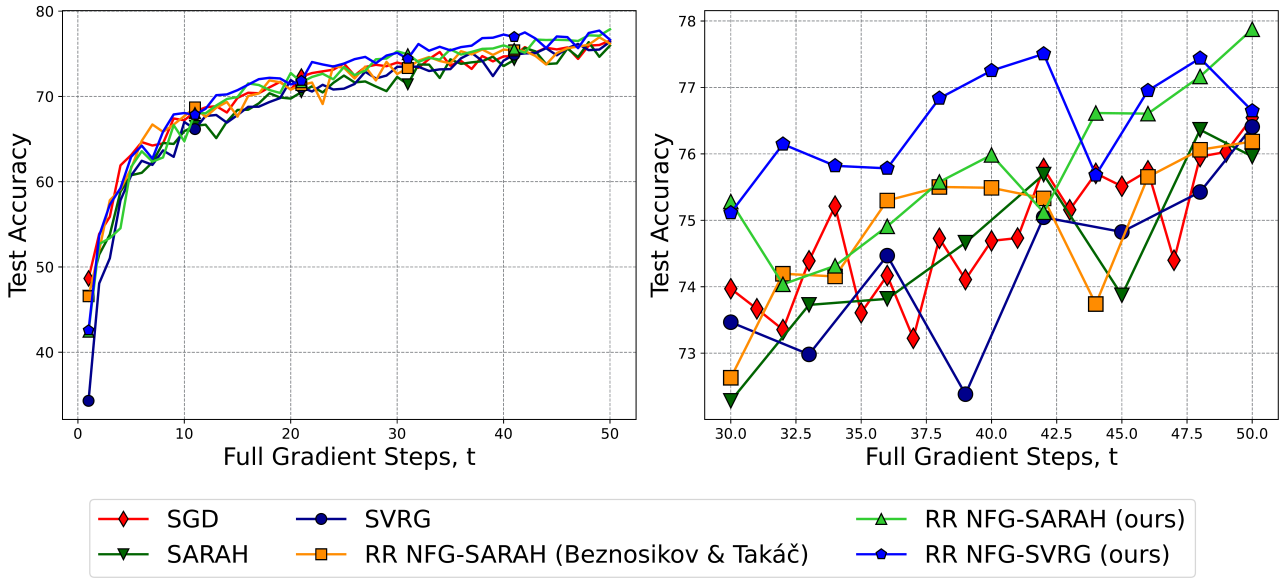


Figure 7: No Full Grad SARAH and SVRG on Tiny ImageNet convergence.

Table 2: Final Accuracy of Variance Reduction Methods on Tiny ImageNet Convergence.

Algorithm	Final accuracy (↑)
SGD	76.545
SARAH	75.961
SVRG	76.407
RR NFG-SARAH [Beznosikov and Takáč, 2023]	76.186
RR NFG-SARAH (ours)	**77.875**
RR NFG-SVRG (ours)	76.646

The results demonstrate the superior performance of our methods compared to classical variance reduction methods and modified version of SARAH [Beznosikov and Takáč, 2023]. The advantage is evident for both low-dimensional problems and complex networks with a large number of parameters.

[2] https://github.com/microsoft/Swin-Transformer/blob/main/MODELHUB.md

B GENERAL INEQUALITIES

We introduce important inequalities that are used in further proofs. Let f adhere to Assumption 1, g adhere to Assumption 2(a). Then for any real number i and for all vectors $x, y, \{x_i\} \in \mathbb{R}^d$ with a positive scalars α, β, the following inequalities hold:

$$2\langle x, y\rangle \leqslant \frac{\|x\|^2}{\alpha} + \alpha\|y\|^2, \tag{Scalar}$$

$$2\langle x, y\rangle = \|x+y\|^2 - \|x\|^2 - \|y\|, \tag{Norm}$$

$$\|x+y\|^2 \leqslant (1+\beta)\|x\|^2 + (1+\frac{1}{\beta})\|y\|^2, \tag{Quad}$$

$$f(x) \leqslant f(y) + \langle \nabla f(y), x-y\rangle + \frac{L}{2}\|x-y\|^2, \tag{Lip}$$

$$\left\|\sum_{i=1}^n x_i\right\|^2 \leqslant n\sum_{i=1}^n \|x_i\|^2 \qquad \text{(Cauchy-Schwarz)}, \tag{CS}$$

$$g(x) - \inf g \leqslant \frac{1}{2\mu}\|\nabla g(x)\|^2 \qquad \text{(Polyak-Lojasiewicz)}. \tag{PL}$$

Here, (Lip) was derived in [Nesterov et al., 2018] in Theorem 2.1.5.

C NO FULL GRAD SVRG

For the convenience of the reader, we provide here the short description of Algorithm 1. If we consider it in epoch $s \neq 0$, one can note that the update rule is nothing but

$$\begin{cases} \text{initial initialization:} \\ \quad \omega_s = x_s^0 = x_{s-1}^n \\ \quad v_s = \frac{1}{n}\sum\limits_{t=0}^{n-1} \nabla f_{\pi_{s-1}^t}(x_{s-1}^t) \\ \text{for all iterations during the epoch :} \\ \quad v_s^t = \nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(\omega_s) + v_s \\ \quad x_s^{t+1} = x_s^t - \gamma v_s^t \end{cases} \tag{9}$$

C.1 Non-convex setting

Lemma 5. *Suppose that Assumptions 1, 2 hold. Let the stepsize $\gamma \leqslant \frac{1}{Ln}$. Then for Algorithm 1 it holds*

$$f(\omega_{s+1}) \leqslant f(\omega_s) - \frac{\gamma n}{2}\|\nabla f(\omega_s)\|^2 + \frac{\gamma n}{2}\left\|\nabla f(\omega_s) - \frac{1}{n}\sum_{t=0}^{n-1} v_s^t\right\|^2.$$

Proof. Using the iteration of Algorithm 1 (9), we have

$$\begin{aligned} f(\omega_{s+1}) &= f(\omega_s - (\omega_s - \omega_{s+1})) \\ &\overset{\text{(Lip)}}{\leqslant} f(\omega_s) + \langle \nabla f(\omega_s), \omega_{s+1} - \omega_s\rangle + \frac{L}{2}\|\omega_{s+1} - \omega_s\|^2 \\ &= f(\omega_s) - \gamma n\left\langle \nabla f(\omega_s), \frac{1}{n}\sum_{t=0}^{n-1} v_s^t\right\rangle + \frac{\gamma^2 n^2 L}{2}\left\|\frac{1}{n}\sum_{t=0}^{n-1} v_s^t\right\|^2 \\ &\overset{\text{(Norm)}}{=} f(\omega_s) - \frac{\gamma n}{2}\left[\|\nabla f(\omega_s)\|^2 + \left\|\frac{1}{n}\sum_{t=0}^{n-1} v_s^t\right\|^2 - \left\|\nabla f(\omega_s) - \frac{1}{n}\sum_{t=0}^{n-1} v_s^t\right\|^2\right] \\ &\quad + \frac{\gamma^2 n^2 L}{2}\left\|\frac{1}{n}\sum_{t=0}^{n-1} v_s^t\right\|^2 \end{aligned}$$

$$= \quad f(\omega_s) - \frac{\gamma n}{2}\left[\|\nabla f(\omega_s)\|^2 - \left\|\nabla f(\omega_s) - \frac{1}{n}\sum_{t=0}^{n-1} v_s^t\right\|^2\right] - \frac{\gamma n}{2}\cdot(1-\gamma n L)\left\|\frac{1}{n}\sum_{t=0}^{n-1} v_s^t\right\|^2,$$

Choosing $\gamma : \frac{\gamma n}{2}(1-\gamma n L) > 0$ and note that such a choice is followed by $\gamma \leqslant \frac{1}{Ln}$. In that way, we make the last term is negative and obtain the result of the lemma. □

Now we want to address the last term in the inequality of Lemma 5. We prove the following lemma.

Lemma 6 (**Lemma 1**). *Suppose that Assumptions 1, 2 hold. Then for Algorithm 1 a valid estimate is*

$$\left\|\nabla f(\omega_s) - \frac{1}{n}\sum_{t=0}^{n-1} v_s^t\right\|^2 \leqslant 2\|\nabla f(\omega_s) - v_s\|^2 + \frac{2L^2}{n}\sum_{t=0}^{n-1}\|x_s^t - \omega_s\|^2.$$

Proof. We straightforwardly move to estimate of the desired norm:

$$\begin{aligned}
\left\|\nabla f(\omega_s) - \frac{1}{n}\sum_{t=0}^{n-1} v_s^t\right\|^2 &\overset{(9)}{=} \left\|\nabla f(\omega_s) - \frac{1}{n}\left(n v_s + \sum_{t=0}^{n-1}\left(\nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(\omega_s)\right)\right)\right\|^2 \\
&\overset{(CS)}{\leqslant} 2\|\nabla f(\omega_s) - v_s\|^2 + \frac{2}{n^2}\left\|\sum_{t=0}^{n-1}\left(\nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(\omega_s)\right)\right\|^2 \\
&\overset{(CS)}{\leqslant} 2\|\nabla f(\omega_s) - v_s\|^2 + \frac{2}{n}\sum_{t=0}^{n-1}\left\|\nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(\omega_s)\right\|^2 \\
&\overset{\text{Ass. 1}}{\leqslant} 2\|\nabla f(\omega_s) - v_s\|^2 + \frac{2L^2}{n}\sum_{t=0}^{n-1}\left\|x_s^t - \omega_s\right\|^2,
\end{aligned} \tag{10}$$

which ends the proof. □

Lemma 7 (**Lemma 2**). *Suppose that Assumptions 1, 2 hold. Let the stepsize $\gamma \leqslant \frac{1}{2Ln}$. Then for Algorithm 1 a valid estimate is*

$$\left\|\nabla f(\omega_s) - \frac{1}{n}\sum_{t=0}^{n-1} v_s^t\right\|^2 \leqslant 8\gamma^2 L^2 n^2\|v_s\|^2 + 32\gamma^2 L^2 n^2\|v_{s-1}\|^2.$$

Proof. To begin with, in Lemma 6, we obtain

$$\left\|\nabla f(\omega_s) - \frac{1}{n}\sum_{t=0}^{n-1} v_s^t\right\|^2 \leqslant 2\|\nabla f(\omega_s) - v_s\|^2 + \frac{2L^2}{n}\sum_{t=0}^{n-1}\|x_s^t - \omega_s\|^2. \tag{11}$$

Let us show what v_s is (here we use Line 6 of Algorithm 1):

$$\begin{aligned}
v_s = \widetilde{v}_{s-1}^n &= \frac{n-1}{n}\widetilde{v}_{s-1}^{n-1} + \frac{1}{n}\nabla f_{\pi_{s-1}^{n-1}}(x_{s-1}^{n-1}) \\
&= \frac{n-1}{n}\cdot\frac{n-2}{n-1}\widetilde{v}_{s-1}^{n-2} + \frac{n-1}{n}\cdot\frac{1}{n-1}\nabla f_{\pi_{s-1}^{n-2}}(x_{s-1}^{n-2}) + \frac{1}{n}\nabla f_{\pi_{s-1}^{n-1}}(x_{s-1}^{n-1}) \\
&= \frac{n-1}{n}\cdot\frac{n-2}{n-1}\cdot\ldots\cdot 0\cdot\widetilde{v}_{s-1}^0 + \frac{1}{n}\sum_{t=0}^{n-1}\nabla f_{\pi_{s-1}^t}(x_{s-1}^t) \\
&\overset{(i)}{=} \frac{1}{n}\sum_{t=0}^{n-1}\nabla f_{\pi_{s-1}^t}(x_{s-1}^t),
\end{aligned} \tag{12}$$

where equation (i) is correct due to initialization $\widetilde{v}_{s-1}^0 = 0$ (Line 12 of Algorithm 1). In that way, using (11) and (12),

$$\left\|\nabla f(\omega_s) - \frac{1}{n}\sum_{t=0}^{n-1} v_s^t\right\|^2 \leqslant 2\left\|\nabla f(\omega_s) - \frac{1}{n}\sum_{t=0}^{n-1} \nabla f_{\pi_{s-1}^t}(x_{s-1}^t)\right\|^2 + \frac{2L^2}{n}\sum_{t=0}^{n-1}\|x_s^t - \omega_s\|^2.$$

Then, using (1),

$$\begin{aligned}
\left\|\nabla f(\omega_s) - \frac{1}{n}\sum_{t=0}^{n-1} v_s^t\right\|^2 &\leqslant 2\left\|\frac{1}{n}\sum_{t=0}^{n-1}\left(\nabla f_{\pi_{s-1}^t}(\omega_s) - \nabla f_{\pi_{s-1}^t}(x_{s-1}^t)\right)\right\|^2 + \frac{2L^2}{n}\sum_{t=0}^{n-1}\|x_s^t - \omega_s\|^2 \\
&\overset{\text{(CS)}}{\leqslant} \frac{2}{n}\sum_{t=0}^{n-1}\|\nabla f_{\pi_{s-1}^t}(\omega_s) - \nabla f_{\pi_{s-1}^t}(x_{s-1}^t)\|^2 + \frac{2L^2}{n}\sum_{t=0}^{n-1}\|x_s^t - \omega_s\|^2 \\
&\overset{\text{Ass. 1}}{\leqslant} \frac{2L^2}{n}\sum_{t=0}^{n-1}\|x_{s-1}^t - \omega_s\|^2 + \frac{2L^2}{n}\sum_{t=0}^{n-1}\|x_s^t - \omega_s\|^2 \\
&\overset{\text{(Quad)}}{\leqslant} \frac{4L^2}{n}\sum_{t=0}^{n-1}\|x_{s-1}^t - \omega_{s-1}\|^2 + \frac{4L^2}{n}\sum_{t=0}^{n-1}\|\omega_s - \omega_{s-1}\|^2 + \frac{2L^2}{n}\sum_{t=0}^{n-1}\|x_s^t - \omega_s\|^2.
\end{aligned} \tag{13}$$

Now we have to bound these three terms. Let us begin with $\sum\limits_{t=0}^{n-1}\|x_s^t - \omega_s\|^2$.

$$\begin{aligned}
\sum_{t=0}^{n-1}\|x_s^t - \omega_s\|^2 &= \gamma^2\sum_{t=0}^{n-1}\left\|\sum_{k=0}^{t-1} v_s^k\right\|^2 \overset{(9)}{=} \gamma^2\sum_{t=0}^{n-1}\left\|tv_s + \sum_{k=0}^{t-1}\left(\nabla f_{\pi_s^k}(x_s^k) - \nabla f_{\pi_s^k}(\omega_s)\right)\right\|^2 \\
&\overset{\text{(CS)}}{\leqslant} 2\gamma^2\sum_{t=0}^{n-1} t^2\|v_s\|^2 + 2\gamma^2\sum_{t=0}^{n-1} t\sum_{k=0}^{t-1}\|\nabla f_{\pi_s^k}(x_s^k) - \nabla f_{\pi_s^k}(\omega_s)\|^2 \\
&\overset{\text{Ass. 1}}{\leqslant} 2\gamma^2 n^3\|v_s\|^2 + 2\gamma^2 L^2 n\sum_{t=0}^{n-1}\sum_{k=0}^{t-1}\|x_s^k - \omega_s\|^2 \\
&\leqslant 2\gamma^2 n^3\|v_s\|^2 + 2\gamma^2 L^2 n^2\sum_{t=0}^{n-2}\|x_s^t - \omega_s\|^2 \\
&\leqslant 2\gamma^2 n^3\|v_s\|^2 + 2\gamma^2 L^2 n^2\sum_{t=0}^{n-1}\|x_s^t - \omega_s\|^2.
\end{aligned}$$

Expressing $\sum\limits_{t=0}^{n-1}\|x_s^t - \omega_s\|^2$ from here, we get

$$\sum_{t=0}^{n-1}\|x_s^t - \omega_s\|^2 \leqslant \frac{2\gamma^2 n^3\|v_s\|^2}{1 - 2\gamma^2 L^2 n^2}.$$

To finish this part of proof it remains for us to choose appropriate γ. In Lemma 5 we require $\gamma \leqslant \frac{1}{Ln}$. There we choose smaller values of $\gamma : \gamma \leqslant \frac{1}{2Ln}$ (with that values all previous transitions is correct). Now we provide final estimation of this norm:

$$\sum_{t=0}^{n-1} \|x_s^t - \omega_s\|^2 \leqslant 4\gamma^2 n^3 \|v_s\|^2. \tag{14}$$

One can note the boundary of the $\sum\limits_{t=0}^{n-1} \|x_{s-1}^t - \omega_{s-1}\|^2$ term is similar because it involves the same sum of norms from the previous epoch.

$$\sum_{t=0}^{n} \|x_{s-1}^t - \omega_{s-1}\|^2 \leqslant 4\gamma^2 n^3 \|v_{s-1}\|^2. \tag{15}$$

It remains for us to estimate the $\sum\limits_{t=0}^{n-1} \|\omega_s - \omega_{s-1}\|^2$ term.

$$\begin{aligned}
\sum_{t=0}^{n-1} \|\omega_s - \omega_{s-1}\|^2 &= \gamma^2 \sum_{t=0}^{n-1} \left\| \sum_{k=0}^{n-1} v_{s-1}^k \right\|^2 \\
&\overset{(9)}{=} \gamma^2 \sum_{t=0}^{n-1} \left\| n v_{s-1} + \sum_{k=0}^{n-1} \left(\nabla f_{\pi_{s-1}^k}(x_{s-1}^k) - \nabla f_{\pi_{s-1}^k}(\omega_{s-1}) \right) \right\|^2 \\
&\overset{(CS)}{\leqslant} 2\gamma^2 \sum_{t=0}^{n-1} n^2 \|v_{s-1}\|^2 \\
&\quad + 2\gamma^2 \sum_{t=0}^{n-1} n \sum_{k=0}^{n-1} \|\nabla f_{\pi_{s-1}^k}(x_{s-1}^k) - \nabla f_{\pi_{s-1}^k}(\omega_{s-1})\|^2 \\
&\overset{\text{Ass. 1}}{\leqslant} 2\gamma^2 n^3 \|v_{s-1}\|^2 + 2\gamma^2 L^2 n \sum_{t=0}^{n-1} \sum_{k=0}^{t-1} \|x_{s-1}^k - \omega_{s-1}\|^2 \\
&\leqslant 2\gamma^2 n^3 \|v_{s-1}\|^2 + 2\gamma^2 L^2 n^2 \sum_{t=0}^{n-2} \|x_{s-1}^t - \omega_{s-1}\|^2 \\
&\leqslant 2\gamma^2 n^3 \|v_{s-1}\|^2 + 2\gamma^2 L^2 n^2 \sum_{t=0}^{n-1} \|x_{s-1}^t - \omega_{s-1}\|^2 \\
&\overset{(15)}{\leqslant} 2\gamma^2 n^3 \|v_{s-1}\|^2 + 8\gamma^4 L^2 n^5 \sum_{t=0}^{n-1} \|x_{s-1}^t - \omega_{s-1}\|^2.
\end{aligned}$$

Using our choice $\gamma \leqslant \frac{1}{2Ln}$, we derive the estimate of the last term:

$$\sum_{t=0}^{n-1} \|\omega_s - \omega_{s-1}\|^2 \leqslant 2\gamma^2 n^3 \|v_{s-1}\|^2 + 2\gamma^2 n^3 \|v_{s-1}\|^2 = 4\gamma^2 n^3 \|v_{s-1}\|^2. \tag{16}$$

Now we can apply the upper bounds obtained in (14) – (16) to (13) and have

$$\begin{aligned}
\left\| \nabla f(\omega_s) - \frac{1}{n} \sum_{t=0}^{n-1} v_s^t \right\|^2 &\leqslant 8\gamma^2 L^2 n^2 \|v_s\|^2 + 16\gamma^2 L^2 n^2 \|v_{s-1}\|^2 + 16\gamma^2 L^2 n^2 \|v_{s-1}\|^2 \\
&= 8\gamma^2 L^2 n^2 \|v_s\|^2 + 32\gamma^2 L^2 n^2 \|v_{s-1}\|^2,
\end{aligned}$$

which ends the proof. □

Theorem 7 (**Theorem 1**). *Suppose Assumptions 1, 2(b) hold. Then Algorithm 1 with $\gamma \leqslant \frac{1}{20Ln}$ to reach ε-accuracy, where $\varepsilon^2 = \frac{1}{S}\sum_{s=1}^{S}\|\nabla f(\omega_s)\|^2$, needs*

$$\mathcal{O}\left(\frac{nL}{\varepsilon^2}\right) \quad \textit{iterations and oracle calls.}$$

Proof. We combine the result of Lemma 5 with the result of Lemma 7 and obtain

$$\begin{aligned} f(\omega_{s+1}) \leqslant f(\omega_s) &- \frac{\gamma n}{2}\|\nabla f(\omega_s)\|^2 \\ &+ \frac{\gamma n}{2}\left(8\gamma^2 L^2 n^2 \|v_s\|^2 + 32\gamma^2 L^2 n^2 \|v_{s-1}\|^2\right). \end{aligned}$$

We subtract $f(x^*)$ from both parts:

$$\begin{aligned} f(\omega_{s+1}) - f(x^*) &\leqslant f(\omega_s) - f(x^*) - \frac{\gamma n}{2}\|\nabla f(\omega_s)\|^2 \\ &\quad + \frac{\gamma n}{2}\left(8\gamma^2 L^2 n^2 \|v_s\|^2 + 32\gamma^2 L^2 n^2 \|v_{s-1}\|^2\right) \\ &= f(\omega_s) - f(x^*) - \frac{\gamma n}{4}\|\nabla f(\omega_s)\|^2 \\ &\quad + \frac{\gamma n}{2}\left(8\gamma^2 L^2 n^2 \|v_s\|^2 + 32\gamma^2 L^2 n^2 \|v_{s-1}\|^2\right) \\ &\quad - \frac{\gamma n}{4}\|\nabla f(\omega_s)\|^2. \end{aligned}$$

Then, transforming the last term by using (Quad) with $\beta = 1$, we get

$$\begin{aligned} f(\omega_{s+1}) - f(x^*) &\leqslant f(\omega_s) - f(x^*) - \frac{\gamma n}{4}\|\nabla f(\omega_s)\|^2 \\ &\quad + \frac{\gamma n}{2}\left(8\gamma^2 L^2 n^2 \|v_s\|^2 + 32\gamma^2 L^2 n^2 \|v_{s-1}\|^2\right) \\ &\quad - \frac{\gamma n}{8}\|v_s\|^2 + \frac{\gamma n}{4}\|v_s - \nabla f(x\omega_s)\|^2. \end{aligned}$$

Using Lemma 7 to $\|v_s - \nabla f(\omega_s)\|^2$ (specially $\frac{4L^2}{n}\cdot(15) + \frac{4L^2}{n}\cdot(16)$),

$$\begin{aligned} f(\omega_{s+1}) - f(x^*) &\leqslant f(\omega_s) - f(x^*) - \frac{\gamma n}{4}\|\nabla f(\omega_s)\|^2 \\ &\quad + \frac{\gamma n}{2}\left(8\gamma^2 L^2 n^2 \|v_s\|^2 + 32\gamma^2 L^2 n^2 \|v_{s-1}\|^2\right) \\ &\quad - \frac{\gamma n}{8}\|v_s\|^2 + \frac{\gamma n}{4}\cdot 32\gamma^2 L^2 n^2 \|v_{s-1}\|^2. \end{aligned}$$

Combining alike expressions,

$$\begin{aligned} f(\omega_{s+1}) - f(x^*) + \frac{\gamma n}{4}\|\nabla f(\omega_s)\|^2 \leqslant f(\omega_s) - f(x^*) &- \frac{\gamma n}{8}\left(1 - 32\gamma^2 L^2 n^2\right)\|v_s\|^2 \\ &+ \gamma n \cdot 24\gamma^2 L^2 n^2 \|v_{s-1}\|^2. \end{aligned} \tag{17}$$

Using $\gamma \leqslant \frac{1}{20Ln}$ (note it is the smallest stepsize from all the steps we used before, so all previous transitions are correct), we get

$$\begin{aligned} f(\omega_{s+1}) - f(x^*) + \frac{1}{10}\gamma n\|v_s\|^2 + \frac{\gamma(n+1)}{4}\|\nabla f(\omega_s)\|^2 \\ \leqslant f(\omega_s) - f(x^*) + \frac{1}{10}\gamma n\|v_{s-1}\|^2. \end{aligned}$$

Next, denoting $\Delta_s = f(\omega_{s+1}) - f(x^*) + \frac{1}{10}\gamma n\|v_s\|^2$, we obtain

$$\frac{1}{S}\sum_{s=1}^{S}\|\nabla f(\omega_s)\|^2 \leqslant \frac{4\left[\Delta_0 - \Delta_S\right]}{\gamma n S}.$$

We choose $\varepsilon^2 = \frac{1}{S}\sum_{s=1}^{S}\|\nabla f(\omega_s)\|^2$ as criteria. Hence, to reach ε-accuracy we need $\mathcal{O}\left(\frac{L}{\varepsilon^2}\right)$ epochs and $\mathcal{O}\left(\frac{nL}{\varepsilon^2}\right)$ iterations. Additionally, we note that the oracle complexity of our algorithm is also equal to $\mathcal{O}(\frac{nL}{\varepsilon^2})$, since at each iteration the algorithm computes the stochastic gradient at only two points. This ends the proof. □

C.2 Strongly convex setting

Theorem 8 (**Theorem 2**). *Suppose Assumptions 1, 2(a) hold. Then Algorithm 1 with $\gamma \leqslant \frac{1}{20Ln}$ to reach ε-accuracy, where $\varepsilon = f(x^0_{S+1}) - f(x^*)$, needs*

$$\mathcal{O}\left(\frac{nL}{\mu}\log\frac{1}{\varepsilon}\right) \quad \textit{iterations and oracle calls.}$$

Proof. Under Assumption 2(a), which states that the function is strongly convex, the (PL) condition is automatically satisfied. Therefore,

$$f(\omega_{s+1}) - f(x^*) + \frac{\gamma\mu n}{2}\left(f(\omega_s) - f(x^*)\right) \leqslant f(\omega_{s+1}) - f(x^*) + \frac{\gamma n}{4}\|\nabla f(\omega_s)\|^2.$$

Thus, using (17),

$$\begin{aligned} f(\omega_{s+1}) - f(x^*) + \frac{\gamma\mu n}{2}\left(f(\omega_s) - f(x^*)\right) &\leqslant f(\omega_s) - f(x^*) \\ &- \frac{\gamma n}{8}\left(1 - 32\gamma^2L^2n^2\right)\|v_s\|^2 + \gamma n\cdot 24\gamma^2L^2n^2\|v_{s-1}\|^2. \end{aligned}$$

Using $\gamma \leqslant \frac{1}{20L(n+1)}$ and assuming $n \geqslant 2$, we get

$$\begin{aligned} f(\omega_{s+1}) - f(x^*) + \frac{1}{10}\gamma n\|v_s\|^2 &\leqslant \left(1 - \frac{\gamma\mu n}{2}\right)\left(f(\omega_s) - f(x^*)\right) \\ &+ \frac{1}{10}\gamma n\cdot\left(1 - \frac{\gamma\mu n}{2}\right)\|v_{s-1}\|^2. \end{aligned}$$

Next, denoting $\Delta_s = f(\omega_{s+1}) - f(x^*) + \frac{1}{10}\gamma n\|v_s\|^2$, we obtain the final convergence over one epoch:

$$\Delta_{s+1} \leqslant \left(1 - \frac{\gamma\mu n}{2}\right)\Delta_s.$$

Going into recursion over all epoch,

$$f(\omega_{S+1}) - f(x^*) \leqslant \Delta_S \leqslant \left(1 - \frac{\gamma\mu n}{2}\right)^{S+1}\Delta_0.$$

We choose $\varepsilon = f(\omega_{S+1}) - f(x^*)$ as criteria. Then to reach ε-accuracy we need $\mathcal{O}\left(\frac{L}{\mu}\log\left(\frac{1}{\varepsilon}\right)\right)$ epochs and $\mathcal{O}\left(\frac{nL}{\mu}\log\left(\frac{1}{\varepsilon}\right)\right)$ iterations. Additionally, we note that the oracle complexity of our algorithm is also equal to $\mathcal{O}\left(\frac{nL}{\mu}\log\left(\frac{1}{\varepsilon}\right)\right)$, since at each iteration the algorithm computes the stochastic gradient at only two points. □

D NO FULL GRAD SARAH

For the convenience of the reader, we provide here the short description of Algorithm 2. If we consider it in epoch $s \neq 0$, one can note that the update rule is nothing but

$$\begin{cases} \text{if iteration } t = 0: \\ \quad x^0_s = x^n_{s-1} \\ \quad v^0_s = v_s = \frac{1}{n}\sum_{t=1}^{n}\nabla f_{\pi^t_{s-1}}(x^t_{s-1}) \\ \quad x^1_s = x^0_s - \gamma v^0_s \\ \text{for rest iterations during the epoch}: \\ \quad v^t_s = v^{t-1}_s + \frac{1}{n}\left(\nabla f_{\pi^t_s}(x^t_s) - \nabla f_{\pi^t_s}(\omega_s)\right) \\ \quad x^{t+1}_s = x^t_s - \gamma v^t_s \end{cases} \tag{18}$$

D.1 Non-convex setting

Lemma 8. *Suppose that Assumptions 1, 2 hold. Let the stepsize $\gamma \leqslant \frac{1}{L(n+1)}$. Then for Algorithm 2 it holds*

$$f(x_{s+1}^0) \leqslant f(x_s^0) - \frac{\gamma(n+1)}{2}\|\nabla f(x_s^0)\|^2 + \frac{\gamma(n+1)}{2}\left\|\nabla f(x_s^0) - \frac{1}{n+1}\sum_{i=0}^{n} v_s^i\right\|^2.$$

Proof. Using the iteration of Algorithm 2 (18), we have

$$\begin{aligned}
f(x_{s+1}^0) &= f(x_s^0 - (x_s^0 - x_{s+1}^0)) \\
&\overset{(\text{Lip})}{\leqslant} f(x_s^0) + \langle \nabla f(x_s^0), x_{s+1}^0 - x_s^0\rangle + \frac{L}{2}\|x_{s+1}^0 - x_s^0|^2 \\
&= f(x_s^0) - \gamma(n+1)\left\langle \nabla f(x_s^0), \frac{1}{n+1}\sum_{t=0}^{n} v_s^t\right\rangle + \frac{\gamma^2(n+1)^2 L}{2}\left\|\frac{1}{n+1}\sum_{t=0}^{n} v_s^t\right\|^2 \\
&\overset{(\text{Norm})}{=} f(x_s^0) - \frac{\gamma(n+1)}{2}\left[\|\nabla f(x_s^0)\|^2 + \left\|\frac{1}{n+1}\sum_{t=0}^{n} v_s^t\right\|^2\right. \\
&\quad \left. - \left\|\nabla f(x_s^0) - \frac{1}{n+1}\sum_{t=0}^{n} v_s^t\right\|^2\right] + \frac{\gamma^2(n+1)^2 L}{2}\left\|\frac{1}{n+1}\sum_{t=0}^{n} v_s^t\right\|^2 \\
&= f(x_s^0) - \frac{\gamma(n+1)}{2}\left[\|\nabla f(x_s^0)\|^2 - \left\|\nabla f(x_s^0) - \frac{1}{n+1}\sum_{t=0}^{n} v_s^t\right\|^2\right] \\
&\quad - \frac{\gamma(n+1)}{2}\cdot(1-\gamma(n+1)L)\left\|\frac{1}{n+1}\sum_{t=0}^{n} v_s^t\right\|^2.
\end{aligned}$$

It remains for us to choose $\gamma : \frac{\gamma(n+1)}{2}(1-\gamma(n+1)L) > 0$ and note that such a choice is followed by $\gamma \leqslant \frac{1}{L(n+1)}$. In that way we make the last term is negative and obtain the result of the lemma. □

Now we want to address the last term in the result of Lemma 8. We prove the following lemma.

Lemma 9 (Lemma 3). *Suppose that Assumptions 1, 2 hold. Then for Algorithm 2 a valid estimate is*

$$\left\|\nabla f(x_s^0) - \frac{1}{n+1}\sum_{t=0}^{n} v_s^t\right\|^2 \leqslant 2\|\nabla f(x_s^0) - v_s\|^2 + \frac{2L^2}{n+1}\sum_{t=1}^{n}\|x_s^t - x_s^{t-1}\|^2.$$

Proof. We claim that

$$\sum_{t=k}^{n} v_s^t = \frac{1}{n}\sum_{t=k+1}^{n}(n-t+1)\left(\nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(x_s^{t-1})\right) + (n-k+1)v_s^k. \tag{19}$$

Let us prove this. We use the method of induction. For $k = n$ it is obviously true. We suppose that it is true for some some fixed $k = \widetilde{k} \geqslant 1$ ($k = 0$ is the first index in the epoch, i.e. start of the recursion) and want to prove that it is true for $k = \widetilde{k} - 1$.

$$\begin{aligned}
\sum_{t=\widetilde{k}-1}^{n} v_s^t &= v_s^{\widetilde{k}-1} + \sum_{t=\widetilde{k}}^{n} v_s^t \\
&= v_s^{\widetilde{k}-1} + \frac{1}{n}\sum_{t=\widetilde{k}+1}^{n}(n-t+1)\left(\nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(x_s^{t-1})\right) + (n-\widetilde{k}+1)v_s^{\widetilde{k}} \\
&\overset{(i)}{=} v_s^{\widetilde{k}-1} + \frac{1}{n}\sum_{t=\widetilde{k}+1}^{n}(n-t+1)\left(\nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(x_s^{t-1})\right)
\end{aligned}$$

$$
\begin{aligned}
&+ (n-\widetilde{k}+1)\left(v_s^{\widetilde{k}-1} + \frac{1}{n}\left(\nabla f_{\pi_s^{\widetilde{k}}}(x_s^{\widetilde{k}}) - \nabla f_{\pi_s^{\widetilde{k}}}(x_s^{\widetilde{k}-1})\right)\right) \\
&= \frac{1}{n}\sum_{t=\widetilde{k}}^{n}(n-t+1)\left(\nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(x_s^{t-1})\right) + (n-\widetilde{k}+2)v_s^{\widetilde{k}-1},
\end{aligned}
$$

where equation (i) is correct due to (18) and $\widetilde{k} \geqslant 1$. In that way, the induction step is proven. It means that (19) is valid. We substitute $k=0$ in (19) and, utilizing $v_s^0 = v_s$, get

$$
\sum_{t=0}^{n} v_s^t = \frac{1}{n}\sum_{t=1}^{n}(n-t+1)\left(\nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(x_s^{t-1})\right) + (n+1)v_s. \tag{20}
$$

Hence, estimating the desired term gives

$$
\begin{aligned}
\left\|\nabla f(x_s^0) - \frac{1}{n+1}\sum_{t=0}^{n} v_s^t\right\|^2 &= \frac{1}{(n+1)^2}\left\|(n+1)\nabla f(x_s^0) - \sum_{t=0}^{n} v_s^t\right\|^2 \\
&\overset{(20)}{=} \frac{1}{(n+1)^2}\Bigg\|(n+1)\nabla f(x_s^0) \\
&\quad -\frac{1}{n}\sum_{t=1}^{n}(n-t+1)\left(\nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(x_s^{t-1})\right) \\
&\quad -(n+1)v_s\Bigg\|^2 \\
&\overset{(CS)}{\leqslant} 2\|\nabla f(x_s^0) - v_s\|^2 \\
&\quad + \frac{2}{(n+1)^2}\left\|\frac{1}{n}\sum_{t=1}^{n}(n-t+1)\left(\nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(x_s^{t-1})\right)\right\|^2 \\
&\overset{(i)}{\leqslant} 2\|\nabla f(x_s^0) - v_s\|^2 \\
&\quad + \frac{2}{(n+1)^2}\left\|\sum_{t=1}^{n}\left(\nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(x_s^{t-1})\right)\right\|^2 \\
&\overset{(CS)}{\leqslant} 2\|\nabla f(x_s^0) - v_s\|^2 + \frac{2}{n+1}\sum_{t=1}^{n}\left\|\nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(x_s^{t-1})\right\|^2 \\
&\overset{\text{Ass. 1}}{\leqslant} 2\|\nabla f(x_s^0) - v_s\|^2 + \frac{2L^2}{n+1}\sum_{t=1}^{n}\left\|x_s^t - x_s^{t-1}\right\|^2,
\end{aligned}
$$

where inequality (i) is correct due to $t \geqslant 1$ holds during the summation. The obtained inequality finishes the proof of the lemma. □

Lemma 10 (**Lemma 4**). *Suppose that Assumptions 1, 2 hold. Let the stepsize $\gamma \leqslant \frac{1}{3L}$. Then for Algorithm 2 a valid estimate is*

$$
\left\|\nabla f(x_s^0) - \frac{1}{n+1}\sum_{t=0}^{n} v_s^t\right\|^2 \leqslant 9\gamma^2 L^2\|v_s\|^2 + 36\gamma^2 L^2 n^2\|v_{s-1}\|^2.
$$

Proof. To begin with, in Lemma 1, we obtain

$$
\left\|\nabla f(x_s^0) - \frac{1}{n+1}\sum_{t=0}^{n} v_s^t\right\|^2 \leqslant 2\|\nabla f(x_s^0) - v_s\|^2 + \frac{2L^2}{n+1}\sum_{t=1}^{n}\|x_s^t - x_s^{t-1}\|^2. \tag{21}
$$

Let us show what v_s is (here we use Line 8 of Algorithm 2):

$$\begin{aligned} v_s = \widetilde{v}_{s-1}^{n+1} &= \frac{n-1}{n}\widetilde{v}_{s-1}^{n} + \frac{1}{n}\nabla f_{\pi_{s-1}^n}(x_{s-1}^n) \\ &= \frac{n-1}{n}\cdot\frac{n-2}{n-1}\widetilde{v}_{s-1}^{n-1} + \frac{n-1}{n}\cdot\frac{1}{n-1}\nabla f_{\pi_{s-1}^{n-1}}(x_{s-1}^{n-1}) + \frac{1}{n}\nabla f_{\pi_{s-1}^n}(x_{s-1}^n) \\ &= \frac{n-1}{n}\cdot\frac{n-2}{n-1}\cdot\ldots\cdot 0\cdot\widetilde{v}_{s-1}^{1} + \frac{1}{n}\sum_{t=1}^{n}\nabla f_{\pi_{s-1}^t}(x_{s-1}^t) \\ &\overset{(i)}{=} \frac{1}{n}\sum_{t=1}^{n}\nabla f_{\pi_{s-1}^t}(x_{s-1}^t), \end{aligned} \tag{22}$$

where equation (i) is correct due to initialization $\widetilde{v}_{s-1}^1 = 0$ (Line 13 of Algorithm 2). In that way, using (21) and (22),

$$\begin{aligned} \left\|\nabla f(x_s^0) - \frac{1}{n+1}\sum_{t=0}^{n} v_s^t\right\|^2 \leqslant\ & 2\left\|\nabla f(x_s^0) - \frac{1}{n}\sum_{t=1}^{n}\nabla f_{\pi_{s-1}^t}(x_{s-1}^t)\right\|^2 \\ & + \frac{2L^2}{n+1}\sum_{t=1}^{n}\|x_s^t - x_s^{t-1}\|^2. \end{aligned}$$

Then, using (1),

$$\begin{aligned} \left\|\nabla f(x_s^0) - \frac{1}{n+1}\sum_{t=0}^{n} v_s^t\right\|^2 &\leqslant 2\left\|\frac{1}{n}\sum_{t=1}^{n}\left[\nabla f_{\pi_{s-1}^t}(x_s^0) - \nabla f_{\pi_{s-1}^t}(x_{s-1}^t)\right]\right\|^2 \\ &\quad + \frac{2L^2}{n+1}\sum_{t=1}^{n}\|x_s^t - x_s^{t-1}\|^2 \\ &\overset{\text{(CS),Ass. 1}}{\leqslant} \frac{2L^2}{n}\sum_{t=1}^{n}\|x_{s-1}^t - x_s^0\|^2 + \frac{2L^2}{n+1}\sum_{t=1}^{n}\|x_s^t - x_s^{t-1}\|^2 \\ &\overset{\text{(Quad)}}{\leqslant} \frac{4L^2}{n}\sum_{t=1}^{n}\|x_{s-1}^t - x_{s-1}^0\|^2 + \frac{4L^2}{n}\sum_{t=1}^{n}\|x_s^0 - x_{s-1}^0\|^2 \\ &\quad + \frac{2L^2}{n+1}\sum_{t=1}^{n}\|x_s^t - x_s^{t-1}\|^2. \end{aligned} \tag{23}$$

Now we have to bound these three terms. Let us begin with the $\sum\limits_{t=1}^{n}\|x_s^t - x_s^{t-1}\|^2$ norm.

$$\sum_{t=1}^{n}\|x_s^t - x_s^{t-1}\|^2 = \gamma^2\sum_{t=1}^{n}\|v_s^{t-1}\|^2 = \gamma^2\sum_{t=0}^{n-1}\|v_s^t\|^2. \tag{24}$$

Now we estimate $\|v_s^t\|^2$. For $t \geqslant 1$:

$$\begin{aligned} \|v_s^t\|^2 &= \left\|v_s^{t-1} + \frac{1}{n}\left(\nabla f_{\pi_s^t}(x_s^t) - \nabla f_{\pi_s^t}(x_s^{t-1})\right)\right\|^2 \\ &\overset{\text{(Quad)}}{\leqslant} \left(1+\frac{1}{\beta}\right)\|v_s^{t-1}\|^2 + \frac{(1+\beta)L^2}{n^2}\|x_s^t - x_s^{t-1}\|^2 \\ &\overset{\text{(Quad)}}{\leqslant} \left(1+\frac{1}{\beta}\right)^2\|v_s^{t-2}\|^2 + \frac{1}{n^2}\left(1+\frac{1}{\beta}\right)(1+\beta)L^2\|x_s^{t-1} - x_s^{t-2}\|^2 \\ &\quad + \frac{1}{n^2}(1+\beta)L^2\|x_s^t - x_s^{t-1}\|^2 \\ &\overset{\text{(Quad)}}{\leqslant} \left(1+\frac{1}{\beta}\right)^t\|v_s\|^2 + \frac{1}{n^2}(1+\beta)L^2\sum_{k=1}^{t}\left(1+\frac{1}{\beta}\right)^{k-1}\|x_s^{t-k+1} - x_s^{t-k}\|^2 \end{aligned}$$

$$\underset{\beta=t}{\overset{(\text{Quad})}{\leqslant}} \left(1+\frac{1}{t}\right)^t \|v_s\|^2 + \frac{1}{n^2}(1+t)\left(1+\frac{1}{t}\right)^t L^2 \sum_{k=1}^{t} \|x_s^k - x_s^{k-1}\|^2.$$

Then, utilizing the property of the exponent $\left(\left(1+\frac{1}{t}\right)^t \leqslant e\right)$ and $t \leqslant n-1$ (24), we get an important inequality (for $0 \leqslant t \leqslant n-1$, since for $t=0$ we have $\|v_s^t\|^2 = \|v_s\|^2$ and desired inequality becomes trivial):

$$\|v_s^t\|^2 \leqslant e\|v_s\|^2 + \frac{eL^2}{n}\sum_{k=1}^{t}\|x_s^k - x_s^{k-1}\|^2. \tag{25}$$

Now we substitute (25) to (24) and obtain

$$\begin{aligned}\sum_{t=1}^{n}\|x_s^t - x_s^{t-1}\|^2 &= \gamma^2\sum_{t=0}^{n-1}\|v_s^t\|^2 \leqslant e\gamma^2 n\|v_s\|^2 + \frac{e\gamma^2L^2}{n}\sum_{t=0}^{n-1}\sum_{k=1}^{t}\|x_s^k - x_s^{k-1}\|^2\\ &\leqslant e\gamma^2 n\|v_s\|^2 + e\gamma^2L^2\sum_{t=1}^{n-1}\|x_s^t - x_s^{t-1}\|^2\\ &\leqslant e\gamma^2 n\|v_s\|^2 + e\gamma^2L^2\sum_{t=1}^{n}\|x_s^t - x_s^{t-1}\|^2.\end{aligned}$$

Straightforwardly expressing $\sum\limits_{t=1}^{n}\|x_s^t - x_s^{t-1}\|^2$ we obtain desired estimation:

$$\sum_{t=1}^{n}\|x_s^t - x_s^{t-1}\|^2 \leqslant \frac{e\gamma^2 n\|v_s\|^2}{1-e\gamma^2L^2} \overset{e<3}{\leqslant} \frac{3\gamma^2 n\|v_s\|^2}{1-3\gamma^2L^2}.$$

To finish this part of proof it remains for us to choose appropriate γ. In Lemma 8 we require $\gamma \leqslant \frac{1}{L(n+1)}$. There we are satisfied with even large values of γ. Let us estimate obtained expression with $\gamma \leqslant \frac{1}{L(n+1)} \leqslant \frac{1}{3L}$. Now we provide final estimation of this norm:

$$\sum_{t=1}^{n}\|x_s^t - x_s^{t-1}\|^2 \leqslant \frac{9}{2}\gamma^2 n\|v_s\|^2. \tag{26}$$

Let us proceed our estimation of (23) with the $\sum\limits_{t=1}^{n}\|x_{s-1}^t - x_{s-1}^0\|^2$ term.

$$\sum_{t=1}^{n}\|x_{s-1}^t - x_{s-1}^0\|^2 = \gamma^2\sum_{t=1}^{n}\left\|\sum_{k=0}^{t-1}v_{s-1}^k\right\|^2 \overset{(\text{CS})}{\leqslant} \gamma^2\sum_{t=1}^{n}t\sum_{k=0}^{t-1}\|v_{s-1}^k\|^2 \leqslant \gamma^2n^2\sum_{t=0}^{n-1}\|v_{s-1}^t\|^2. \tag{27}$$

Note, that we have already estimated $\|v_s^t\|^2$ term for $0 \leqslant t \leqslant n-1$ (25). Furthermore, we can make the same estimate for the terms in the $(s-1)$-th epoch and write

$$\|v_{s-1}^t\|^2 \leqslant e\|v_{s-1}\|^2 + \frac{eL^2}{n}\sum_{k=1}^{t}\|x_{s-1}^k - x_{s-1}^{k-1}\|^2. \tag{28}$$

Now we substitute (28) to (27) to obtain

$$\begin{aligned}\sum_{t=1}^{n}\|x_{s-1}^t - x_{s-1}^0\|^2 &\leqslant \gamma^2n^2\sum_{t=0}^{n-1}\left(e\|v_{s-1}\|^2 + \frac{eL^2}{n}\sum_{k=1}^{t}\|x_{s-1}^k - x_{s-1}^{k-1}\|^2\right)\\ &\leqslant \gamma^2n^3e\|v_{s-1}\|^2 + e\gamma^2L^2n\sum_{t=0}^{n-1}\sum_{k=1}^{t}\|x_{s-1}^k - x_{s-1}^{k-1}\|^2\\ &\leqslant \gamma^2n^3e\|v_{s-1}\|^2 + e\gamma^2L^2n^2\sum_{t=1}^{n-1}\|x_{s-1}^t - x_{s-1}^{t-1}\|^2\end{aligned}$$

$$\leqslant \gamma^2 n^3 e \|v_{s-1}\|^2 + e\gamma^2 L^2 n^2 \sum_{t=1}^{n} \|x_{s-1}^t - x_{s-1}^{t-1}\|^2. \tag{29}$$

Note, that we have already estimated the $\sum\limits_{t=1}^{n} \|x_s^t - x_s^{t-1}\|^2$ term (26). Furthermore, we can make the same estimate for the term in the $(s-1)$-th epoch and write

$$\sum_{t=1}^{n} \|x_{s-1}^t - x_{s-1}^{t-1}\|^2 \leqslant \frac{9}{2}\gamma^2 n \|v_{s-1}\|^2. \tag{30}$$

Substituting (30) to (29) we derive

$$\sum_{t=1}^{n} \|x_{s-1}^t - x_{s-1}^0\|^2 \leqslant 3\gamma^2 n^3 \|v_{s-1}\|^2 + \frac{27}{2}\gamma^4 L^2 n^3 \|v_{s-1}\|^2.$$

Using our $\gamma \leqslant \frac{1}{3L}$ choice,

$$\sum_{t=1}^{n} \|x_{s-1}^t - x_{s-1}^0\|^2 \leqslant 3\gamma^2 n^3 \|v_{s-1}\|^2 + \frac{3}{2}\gamma^2 n^3 \|v_{s-1}\|^2 = \frac{9}{2}\gamma^2 n^3 \|v_{s-1}\|^2. \tag{31}$$

It remains for us to estimate the $\sum\limits_{t=1}^{n} \|x_s^0 - x_{s-1}^0\|^2$ term. The estimate is quite similar to the previous one:

$$\sum_{t=1}^{n} \|x_s^0 - x_{s-1}^0\|^2 = \gamma^2 \sum_{t=1}^{n} \left\| \sum_{k=0}^{n-1} v_{s-1}^{k-1} \right\|^2 \overset{\text{(Quad)}}{\leqslant} \gamma^2 \sum_{t=1}^{n} n \sum_{k=0}^{n-1} \|v_{s-1}^k\|^2 \leqslant \gamma^2 n^2 \sum_{t=0}^{n-1} \|v_{s-1}^t\|^2.$$

We obtain the estimate as in (27). Thus, proceed similarly as we did for the previous term, we obtain

$$\sum_{t=1}^{n} \|x_s^0 - x_{s-1}^0\|^2 \leqslant \frac{9}{2}\gamma^2 n^3 \|v_{s-1}\|^2. \tag{32}$$

Now we can apply the upper bounds obtained in (26), (31), (32) to (23) and have

$$\begin{aligned} \left\| \nabla f(\omega_s) - \frac{1}{n+1} \sum_{i=0}^{n} v_s^i \right\|^2 &\leqslant 18\gamma^2 L^2 n^2 \|v_{s-1}\|^2 + 18\gamma^2 L^2 n^2 \|v_{s-1}\|^2 + 9\gamma^2 L^2 \|v_s\|^2 \\ &= 9\gamma^2 L^2 \|v_s\|^2 + 36\gamma^2 L^2 n^2 \|v_{s-1}\|^2, \end{aligned}$$

which ends the proof. □

Theorem 9 (**Theorem 3**). *Suppose Assumptions 1, 2(b) hold. Then Algorithm 2 with $\gamma \leqslant \frac{1}{20L(n+1)}$ to reach ε-accuracy, where $\varepsilon^2 = \frac{1}{S}\sum\limits_{s=1}^{S} \|\nabla f(x_s^0)\|^2$, needs*

$$\mathcal{O}\left(\frac{nL}{\varepsilon^2}\right) \quad \textit{iterations and oracle calls.}$$

Proof. We combine the result of Lemma 8 with the result of Lemma 10 and obtain

$$\begin{aligned} f(x_{s+1}^0) \leqslant f(x_s^0) &- \frac{\gamma(n+1)}{2} \|\nabla f(x_s^0)\|^2 \\ &+ \frac{\gamma(n+1)}{2} \left(9\gamma^2 L^2 \|v_s\|^2 + 36\gamma^2 L^2 n^2 \|v_{s-1}\|^2\right). \end{aligned}$$

We subtract $f(x^*)$ from both parts:

$$f(x_{s+1}^0) - f(x^*) \leqslant f(x_s^0) - f(x^*) - \frac{\gamma(n+1)}{2} \|\nabla f(x_s^0)\|^2$$

$$
\begin{aligned}
&+ \frac{\gamma(n+1)}{2}\left(9\gamma^2L^2\|v_s\|^2 + 36\gamma^2L^2n^2\|v_{s-1}\|^2\right)\\
&= f(x_s^0) - f(x^*) - \frac{\gamma(n+1)}{4}\|\nabla f(x_s^0)\|^2\\
&+ \frac{\gamma(n+1)}{2}\left(9\gamma^2L^2\|v_s\|^2 + 36\gamma^2L^2n^2\|v_{s-1}\|^2\right)\\
&- \frac{\gamma(n+1)}{4}\|\nabla f(x_s^0)\|^2.
\end{aligned}
$$

Then, transforming the last term by using (Quad) with $\beta = 1$, we get

$$
\begin{aligned}
f(x_{s+1}^0) - f(x^*) \leqslant f(x_s^0) - f(x^*) &- \frac{\gamma(n+1)}{4}\|\nabla f(x_s^0)\|^2\\
&+ \frac{\gamma(n+1)}{2}\left(9\gamma^2L^2\|v_s\|^2 + 36\gamma^2L^2n^2\|v_{s-1}\|^2\right)\\
&- \frac{\gamma(n+1)}{8}\|v_s\|^2 + \frac{\gamma(n+1)}{4}\|v_s - \nabla f(x_s^0)\|^2.
\end{aligned}
$$

Using Lemma 10 to $\|v_s - \nabla f(x_s^0)\|^2$ (specially $\frac{4L^2}{n}\cdot(31) + \frac{4L^2}{n}\cdot(32)$),

$$
\begin{aligned}
f(x_{s+1}^0) - f(x^*) \leqslant f(x_s^0) - f(x^*) &- \frac{\gamma(n+1)}{4}\|\nabla f(x_s^0)\|^2\\
&+ \frac{\gamma(n+1)}{2}\left(9\gamma^2L^2\|v_s\|^2 + 36\gamma^2L^2n^2\|v_{s-1}\|^2\right)\\
&- \frac{\gamma(n+1)}{8}\|v_s\|^2 + \frac{\gamma(n+1)}{4}\cdot 36\gamma^2L^2n^2\|v_{s-1}\|^2.
\end{aligned}
$$

Combining alike expressions,

$$
\begin{aligned}
f(x_{s+1}^0) - f(x^*) + \frac{\gamma(n+1)}{4}\|\nabla f(x_s^0)\|^2 \leqslant f(x_s^0) - f(x^*) &- \frac{\gamma(n+1)}{8}\left(1 - 36\gamma^2L^2\right)\|v_s\|^2\\
&+ \gamma(n+1)\cdot 27\gamma^2L^2n^2\|v_{s-1}\|^2. \qquad (33)
\end{aligned}
$$

Using $\gamma \leqslant \frac{1}{20L(n+1)}$ (note it is the smallest stepsize from all the steps we used before, so all previous transitions are correct), we get

$$
\begin{aligned}
f(x_{s+1}^0) - f(x^*) + \frac{1}{10}\gamma(n+1)\|v_s\|^2 + \frac{\gamma(n+1)}{4}\|\nabla f(\omega_s)\|^2\\
\leqslant f(x_s^0) - f(x^*) + \frac{1}{10}\gamma(n+1)\|v_{s-1}\|^2.
\end{aligned}
$$

Next, denoting $\Delta_s = f(x_{s+1}^0) - f(x^*) + \frac{1}{10}\gamma(n+1)\|v_s\|^2$, we obtain

$$
\frac{1}{S}\sum_{s=1}^{S}\|\nabla f(x_s^0)\|^2 \leqslant \frac{4\left[\Delta_0 - \Delta_S\right]}{\gamma(n+1)S}.
$$

We choose $\varepsilon^2 = \frac{1}{S}\sum\limits_{s=1}^{S}\|\nabla f(x_s^0)\|^2$ as criteria. Hence, to reach ε-accuracy we need $\mathcal{O}\left(\frac{L}{\varepsilon^2}\right)$ epochs and $\mathcal{O}\left(\frac{nL}{\varepsilon^2}\right)$ iterations. Additionally, we note that the oracle complexity of our algorithm is also equal to $\mathcal{O}(\frac{nL}{\varepsilon^2})$, since at each iteration the algorithm computes the stochastic gradient at only two points. This ends the proof. □

D.2 Strongly convex setting

Theorem 10 (**Theorem 4**). *Suppose Assumptions 1, 2(a) hold. Then Algorithm 2 with $\gamma \leqslant \frac{1}{20L(n+1)}$ to reach ε-accuracy, where $\varepsilon = f(x_{S+1}^0) - f(x^*)$, needs*

$$
\mathcal{O}\left(\frac{nL}{\mu}\log\frac{1}{\varepsilon}\right) \quad \textit{iterations and oracle calls.}
$$

Proof. Under Assumption 2(a), which states that the function is strongly convex, the (PL) condition is automatically satisfied. Therefore,

$$f(x_{s+1}^0) - f(x^*) + \frac{\gamma\mu(n+1)}{2}\left(f(x_s^0) - f(x^*)\right) \leqslant f(x_{s+1}^0) - f(x^*) + \frac{\gamma(n+1)}{4}\|\nabla f(x_s^0)\|^2.$$

Thus, using (33),

$$\begin{aligned} f(x_{s+1}^0) - f(x^*) + \frac{\gamma\mu(n+1)}{2}\left(f(x_s^0) - f(x^*)\right) &\leqslant f(x_s^0) - f(x^*) \\ &- \frac{\gamma(n+1)}{8}\left(1 - 36\gamma^2L^2\right)\|v_s\|^2 + \gamma(n+1)\cdot 27\gamma^2L^2n^2\|v_{s-1}\|^2. \end{aligned}$$

Using $\gamma \leqslant \frac{1}{20L(n+1)}$ and assuming $n \geqslant 2$, we get

$$\begin{aligned} f(x_{s+1}^0) - f(x^*) + \frac{1}{10}\gamma(n+1)\|v_s\|^2 &\leqslant \left(1 - \frac{\gamma\mu(n+1)}{2}\right)(f(\omega_s) - f(x^*)) \\ &+ \frac{1}{10}\gamma(n+1)\cdot\left(1 - \frac{\gamma\mu(n+1)}{2}\right)\|v_{s-1}\|^2. \end{aligned}$$

Next, denoting $\Delta_s = f(x_{s+1}^0) - f(x^*) + \frac{1}{10}\gamma(n+1)\|v_s\|^2$, we obtain the final convergence over one epoch:

$$\Delta_{s+1} \leqslant \left(1 - \frac{\gamma\mu(n+1)}{2}\right)\Delta_s.$$

Going into recursion over all epoch,

$$f(x_{S+1}^0) - f(x^*) \leqslant \Delta_S \leqslant \left(1 - \frac{\gamma\mu(n+1)}{2}\right)^{S+1}\Delta_0.$$

We choose $\varepsilon = f(x_{S+1}^0) - f(x^*)$ as criteria. Then to reach ε-accuracy we need $\mathcal{O}\left(\frac{L}{\mu}\log\left(\frac{1}{\varepsilon}\right)\right)$ epochs and $\mathcal{O}\left(\frac{nL}{\mu}\log\left(\frac{1}{\varepsilon}\right)\right)$ iterations. Additionally, we note that the oracle complexity of our algorithm is also equal to $\mathcal{O}\left(\frac{nL}{\mu}\log\left(\frac{1}{\varepsilon}\right)\right)$, since at each iteration the algorithm computes the stochastic gradient at only two points. □

E LOWER BOUNDS

In this section we provide the proof of the lower bound on the amount of oracle calls in the class of the first-order algorithms with shuffling heuristic that find the solution of the non-convex objective finite-sum function. We follow the classical way by presenting the example of function and showing the minimal number of oracles needs to solve the problem. We consider the following function:

$$l(x) = -\Psi(1)\Phi([x]_1) + \sum_{j=2}^{d}\left(\Psi(-[x]_{j-1})\Phi(-[x]_j) - \Psi([x]_{j-1})\Phi([x]_j)\right),$$

where $[x]_j$ is the j-th coordinate of the vector $x \in \mathbb{R}^d$,

$$\Psi(z) = \begin{cases} 0, & \text{if } z \leqslant \frac{1}{2} \\ \exp\left(1 - \frac{1}{(2z-1)^2}\right), & \text{if } z > \frac{1}{2} \end{cases}$$

and

$$\Phi(z) = \sqrt{e}\int_{-\infty}^{z}\exp\left(-\frac{t^2}{2}\right)dt.$$

We also define the following function:

$$\operatorname{prog}(x) = \begin{cases} 0, & \text{if } x = 0 \\ \max\limits_{1 \leqslant j \leqslant d} \{j : [x]_j \neq 0\}, & \text{otherwise} \end{cases},$$

where $x \in \mathbb{R}^d$. In the work [Arjevani et al., 2023] it was shown, that function $l(x)$ satisfies the following properties:

$$\begin{aligned}
&\forall x \in \mathbb{R}^d \quad l(0) - \inf_x l(x) \leqslant \Delta_0 d, \\
&l(x) \text{ is } L_0\text{-smooth with } L_0 = 152, \\
&\forall x \in \mathbb{R}^d \quad \|l(x)\|_\infty \leqslant G_0 \text{ with } G_0 = 23, \\
&\forall x \in \mathbb{R}^d : [x]_d = 0 \quad \|l(x)\|_\infty \geqslant 1, \\
&l(x) \text{ is the zero-chain function, i.e., } \operatorname{prog}(\nabla l(x)) \leqslant \operatorname{prog}(x) + 1.
\end{aligned}$$

Lemma 11 (Lemma C.9 from [Metelev et al., 2024]). *Each $l_j(x)$ is L_0-smooth, where*

$$l_j(x) = \begin{cases} -\Psi(1)\Phi([x]_1), & \textit{if } j = 1 \\ \Psi(-[x]_{j-1})\Phi(-[x]_j) - \Psi([x]_{j-1})\Phi([x]_j), & \textit{otherwise} \end{cases}.$$

E.1 Proof of Theorem 5

Theorem 11 (**Theorem 5**). *For any $L > 0$ there exists a problem (1) which satisfies Assumption 1, such that for any output of first-order algorithm, number of oracle calls N_c required to reach ε-accuracy is lower bounded as*

$$N_c = \Omega\left(\frac{L\Delta}{\varepsilon^2}\right).$$

Proof. To begin with, we need to decompose the function $l(x)$ to the finite-sum:

$$l(x) = \sum_{j=1}^{d} l_j(x),$$

where index j responds the definition of $l(x)$, i.e.,

$$l_j(x) = \begin{cases} -\Psi(1)\Phi([x]_1), & \text{if } j = 1 \\ \Psi(-[x]_{j-1})\Phi(-[x]_j) - \Psi([x]_{j-1})\Phi([x]_j), & \text{otherwise} \end{cases}.$$

Now we design the following objective function:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x),$$

where $f_i(x) = \frac{LC^2}{L_0} \sum\limits_{j \equiv i \mod n} l_j\left(\frac{x}{C}\right)$. Since each $l_j(\cdot)$ is L_0-smooth (according to Lemma 11) and for $j \equiv i \mod n$ the gradients $\nabla l_j(x)$ are separable, than for any $x_1, x_2 \in \mathbb{R}^d$ it implies

$$\begin{aligned}
\|\nabla f_i(x_1) - \nabla f_i(x_2)\|^2 &= \frac{L^2C^2}{L_0^2}\left\| \sum_{j \equiv i \mod n} \left(\nabla l_j\left(\frac{x_1}{C}\right) - \nabla l_j\left(\frac{x_2}{C}\right)\right)\right\|^2 \\
&\leqslant \frac{L^2C^2}{L_0^2}\frac{L_0^2}{C^2}\|x_1 - x_2\|^2 = L^2\|x_1 - x_2\|^2.
\end{aligned}$$

It means, each function $f_i(x)$ is L-smooth. Moreover, since $f(x) = \frac{LC^2}{nL_0} l\left(\frac{x}{C}\right)$,

$$\Delta = f(0) - \inf_x f(x) = \frac{LC^2}{nL_0}\left(l(0) - \inf_x l\left(\frac{x}{C}\right)\right)$$

$$= \quad \frac{LC^2}{nL_0}\left(l(0) - \inf_x l(x)\right) \leqslant \frac{LC^2\Delta_0 d}{nL_0}. \tag{34}$$

Now we show, how many oracle calls we need to have progress in one coordinate fo vector x. At the current moment, we need a specific piece of function, because according to structure of $l(x)$, each gradient estimation can "defreeze" at most one component and only a computation on a certain block makes it possible. Formally, since $\frac{1}{n}\sum\limits_{i=1}^{n} f_i(x) = \frac{LC}{dL_0} l\left(\frac{x}{C}\right)$,

$$\text{prog}(\nabla f_i(x)) \begin{cases} = \text{prog}(x) + 1, & \text{if } i = \text{prog}(x) \mod n \\ \leqslant \text{prog}(x), & \text{otherwise} \end{cases}.$$

Now, we need to show the probability of choosing the necessary piece of function, according to the shuffling heuristic. This probability at the first iteration of the epoch, i.e., iteration t, such that $t \mod n = 1$, is obviously $\frac{1}{n}$. At the second iteration of the epoch – $\frac{n-1}{n} \cdot \frac{1}{n-1} = \frac{1}{n}$. Thus, at the k-th iteration of the epoch, the desired probably is $\frac{n-1}{n} \cdot \frac{n-2}{n-1} \cdot \ldots \cdot \frac{1}{n-k+1} = \frac{1}{n}$. In that way, the expected amount of gradient calculations though the epoch is

$$\sum_{i=1}^{n} \frac{i}{n} = \frac{n+1}{2} \leqslant n.$$

Since epochs is symmetrical in a sense of choosing indices, we need to perform n oracle calls at each moment of training. Thus, after T oracle calls, we can change only $\frac{T}{n}$ coordinate of vector x. Now, we can write the final estimate:

$$\begin{aligned} \mathbb{E}\|\nabla f(\hat{x})\|_2^2 \quad &\geqslant \quad \mathbb{E}\|\nabla f(\hat{x})\|_\infty^2 \geqslant \min_{[x]_d=0} \|\nabla f(\hat{x})\|_\infty^2 = \frac{L^2C^2}{n^2L_0^2}\left\|\nabla l\left(\frac{\hat{x}}{C}\right)\right\|_\infty^2 \geqslant \frac{L^2C^2}{n^2L_0^2} \\ &\overset{(34)}{\geqslant} \quad \frac{L\Delta}{nL_0\Delta_0 d} = \frac{L\Delta}{L_0\Delta_0 T}. \end{aligned}$$

Thus, lower bound on T is $\Omega\left(\frac{L\Delta}{\varepsilon^2}\right)$. □

E.2 Proof of Theorem 6

Before we start the proof, let us introduce other assumptions of smoothness for the complete analysis.

Assumption 3 (Smoothness of each f_i)**.** *Each function f_i is L_i-smooth, i.e., it satisfies*

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L_i\|x - y\|$$

for any $x, y \in \mathbb{R}^d$.

Assumption 4 (Average smoothness of f)**.** *Function f is $\hat{L}$-average smooth, i.e., it satisfies*

$$\mathbb{E}_i\left[\|\nabla f_i(x) - \nabla f_i(y)\|^2\right] \leq \hat{L}^2\|x - y\|^2$$

for any $x, y \in \mathbb{R}^d$.

Here we also assume that L_i with $i = 1, \ldots, n$ and $\hat{L}$ are *effective*: it means that these constants cannot be reduced.

If $\{f_i\}_{i=1}^n$ satisfies Assumption 1, it automatically leads to the satisfaction of Assumption 3, since L_i can be chosen as L. Nevertheless, the effective constant of smoothness for f_i can be less than L. As a consequence, we obtain the next result.

Lemma 12. *Suppose that Assumption 1 holds. Then, the set $\{f_i\}_{i=1}^n$ satisfies Assumptions 3 and 4. Moreover,*

$$\hat{L} \leq L,$$

where $\hat{L}$ and L are chosen effectively.

Proof. Let L_i be the constant of smoothness of f_i. Therefore, $L_i \leq L$, and L is defined as $\max_i L_i$. Moreover, $\hat{L}^2$ is defined as

$$\hat{L}^2 = \sum_{i=1}^{n} w_i L_i^2,$$

where $\{w_i\}_{i=1}^n$ is probabilities for the sampling of f_i, i.e. $w = (w_1, \ldots, w_n)$ formalizes the discrete distribution over indices i (the most common case: $w_i = \frac{1}{n}$; nevertheless, we consider an unified option). As a result, we have

$$\hat{L}^2 = \sum_{i=1}^{n} w_i L_i^2 \leq \sum_{i=1}^{n} w_i L^2 = L^2.$$

This concludes the proof. □

Now we are ready to proof the Theorem 6.

Theorem 12 (**Theorem 6**)**.** *For any $L > 0$ there is **no** problem (1) which satisfies Assumption 1, such that for any output of first-order algorithm, number of oracle calls N_c required to reach ε-accuracy is lower bounded with $p > \frac{1}{2}$:*

$$N_c = \Omega\left(\frac{n^p L \Delta}{\varepsilon^2}\right).$$

Proof. Let us assume that we can find the problem (1) which satisfies Assumption 1, such that for any output of first-order algorithm, number of oracle calls N_c required to reach ε-accuracy is lower bounded as

$$N_c = \Omega\left(\frac{n^p L \Delta}{\varepsilon^2}\right)$$

with $p > \frac{1}{2}$. Applying Lemma 12, one can obtain

$$N_c = \Omega\left(\frac{n^p L \Delta}{\varepsilon^2}\right) \geq \Omega\left(\frac{n^p \hat{L} \Delta}{\varepsilon^2}\right),$$

which contradict existing results of upper bound in terms of n under Assumption 4 (e.g. [Fang et al., 2018]). This finishes the proof. □