Journal of Machine Learning Research 21 (2020) 1-63 Submitted 7/18; Revised 4/20; Published 5/20

Stochastic Nested Variance Reduction for Nonconvex Optimization

Dongruo Zhou DRZHOU@CS.UCLA.EDU
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095, USA

Pan Xu PANXU@CS.UCLA.EDU
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095, USA

Quanquan Gu QGU@CS.UCLA.EDU
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095, USA

Editor: Animashree Anandkumar

Abstract

We study nonconvex optimization problems, where the objective function is either an average of n nonconvex functions or the expectation of some stochastic function. We propose a new stochastic gradient descent algorithm based on nested variance reduction, namely, Stochastic Nested Variance-Reduced Gradient descent (SNVRG). Compared with conventional stochastic variance reduced gradient (SVRG) algorithm that uses two reference points to construct a semi-stochastic gradient with diminishing variance in each iteration, our algorithm uses $K+1$ nested reference points to build a semi-stochastic gradient to further reduce its variance in each iteration. For smooth nonconvex functions, SNVRG converges to an ϵ-approximate first-order stationary point within $\widetilde{O}(n \wedge \epsilon^{-2} + \epsilon^{-3} \wedge n^{1/2}\epsilon^{-2})$[1] number of stochastic gradient evaluations. This improves the best known gradient complexity of SVRG $O(n + n^{2/3}\epsilon^{-2})$ and that of SCSG $O(n \wedge \epsilon^{-2} + \epsilon^{-10/3} \wedge n^{2/3}\epsilon^{-2})$. For gradient dominated functions, SNVRG also achieves better gradient complexity than the state-of-the-art algorithms.

Based on SNVRG, we further propose two algorithms that can find local minima faster than state-of-the-art algorithms in both finite-sum and general stochastic (online) nonconvex optimization. In particular, for finite-sum optimization problems, the proposed SNVRG + Neon2$^{\text{finite}}$ algorithm achieves $\widetilde{O}(n^{1/2}\epsilon^{-2} + n\epsilon_H^{-3} + n^{3/4}\epsilon_H^{-7/2})$ gradient complexity to converge to an (ϵ, ϵ_H)-second-order stationary point, which outperforms SVRG+Neon2$^{\text{finite}}$ (Allen-Zhu and Li, 2018), the best existing algorithm, in a wide regime. For general stochastic optimization problems, the proposed SNVRG + Neon2$^{\text{online}}$ achieves $\widetilde{O}(\epsilon^{-3} + \epsilon_H^{-5} + \epsilon^{-2}\epsilon_H^{-3})$ gradient complexity, which is better than both SVRG + Neon2$^{\text{online}}$ (Allen-Zhu and Li, 2018) and Natasha2 (Allen-Zhu, 2018a) in certain regimes. Thorough experimental results on different nonconvex optimization problems back up our theory.

Keywords: Nonconvex Optimization, Finding Local Minima, Variance Reduction

1. $\widetilde{O}(\cdot)$ hides the logarithmic factors, and $a \wedge b$ means $\min(a, b)$.

1. Introduction

We study the following nonconvex optimization problem: $\min_{\mathbf{x}\in\mathbb{R}^d} F(\mathbf{x})$, where F is a nonconvex smooth function. A popular example of this problem is the finite-sum optimization, where the loss function is a sum of n nonconvex component functions:

$$\min_{\mathbf{x}\in\mathbb{R}^d} F(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}), \tag{1}$$

where each f_i is defined on a different data point. The finite-sum optimization problem (1) is often regarded as the offline learning setting in the literature (Allen-Zhu and Li, 2018; Fang et al., 2018). A closely related variant of the finite-sum optimization problem in (1) is the following general stochastic optimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^d} F(\mathbf{x}) = \mathbb{E}_{\xi\sim\mathcal{D}}[F(\mathbf{x};\xi)], \tag{2}$$

where ξ is a random variable drawn from some fixed but unknown distribution $\mathcal{D}$ and $F(\mathbf{x};\xi)$ is a nonconvex smooth function indexed by ξ. The general stochastic optimization problem defined in (2) encloses innumerable large-scale machine learning applications which keep generating oceans of data samples. Therefore, (2) is also referred to as the online learning setting (Allen-Zhu and Li, 2018).

For either (1) or (2), finding the global minimum of such nonconvex optimization problems can be generally NP hard (Hillar and Lim, 2013). Therefore, instead of finding the global minimum, various optimization methods have been developed to find an ϵ-approximate first-order stationary point of (1) and (2), i.e., a point $\mathbf{x}$ satisfying $\|\nabla F(\mathbf{x})\|_2 \leq \epsilon$, where $\epsilon > 0$ is a predefined precision parameter. This vast body of literature consists of gradient descent (GD), stochastic gradient descent (SGD) (Robbins and Monro, 1951), stochastic variance reduced gradient (SVRG) (Reddi et al., 2016a; Allen-Zhu and Hazan, 2016), StochAstic Recursive grAdient algoritHm (SARAH) (Nguyen et al., 2017b) and stochastically controlled stochastic gradient (SCSG) (Lei et al., 2017). Among all the aforementioned first-order methods, the stochastically controlled stochastic gradient (SCSG) proposed by Lei et al. (2017) achieves the lowest gradient complexity[2] $O(n \wedge \epsilon^{-2} + \epsilon^{-10/3} \wedge (n^{2/3}\epsilon^{-2}))$, which, to the best of our knowledge, is the state-of-the-art gradient complexity under the smoothness (i.e., gradient Lipschitzness) and bounded stochastic gradient variance assumptions. The key idea behind variance reduction is that the gradient complexity can be saved if the algorithm use history information as *reference*. For instance, the representative variance reduction method SVRG is based on a semi-stochastic gradient that is defined by two reference points. Since the the variance of this semi-stochastic gradient will diminish when the iterate gets closer to the minimizer, it therefore accelerates the convergence of stochastic gradient method. A natural and long standing question is:

Is there still room for improvement in nonconvex finite-sum optimization without making additional assumptions beyond smoothness and bounded stochastic gradient variance?

2. We usually use gradient complexity, the number of stochastic gradient evaluations, to measure the convergence speed of different first-order algorithms.

In this paper, we provide an affirmative answer to the above question, by showing that the dependence on n in the gradient complexity of SVRG (Reddi et al., 2016a; Allen-Zhu and Hazan, 2016) and SCSG (Lei et al., 2017) can be further reduced. We propose a novel algorithm namely Stochastic Nested Variance-Reduced Gradient descent (SNVRG). Similar to SVRG and SCSG, our proposed algorithm works in a multi-epoch way. Nevertheless, the technique we developed is highly nontrivial. At the core of our algorithm is the multiple reference points-based variance reduction technique in each iteration. In detail, inspired by SVRG and SCSG, which uses two reference points to construct a semi-stochastic gradient with diminishing variance, our algorithm uses $K+1$ reference points to construct a semi-stochastic gradient, whose variance decays faster than that of the semi-stochastic gradient used in SVRG and SCSG.

Due to the nonconvexity of the objective function $F(\mathbf{x})$, first-order stationary points are not always satisfying since they can be saddle points and even local maxima. To avoid such unsatisfactory stationary points, one can further pursue an (ϵ, ϵ_H)-approximate second-order stationary point (Nesterov and Polyak, 2006) of (1) and (2), namely a point $\mathbf{x}$ that satisfies

$$\|\nabla F(\mathbf{x})\|_2 \leq \epsilon, \text{ and } \lambda_{\min}\big(\nabla^2 F(\mathbf{x})\big) \geq -\epsilon_H, \tag{3}$$

where $\epsilon, \epsilon_H \in (0,1)$ are predefined precision parameters and $\lambda_{\min}(\cdot)$ denotes the minimum eigenvalue of a matrix. An $(\epsilon, \sqrt{\epsilon})$-approximate second-order stationary point is considered as an approximate local minimum of the optimization problem (Nesterov and Polyak, 2006). In many tasks such as training a deep neural network, matrix completion and matrix sensing, one have found that local minima have a very good generalization performance (Choromanska et al., 2015; Dauphin et al., 2014) or all local minima are global minima (Ge et al., 2016; Bhojanapalli et al., 2016; Zhang et al., 2018). Although it has been proved that first-order method such as GD can converge to local minima asymptotically (Lee et al., 2016, 2019), there is no result in the literature that establishes the convergence rate of vanilla GD/SGD algorithms to local minima. Recently, there has emerged a large body of work (Xu et al., 2018b; Allen-Zhu and Li, 2018; Jin et al., 2018; Daneshmand et al., 2018) that only use first-order oracles to find the negative curvature direction. Specifically, Xu et al. (2018b) proposed a negative curvature originated from noise (NEON) algorithm that can extract the negative curvature direction based on gradient evaluation, which saves Hessian-vector computation. Later, Allen-Zhu and Li (2018) proposed a Neon2 algorithm, which further reduces the number of (stochastic) gradient evaluations required by NEON. Equipped with NEON and Neon2, many aforementioned algorithms such as GD, SGD, SVRG, SCSG for finding the first-order stationary point can be turned into local minimum finding ones (Xu et al., 2018b; Allen-Zhu and Li, 2018; Yu et al., 2017, 2018).

Based on the SNVRG algorithm we proposed for finding the first-order stationary point in nonconvex optimization, we take a step further to propose faster algorithms for finding the second-order stationary point. More specifically, we present two novel algorithms that can find local minima faster than existing algorithms (Xu et al., 2018b; Allen-Zhu and Li, 2018; Yu et al., 2018) in a wide regime for both finite-sum and stochastic optimization. The proposed algorithms essentially use Neon2 (Allen-Zhu and Li, 2018) to turn One-epoch-SNVRG into a local minimum finder.

1.1. Contribution

We summarize the major contributions of this paper as follows:

- We propose a stochastic nested variance reduced gradient (SNVRG) algorithm for nonconvex optimization, which reduces the dependence of the gradient complexity on n compared with SVRG and SCSG.

- We show that our proposed algorithm is able to find an ϵ-approximate stationary point with $\widetilde{O}(n \wedge \epsilon^{-2} + \epsilon^{-3} \wedge n^{1/2}\epsilon^{-2})$ stochastic gradient evaluations, which outperforms all existing first-order algorithms such as GD, SGD, SVRG and SCSG. A detailed comparison is demonstrated in Figure 1.

- As a by-product, when F is a τ-gradient dominated function, a variant of our algorithm can achieve an ϵ-approximate global minimizer (i.e., $F(\mathbf{x}) - \min_{\mathbf{y}} F(\mathbf{y}) \le \epsilon$) within $\widetilde{O}\big(n \wedge \tau\epsilon^{-1} + \tau(n \wedge \tau\epsilon^{-1})^{1/2}\big)$ stochastic gradient evaluations, which also outperforms the state-of-the-art.

- For the finite-sum optimization setting (1), we propose an algorithm, SNVRG + Neon2$^{\text{finite}}$, that can find an (ϵ, ϵ_H) second-order stationary point of the finite-sum problem (1) within $\widetilde{O}(n^{1/2}\epsilon^{-2} + n\epsilon_H^{-3} + n^{3/4}\epsilon_H^{-7/2})$ stochastic gradient evaluations, which is evidently faster than the best existing algorithm SVRG + Neon2$^{\text{finite}}$ (Allen-Zhu and Li, 2018) that attains $\widetilde{O}(n^{2/3}\epsilon^{-2} + n\epsilon_H^{-3} + n^{3/4}\epsilon_H^{-7/2})$ gradient complexity in a wide regime. A thorough comparison is illustrated in Figure 3.

- For the general stochastic optimization setting (2), we propose an algorithm, SNVRG+ Neon2$^{\text{online}}$, that can find an (ϵ, ϵ_H) second-order stationary point of (2) within $\widetilde{O}(\epsilon^{-3} + \epsilon_H^{-5} + \epsilon^{-2}\epsilon_H^{-3})$ stochastic gradient evaluations, which is again faster than the state-of-the-art algorithms such as SCSG+Neon2$^{\text{online}}$ (Allen-Zhu and Li, 2018) with $\widetilde{O}(\epsilon^{-10/3} + \epsilon_H^{-5} + \epsilon^{-2}\epsilon_H^{-3})$ gradient complexity, and Natasha2 (Allen-Zhu, 2018a) with $\widetilde{O}(\epsilon^{-3.25} + \epsilon^{-3}\epsilon_H + \epsilon_H^{-5})$ gradient complexity in certain regime. A detailed comparison is demonstrated in Figure 4.

- We also show that our proposed algorithms can find local minima even faster when the objective function enjoys the third-order smoothness property. We prove that our proposed algorithms achieve faster convergence rates to a local minimum than the FLASH algorithm proposed in Yu et al. (2018), which also exploits the third-order smoothness of objective functions for both finite-sum and general stochastic optimization problems.

A short version of this paper (Zhou et al., 2018b) has been published in NeurIPS 2018, which proposes the SNVRG algorithm for finding first-order stationary points. This longer version adds new algorithms that turn SNVRG into local minima finding algorithms.

The remainder of this paper is organized as follows: In Section 2 we review the relevant work in the literature. We present preliminary definitions in Section 3. We then present our SNVRG algorithm in Section 4. We present our main theoretical results for finding stationary points in Section 5. We further present two algorithms based on SNVRG to find local minima in Section 6. The theoretical analysis for finding local minima for second-order smooth functions is in Section 7 and that for third-order smooth functions in Section 8. Experiments on validating the advantage of SNVRG is provided in Section 9. We conclude the paper with Section 10.

Notation: Denote $\mathbf{A} = [A_{ij}] \in \mathbb{R}^{d\times d}$ as a matrix and $\mathbf{x} = (x_1, ..., x_d)^\top \in \mathbb{R}^d$ as a vector. $\|\mathbf{v}\|_2$ denotes the 2-norm of a vector $\mathbf{v} \in \mathbb{R}^d$. We use $\langle \cdot, \cdot \rangle$ to represent the inner product. For two sequences $\{a_n\}$ and $\{b_n\}$, we denote $a_n = O(b_n)$ if there is a constant $0 < C < +\infty$ such that $a_n \leq C\, b_n$, denote $a_n = \Omega(b_n)$ if there is a constant $0 < C < +\infty$, such that $a_n \geq C\, b_n$, and use $\widetilde{O}(\cdot)$ to hide logarithmic factors. We also write $a_n \lesssim b_n$ (or $a_n \gtrsim b_n$) if a_n is less than (or larger than) b_n up to a constant. We denote the product $c_a c_{a+1} \dots c_b$ term as $\prod_{i=a}^b c_i$. In addition, if $a > b$, we define $\prod_{i=a}^b c_i = 1$. In this paper, $\lfloor \cdot \rfloor$ represents the floor function and $\log(x)$ represents the logarithm of x to base 2. $a \wedge b$ means $\min(a, b)$. We denote by $\mathbb{1}\{\mathcal{E}\}$ the indicator function such that $\mathbb{1}\{\mathcal{E}\} = 1$ if the event $\mathcal{E}$ is true, and $\mathbb{1}\{\mathcal{E}\} = 0$ otherwise.

2. Related Work

In this section, we review and discuss the relevant work in the literature of nonconvex optimization for solving problems (1) and (2).

Finding first-order stationary points For nonconvex optimization, it is well-known that Gradient Descent (GD) can converge to an ϵ-approximate stationary point with $O(n \cdot \epsilon^{-2})$ (Nesterov, 2013) number of stochastic gradient evaluations. GD needs to calculate the full gradient at each iteration, which is a heavy load when $n \gg 1$. Stochastic gradient descent (SGD) (Robbins and Monro, 1951; Nesterov, 2013) and its variants (Ghadimi and Lan, 2013, 2016; Ghadimi et al., 2016) achieve $O(1/\epsilon^4)$ gradient complexity under the assumption that the stochastic gradient has a bounded variance. Inspired by the great success of various variance reduced techniques in convex finite-sum optimization such as Stochastic Average Gradient (SAG) (Roux et al., 2012), Stochastic Variance Reduced Gradient (SVRG) (Johnson and Zhang, 2013; Xiao and Zhang, 2014), SAGA (Defazio et al., 2014a), Stochastic Dual Coordinate Ascent (SDCA) (Shalev-Shwartz and Zhang, 2013), Finito (Defazio et al., 2014b) and Batching SVRG (Harikandeh et al., 2015), Garber and Hazan (2015); Shalev-Shwartz (2016) first analyzed the convergence of SVRG under nonconvex setting, where F is still convex but each component function f_i can be nonconvex. The analysis for the general nonconvex function F was done by Reddi et al. (2016a); Allen-Zhu and Hazan (2016), which shows that SVRG can converge to an ϵ-approximate stationary point with $O(n^{2/3} \cdot \epsilon^{-2})$ number of stochastic gradient evaluations. This result is strictly better than that of GD. Nguyen et al. (2017a,b) proposed StochAstic Recursive grAdient algoritHm (SARAH) with recursive estimators for finding first-order stationary points with $O(n + L^2/\epsilon^4)$ stochastic gradient evaluations. Lei et al. (2017) proposed a new variance reduction algorithm, i.e., the stochastically controlled stochastic gradient (SCSG) algorithm, which finds a first-order stationary point within $O(\min\{\epsilon^{-10/3}, n^{2/3}\epsilon^{-2}\})$ stochastic gradient evaluations for finite-sum optimization in (1), and outperforms SVRG when the number of component functions n is large.

The literature of finding local minima in nonconvex optimization can be roughly divided into three categories according to the oracles they use: Hessian-based, Hessian-vector product-based and gradient-based (Hessian-free). We review each category in the sequel accordingly.

Finding local minima using Hessian matrix The most popular algorithm using Hessian matrix to find an $(\epsilon, \sqrt{\epsilon})$-approximate local minimum is the cubic regularized Newton's method (Nesterov and Polyak, 2006), which attains $O(\epsilon^{-3/2})$ iteration complexity. The trust region method is proved to achieve the same iteration complexity (Curtis et al., 2017). To alleviate the computation burden of evaluating full gradients and Hessian matrices in large-scale optimization problems, subsampled cubic regularization and trust-region methods (Kohler and Lucchi, 2017; Xu et al., 2019) were proposed and proved to enjoy the same iteration complexity as their original versions with full gradients and Hessian matrices. Recently, stochastic variance reduced cubic regularization method (SVRC) (Zhou et al., 2018a) was proposed, which achieves the best-known second-order oracle complexity among existing cubic regularization methods.

Finding local minima using Hessian-vector product Another line of research uses Hessian-vector products to find the second-order stationary points. Carmon et al. (2018); Agarwal et al. (2017) independently proposed two algorithms that can find an $(\epsilon, \sqrt{\epsilon})$-approximate local minimum within $O(\epsilon^{-7/4})$ full gradient and Hessian-product evaluations. Agarwal et al. (2017) also showed that their algorithm only needs $O(n\epsilon^{-3/2} + n^{3/4}\epsilon^{7/4})$ stochastic gradient and Hessian-vector product evaluations for finite-sum optimization problems (1). Reddi et al. (2018) proposed a generic algorithmic framework that uses both first-order and second-order methods to find the local minimum within $O(n^{2/3}\epsilon^{-2} + n\epsilon^{-3/2} + n^{3/4}\epsilon^{7/4})$ stochastic gradient and Hessian-product evaluations. Allen-Zhu (2018a) proposed the Natasha2 algorithm which finds an $(\epsilon, \sqrt{\epsilon})$-approximate second-order stationary point within $O(\epsilon^{-7/2})$ stochastic gradient and Hessian-vector product evaluations.

Finding local minima using gradient The last line of research uses purely gradient information to find the local minima. The local minima finding algorithms proposed in this paper also fall into this category. Ge et al. (2015); Levy (2016) studied the perturbed GD and SGD algorithms for escaping saddle points, where isotropic noise is added into the gradient or stochastic gradient at each iteration or whenever the gradient is sufficiently small. Jin et al. (2017) further proposed a perturbed accelerated gradient descent, which can finds the second-order stationary point even faster. Xu et al. (2018b) showed that perturbed gradient or stochastic gradient descent can help find the negative curvature direction without using Hessian matrix and proposed the NEON algorithm that extracts the negative curvature using only first-order information. Later Allen-Zhu and Li (2018) developed the Neon2 algorithm, which improves upon on Neon, and turns Natasha2 (Allen-Zhu, 2018a) into a first-order method to find the local minima. Yu et al. (2017) proposed the gradient descent with one-step escaping algorithm (GOSE) that saves negative curvature computation and Yu et al. (2018) proposed the FLASH algorithm that exploits the third-order smoothness of the objective function. Very recently, Daneshmand et al. (2018) proved that SGD with periodically changing step size can escape from saddle points under an additional correlated negative curvature (CNC) assumption on the stochastic gradient. In another line of research, Zhang et al. (2017); Chen et al. (2020) studied the hitting time of SGLD to a local minimum of nonconvex functions. And Raginsky et al. (2017); Xu et al. (2018a) studied the global convergence of a family of Langevin dynamics based algorithms for nonconvex optimization.

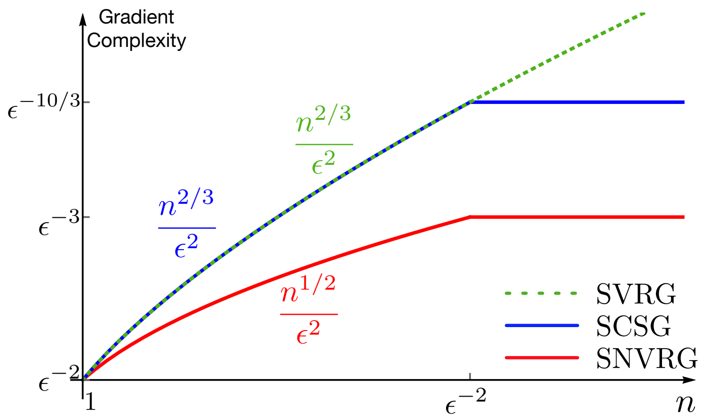


Figure 1: Comparison of gradient complexities.

To give a thorough comparison of our proposed SNVRG algorithm with existing algorithms for nonconvex finite-sum optimization, we summarize the gradient complexity of the most relevant algorithms in Table 1 for finding first-order stationary points and in Table 2 for finding local minimum using first-order information. We also present the gradient complexities of first-order local minimum finding algorithms in Table 2. According to Table 1, the proposed SNVRG algorithm achieves the lowest gradient complexity to find an ϵ-approximate first-order stationary point for both nonconvex functions and gradient dominant functions. We can also see from Table 2 that our proposed algorithms $\text{SNVRG} + \text{Neon2}^{\text{finite}}$ and $\text{SNVRG} + \text{Neon2}^{\text{online}}$ outperform all other first-order algorithms in finding an (ϵ, ϵ_H)-approximate second-order stationary point for nonconvex optimization problems in a wide regime, for both finite-sum and general stochastic optimization.

Follow-up work after this paper After the first appearance of our SNVRG algorithm in a conference paper (Zhou et al., 2018b), there have emerged a considerable amount of exciting work on this topic. Fang et al. (2018) concurrently proposed the Stochastic Path-Integrated Differential EstimatoR (SPIDER), which uses recursive update to define the semi-stochastic gradient in the variance reduction algorithm. They proved that SPIDER achieves $O(n^{1/2}\epsilon^{-2} \wedge \epsilon^{-3})$ gradient complexity for finding an ϵ-approximate stationary point in nonconvex optimization. Wang et al. (2019) proposed an improved analysis for SPIDER (also called SpiderBoost) and SPIDER with momentum. Nguyen et al. (2019) proposed an improved analysis for SARAH. Tran-Dinh et al. (2019) proposed a hybrid method which combines SARAH (Nguyen et al., 2017a) and SGD. Note that all the aforementioned algorithms enjoy a similar convergence rate to SPIDER (Fang et al., 2018). Fang et al. (2018); Zhou and Gu (2019) also showed that both SPIDER and SNVRG are near optimal with respect to the gradient complexity. In a recent work, Fang et al. (2019) proposed a tighter analysis of the gradient complexity for SGD to escape saddle points.

3. Preliminaries

In this section, we present some definitions that will be used throughout this paper.

Table 1: Comparisons on gradient complexity of different algorithms. The second column shows the gradient complexity for a nonconvex and smooth function to achieve an ϵ-approximate stationary point (i.e., $\|\nabla F(\mathbf{x})\|_2 \leq \epsilon$). The third column presents the gradient complexity for a gradient dominant function to achieve an ϵ-approximate global minimizer (i.e., $F(\mathbf{x}) - \min_{\mathbf{x}} F(\mathbf{x}) \leq \epsilon$). The last column presents the space complexity of all algorithms.

Algorithm	nonconvex	gradient dominant	Hessian Lipschitz
GD	$O\big(\frac{n}{\epsilon^2}\big)$	$\widetilde{O}(\tau n)$	No
SGD	$O\big(\frac{1}{\epsilon^4}\big)$	$O\big(\frac{1}{\epsilon^4}\big)$	No
SVRG (Reddi et al., 2016a)	$O\big(\frac{n^{2/3}}{\epsilon^2}\big)$	$\widetilde{O}(n + \tau n^{2/3})$	No
SCSG (Lei et al., 2017)	$O\big(\frac{1}{\epsilon^{10/3}} \wedge \frac{n^{2/3}}{\epsilon^2}\big)$	$\widetilde{O}\Big(n \wedge \frac{\tau}{\epsilon} + \tau\big(n \wedge \frac{\tau}{\epsilon}\big)^{2/3}\Big)$	No
GNC-AGD (Carmon et al., 2017)	$\widetilde{O}\big(\frac{n}{\epsilon^{1.75}}\big)$	N/A	Needed
Natasha 2 (Allen-Zhu, 2018a)	$\widetilde{O}\big(\frac{1}{\epsilon^{3.25}}\big)$	N/A	Needed
SNVRG (Algorithm 2 & 3)	$\widetilde{O}\big(\frac{1}{\epsilon^3} \wedge \frac{n^{1/2}}{\epsilon^2}\big)$	$\widetilde{O}\Big(n \wedge \frac{\tau}{\epsilon} + \tau\big(n \wedge \frac{\tau}{\epsilon}\big)^{1/2}\Big)$	No

Definition 1 (Smoothness) *$f : \mathbb{R}^d \to \mathbb{R}$ is L_1-smooth for some constant $L_1 > 0$, if it is differentiable and satisfies*

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq L_1\|\mathbf{x} - \mathbf{y}\|_2, \quad \textit{for any } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d. \tag{4}$$

Definition 1 implies that if f is L-smooth, we have for any $\mathbf{x}, \mathbf{h} \in \mathbb{R}^d$

$$f(\mathbf{x} + \mathbf{h}) \leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{h} \rangle + \frac{L}{2}\|\mathbf{h}\|_2^2. \tag{5}$$

Definition 2 (Hessian Lipschitzness) *$f : \mathbb{R}^d \to \mathbb{R}$ is L_2-Hessian Lipschitz for some constant $L_2 > 0$, if it is twice-differentiable and satisfies*

$$\|\nabla^2 f(\mathbf{x}) - \nabla^2 f(\mathbf{y})\|_2 \leq L_2\|\mathbf{x} - \mathbf{y}\|_2, \quad \textit{for any } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d.$$

The above two smoothness conditions are widely used in nonconvex optimization problems (Nesterov and Polyak, 2006). We will call them first-order smoothness and second-order smoothness respectively in this paper. As shown in Carmon et al. (2017); Yu et al. (2018), when the objective function has additionally third-order smoothness, one can design algorithms that find local minima even faster. Following Yu et al. (2018), we denote the three-way tensor $\nabla^3 f(\mathbf{x}) \in \mathbb{R}^{d\times d\times d}$ as the third-order derivative of f.

Definition 3 (Third-order Derivative) *The third-order derivative of function $f : \mathbb{R}^d \to \mathbb{R}$ is defined as a three-way tensor $\nabla^3 f(\mathbf{x}) \in \mathbb{R}^{d\times d\times d}$, where*

$$[\nabla^3 f(\mathbf{x})]_{ijk} = \frac{\partial}{\partial x_i \partial x_j \partial x_k} f(\mathbf{x}), \quad i, j, k = 1, \ldots, d \textit{ and } \mathbf{x} \in \mathbb{R}^d.$$

Table 2: Comparisons on gradient complexities of different algorithms to find an (ϵ, ϵ_H)-approximate second-order stationary point in both finite-sum and general stochastic optimization settings. The last column indicates whether the algorithm exploits the third-order smoothness of the objective function.

Setting	algorithm	gradient complexity	3^{rd}-order smooth
Finite-Sum	PGD (Jin et al., 2017)	$\widetilde{O}\big(\frac{n}{\epsilon^2}\big)$ (for $\epsilon_H \geq \epsilon^{1/2}$)	No
	SVRG + Neon2$^{\text{finite}}$ (Allen-Zhu and Li, 2018)	$\widetilde{O}\Big(\frac{n^{2/3}}{\epsilon^2} + \frac{n}{\epsilon_H^3} + \frac{n^{3/4}}{\epsilon_H^{7/2}}\Big)$	No
	FLASH (Yu et al., 2018)	$\widetilde{O}\Big(\frac{n^{2/3}}{\epsilon^2} + \frac{n}{\epsilon_H^2} + \frac{n^{3/4}}{\epsilon_H^{5/2}}\Big)$	Needed
	SNVRG + Neon2$^{\text{finite}}$ (Algorithm 4)	$\widetilde{O}\Big(\frac{n^{1/2}}{\epsilon^2} + \frac{n}{\epsilon_H^3} + \frac{n^{3/4}}{\epsilon_H^{7/2}}\Big)$	No
	SNVRG + Neon2$^{\text{finite}}$ (Algorithm 4)	$\widetilde{O}\Big(\frac{n^{1/2}}{\epsilon^2} + \frac{n}{\epsilon_H^2} + \frac{n^{3/4}}{\epsilon_H^{5/2}}\Big)$	Needed
Stochastic	Perturbed SGD (Ge et al., 2015)	$\widetilde{O}\Big(\frac{\text{poly}(d)}{\epsilon^4}\Big)$ (for $\epsilon_H \geq \epsilon^{1/4}$)	No
	CNC-SGD (Daneshmand et al., 2018)	$\widetilde{O}\Big(\frac{1}{\epsilon^4} + \frac{1}{\epsilon_H^{10}}\Big)$	No
	Natasha2+Neon2$^{\text{online}}$ (Allen-Zhu, 2018a)	$\widetilde{O}\Big(\frac{1}{\epsilon^{3.25}} + \frac{1}{\epsilon^3\epsilon_H} + \frac{1}{\epsilon_H^5}\Big)$	No
	SCSG+Neon2$^{\text{online}}$ (Allen-Zhu and Li, 2018)	$\widetilde{O}\Big(\frac{1}{\epsilon^{10/3}} + \frac{1}{\epsilon^2\epsilon_H^3} + \frac{1}{\epsilon_H^5}\Big)$	No
	FLASH (Yu et al., 2018)	$\widetilde{O}\Big(\frac{1}{\epsilon^{10/3}} + \frac{1}{\epsilon^2\epsilon_H^2} + \frac{1}{\epsilon_H^4}\Big)$	Needed
	SNVRG + Neon2$^{\text{online}}$ (Algorithm 5)	$\widetilde{O}\Big(\frac{1}{\epsilon^3} + \frac{1}{\epsilon^2\epsilon_H^3} + \frac{1}{\epsilon_H^5}\Big)$	No
	SNVRG + Neon2$^{\text{online}}$ (Algorithm 5)	$\widetilde{O}\Big(\frac{1}{\epsilon^3} + \frac{1}{\epsilon^2\epsilon_H^2} + \frac{1}{\epsilon_H^4}\Big)$	Needed

Now we are ready to present the formal definition of third-order smoothness, which has been explored in Anandkumar and Ge (2016); Carmon et al. (2017); Yu et al. (2018). It is also called third-order derivative Lipschitzness in Carmon et al. (2017).

Definition 4 (Third-order Smoothness) $f: \mathbb{R}^d \rightarrow \mathbb{R}$ *is* L_3*-third-order smooth for some constant* $L_3 > 0$*, if it is thrice-differentiable and satisfies*

$$\|\nabla^3 f(\mathbf{x}) - \nabla^3 f(\mathbf{y})\|_F \leq L_3 \|\mathbf{x} - \mathbf{y}\|_2, \quad \textit{for any } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d.$$

The following definition characterizes the distance between the initial point of an algorithm and the minimizer of function f.

Definition 5 (Optimal Gap) *The optimal gap of f at point $\mathbf{x}_0$ is denoted by Δ_f and*

$$f(\mathbf{x}_0) - \min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) \le \Delta_f.$$

W.L.O.G., we assume $\Delta_f < +\infty$.

Definition 6 *$f : \mathbb{R}^d \to \mathbb{R}$ is λ-strongly convex for some constant $\lambda > 0$, if it satisfies*

$$f(\mathbf{x}+\mathbf{h}) \ge f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{h}\rangle + \frac{\lambda}{2}\|\mathbf{h}\|_2^2, \quad \text{for any } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d. \tag{6}$$

While the above definitions are based on a general function f, the following two definitions rely on the finite-sum structure of F defined in (1).

Definition 7 *A function F with finite-sum structure in (1) is said to have stochastic gradients with bounded variance σ^2, if for any $\mathbf{x} \in \mathbb{R}^d$, we have*

$$\mathbb{E}_i\|\nabla f_i(\mathbf{x}) - \nabla F(\mathbf{x})\|_2^2 \le \sigma^2, \tag{7}$$

where i a random index uniformly chosen from $[n]$ and $\mathbb{E}_i$ denotes the expectation over such i.

σ^2 is called the upper bound on the variance of stochastic gradients (Lei et al., 2017).

Definition 8 *A function F with finite-sum structure in (1) is said to have averaged L-Lipschitz gradient, if for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, we have*

$$\mathbb{E}_i\|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})\|_2^2 \le L^2\|\mathbf{x}-\mathbf{y}\|_2^2, \tag{8}$$

where i is a random index uniformly chosen from $[n]$ and $\mathbb{E}_i$ denotes the expectation over the choice.

It should be noted that the smoothness condition of each f_i in Definition 1 will directly imply the averaged L-Lipschitz gradient for F.

We also consider a class of functions namely gradient dominated functions (Polyak, 1963), which is formally defined as follows:

Definition 9 *We say function f is τ-gradient dominated if for any $\mathbf{x} \in \mathbb{R}^d$, we have*

$$f(\mathbf{x}) - f(\mathbf{x}^*) \le \tau \cdot \|\nabla f(\mathbf{x})\|_2^2, \tag{9}$$

where $\mathbf{x}^* \in \mathbb{R}^d$ is the global minimum of f.

Note that gradient dominated condition is also known as the Polyak-Lojasiewicz (P-L) condition (Polyak, 1963), and is not necessarily convex. It is weaker than strong convexity as well as other popular conditions that appear in the optimization literature (Karimi et al., 2016).

Inspired by the SCSG algorithm (Lei et al., 2017), we will use the property of geometric distribution in our algorithm design. The definition of geometric random variable is as follows.

Definition 10 (Geometric Distribution) *A random variable X follows a geometric distribution with parameter p, denoted as Geom(p), if it holds that*

$$\mathbb{P}(X = k) = p(1-p)^k, \quad \forall k = 0, 1, \dots.$$

Definition 11 (Sub-Gaussian Stochastic Gradient) *We say a function F has σ^2-sub-Gaussian stochastic gradient $\nabla F(\mathbf{x}; \xi)$ for any $\mathbf{x} \in \mathbb{R}^d$ and random variable $\xi \sim \mathcal{D}$, if it satisfies*

$$\mathbb{E}\bigg[\exp\bigg(\frac{\|\nabla F(\mathbf{x}; \xi) - \nabla f(\mathbf{x})\|_2^2}{\sigma^2}\bigg)\bigg] \leq \exp(1).$$

Note that Definition 11 implies $\mathbb{E}[\|\nabla F(\mathbf{x}; \xi) - \nabla f(\mathbf{x})\|_2^2] \leq 2\sigma^2$ (Vershynin, 2010). In the finite-sum optimization setting (1), we call $\nabla f_i(\mathbf{x})$ a stochastic gradient of function F for a randomly chosen index $i \in [n]$, and we say F has σ^2-sub-Gaussian stochastic gradient if $\mathbb{E}[\|\nabla f_i(\mathbf{x}) - \nabla F(\mathbf{x})\|_2^2] \leq 2\sigma^2$.

4. Stochastic Nested Variance-Reduced Gradient Descent

In this section, we present our nested stochastic variance reduction algorithm, namely, SNVRG for finding first-order stationary points in nonconvex optimization.

One-epoch-SNVRG: We first present the key component of our main algorithm, One-epoch-SNVRG, which is displayed in Algorithm 1. The most innovative part of Algorithm 1 attributes to the $K+1$ *reference points* and $K+1$ *reference gradients*. Note that when $K = 1$, Algorithm 1 reduces to one epoch of SVRG algorithm (Johnson and Zhang, 2013; Reddi et al., 2016a; Allen-Zhu and Hazan, 2016). To better understand our One-epoch-SNVRG algorithm, it would be helpful to revisit the original SVRG which is a special case of our algorithm. For the finite-sum optimization problem in (1), the original SVRG takes the following updating formula

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \mathbf{v}_t = \mathbf{x}_t - \eta\big(\nabla F(\widetilde{\mathbf{x}}) + \nabla f_{i_t}(\mathbf{x}_t) - \nabla f_{i_t}(\widetilde{\mathbf{x}})\big),$$

where $\eta > 0$ is the step size, i_t is a random index uniformly chosen from $[n]$ and $\widetilde{\mathbf{x}}$ is a snapshot for $\mathbf{x}_t$ after every T_1 iterations. There are two reference points in the update formula at $\mathbf{x}_t$: $\mathbf{x}_t^{(0)} = \widetilde{\mathbf{x}}$ and $\mathbf{x}_t^{(1)} = \mathbf{x}_t$. Note that $\widetilde{\mathbf{x}}$ is updated every T_1 iterations, namely, $\widetilde{\mathbf{x}}$ is set to be $\mathbf{x}_t$ only when $(t \mod T_1) = 0$. Moreover, in the semi-stochastic gradient $\mathbf{v}_t$, there are also two reference gradients and we denote them by $\mathbf{g}_t^{(0)} = \nabla F(\widetilde{\mathbf{x}})$ and $\mathbf{g}_t^{(1)} = \nabla f_{i_t}(\mathbf{x}_t) - \nabla f_{i_t}(\widetilde{\mathbf{x}}) = \nabla f_{i_t}(\mathbf{x}_t^{(1)}) - \nabla f_{i_t}(\mathbf{x}_t^{(0)})$.

Back to our One-epoch-SNVRG, we can define similar reference points and reference gradients as that in the special case of SVRG. Specifically, for $t = 0, \dots, \prod_{l=1}^K T_l - 1$, each point $\mathbf{x}_t$ has $K+1$ reference points $\{\mathbf{x}_t^{(l)}\}, l = 0, \dots, K$, which is set to be $\mathbf{x}_t^{(l)} = \mathbf{x}_{t^l}$ with index t^l defined as

$$t^l = \bigg\lfloor \frac{t}{\prod_{k=l+1}^K T_k} \bigg\rfloor \cdot \prod_{k=l+1}^K T_k. \tag{10}$$

Algorithm 1 One-epoch-SNVRG($\mathbf{x}_0,\ F, K, M, \{T_l\}, \{B_l\}, B_0$)

Input: initial point $\mathbf{x}_{-1}^{(l)} \leftarrow \mathbf{x}_0, l \in [K]$; function F; loop number K; step size parameter M; loop parameters $\{T_l\}$; batch parameters $\{B_l\}$, base batch size B_0.
Option I $T = \prod_{l=1}^{K} T_l$
Option II $T \sim \mathrm{Geom}(1/(1+\prod_{l=1}^{K} T_l))$
for $t = 0, \ldots, T-1$ **do**
  $r = \min\{j : 0 = (t \mod \prod_{l=j+1}^{K} T_l),\ 0 \le j \le K\}$
  $\{\mathbf{x}_t^{(l)}\} \leftarrow$ Update_reference_points($\{\mathbf{x}_{t-1}^{(l)}\}, \mathbf{x}_t, r), 0 \le l \le K$.
  $\{\mathbf{g}_t^{(l)}\} \leftarrow$ Update_reference_gradients($\{\mathbf{g}_{t-1}^{(l)}\}, \{\mathbf{x}_t^{(l)}\}, r), 0 \le l \le K$.
  $\mathbf{v}_t \leftarrow \sum_{l=0}^{K} \mathbf{g}_t^{(l)}$
  $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - 1/(10M) \cdot \mathbf{v}_t$
end for
$\mathbf{x}_{\mathrm{out}} \leftarrow$ uniformly random choice from $\{\mathbf{x}_t\}$, where $0 \le t < \prod_{l=1}^{K} T_l$
Output: $[\mathbf{x}_{\mathrm{out}}, \mathbf{x}_T]$

Function: Update_reference_points($\{\mathbf{x}_{\mathrm{old}}^{(l)}\}, \mathbf{x}, r$)
$\mathbf{x}_{\mathrm{new}}^{(l)} \leftarrow \mathbf{x}_{\mathrm{old}}^{(l)}, 0 \le l \le r-1$; $\mathbf{x}_{\mathrm{new}}^{(l)} \leftarrow \mathbf{x}, r \le l \le K$
return $\{\mathbf{x}_{\mathrm{new}}^{(l)}\}$

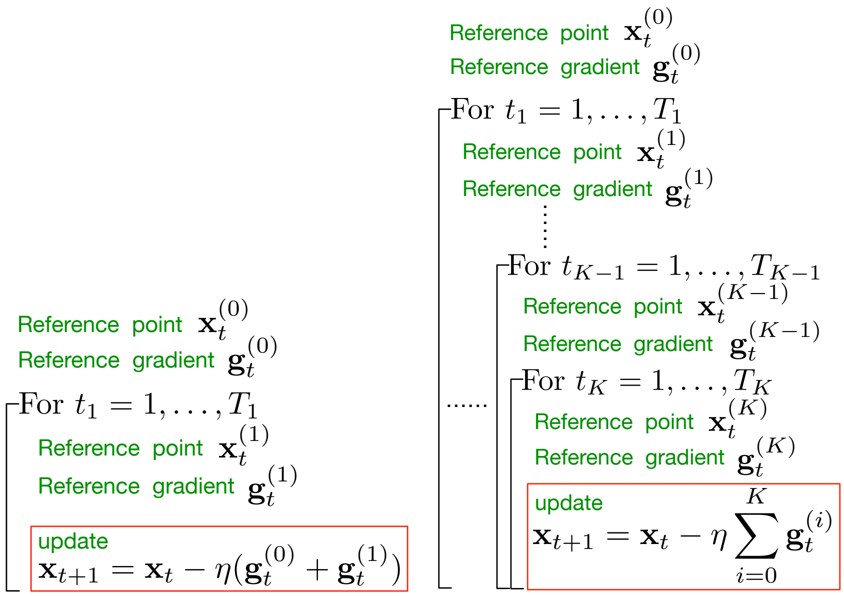
Function: Update_reference_gradients($\{\mathbf{g}_{\mathrm{old}}^{(l)}\}, \{\mathbf{x}_{\mathrm{new}}^{(l)}\}, r$)
if $r > 0$ **then**
  $\mathbf{g}_{\mathrm{new}}^{(l)} \leftarrow \mathbf{g}_{\mathrm{old}}^{(l)}, 0 \le l < r$; $\mathbf{g}_{\mathrm{new}}^{(l)} \leftarrow 0, r+1 \le l \le K$
  Uniformly generate index set $I \subset [n]$ without replacement, $|I| = B_r$
  $\mathbf{g}_{\mathrm{new}}^{(r)} \leftarrow 1/B_r \sum_{i \in I} \big[\nabla f_i(\mathbf{x}_{\mathrm{new}}^{(r)}) - \nabla f_i(\mathbf{x}_{\mathrm{new}}^{(r-1)})\big]$
else
  Uniformly generate index set $I \subset [n]$ without replacement, $|I| = B_0$
  $\mathbf{g}_{\mathrm{new}}^{(0)} \leftarrow 1/B_0 \sum_{i \in I} \nabla f_i(\mathbf{x}_{\mathrm{new}}^{(0)})$; $\mathbf{g}_{\mathrm{new}}^{(l)} \leftarrow 0, 1 \le l \le K$
end if
return $\{\mathbf{g}_{\mathrm{new}}^{(l)}\}$.

Specially, note that we have $\mathbf{x}_t^{(0)} = \mathbf{x}_0$ and $\mathbf{x}_t^{(K)} = \mathbf{x}_t$ for all $t = 0, \ldots, \prod_{l=1}^{K} T_l - 1$. Similarly, $\mathbf{x}_t$ also has $K+1$ reference gradients $\{\mathbf{g}_t^{(l)}\}$, which can be defined based on the reference points $\{\mathbf{x}_t^{(l)}\}$:

$$\mathbf{g}_t^{(0)} = \frac{1}{B} \sum_{i \in I} \nabla f_i(\mathbf{x}_0), \qquad \mathbf{g}_t^{(l)} = \frac{1}{B_l} \sum_{i \in I_l} \big[\nabla f_i(\mathbf{x}_t^{(l)}) - \nabla f_i(\mathbf{x}_t^{(l-1)})\big], l = 1, \ldots, K, \tag{11}$$

where I, I_l are random index sets with $|I| = B, |I_l| = B_l$ and are uniformly generated from $[n]$ without replacement. Based on the reference points and reference gradients, we then update $\mathbf{x}_{t+1} = \mathbf{x}_t - 1/(10M) \cdot \mathbf{v}_t$, where $\mathbf{v}_t = \sum_{l=0}^{K} \mathbf{g}_t^{(l)}$ and M is the step size parameter. The illustration of reference points and gradients of SNVRG is displayed in Figure 2(b).

We remark that it would be a huge waste for us to re-evaluate $\mathbf{g}_t^{(l)}$ at each iteration. Fortunately, due to the fact that each reference point is only updated after a long period,



(a) SVRG

(b) SNVRG

Figure 2: Illustration of reference points and gradients in SVRG and SNVRG.

we can maintain $\mathbf{g}_t^{(l)} = \mathbf{g}_{t-1}^{(l)}$ and only need to update $\mathbf{g}_t^{(l)}$ when $\mathbf{x}_t^{(l)}$ has been updated as is suggested by Line 20 in Algorithm 1.

SNVRG: Using One-epoch-SNVRG (Algorithm 1) as a building block, we now present our main algorithm: Algorithm 2, for finding an ϵ-approximate stationary point in nonconvex finite-sum optimization. At each iteration of Algorithm 2, it executes One-epoch-SNVRG (Algorithm 1) which takes $\mathbf{z}_{s-1}$ as its input and outputs $[\mathbf{y}_s, \mathbf{z}_s]$. We choose $\mathbf{y}_{\text{out}}$ as the output of Algorithm 2 uniformly from $\{\mathbf{y}_s\}$, for $s = 1, \ldots, S$.

SNVRG-PL: In addition, when function F in (1) is gradient dominated as defined in Definition 9 (P-L condition), it has been proved that the global minimum can be found by SGD (Karimi et al., 2016), SVRG (Reddi et al., 2016a) and SCSG (Lei et al., 2017) very efficiently. Following a similar trick used in Reddi et al. (2016a), we present Algorithm 3 on top of Algorithm 2, to find the global minimum in this setting. We call Algorithm 3 SNVRG-PL, because gradient dominated condition is also known as Polyak-Lojasiewicz (PL) condition (Polyak, 1963).

Space complexity: We briefly compare the space complexity between our algorithms and other variance reduction based algorithms. SVRG and SCSG needs $O(d)$ space complexity to store one reference gradient, SAGA (Defazio et al., 2014a) needs to store reference gradients for each component functions, and its space complexity is $O(nd)$ without using any trick. For our algorithm SNVRG, we need to store K reference gradients, thus its space complexity is $O(Kd)$. In our theory, we will show that $K = O(\log \log n)$. Therefore, the space complexity of our algorithm is actually $\widetilde{O}(d)$, which is almost comparable to that of SVRG and SCSG.

Algorithm 2 SNVRG($\mathbf{z}_0, F, K, M, \{T_l\}, \{B_l\}, B_0, S$)

1: **Input:** initial point $\mathbf{z}_0$; function F; loop numbers K, S; step size parameter M; loop parameters $\{T_l\}$; batch parameters $\{B_l\}$; base batch size B_0.
2: **for** $s = 1, \ldots, S$ **do**
3: $[\mathbf{y}_s, \mathbf{z}_s] =$ One-epoch-SNVRG($\mathbf{z}_{s-1}, F, K, M, \{T_l\}, \{B_l\}, B_0$) ▷ Algorithm 1 with **Option I**
4: **end for**
5: **Output:** Uniformly choose $\mathbf{y}_{\text{out}}$ from $\{\mathbf{y}_s\}, 1 \le s \le S$.

Algorithm 3 SNVRG-PL($\mathbf{z}_0, F, K, M, \{T_l\}, \{B_l\}, B_0, S, U$)

1: **Input:** initial point $\mathbf{z}_0$; function F; loop number K, S; step size parameter M; loop parameters $\{T_l\}$; batch parameters $\{B_l\}$; base batch size B_0; outer loop number U.
2: **for** $u = 1, \ldots, U$ **do**
3: $\mathbf{z}_u =$ SNVRG($\mathbf{z}_{u-1}, F, K, M, \{T_l\}, \{B_l\}, B_0, S$) ▷ Algorithm 2
4: **end for**
5: **Output:** $\mathbf{z}_{\text{out}} = \mathbf{z}_U$.

5. Theoretical Analysis of SNVRG

In this section, we provide the convergence analysis of SNVRG. We will assume that F has the finite-sum structure in (1) throughout this section.

5.1. Convergence of SNVRG

The following theorem shows the gradient complexity for Algorithm 2 to find an ϵ-approximate stationary point with a constant base batch size B_0.

Theorem 12 *Suppose that F has averaged L-Lipschitz gradient and stochastic gradients with bounded variance σ^2. In Algorithm 2, let $B_0 = n \wedge (2C\sigma^2/\epsilon^2)$ and suppose $B_0 > 4$, $S = 1 \vee (2CL\Delta_F/(B_0^{1/2}\epsilon^2))$ and $C = 6000$. The rest parameters $(K, M, \{B_l\}, \{T_l\})$ are chosen as follows:*

$$\begin{aligned}
&K = \lfloor \log\log B_0 \rfloor, \\
&M = 6L_1, \\
&T_1 = \big\lfloor B_0^{2^{-K}} \big\rfloor, \quad T_l = \big\lfloor B_0^{2^{l-K-2}} \big\rfloor, \text{ for } 2 \le l \le K, \\
&B_l = 6^{K-l+1}\bigg(\prod_{s=l}^{K} T_s\bigg)^2, \text{ for } 1 \le l \le K.
\end{aligned} \tag{12}$$

Then the output $\mathbf{y}_{out}$ of Algorithm 2 satisfies $\mathbb{E}[\|\nabla F(\mathbf{y}_{out})\|_2^2] \le \epsilon^2$ with less than

$$O\bigg(\log^3\bigg(\frac{\sigma^2}{\epsilon^2} \wedge n\bigg)\bigg[\frac{\sigma^2}{\epsilon^2} \wedge n + \frac{L\Delta_F}{\epsilon^2}\bigg[\frac{\sigma^2}{\epsilon^2} \wedge n\bigg]^{1/2}\bigg]\bigg) \tag{13}$$

stochastic gradient computations, where $\Delta_F = F(\mathbf{z}_0) - F^*$.

Remark 13 *If we treat σ^2, L and Δ_F as constants, and assume $\epsilon \ll 1$, then* (13) *can be simplified to $\widetilde{O}(\epsilon^{-3} \wedge n^{1/2}\epsilon^{-2})$. This gradient complexity is strictly better than $O(\epsilon^{-10/3} \wedge n^{2/3}\epsilon^{-2})$, which is achieved by SCSG (Lei et al., 2017). Specifically, when $n \lesssim 1/\epsilon^2$, our proposed SNVRG is faster than SCSG by a factor of $n^{1/6}$; when $n \gtrsim 1/\epsilon^2$, SNVRG is faster than SCSG by a factor of $\epsilon^{-1/3}$. Moreover, SNVRG also outperforms Natasha 2 (Allen-Zhu, 2018a) which attains $\widetilde{O}(\epsilon^{-3.25})$ gradient complexity and needs the additional Hessian Lipschitz condition.*

5.2. Convergence of SNVRG-PL

We now consider the case when F is a τ-gradient dominated function. In general, we are able to find an ϵ-approximate global minimizer of F instead of only an ϵ-approximate stationary point. Algorithm 3 uses Algorithm 2 as a component.

Theorem 14 *Suppose that F has averaged L-Lipschitz gradient and stochastic gradients with bounded variance σ^2, F is a τ-gradient dominated function. In Algorithm 3, let the base batch size $B_0 = n \wedge (4C_1\tau\sigma^2/\epsilon)$ and suppose $B_0 > 4$, the number of epochs for SNVRG $S = 1 \vee (2C_1\tau L/B_0^{1/2})$ and the number of epochs $U = \log(2\Delta_F/\epsilon)$. The rest parameters $(K, M, \{B_l\}, \{T_l\})$ are chosen as the same in Lemma 28. Then the output $\mathbf{z}_{out}$ of Algorithm 3 satisfies $\mathbb{E}\big[F(\mathbf{z}_{out}) - F^*\big] \le \epsilon$ within*

$$O\bigg(\log^3\bigg(n \wedge \frac{\tau\sigma^2}{\epsilon}\bigg)\log\frac{\Delta_F}{\epsilon}\bigg[n \wedge \frac{\tau\sigma^2}{\epsilon} + \tau L\bigg[n \wedge \frac{\tau\sigma^2}{\epsilon}\bigg]^{1/2}\bigg]\bigg) \tag{14}$$

stochastic gradient computations, where $\Delta_F = F(\mathbf{z}_0) - F^*$

Remark 15 *If we treat σ^2, L and Δ_F as constants, then the gradient complexity in* (14) *turns into $\widetilde{O}(n \wedge \tau\epsilon^{-1} + \tau(n \wedge \tau\epsilon^{-1})^{1/2})$. Compared with nonconvex SVRG (Reddi et al., 2016b) which achieves $\widetilde{O}(n + \tau n^{2/3})$ gradient complexity, our SNVRG-PL is strictly better than SVRG in terms of the first summand and is faster than SVRG at least by a factor of $n^{1/6}$ in terms of the second summand. Compared with a more general variant of SVRG, namely, the SCSG algorithm (Lei et al., 2017), which attains $\widetilde{O}\big(n \wedge \tau\epsilon^{-1} + \tau(n \wedge \tau\epsilon^{-1})^{2/3}\big)$ gradient complexity, SNVRG-PL also outperforms it by a factor of $(n \wedge \tau\epsilon^{-1})^{1/6}$.*

If we further assume that F is λ-strongly convex, then it is easy to verify that F is also $1/(2\lambda)$-gradient dominated. As a direct consequence, we have the following corollary:

Corollary 16 *Under the same conditions and parameter choices as Theorem 14. If we additionally assume that F is λ-strongly convex, then Algorithm 3 will outputs an ϵ-approximate global minimizer within*

$$\widetilde{O}\bigg(n \wedge \frac{\lambda\sigma^2}{\epsilon} + \kappa \cdot \bigg[n \wedge \frac{\lambda\sigma^2}{\epsilon}\bigg]^{1/2}\bigg) \tag{15}$$

stochastic gradient computations, where $\kappa = L/\lambda$ is the condition number of F.

Remark 17 *Corollary 16 suggests that when we regard λ and σ^2 as constants and set $\epsilon \ll 1$, Algorithm 3 is able to find an ϵ-approximate global minimizer within $\widetilde{O}(n+n^{1/2}\kappa)$ stochastic* *gradient computations, which matches SVRG*^{1ep} *in Katyusha X (Allen-Zhu, 2018b). Using catalyst techniques (Lin et al., 2015) or Katyusha momentum (Allen-Zhu, 2017), it can be further accelerated to* $\widetilde{O}(n + n^{3/4}\sqrt{\kappa})$*, which matches the best-known convergence rate (Shalev-Shwartz, 2016; Allen-Zhu, 2018b).*

6. Stochastic Nested Variance Reduction for Finding Local Minima

In this section, we present our algorithms that are built upon One-epoch-SNVRG (Algorithm 1) and Neon2 (Allen-Zhu and Li, 2018) to find a local minimum in nonconvex optimization faster than existing methods. It is worth noting that to find local minima, we employ a different choice of the number of iteration T which is chosen to be a random variable following a geometric distribution (Algorithm 1 with Option II) rather than fixed. We will show in the next section that these differences are essential in the theoretical analysis of finding local minima.

6.1. SNVRG + Neon2: Finding Local Minima

We propose two different algorithms for solving the finite-sum optimization problem in (1) and the general stochastic optimization problem in (2) respectively.

To solve the finite-sum optimization problem (1), we propose the SNVRG + Neon2$^{\text{finite}}$ algorithm to find the local minimum, which is displayed in Algorithm 4. At each iteration of 4, it first determines whether the current point is a first-order stationary point (Line 4) or not. If not, it will run Algorithm 1 (One-epoch-SNVRG) in order to find a first-order stationary point. Once obtaining a first-order stationary point, it will call Neon2$^{\text{finite}}$ to find the negative curvature direction to escape any potential non-degenerate saddle point. According to Xu et al. (2018b); Allen-Zhu and Li (2018), Neon-type algorithms can output such a direction with probability $1-\delta$ for some failure probability $\delta \in (0,1)$. If Neon2$^{\text{finite}}$ does not find such a direction, it will output $\widehat{\mathbf{v}} = \perp$ and Algorithm 4 terminates and outputs $\mathbf{z}_{u-1}$ (Line 9) since it has already reached a second-order stationary point according to (3). If Neon2$^{\text{finite}}$ finds a negative curvature direction $\widehat{\mathbf{v}} \neq \perp$, Algorithm 4 will perform one step of negative curvature descent in the direction of $\widehat{\mathbf{v}}$ or $-\widehat{\mathbf{v}}$ (Line 12) to escape the non-degenerate saddle point. The direction can also be chosen in the same way as in Carmon et al. (2017) via comparing the function values at the two resulting points. Here to reduce the computational complexity, we follow Xu et al. (2018b) and generate a Rademacher random variable to decide the direction, which leads to the same result in expectation.

To solve the general stochastic optimization problem in (2), we propose the SNVRG + Neon2$^{\text{online}}$ algorithm to find the local minimum, which is displayed in Algorithm 5. It is almost the same as Algorithm 4 used in the finite-sum nonconvex optimization setting except that it uses a subsampled gradient to determine whether we have obtained a first-order stationary point (Line 5 in Algorithm 5) and it uses Neon2$^{\text{online}}$ (Line 8) to find the negative curvature direction to escape the potential saddle points. Algorithm 5 will terminate and output the current iterate if no negative curvature direction is found (Line 10).

Note that both Algorithms 4 and 5 are only based on the gradient information of the objective function and are therefore first-order optimization algorithms. As we will show in the next two sections, our proposed algorithms push the frontier of first-order stochastic

Algorithm 4 SNVRG + Neon2$^{\text{finite}}$($\mathbf{z}_0, F, K, M, \{T_l\}, \{B_l\}, B_0, U, \epsilon, \epsilon_H, \delta, \eta, L_1, L_2$)

Input: initial point $\mathbf{z}_0$; function F; loop number K; step size parameter M; loop parameters $\{T_l\}$; batch parameters $\{B_l\}$; base batch size B_0; gradient accuracy ϵ; Hessian accuracy ϵ_H; failure probability δ; negative curvature descent step size η; gradient Lipschitz parameter L_1; Hessian Lipschitz parameter L_2.
for $u = 1, \ldots, U$ **do**
  $\mathbf{g}_{u-1} = \nabla F(\mathbf{z}_{u-1})$
  if $\|\mathbf{g}_{u-1}\|_2 \geq \epsilon$ **then**
    $\mathbf{z}_u =$ One-epoch-SNVRG($\mathbf{z}_{u-1}$,$F, K, M, \{T_l\}, \{B_l\}, B_0$) ▷ Algorithm 1 with **Option II**
  else
    $\mathbf{v} =$ Neon2$^{\text{finite}}$($F, \mathbf{z}_{u-1}, L_1, L_2, \delta, \epsilon_H$)
    if $\mathbf{v} = \perp$ **then**
      return $\mathbf{z}_{u-1}$
    else
      Generate a Rademacher random variable ζ
      $\mathbf{z}_u \leftarrow \mathbf{z}_{u-1} + \zeta\eta\widehat{\mathbf{v}}$
    end if
  end if
end for
return

optimization algorithms for finding local minima (Xu et al., 2018b; Allen-Zhu and Li, 2018; Allen-Zhu, 2018a; Yu et al., 2017, 2018).

7. Theoretical Analysis of SNVRG for Finding Local Minima

In this section, we provide the main theoretical results for finding local minima using SNVRG.

7.1. Finite-Sum Optimization Problems

We start with the nonconvex finite-sum optimization problem (1). The following theorem provides the gradient complexity of Algorithm 4 in finding an approximate local minimum.

Theorem 18 *Suppose that $F = 1/n \sum_{i=1}^n f_i$, where each f_i is L_1-smooth and L_2-Hessian Lipschitz continuous. Let $0 < \epsilon, \epsilon_H < 1$, $\delta = \epsilon_H^3/(144 L_2^2 \Delta_F)$ and $U = 24 L_2^2 \Delta_F \epsilon_H^{-3} + 1800 L_1 \Delta_F \epsilon^{-2} n^{-1/2}$. Set $B_0 = n, M = 6L_1$ and all the rest parameters of One-epoch-SNVRG as in* (12) *Lemma 29. Choose step size $\eta = \epsilon_H/L_2$. Then with probability at least $1/4$, SNVRG + Neon2$^{\text{finite}}$ will find an (ϵ, ϵ_H)-second-order stationary point within*

$$\widetilde{O}\bigg(\frac{\Delta_F n L_2^2}{\epsilon_H^3} + \frac{\Delta_F n^{3/4} L_1^{1/2} L_2^2}{\epsilon_H^{7/2}} + \frac{\Delta_F n^{1/2} L_1}{\epsilon^2}\bigg) \tag{16}$$

stochastic gradient evaluations.

Algorithm 5 SNVRG + $\text{Neon2}^{\text{online}}(\mathbf{z}_0, F, K, M, \{T_l\}, \{B_l\}, U, \epsilon, \epsilon_H, \delta, \eta, L_1, L_2)$

Input: initial point $\mathbf{z}_0$; function F; loop number K; step size parameter M; loop parameters $\{T_l\}$; batch parameters $\{B_l\}$; base batch size B_0; gradient accuracy ϵ; Hessian accuracy ϵ_H; failure probability δ; negative curvature descent step size η; gradient Lipschitz parameter L_1; Hessian Lipschitz parameter L_2.
for $u = 1, \ldots, U$ **do**
  Uniformly generate index set $I \subset [n]$ without replacement, $|I| = B_0$
  $\mathbf{g}_{u-1} = 1/B_0 \sum_{i \in I} \nabla f_i(\mathbf{z}_{u-1})$
  if $\|\mathbf{g}_{u-1}\|_2 \geq \epsilon/2$ **then**
    $\mathbf{z}_u = \text{One-epoch-SNVRG}(\mathbf{z}_{u-1}, F, K, M, \{T_l\}, \{B_l\}, B_0)$ ▷ Algorithm 1 with **Option II**
  else
    $\mathbf{v} = \text{Neon2}^{\text{online}}(F, \mathbf{z}_{u-1}, L_1, L_2, \delta, \epsilon_H)$
    if $\mathbf{v} = \perp$ **then**
      return $\mathbf{z}_{u-1}$
    else
      Generate a Rademacher random variable ζ
      $\mathbf{z}_u \leftarrow \mathbf{z}_{u-1} + \zeta \eta \widehat{\mathbf{v}}$
    end if
  end if
end for
return

Remark 19 *Note that the gradient complexity in Theorem 18 holds with constant probability* $1/4$*. In practice, we can repeatedly run Algorithm 4 for* $\log(1/p)$ *times to achieve a result that holds with probability at least* $1-p$ *for any* $p \in (0,1)$*. Similar boosting techniques have also been used in Yu et al. (2017); Allen-Zhu and Li (2018); Yu et al. (2018).*

Remark 20 *For finite-sum nonconvex optimization, Theorem 18 suggests that the gradient complexity of Algorithm 4 (SNVRG +* $\text{Neon2}^{\text{finite}}$*) is* $\widetilde{O}(n^{1/2}\epsilon^{-2} + n\epsilon_H^{-3} + n^{3/4}\epsilon_H^{-7/2})$*. In contrast, the gradient complexity of other state-of-the-art local minimum finding algorithms (SVRG +* $\text{Neon2}^{\text{finite}}$*) (Allen-Zhu and Li, 2018) is* $\widetilde{O}(n^{2/3}\epsilon^{-2} + n\epsilon_H^{-3} + n^{3/4}\epsilon_H^{-7/2})$*. Our algorithm is strictly better than that of Allen-Zhu and Li (2018) in terms of the first term in the big O notation.*

If we choose $\epsilon_H = \sqrt{\epsilon}$*, the gradient complexity of our algorithm to find an* $(\epsilon, \sqrt{\epsilon})$*-approximate local minimum turns out to be* $O(n^{1/2}\epsilon^{-2} + n\epsilon^{-3/2} + n^{3/4}\epsilon^{-7/4})$ *and that of SVRG +* $\text{Neon2}^{\text{finite}}$ *is* $O(n^{2/3}\epsilon^{-2} + n\epsilon^{-3/2} + n^{3/4}\epsilon^{-7/4})$*. We compare these two algorithms in Figure 3 when* $\epsilon_H = \sqrt{\epsilon}$ *and make the following comments:*

- *When* $n \gtrsim \epsilon^{-3/2}$*, the gradient complexities of both algorithms are in the same order of* $\widetilde{O}(n\epsilon^{-3/2})$*.*
- *When* $\epsilon^{-1} \lesssim n \lesssim \epsilon^{-3/2}$*, SNVRG +* $\text{Neon2}^{\text{finite}}$ *enjoys* $\widetilde{O}(n\epsilon^{-3/2})$ *gradient complexity, which is strictly better than that of SVRG +* $\text{Neon2}^{\text{finite}}$*, i.e.,* $\widetilde{O}(n^{2/3}\epsilon^{-2})$*.*

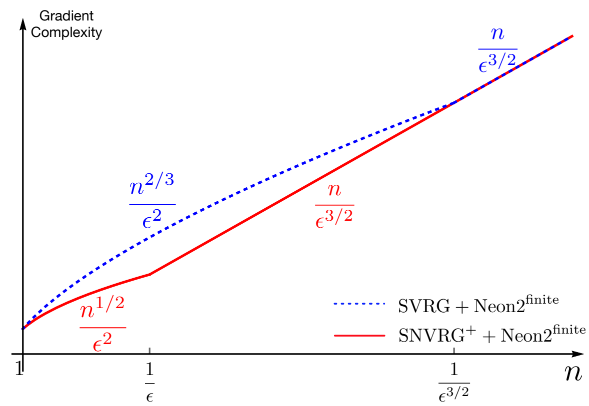


Figure 3: Comparison of gradient complexities between SNVRG + Neon2$^{\text{finite}}$ and SVRG + Neon2$^{\text{finite}}$ for finding an $(\epsilon, \sqrt{\epsilon})$-approximate local minimum in finite-sum optimization problems.

- *Lastly, when $n \lesssim \epsilon^{-1}$, SNVRG + Neon2$^{\textit{finite}}$ achieves $\widetilde{O}(n^{1/2}\epsilon^{-2})$ gradient complexity, which is again better than the gradient complexity of SVRG + Neon2$^{\textit{finite}}$, $\widetilde{O}(n^{2/3}\epsilon^{-2})$, by a factor of $\widetilde{O}(n^{1/6})$.*

In short, our algorithms beats SVRG + Neon2$^{\textit{finite}}$ when $n \lesssim \epsilon^{-3/2}$.

7.2. General Stochastic Optimization Problems

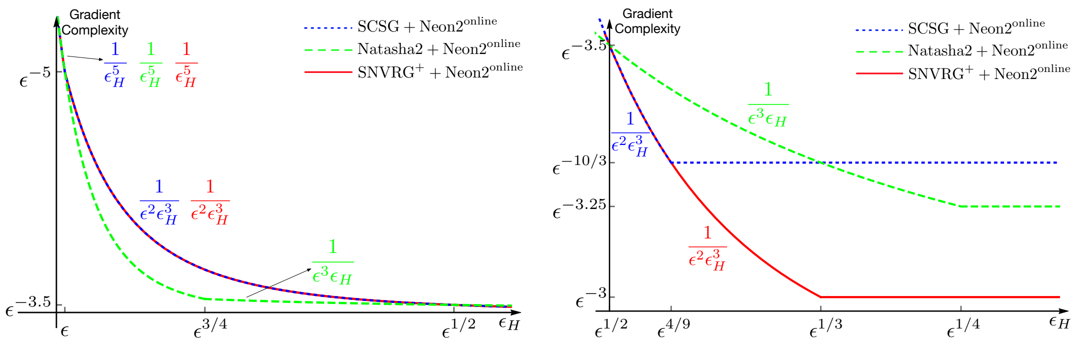
Now we consider the general stochastic optimization problem (2). Recall that in this setting we will call $\nabla F(\mathbf{x}; \xi_i)$ the stochastic gradient at point $\mathbf{x}$ for some random variable ξ_i and index i. Note that the finite-sum optimization problem (1) can be viewed as a special case of the general stochastic optimization problem (2). When n is very large, we can avoid using the full batch size n as suggested in Theorem 21 and instead use batch size $\widetilde{O}(1/\epsilon^2)$ as suggested in the following theorem, by applying Algorithm 5 for the general stochastic setting.

Theorem 21 *Suppose that $F(\mathbf{x}) = \mathbb{E}_{\xi \in \mathcal{D}} F(\mathbf{x}; \xi)$ has σ^2-sub-Gaussian stochastic gradient, where each $F(\mathbf{x}; \xi)$ is L_1-smooth and L_2-Hessian Lipschitz continuous. Let $0 < \epsilon, \epsilon_H < 1$ and*

$$B_0 = \sigma^2 \epsilon^{-2} \cdot \max\bigg\{64\Big(1 + \log\big[2500 C_1 \max\big\{54\sigma^2 L_1^{-1} L_2^2 \epsilon_H^{-3}, 6\big\} \Delta_F L_1 \epsilon^{-2}\big]\Big), 96 C_1\bigg\}, \quad (17)$$

where $C_1 = 200$. Define $\rho = \max\{54\sigma^2 L_1^{-1} L_2^2 \epsilon_H^{-3} B_0^{-1/2}, 6\}$. Set $\delta = 1/(3000 \Delta_F L_2^2 \epsilon_H^{-3})$, the number of epochs $U = 216 \Delta_F L_2^2 \epsilon_H^{-3} + 96 C_1 \rho \Delta_F L_1 B_0^{-1/2} \epsilon^{-2}$, the step size of Algorithm 5 $\eta = \epsilon_H / L_2$ and the step size of One-epoch-SNVRG $M = 2\rho L_1$. Choose all the rest parameters of One-epoch-SNVRG as in Lemma 29. Then with probability at least $1/4$, SNVRG + Neon2$^{\textit{online}}$ will find an (ϵ, ϵ_H)-second-order stationary point within

$$\widetilde{O}\bigg(\frac{\Delta_F L_1^2 L_2^2}{\epsilon_H^5} + \frac{\Delta_F \sigma^2 L_2^2}{\epsilon_H^3 \epsilon^2} + \frac{\Delta_F \sigma L_1}{\epsilon^3}\bigg) \quad (18)$$



(a) $\epsilon_H < \sqrt{\epsilon}$ (b) $\epsilon_H \geq \sqrt{\epsilon}$

Figure 4: Comparison of gradient complexities among SNVRG + Neon2$^{\text{online}}$, SCSG + Neon2$^{\text{online}}$ and Natasha2+Neon2$^{\text{online}}$ for finding an (ϵ, ϵ_H)-approximate second-order stationary point: (a) the comparison when $\epsilon_H < \sqrt{\epsilon}$, and (b) the comparison when $\epsilon_H \geq \sqrt{\epsilon}$.

stochastic gradient evaluations.

The gradient complexity in Theorem 21 again holds with constant probability 1/4 and we can boost it to a high probability using the same trick as we discussed in Remark 19.

Remark 22 *Theorem 21 suggests that the gradient complexity of Algorithm 5 is $\widetilde{O}(\epsilon^{-3} + \epsilon_H^{-5} + \epsilon^{-2}\epsilon_H^{-3})$. In contrast, the gradient complexity of SCSG+Neon2$^{\text{online}}$ (Allen-Zhu and Li, 2018) is $\widetilde{O}(\epsilon^{-10/3} + \epsilon_H^{-5} + \epsilon^{-2}\epsilon_H^{-3})$ and that of Natasha2+Neon2$^{\text{online}}$ (Allen-Zhu, 2018a) is $\widetilde{O}(\epsilon^{-3.25} + \epsilon_H^{-5} + \epsilon^{-3}\epsilon_H)$. Our algorithm is evidently faster than these two algorithms in the first term in the big O notation. We visualize the gradient complexities of these three algorithms in Figure 4. To better visualize the differences, we divide the entire regime of ϵ_H into two regimes: (a) $\epsilon_H < \sqrt{\epsilon}$ and (b) $\epsilon_H \geq \sqrt{\epsilon}$, and plot them separately in Figures 4(a) and 4(b). From Figure 4, we have the following discussion.*

- *When $\epsilon_H \leq \epsilon$, all three algorithms achieve $\widetilde{O}(\epsilon_H^{-5})$ gradient complexity.*
- *When $\epsilon < \epsilon_H < \sqrt{\epsilon}$, both SNVRG + Neon2$^{\text{online}}$ and SCSG+Neon2$^{\text{online}}$ attain $\widetilde{O}(\epsilon^{-2}\epsilon_H^{-3})$ gradient complexity and are worse than Natasha2+Neon2$^{\text{online}}$, which has $\widetilde{O}(\epsilon_H^{-5})$ gradient complexity for $\epsilon_H \in (\epsilon, \epsilon^{3/4})$ and $\widetilde{O}(\epsilon^{-3}\epsilon_H^{-1})$ gradient complexity for $\epsilon_H \in (\epsilon^{3/4}, \epsilon^{1/2})$.*
- *When $\sqrt{\epsilon} \leq \epsilon_H \leq \epsilon^{4/9}$, SNVRG + Neon2$^{\text{online}}$ and SCSG+Neon2$^{\text{online}}$ still attain $\widetilde{O}(\epsilon^{-2}\epsilon_H^{-3})$ gradient complexity but perform better than Natasha2+Neon2$^{\text{online}}$, which has $\widetilde{O}(\epsilon^{-3}\epsilon_H^{-1})$ gradient complexity.*
- *When $\epsilon_H \geq \epsilon^{4/9}$, SNVRG + Neon2$^{\text{online}}$ enjoys a smaller gradient complexity than both SCSG+Neon2$^{\text{online}}$ and Natasha2+Neon2$^{\text{online}}$.*

In particular, when $\epsilon_H = \epsilon^{1/3}$, the gradient complexity of our algorithm SNVRG+Neon2^{online} is smaller than that of SCSG+Neon2^{online} and Natasha2+Neon2^{online} by a factor of $O(\epsilon^{1/3})$. And when $\epsilon_H \geq \epsilon^{1/4}$, SNVRG+Neon2^{online} is faster than Natasha2+Neon2^{online} by a factor of $O(\epsilon^{1/4})$.

8. Theoretical Analysis of SNVRG for Finding Local Minima with Third-Order Smoothness

As we mentioned before, it has been shown that the third-order smoothness of the objective function F can help accelerate the convergence of nonconvex optimization (Carmon et al., 2017; Yu et al., 2018). For the intuition of the acceleration by third-order smoothness, we refer readers to the detailed exhibition and discussion in Yu et al. (2018). In this section, we will show that our local minimum finding algorithms (Algorithms 4 and 5) can find local minima faster provided this additional condition.

8.1. Finite-Sum Optimization Problems

We first consider the finite-sum optimization problem in (1). The following theorem spells out the gradient complexity of Algorithm 4 under additional third-order smoothness.

Theorem 23 *Suppose that $F = 1/n \sum_{i=1}^n f_i$, where each f_i is L_1-smooth, L_2-Hessian Lipschitz continuous and F is L_3-third-order smooth. Let $0 < \epsilon, \epsilon_H < 1$, $\delta = \epsilon_H^2/(72L_3\Delta_F)$ and $U = 12L_3\Delta_F\epsilon_H^{-2} + 1800CL_1\Delta_F\epsilon^{-2}n^{-1/2}$. Set $B_0 = n, M = 6L_1$ and all the rest parameters of One-epoch-SNVRG as in Lemma 29. Choose the step size as $\eta = \sqrt{3\epsilon_H/L_3}$. Then with probability at least $1/4$, SNVRG + Neon2^{finite} will find an (ϵ, ϵ_H)-second-order stationary point within*

$$\widetilde{O}\bigg(\frac{\Delta_F n L_3}{\epsilon_H^2} + \frac{\Delta_F n^{3/4} L_1^{1/2} L_3}{\epsilon_H^{5/2}} + \frac{\Delta_F n^{1/2} L_1}{\epsilon^2}\bigg) \tag{19}$$

stochastic gradient evaluations.

Similar to previous discussions, we can repeatedly run Algorithm 4 for $\log(1/p)$ times to boost its confidence to $1 - p$ for any $p \in (0, 1)$.

Remark 24 *Compared with step size $\eta = \epsilon_H/L_2$ used in the negative curvature descent step (Line 12) of Algorithm 4 in Theorem 18 without third-order smoothness, the step size in Theorem 23 is chosen to be $\eta = \sqrt{\epsilon_H/L_3}$ where L_3 is the third-order smoothness parameter. Note that when $\epsilon_H \ll 1$, the step size we choose under third-order smoothness assumption is much bigger than that under only second-order smoothness assumption. As is pointed out by Yu et al. (2018), the key advantage of third-order smoothness condition is that it enables us to choose a larger step size and therefore achieve much more function value decrease in the negative curvature descent step (Line 12 of Algorithm 4).*

Remark 25 *Theorem 23 suggests that the gradient complexity of SNVRG+ Neon2^{finite} under third-order smoothness is $\widetilde{O}(n^{1/2}\epsilon^{-2} + n\epsilon_H^{-2} + n^{3/4}\epsilon_H^{-5/2})$. In stark contrast, the gradient complexity of the state-of-the-art finite-sum local minimum finding algorithm with third-order smoothness assumption (FLASH) (Yu et al., 2018) is $\widetilde{O}(n^{2/3}\epsilon^{-2} + n\epsilon_H^{-2} + n^{3/4}\epsilon_H^{-5/2})$.*

Clearly, our algorithm is strictly better than the FLASH algorithm (Yu et al., 2018) in the first term of the gradient complexity.

Specifically, if we choose $\epsilon_H = \sqrt{\epsilon}$, SNVRG+ Neon2$^{\text{finite}}$ is faster for finding an $(\epsilon, \sqrt{\epsilon})$-approximate local minimum than FLASH by a factor of $O(1/\epsilon^{1/6})$ when $n \lesssim \epsilon^{-2}$. SNVRG+ Neon2$^{\text{finite}}$ is also strictly faster than FLASH when $\epsilon^{-2} \lesssim n \lesssim \epsilon^{-3}$ and will match FLASH when $n \gtrsim \epsilon^{-3}$. We show this comparison in Figure 5, which clearly demonstrates that the gradient complexity of SNVRG+ Neon2$^{\text{finite}}$ is much smaller than that of FLASH in a very wide regime.

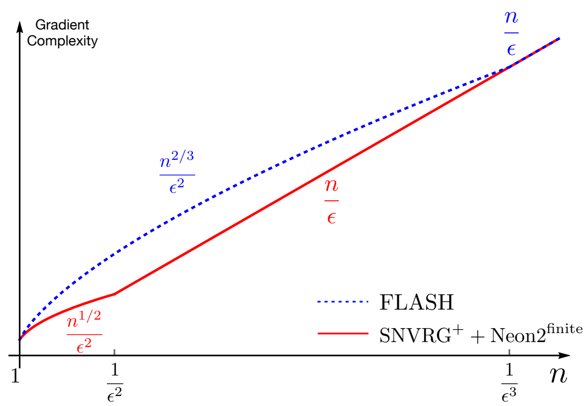


Figure 5: Comparison of gradient complexities between SNVRG + Neon2$^{\text{finite}}$ and FLASH for finding an $(\epsilon, \sqrt{\epsilon})$-approximate local minimum in finite-sum nonconvex optimization problems.

8.2. General Stochastic Optimization Problems

Now we turn to the general stochastic optimization problem (2). We characterize the gradient complexity of Algorithm 5 under third-order smoothness in the following theorem.

Theorem 26 *Suppose that $F(\mathbf{x}) = \mathbb{E}_{\xi \in \mathcal{D}} F(\mathbf{x}; \xi)$ has σ^2-sub-Gaussian stochastic gradient, where each $F(\mathbf{x}; \xi)$ is L_1-smooth, L_2-Hessian Lipschitz continuous and $F(\mathbf{x})$ is L_3-third-order smooth. Let $0 < \epsilon, \epsilon_H < 1$, and*

$$B_0 = \sigma^2 \epsilon^{-2} \cdot \max\bigg\{64\Big(1 + \log\big[2500 C_1 \max\{36\sigma^2 L_1^{-1} L_3 \epsilon_H^{-2}, 6\} \Delta_F L_1 \epsilon^{-2}\big]\Big), 96 C_1\bigg\}, \quad (20)$$

where $C_1 = 200$. Define $\rho = \max\{36\sigma^2 L_1^{-1} L_3 \epsilon_H^{-2} B_0^{-1/2}, 6\}$. Let $\delta = 1/(1000 \Delta_F L_3 \epsilon_H^{-2})$, the number of epochs $U = 72\Delta_F L_3 \epsilon_H^{-2} + 96 C_1 \rho \Delta_F L_1 B_0^{-1/2} \epsilon^{-2}$, the step size $\eta = \sqrt{\epsilon_H / L_3}$ and the step size of One-epoch-SNVRG $M = 2\rho L_1$. Choose all the rest parameters of One-epoch-SNVRG the same as in Lemma 29. Then with probability at least $1/4$, SNVRG+ Neon2$^{\text{online}}$ will find an (ϵ, ϵ_H)-second-order stationary point within

$$\widetilde{O}\bigg(\frac{\Delta_F L_1^2 L_3}{\epsilon_H^4} + \frac{\Delta_F \sigma^2 L_3}{\epsilon_H^2 \epsilon^2} + \frac{\Delta_F \sigma L_1}{\epsilon^3}\bigg) \quad (21)$$

stochastic gradient evaluations.

Remark 27 *Theorem 26 suggests that the gradient complexity of Algorithm 5 under third-order smoothness is $\widetilde{O}(\epsilon^{-3} + \epsilon_H^{-4} + \epsilon^{-2}\epsilon_H^{-2})$. As a comparison, the gradient complexity of existing best stochastic local minimum finding algorithm with third-order smoothness assumption (FLASH) (Yu et al., 2018) is $\widetilde{O}(\epsilon^{-10/3} + \epsilon_H^{-4} + \epsilon^{-2}\epsilon_H^{-2})$. The gradient complexity of SNVRG + Neon2^{online} is faster than that of FLASH in the first term. We illustrate the comparison of gradient complexities of both algorithms in Figure 6. It is evident that when $\epsilon_H \geq \epsilon^{2/3}$, our algorithm SNVRG + Neon2^{online} always enjoys a lower gradient complexity than FLASH. In addition, if we choose $\epsilon_H = \sqrt{\epsilon}$, our algorithm is faster for finding an $(\epsilon, \sqrt{\epsilon})$-approximate local minimum than FLASH (Yu et al., 2018) by a factor of $O(\epsilon^{-1/3})$.*

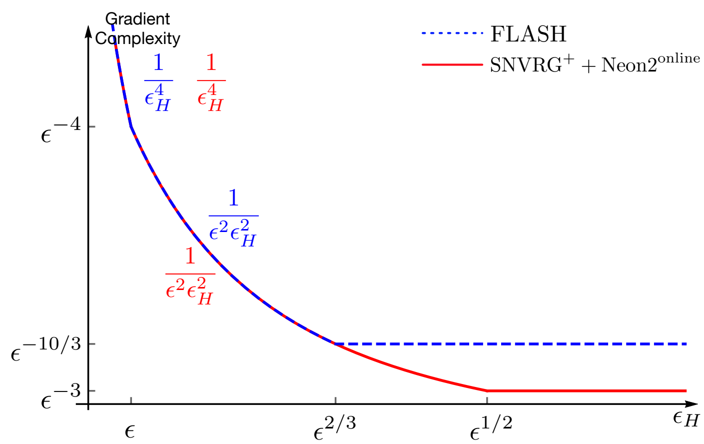


Figure 6: Comparison of gradient complexities between SNVRG + Neon2$^{\text{online}}$ and FLASH for finding an (ϵ, ϵ_H)-approximate second-order stationary point in general stochastic problems.

9. Experiments

In this section, we conduct experiments to validate the superiority of the proposed algorithms. In the first part of this section, we compare our algorithm SNVRG with existing baseline algorithms on training a convolutional neural network for image classification. In the second part of this section, we consider a symmetric matrix sensing problem, where many saddle points exist and thus the proposed SNVRG + Neon2$^{\text{online}}$ is compared with vanilla SNVRG, SGD+NEON and SCSG+Neon2$^{\text{online}}$.

9.1. SNVRG for Training CNNs for Image Classification

We compare the performance of the following algorithms: SGD; SGD with momentum (Qian, 1999) (denoted by SGD-momentum); ADAM (Kingma and Ba, 2014); SCSG Lei et al. (2017). It is worth noting that SCSG is a special case of SNVRG when the number of nested loops $K = 1$. Due to the memory cost, we did not compare Gradient Descent (GD) or SVRG which need to calculate the full gradient. Although our theoretical analysis holds for general K nested loops, it suffices to choose $K = 2$ in SNVRG to illustrate the effectiveness of the nested structure for the simplification of implementation. In this case, we have 3 reference points and gradients. All experiments are conducted on Amazon AWS p2.xlarge servers which comes with Intel Xeon E5 CPU and NVIDIA Tesla K80 GPU (12G GPU RAM). All algorithm are implemented in Pytorch platform version 0.4.0 within Python 3.6.4.

Datasets We use three image datasets: (1) The MNIST dataset (Schölkopf and Smola, 2002) consists of handwritten digits and has $50,000$ training examples and $10,000$ test examples. The digits have been size-normalized to fit the network, and each image is 28 pixels by 28 pixels. (2) CIFAR10 dataset (Krizhevsky, 2009) consists of images in 10 classes and has $50,000$ training examples and $10,000$ test examples. The digits have been size-normalized to fit the network, and each image is 32 pixels by 32 pixels. (3) SVHN dataset Netzer et al. (2011) consists of images of digits and has $531,131$ training examples and $26,032$ test examples. The digits have been size-normalized to fit the network, and each image is 32 pixels by 32 pixels.

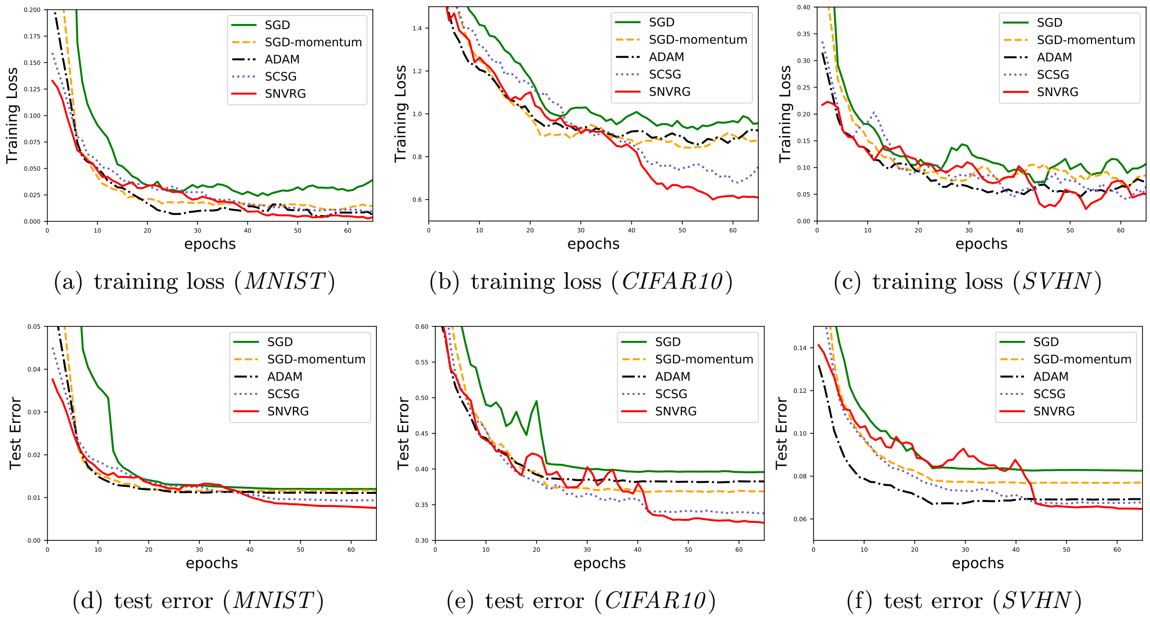
CNN Architecture We use the standard LeNet (LeCun et al., 1998), which has two convolutional layers with 6 and 16 filters of size 5 respectively, followed by three fully-connected layers with output size 120, 84 and 10. We apply max pooling after each convolutional layer.

Implementation Details & Parameter Tuning We did not use the random data augmentation which is set as default by Pytorch, because it will apply random transformation (e.g., clip and rotation) at the beginning of each epoch on the original image dataset, which will ruin the finite-sum structure of the loss function. We set our grid search rules for all three datasets as follows. For SGD, we search the batch size from $\{256, 512, 1024, 2048\}$ and the initial step sizes from $\{1, 0.1, 0.01\}$. For SGD-momentum, we set the momentum parameter as 0.9. We search its batch size from $\{256, 512, 1024, 2048\}$ and the initial learning rate from $\{1, 0.1, 0.01\}$. For ADAM, we search the batch size from $\{256, 512, 1024, 2048\}$ and the initial learning rate from $\{0.01, 0.001, 0.0001\}$. For SCSG and SNVRG, we choose loop parameters $\{T_l\}$ which satisfy $B_l \cdot \prod_{j=1}^{l} T_j = B$ automatically. In addition, for SCSG, we set the batch sizes $(B, B_1) = (B, B/b)$, where b is the batch size ratio parameter. We search B from $\{256, 512, 1024, 2048\}$ and we search b from $\{2, 4, 8\}$. We search its initial learning rate from $\{1, 0.1, 0.01\}$. For our proposed SNVRG algorithm, we set the batch sizes $(B, B_1, B_2) = (B, B/b, B/b^2)$, where b is the batch size ratio parameter. We search B from $\{256, 512, 1024, 2048\}$ and b from $\{2, 4, 8\}$. We search its initial learning rate from $\{1, 0.1, 0.01\}$.

9.1.1. Experimental Results with Learning Rate Decaying

In this section, we first present the experimental results with learning rate decay. In particular, following the convention of deep learning practice, we apply learning rate decay schedule to each algorithm with the learning rate decayed by 0.1 every 20 epochs.

We plotted the training loss and test error for different algorithms on each dataset in Figure 7. The results on MNIST are presented in Figures 7(a) and 7(d); the results on CIFAR10 are in Figures 7(b) and 7(e); and the results on SVHN dataset are shown in



(a) training loss (*MNIST*) (b) training loss (*CIFAR10*) (c) training loss (*SVHN*)

(d) test error (*MNIST*) (e) test error (*CIFAR10*) (f) test error (*SVHN*)

Figure 7: Experiment results on different datasets with learning rate decay. (a) and (d) depict the training loss and test error (top-1 error) v.s. data epochs for training LeNet on MNIST dataset. (b) and (e) depict the training loss and test error v.s. data epochs for training LeNet on CIFAR10 dataset. (c) and (f) depict the training loss and test error v.s. data epochs for training LeNet on SVHN dataset.

Figures 7(c) and 7(f). It can be seen that with learning rate decay schedule, our algorithm SNVRG outperforms all baseline algorithms, which confirms that the use of nested reference points and gradients can accelerate the nonconvex finite-sum optimization.

We would like to emphasize that, while this experiment is on training convolutional neural networks, the major goal of this experiment is to illustrate the advantage of our algorithm and corroborate our theory, rather than claiming a state-of-the-art algorithm for training deep neural networks.

9.1.2. Experimental Results without Learning Rate Decay

We also conducted experiments comparing different algorithms without the learning rate decay schedule. The parameters are tuned by the same grid search described in Section 9. In particular, we summarize the parameters of different algorithms used in our experiments with and without learning rate decay for MNIST in Table 3, CIFAR10 in Table 4, and SVHN in Table 5. We plotted the training loss and test error for each dataset without learning rate decay in Figure 8. The results on MNIST are presented in Figures 8(a) and 8(d); the results on CIFAR10 are in Figures 8(b) and 8(e); and the results on SVHN dataset are shown in Figures 8(c) and 8(f). It can be seen that without learning decay, our algorithm

SNVRG still outperforms all the baseline algorithms except for the training loss on SVHN dataset. However, SNVRG still performs the best in terms of test error on SVHN dataset. These results again suggest that SNVRG can beat the state-of-the-art in practice, which backups our theory.

Table 3: Parameter settings of all algorithms on MNIST dataset.

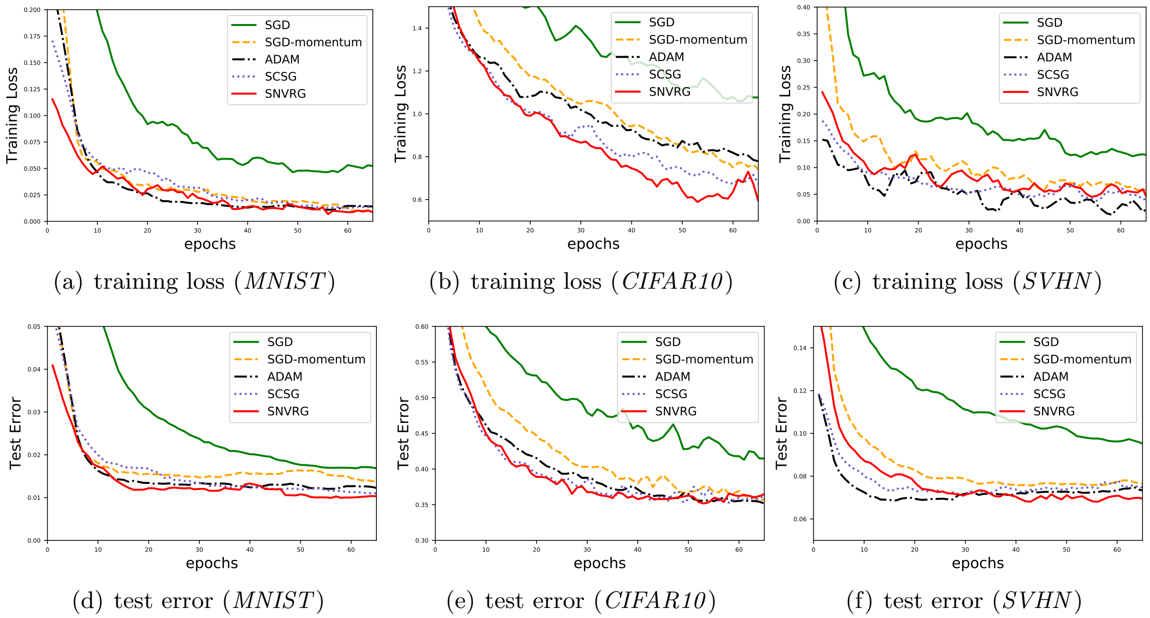
Algorithm	With Learning Rate Decay			Without Learning Rate Decay		
	Initial learning rate η	Batch size B	Batch size ratio b	learning rate η	Batch size B	Batch size ratio b
SGD	0.1	1024	N/A	0.01	1024	N/A
SGD-momentum	0.01	1024	N/A	0.1	1024	N/A
ADAM	0.001	1024	N/A	0.001	1024	N/A
SCSG	0.01	512	8	0.01	512	8
SNVRG	0.01	512	8	0.01	512	8

Table 4: Parameter settings of all algorithms on CIFAR10 dataset.

Algorithm	With Learning Rate Decay			Without Learning Rate Decay		
	Initial learning rate η	Batch size B	Batch size ratio b	learning rate η	Batch size B	Batch size ratio b
SGD	0.1	1024	N/A	0.01	512	N/A
SGD-momentum	0.01	1024	N/A	0.01	2048	N/A
ADAM	0.001	1024	N/A	0.001	2048	N/A
SCSG	0.01	512	8	0.01	512	8
SNVRG	0.01	1024	8	0.01	512	4

Table 5: Parameter settings of all algorithms on SVHN dataset.

Algorithm	With Learning Rate Decay			Without Learning Rate Decay		
	Initial learning rate η	Batch size B	Batch size ratio b	learning rate η	Batch size B	Batch size ratio b
SGD	0.1	2048	N/A	0.01	1024	N/A
SGD-momentum	0.01	2048	N/A	0.01	2048	N/A
ADAM	0.001	1024	N/A	0.001	512	N/A
SCSG	0.01	512	4	0.1	1024	4
SNVRG	0.01	512	8	0.01	512	4



(a) training loss (*MNIST*) (b) training loss (*CIFAR10*) (c) training loss (*SVHN*)

(d) test error (*MNIST*) (e) test error (*CIFAR10*) (f) test error (*SVHN*)

Figure 8: Experimental results on different datasets without learning rate decay. (a) and (d) depict the training loss and test error (top-1 error) v.s. data epochs for training LeNet on MNIST dataset. (b) and (e) depict the training loss and test error v.s. data epochs for training LeNet on CIFAR10 dataset. (c) and (f) depict the training loss and test error v.s. data epochs for training LeNet on SVHN dataset.

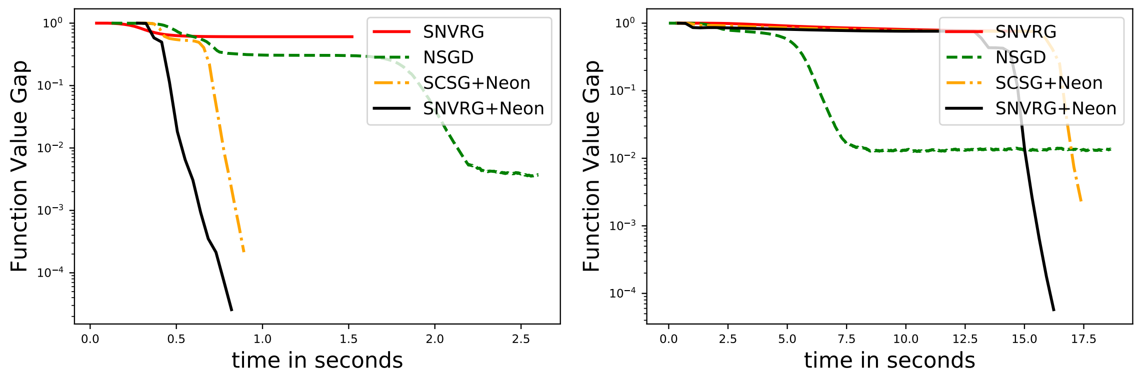
9.2. Experimental Results for Escaping Saddle Points

In this section, we conduct experiments to validate the superiority of our proposed algorithms for escaping from saddle points. We consider the matrix sensing problem, which is defined as follows:

$$\min_{\mathbf{U}\in\mathbb{R}^{d\times r}} f(\mathbf{U}) = \frac{1}{2n}\sum_{i=1}^{n}(\langle \mathbf{A}_i, \mathbf{U}\mathbf{U}^\top\rangle - b_i)^2, \tag{22}$$

where $\{\mathbf{A}_i\}_{i=1}^n$ are sensing matrices, $b_i = \langle \mathbf{A}_i, \mathbf{M}^*\rangle$ is the i-th observation, and $\mathbf{M}^* = \mathbf{U}^*(\mathbf{U}^*)^\top$ is the underlying unknown low-rank matrix. Following the same setting in Yu et al. (2018), we consider two matrix sensing problems: (1) $d = 50, r = 3$ and (2) $d = 100, r = 3$. We generate $n = 20d$ sensing matrices $\{\mathbf{A}_i\}_{i=1}^n$, where each entry of $\mathbf{A}_i$ follows the standard normal distribution. We generate $\mathbf{U}^*$ randomly where each row of $\mathbf{U}^*$ follows the standard normal distribution. We generate $\mathbf{u}_0$ from standard normal distribution and set the initial point as $\mathbf{U}_0 = [\mathbf{u}_0, \mathbf{0}, \ldots, \mathbf{0}]$.

We compare our algorithm SNVRG+Neon with following baselines for nonconvex optimization problems: SNVRG, noisy stochastic gradient descent (NSGD) (Ge et al., 2015), and Stochastically Controlled Stochastic Gradient with Neon (SCSG-Neon) (Xu et al.,



(a) $d = 50, r = 3$

(b) $d = 100, r = 3$

Figure 9: Experimental results on matrix sensing problems. (a) depicts matrix sensing problem with $d = 50, r = 3$. (b) depicts matrix sensing problem with $d = 100, r = 3$.

2018b; Allen-Zhu and Li, 2018). For the simplicity, we choose the gradient batch size to be 100 for all algorithms. For SCSG-Neon, we set the outer batch size to be n. For SNVRG and SNVRG+Neon, we choose $K = 2$ and set $(B, B_1) = (n, n/5)$. We apply Oja's algorithm (Oja, 1982) to calculate the negative curvature with a Hessian mini-batch size of 100. We perform a grid search over step sizes for all algorithms. We report the objective function value versus CPU running time.

The experimental results are shown in Figures 9(a) and 9(b). From the figures we can see that without adding additional noise or using negative curvature information, SNVRG tends to get stuck in saddle points. In sharp contrast, NSGD, SCSG-Neon and SNVRG-Neon are able to escape from saddle points. We also notice that SNVRG-Neon outperforms all other baseline algorithms in both problem settings.

10. Summary and Conclusion

In this work, we study nonconvex optimization problems (1) and (2). In the first part of this paper (Sections 4 and 5), we propose the stochastic nested variance-reduced gradient descent algorithm (SNVRG) for finding an ϵ-approximate first-order stationary point of the nonconvex optimization problems. SNVRG is a natural extension of the original SVRG algorithm proposed by Johnson and Zhang (2013) but utilizes multiple reference points and reference gradients to reduce the variance in the semi-stochastic gradient used in the update rule. We prove that SNVRG converges to an ϵ-approximate first-order stationary point after $\widetilde{O}(n \wedge \epsilon^{-2} + \epsilon^{-3} \wedge n^{1/2}\epsilon^{-2})$ stochastic gradient evaluations. This gradient complexity improves all existing first-order methods in nonconvex optimization such as SGD (Robbins and Monro, 1951), SVRG (Reddi et al., 2016a; Allen-Zhu and Hazan, 2016) and SCSG (Lei et al., 2017), and matches the lower bound provided in Fang et al. (2018); Zhou and Gu (2019).

In the second part of this paper (Sections 6, 7 and 8), we integrate SNVRG with recently proposed NEON/Neon2 algorithms (Xu et al., 2018b; Allen-Zhu and Li, 2018) and propose a class of algorithms that can find local minima, i.e., $(\epsilon, \sqrt{\epsilon})$-approximate second-order stationary points of the nonconvex optimization problems. The proposed algorithms $\text{SNVRG} + \text{Neon2}^{\text{finite}}$ and $\text{SNVRG} + \text{Neon2}^{\text{online}}$ achieve the state-of-the-art gradient complexities for finding local minima in nonconvex optimization. Detailed comparison is presented in Table 2. Furthermore, we provide an alternative analysis of these two algorithms when the objective function enjoys the third-order smoothness property (Anandkumar and Ge, 2016; Carmon et al., 2017; Yu et al., 2018). With this property, we prove that $\text{SNVRG} + \text{Neon2}^{\text{finite}}$ and $\text{SNVRG} + \text{Neon2}^{\text{online}}$ attain lower gradient complexities and can find local minima more efficiently.

Acknowledgement

We would like to thank the anonymous reviewers for their helpful comments. This research was sponsored in part by the National Science Foundation BIGDATA IIS-1855099, IIS-1904183 and IIS-1906169. We also thank AWS for providing cloud computing credits associated with the NSF BIGDATA award. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

Appendix A. Proof of Main Theory for Finding Stationary Points

In this section, we provide the proofs of our theoretical analysis in Section 5 for finding first-order stationary points.

A.1. Proof of Theorem 12

We start with the following supporting lemma that characterizes the function value decrease of One-epoch-SNVRG (Algorithm 1).

Lemma 28 *Suppose that F has averaged L-Lipschitz gradient. Suppose that $B_0 \geq 4$ and the rest parameters $(K, M, \{B_l\}, \{T_l\})$ of Algorithm 1 are chosen the same as in (12). Then Algorithm 1 with Option I satisfies*

$$\mathbb{E}\|\nabla F(\mathbf{x}_{out})\|_2^2 \leq C\bigg(\frac{L}{B_0^{1/2}} \cdot \mathbb{E}\big[F(\mathbf{x}_0) - F(\mathbf{x}_T)\big] + \frac{\sigma^2}{B_0} \cdot \mathbb{1}(B_0 < n)\bigg) \tag{23}$$

within $1 \vee (10B_0 \log^3 B_0)$ stochastic gradient computations, where $T = \prod_{l=1}^{K} T_l$, $C = 6000$ is a constant and $\mathbb{1}(\cdot)$ is the indicator function.

Now we prove our main theorem which spells out the gradient complexity of SNVRG.
Proof [Proof of Theorem 12] By (23) we have

$$\mathbb{E}\|\nabla F(\mathbf{y}_s)\|_2^2 \leq C\bigg(\frac{L}{B_0^{1/2}} \cdot \mathbb{E}\big[F(\mathbf{z}_{s-1}) - F(\mathbf{z}_s)\big] + \frac{\sigma^2}{B_0} \cdot \mathbb{1}(B_0 < n)\bigg), \tag{24}$$

where $C = 6000$. Taking summation for (24) over s from 1 to S, we have

$$\sum_{s=1}^{S} \mathbb{E}\|\nabla F(\mathbf{y}_s)\|_2^2 \leq C\bigg(\frac{L}{B_0^{1/2}} \cdot \mathbb{E}\big[F(\mathbf{z}_0) - F(\mathbf{z}_S)\big] + \frac{\sigma^2}{B_0} \cdot \mathbb{1}(B_0 < n) \cdot S\bigg). \tag{25}$$

Dividing both sides of (25) by S, we immediately obtain

$$\mathbb{E}\|\nabla F(\mathbf{y}_{\text{out}})\|_2^2 \leq C\bigg(\frac{L\mathbb{E}\big[F(\mathbf{z}_0) - F^*\big]}{SB_0^{1/2}} + \frac{\sigma^2}{B_0} \cdot \mathbb{1}(B_0 < n)\bigg) \tag{26}$$

$$= C\bigg(\frac{L\Delta_F}{SB_0^{1/2}} + \frac{\sigma^2}{B_0} \cdot \mathbb{1}(B_0 < n)\bigg), \tag{27}$$

where (26) holds because $F(\mathbf{z}_S) \geq F^*$ and by the definition $\Delta_F = F(\mathbf{z}_0) - F^*$. By the choice of parameters in Theorem 12, we have $B_0 = n \wedge (2C\sigma^2/\epsilon^2), S = 1 \vee (2CL\Delta_F/(B_0^{1/2}\epsilon^2))$, which implies

$$\mathbb{1}(B_0 < n) \cdot \sigma^2/B_0 \leq \epsilon^2/(2C), \quad \text{and} \quad L\Delta_F/(SB_0^{1/2}) \leq \epsilon^2/(2C). \tag{28}$$

Submitting (28) into (27), we have $\mathbb{E}\|\nabla F(\mathbf{y}_{\text{out}})\|_2^2 \leq 2C\epsilon^2/(2C) = \epsilon^2$. By Lemma 23, we have that each One-epoch-SNVRG takes less than $7B_0 \log^3 B_0$ stochastic gradient computations. Since we have total S epochs, so the total gradient complexity of Algorithm 2 is less than

$$S \cdot 7B_0 \log^3 B_0 \leq 7B_0 \log^3 B_0 + \frac{L\Delta_F}{\epsilon^2} \cdot 7B_0^{1/2} \log^3 B_0$$
$$= O\bigg(\log^3\bigg(\frac{\sigma^2}{\epsilon^2} \wedge n\bigg)\bigg[\frac{\sigma^2}{\epsilon^2} \wedge n + \frac{L\Delta_F}{\epsilon^2}\bigg[\frac{\sigma^2}{\epsilon^2} \wedge n\bigg]^{1/2}\bigg]\bigg),$$

which leads to the conclusion. ■

A.2. Proof of Theorem 14

We then prove the main theorem on gradient complexity of SNVRG under gradient dominance condition (Algorithm 3).

Proof [Proof of Theorem 14] Following the proof of Theorem 12, we obtain a similar inequality with (26):

$$\mathbb{E}\|\nabla F(\mathbf{z}_{u+1})\|_2^2 \leq C\bigg(\frac{L\mathbb{E}[F(\mathbf{z}_u) - F^*]}{SB_0^{1/2}} + \frac{\sigma^2}{B_0} \cdot \mathbb{1}(B_0 < n)\bigg). \tag{29}$$

Since F is a τ-gradient dominated function, we have $\mathbb{E}\|\nabla F(\mathbf{z}_{u+1})\|_2^2 \geq 1/\tau \cdot \mathbb{E}[F(\mathbf{z}_{u+1}) - F^*]$ by Definition 9. Plugging this inequality into (29) yields

$$\mathbb{E}\big[F(\mathbf{z}_{u+1}) - F^*\big] \leq \frac{C\tau L}{SB_0^{1/2}} \cdot \mathbb{E}\big[F(\mathbf{z}_u) - F^*\big] + \frac{C\tau\sigma^2}{B_0} \cdot \mathbb{1}(B_0 < n)$$

$$\leq \frac{1}{2}\mathbb{E}\big[F(\mathbf{z}_u) - F^*\big] + \frac{\epsilon}{4}, \tag{30}$$

where the second inequality holds due to the choice of parameters $B_0 = n \wedge (4C_1\tau\sigma^2/\epsilon)$ and $S = 1 \vee (2C_1\tau L/B_0^{1/2})$ for Algorithm 3 in Theorem 14. By (30) we can derive

$$\mathbb{E}\big[F(\mathbf{z}_{u+1}) - F^*\big] - \frac{\epsilon}{2} \leq \frac{1}{2}\bigg(\mathbb{E}\big[F(\mathbf{z}_u) - F^*\big] - \frac{\epsilon}{2}\bigg),$$

which immediately implies

$$\mathbb{E}\big[F(\mathbf{z}_U) - F^*\big] - \frac{\epsilon}{2} \leq \frac{1}{2^U}\bigg(\Delta_F - \frac{\epsilon}{2}\bigg) \leq \frac{\Delta_F}{2^U}. \tag{31}$$

Plugging the number of epochs $U = \log(2\Delta_F/\epsilon)$ into (31), we obtain $\mathbb{E}\big[F(\mathbf{z}_U) - F^*\big] \leq \epsilon$. Note that each epoch of Algorithm 3 needs at most $S \cdot 7B_0 \log^3 B_0$ stochastic gradient computations by Theorem 12 and Algorithm 3 has U epochs, which implies the total stochastic gradient complexity

$$U \cdot S \cdot 7B_0 \log^3 B_0 = O\Bigg(\log^3\bigg(n \wedge \frac{\tau\sigma^2}{\epsilon}\bigg)\log\frac{\Delta_F}{\epsilon}\bigg[n \wedge \frac{\tau\sigma^2}{\epsilon} + \tau L\bigg[n \wedge \frac{\tau\sigma^2}{\epsilon}\bigg]^{1/2}\bigg]\Bigg),$$

which completes the proof. ∎

Appendix B. Proof of Main Theory for Finding Local Minima

In this section, we provide the proofs of gradient complexities of our proposed algorithms SNVRG + Neon2$^{\text{finite}}$ and SNVRG + Neon2$^{\text{online}}$.

B.1. Proof of Theorem 18

It is worth noting that in order to find local minima we apply One-epoch-SNVRG with Option II which samples the total number of epochs T from a geometric distribution. Similar to the analysis for finding first-order stationary points, we also have the following supporting lemma about the function value decrease of Algorithm 1.

Lemma 29 *Suppose that each f_i is L_1-smooth and F has σ^2-sub-Gaussian stochastic gradient. In Algorithm 1, suppose that $B_0 \geq 4$ and the rest parameters $(K, M, \{B_l\}, \{T_l\})$ of Algorithm 1 are chosen the same as in* (12)*. Then Algorithm 1 with Option II satisfies*

$$\mathbb{E}\|\nabla F(\mathbf{x}_T)\|_2^2 \leq C\bigg(\frac{M}{B_0^{1/2}} \cdot \mathbb{E}[F(\mathbf{x}_0) - F(\mathbf{x}_T)] + \frac{2\sigma^2}{B_0} \cdot \mathbb{1}\{B_0 < n\}\bigg), \tag{32}$$

where $C = 1000$. In addition, the total number of stochastic gradient computations $\mathcal{T}$ by Algorithm 1 satisfies $\mathbb{E}\mathcal{T} \leq 10B_0 \log^3 B_0$.

Remark 30 *Note that Lemma 28 is regarding $\mathbf{x}_{out}$, which is a uniformly chosen iterate from $\mathbf{x}_1, \ldots, \mathbf{x}_T$. In contrast, Lemma 29 is regarding the last iterate of $\mathbf{x}_T$ in Algorithm 1. This difference leads to the nonergodic-type and ergodic-type guarantees of One-epoch-SNVRG which plays different roles in the analysis of stationary point finding algorithms and local minimum finding algorithms.*

Remark 31 *For simplicity, we use $\nabla f_i(\mathbf{x})$ to denote the stochastic gradient at point $\mathbf{x}$ in our One-epoch-SNVRG algorithm (Lines 20 and 23 in Algorithm 1) and the analysis of Lemma 29. However, we emphasize that One-epoch-SNVRG also works in the general stochastic optimization setting if we replace $\nabla f_i(\mathbf{x})$ with $\nabla F(\mathbf{x};\xi_i)$ for any index i. And the theoretical result in Lemma 29 still holds.*

When $F(\mathbf{x})$ has the finite-sum structure in (1), we choose $B_0 = n, M = 6L_1$ in One-epoch-SNVRG. Lemma 29 straightforwardly implies the following corollary.

Corollary 32 *Suppose that each f_i is L_1-smooth. We choose $B_0 = n$, and let other parameters be chose as in Lemma 29. Then the output of Algorithm 1 with Option II satisfies*

$$\mathbb{E}\|\nabla F(\mathbf{x}_T)\|_2^2 \le \frac{CL_1}{n^{1/2}} \cdot \mathbb{E}\big[F(\mathbf{x}_0) - F(\mathbf{x}_T)\big],$$

where $C = 6000$. Let $\mathcal{T}$ be the total amount of stochastic gradient computations of Algorithm 1, then we have $\mathbb{E}\mathcal{T} \le 10n\log^3 n$.

The following lemma shows that based on $\text{Neon2}^{\text{finite}}$ the negative curvature descent step of Algorithm 4 (Line 12) enjoys sufficient function value decrease. The proof can be found in Theorem 5 and Claim C.2 in Allen-Zhu and Li (2018).

Lemma 33 (Allen-Zhu and Li (2018)) *Suppose that $F = 1/n\sum_{i=1}^n f_i$, each f_i is L_1-smooth and L_2-Hessian Lipschitz continuous. Let $\epsilon_H \in (0,1)$ and set $\eta = \epsilon_H/L_2$. Suppose that $\lambda_{min}(\nabla^2 F(\mathbf{z}_{u-1})) < -\epsilon_H$ and that at the u-th iteration Algorithm 4 executes the $\text{Neon2}^{\text{finite}}$ algorithm (Line 7). Then with probability $1-\delta$ it holds that*

$$\mathbb{E}_\zeta\big[F(\mathbf{z}_u) - F(\mathbf{z}_{u-1})\big] \le -\epsilon_H^3/(12L_2^2).$$

In addition, $\text{Neon2}^{\text{finite}}$ takes $O\big((n + n^{3/4}\sqrt{L_1/\epsilon_H})\log^2(d/\delta)\big)$ stochastic gradient computations.

Proof [Proof of Theorem 18] Let $\mathcal{I} = \{1, \ldots, U\}$ be the index set of all iterations. We denote $\mathcal{I}_1$ and $\mathcal{I}_2$ as the index sets such that $\mathbf{z}_u$ is obtained from $\text{Neon2}^{\text{finite}}$ for all $u \in \mathcal{I}_1$ and $\mathbf{z}_{u'}$ is the output by SNVRG for all $u' \in \mathcal{I}_2$. Obviously we have $U = |\mathcal{I}_1| + |\mathcal{I}_2|$. We will calculate $|\mathcal{I}_1|, |\mathcal{I}_2|$ separately. For $|\mathcal{I}_1|$, by Lemma 33, with probability $1-\delta$, we have

$$\mathbb{E}\big[F(\mathbf{z}_u) - F(\mathbf{z}_{u-1})\big] \le -\epsilon_H^3/(12L_2^2), \qquad \text{for } u \in \mathcal{I}_1. \tag{33}$$

Summing up (33) over $u \in \mathcal{I}_1$, then with probability $1 - \delta\cdot|\mathcal{I}_1|$ we have

$$|\mathcal{I}_1|\cdot\epsilon_H^3/(12L_2^2) \le \sum_{u\in\mathcal{I}_1}\mathbb{E}\big[F(\mathbf{z}_{u-1}) - F(\mathbf{z}_u)\big] \le \sum_{u\in\mathcal{I}}\mathbb{E}\big[F(\mathbf{z}_{u-1}) - F(\mathbf{z}_u)\big] \le \Delta_F, \tag{34}$$

where the second inequality holds because by Corollary 32 it holds that

$$0 \le \mathbb{E}\|\nabla F(\mathbf{z}_u)\|_2^2 \le \frac{CL_1}{n^{1/2}}\mathbb{E}\big[F(\mathbf{z}_{u-1}) - F(\mathbf{z}_u)\big], \qquad \text{for all } u \in \mathcal{I}_2. \tag{35}$$

By (34), we have

$$|\mathcal{I}_1| \le 12L_2^2\Delta_F/\epsilon_H^3.$$

To calculate $|\mathcal{I}_2|$, we further decompose $\mathcal{I}_2$ into two disjoint sets such that $\mathcal{I}_2 = \mathcal{I}_2^1 \cup \mathcal{I}_2^2$, where $\mathcal{I}_2^1 = \{u \in \mathcal{I}_2 : \|\mathbf{g}_u\|_2 > \epsilon\}$, $\mathcal{I}_2^2 = \{u \in \mathcal{I}_2 : \|\mathbf{g}_u\|_2 \le \epsilon\}$. It is worth noting that if $u \in \mathcal{I}_2^2$ such that $\|\mathbf{g}_u\|_2 \le \epsilon$, then Algorithm 4 will execute Neon2$^{\text{finite}}$ and a negative curvature descent step, which means $u+1 \in \mathcal{I}_1$ by definition. Thus, it always holds that $|\mathcal{I}_2^2| \le |\mathcal{I}_1|$. For $|\mathcal{I}_2^1|$, note that $\mathbf{x}_0 = \mathbf{z}_{u-1}$ and $\mathbf{x}_T = \mathbf{z}_u$ in Corollary 32, which directly implies

$$\begin{aligned}
\sum_{u\in\mathcal{I}_2^1} \mathbb{E}\|\nabla F(\mathbf{z}_u)\|_2^2 &\le \sum_{u\in\mathcal{I}_2^1} \frac{CL_1}{n^{1/2}}\mathbb{E}\big[F(\mathbf{z}_{u-1}) - F(\mathbf{z}_u)\big] \\
&\le \sum_{u\in\mathcal{I}} \frac{CL_1}{n^{1/2}}\mathbb{E}\big[F(\mathbf{z}_{u-1}) - F(\mathbf{z}_u)\big] \\
&\le \frac{CL_1}{n^{1/2}} \cdot \Delta_F,
\end{aligned} \tag{36}$$

where the second inequality holds because $\mathbb{E}[F(\mathbf{z}_{u-1}) - F(\mathbf{z}_u)] \ge 0$ for $u \in \mathcal{I}_1 \cup \mathcal{I}_2$ by (33) and (35). Applying Markov's inequality, with probability at least $2/3$, we have

$$\sum_{u\in\mathcal{I}_2^1} \|\nabla F(\mathbf{z}_u)\|_2^2 \le \frac{3CL_1\Delta_F}{n^{1/2}}.$$

Since for any $u \in \mathcal{I}_2^1$, we have $\|\nabla F(\mathbf{z}_u)\|_2 = \|\mathbf{g}_u\|_2 > \epsilon$, with probability at least $2/3$ it holds that

$$|\mathcal{I}_2^1| \le \frac{3CL_1\Delta_F}{\epsilon^2 n^{1/2}}.$$

Thus, the total number of iterations is $U = |\mathcal{I}_1| + |\mathcal{I}_2| \le 2|\mathcal{I}_1| + |\mathcal{I}_2^1| \le 24L_2^2\Delta_F\epsilon_H^{-3} + 3CL_1\Delta_F\epsilon^{-2}n^{-1/2}$.

We now calculate the gradient complexity of Algorithm 4. By Corollary 32 one single call of One-epoch-SNVRG needs at most $20n\log^3 n$ stochastic gradient computations and by Lemma 33 one single call of Neon2$^{\text{finite}}$ needs $O\big((n + n^{3/4}\sqrt{L_1/\epsilon_H})\log^2(d/\delta)\big)$ stochastic gradient computations. In addition, we need to compute $\mathbf{g}_u$ at each iteration of Algorithm 4 (Line 3), which takes $O(n)$ stochastic gradient computations. Thus, the expectation of the total amount of stochastic gradient computations, denoted by $\mathbb{E}T_{\text{total}}$, can be upper bounded by

$$\begin{aligned}
&|\mathcal{I}_1| \cdot O\big((n + n^{3/4}\sqrt{L_1/\epsilon_H})\log^2(d/\delta)\big) + |\mathcal{I}_2| \cdot O(n\log^3 n) + |\mathcal{I}| \cdot O(n) \\
&= |\mathcal{I}_1| \cdot \widetilde{O}\big(n + n^{3/4}\sqrt{L_1/\epsilon_H}\big) + (|\mathcal{I}_2^1| + |\mathcal{I}_2^2|) \cdot \widetilde{O}(n)
\end{aligned}$$

$$= |\mathcal{I}_1| \cdot \widetilde{O}\big(n + n^{3/4}\sqrt{L_1/\epsilon_H}\big) + (|\mathcal{I}_2^1| + |\mathcal{I}_1|) \cdot \widetilde{O}(n). \tag{37}$$

We further plug the upper bound for $|\mathcal{I}_1|$ and $|\mathcal{I}_2^1|$ into (37) and obtain

$$\begin{aligned}\mathbb{E}T_{\text{total}} &= O(L_2^2\Delta_F\epsilon_H^{-3}) \cdot \widetilde{O}\big(n + n^{3/4}\sqrt{L_1/\epsilon_H}\big) + O(L_1\Delta_F\epsilon^{-2}n^{-1/2})\widetilde{O}(n) \\ &= \widetilde{O}\big(\Delta_F n L_2^2\epsilon_H^{-3} + \Delta_F n^{3/4}L_1^{1/2}L_2^2\epsilon_H^{-7/2} + \Delta_F n^{1/2}L_1\epsilon^{-2}\big).\end{aligned}$$

Finally, applying Markov inequality, with probability 2/3, it holds that

$$T_{\text{total}} = \widetilde{O}\big(\Delta_F n L_2^2\epsilon_H^{-3} + \Delta_F n^{3/4}L_1^{1/2}L_2^2\epsilon_H^{-7/2} + \Delta_F n^{1/2}L_1\epsilon^{-2}\big).$$

Since $|\mathcal{I}_1|\delta = |\mathcal{I}_1|/(144 \cdot L_2^2\Delta_F\epsilon_H^{-3}) \le 1/12$, then by the union bound, with probability $1-1/3-1/3-|\mathcal{I}_1|\delta \ge 1/4$, SNVRG+Neon2$^{\text{finite}}$ will find an (ϵ, ϵ_H)-second order stationary point within

$$\widetilde{O}\big(\Delta_F n L_2^2\epsilon_H^{-3} + \Delta_F n^{3/4}L_1^{1/2}L_2^2\epsilon_H^{-7/2} + \Delta_F n^{1/2}L_1\epsilon^{-2}\big)$$

stochastic gradient computations. ■

B.2. Proof of Theorem 21

For general stochastic problem in (2), we denote $\nabla F(\mathbf{x};\xi_i)$ as one subsampled gradient at $\mathbf{x}$ for any random variable ξ and index i in One-epoch-SNVRG. Base on Lemma 28 we have the next corollary.

Corollary 34 *Suppose that for each ξ, $F(\mathbf{x};\xi)$ is L_1-smooth and has σ^2-sub-Gaussian stochastic gradient. We choose $M = 2\rho L_1$ and suppose that $n \gg O(\epsilon^{-2})$ and $B_0 < n$. Then the output of Algorithm 1 with Option II satisfies*

$$\mathbb{E}\|\nabla F(\mathbf{x}_T)\|_2^2 \le C_1\bigg(\frac{\rho L_1}{B_0^{1/2}} \cdot \mathbb{E}\big[F(\mathbf{x}_0) - F(\mathbf{x}_T)\big] + \frac{\sigma^2}{B_0}\bigg),$$

where $C_1 = 2000$. The total amount of stochastic gradient computations of Algorithm 1 is $\mathbb{E}\mathcal{T} \le 20B_0\log^3 B_0$.

The following lemma shows that based on Neon2$^{\text{online}}$ the negative curvature descent step of Algorithm 5 (Line 13) enjoys sufficient function value decrease. More detailed about Neon2$^{\text{online}}$ can be found in Algorithm 7. The proof can be found as a combination of Theorem 1, Lemma 3.1 and Claim C.2 in Allen-Zhu and Li (2018).

Lemma 35 (Allen-Zhu and Li (2018)) *Suppose that $F(\mathbf{x}) = \mathbb{E}_{\xi\in\mathcal{D}}F(\mathbf{x};\xi)$ and each $F(\mathbf{x};\xi)$ is L_1-smooth, L_2-Hessian Lipschitz continuous. Let $\epsilon_H \in (0,1)$ and set $\eta = \epsilon_H/L_2$. Suppose that $\lambda_{min}(\nabla^2 F(\mathbf{z}_{u-1})) < -\epsilon_H$ and that at the u-th iteration Algorithm 5 executes the Neon2^{online} algorithm (Line 8). Then with probability $1-\delta$ it holds that*

$$\mathbb{E}_\zeta\big[F(\mathbf{z}_u) - F(\mathbf{z}_{u-1})\big] \le -\epsilon_H^3/(12L_2^2).$$

In addition, Neon2^{online} takes $O(L_1^2/\epsilon_H^2\log^2(d/\delta))$ stochastic gradient computations.

We also need the following concentration inequality in our proof.

Lemma 36 (Ghadimi et al. (2016)) *Suppose the stochastic gradient $\nabla F(\mathbf{x};\xi)$ is σ^2-sub-Gaussian. Let $\nabla F_{\mathcal{S}}(\mathbf{x}) = 1/|\mathcal{S}|\sum_{i\in\mathcal{S}}\nabla F(\mathbf{x};\xi_i)$, where $\mathcal{S}$ is a subsampled gradient of $F(\mathbf{x})$. If the sample size satisfies $|\mathcal{S}| = 2\sigma^2/\epsilon^2(1+\log^{1/2}(1/\delta))^2$, then with probability at least $1-\delta$,*

$$\|\nabla F_{\mathcal{S}}(\mathbf{x}) - \nabla F(\mathbf{x})\|_2 \le \epsilon.$$

Proof [Proof of Theorem 21] Denote $\delta_0 = 1/(2500C_1\rho\Delta_F L_1 B_0^{-1/2}\epsilon^{-2})$. Then by the choice of B_0 in (17) it holds that $B_0 > 32\sigma^2/\epsilon^2(1+\log^{1/2}(1/\delta_0))^2$. Let $\mathcal{I} = \{1,\ldots,U\}$ be the index set of all iterations. We use $\mathcal{I}_1$ and $\mathcal{I}_2$ to represent the index set of iterates where the $\mathbf{z}_u$ is obtained from Neon2$^{\text{finite}}$ and SNVRG respectively. From Lemma 35, we have that with probability at least $1-\delta$ that

$$\mathbb{E}\big[F(\mathbf{z}_u) - F(\mathbf{z}_{u-1})\big] \le -\epsilon_H^3/(12L_2^2), \qquad \text{for } u\in\mathcal{I}_1. \tag{38}$$

By Corollary 34, we have

$$\mathbb{E}\|\nabla F(\mathbf{z}_u)\|_2^2 \le C_1\bigg(\frac{\rho L_1}{B_0^{1/2}}\cdot\mathbb{E}\big[F(\mathbf{z}_{u-1}) - F(\mathbf{z}_u)\big] + \frac{\sigma^2}{B_0}\bigg), \qquad \text{for } u\in\mathcal{I}_2. \tag{39}$$

where $C_1 = 200$. We further decompose $\mathcal{I}_2 = \mathcal{I}_2^1\cup\mathcal{I}_2^2$, where $\mathcal{I}_2^1 = \{u\in\mathcal{I}_2 : \|\mathbf{g}_u\|_2 > \epsilon/2\}$ and $\mathcal{I}_2^1 = \{u\in\mathcal{I}_2 : \|\mathbf{g}_u\|_2 \le \epsilon/2\}$. (39) immediately implies the following two inequalities:

$$\mathbb{E}\big[F(\mathbf{z}_u) - F(\mathbf{z}_{u-1})\big] \le -\frac{B_0^{1/2}}{C_1\rho L_1}\mathbb{E}\|\nabla F(\mathbf{z}_u)\|_2^2 + \frac{\sigma^2}{\rho L_1 B_0^{1/2}}, \qquad u\in\mathcal{I}_2^1, \tag{40}$$

$$\mathbb{E}\big[F(\mathbf{z}_u) - F(\mathbf{z}_{u-1})\big] \le \frac{\sigma^2}{\rho L_1 B_0^{1/2}}, \qquad u\in\mathcal{I}_2^2. \tag{41}$$

Summing up (38) over $u\in\mathcal{I}_1$, (40) over $u\in\mathcal{I}_2^1$ and (41) over $u\in\mathcal{I}_2^2$, we have

$$\begin{aligned}\mathbb{E}\bigg[\sum_{u\in\mathcal{I}}F(\mathbf{z}_{u-1}) - F(\mathbf{z}_u)\bigg] &\ge \frac{|\mathcal{I}_1|\epsilon_H^3}{12L_2^2} + \frac{B_0^{1/2}}{C_1\rho L_1}\sum_{u\in\mathcal{I}_2^1}\mathbb{E}\|\nabla F(\mathbf{z}_u)\|_2^2 \\ &\quad - \sum_{u\in\mathcal{I}_2^1}\frac{\sigma^2}{\rho L_1 B_0^{1/2}} - \sum_{u\in\mathcal{I}_2^2}\frac{\sigma^2}{\rho L_1 B_0^{1/2}}.\end{aligned} \tag{42}$$

Since for any $u\in\mathcal{I}_2^2$ we have $\|\mathbf{g}_u\|_2 \le \epsilon/2$, Algorithm 5 will execute Neon2$^{\text{online}}$ at the u-th iteration, which indicates $|\mathcal{I}_2^2| \le |\mathcal{I}_1|$. Combining this with (42) and by the definition of Δ_F, with probability at least $1-|\mathcal{I}_1|\delta$, we have

$$\frac{|\mathcal{I}_1|\epsilon_H^3}{12L_2^2} + \frac{B_0^{1/2}}{C_1\rho L_1}\sum_{u\in\mathcal{I}_2^1}\mathbb{E}\|\nabla F(\mathbf{z}_u)\|_2^2 \le \Delta_F + (|\mathcal{I}_1| + |\mathcal{I}_2^1|)\frac{\sigma^2}{\rho L_1 B_0^{1/2}}. \tag{43}$$

Using Markov inequality, then with probability at least $2/3$, it holds that

$$\frac{|\mathcal{I}_1|\epsilon_H^3}{12L_2^2} + \frac{B_0^{1/2}}{C_1\rho L_1}\sum_{u\in\mathcal{I}_2^1}\|\nabla F(\mathbf{z}_u)\|_2^2 \leq 3\bigg(\Delta_F + (|\mathcal{I}_1| + |\mathcal{I}_2^1|)\frac{\sigma^2}{\rho L_1 B_0^{1/2}}\bigg). \tag{44}$$

By Lemma 36, for any $u \in \mathcal{I}_2^1$, with probability at least $1-\delta_0$, it holds that $\|\nabla F(\mathbf{z}_u)-\mathbf{g}_u\|_2 < \epsilon/4$ if $B_0 \geq 32\sigma^2/\epsilon^2(1+\log^{1/2}(1/\delta_0))^2$, which further indicates that $\|\nabla F(\mathbf{z}_u)\|_2 > \epsilon/4$. Thus, applying union bound yields that with probability at least $1 - |\mathcal{I}_1|\delta - 1/3 - |\mathcal{I}_2^1|\delta_0$ we have

$$\frac{|\mathcal{I}_1|\epsilon_H^3}{12L_2^2} + \frac{|\mathcal{I}_2^1|B_0^{1/2}\epsilon^2}{16C_1\rho L_1} \leq 3\Delta_F + \frac{3|\mathcal{I}_2^1|\sigma^2}{\rho L_1 B_0^{1/2}} + \frac{3|\mathcal{I}_1|\sigma^2}{\rho L_1 B_0^{1/2}}. \tag{45}$$

Recall that in Theorem 21 we set $\rho = \max\{54\sigma^2 L_1^{-1}L_2^2\epsilon_H^{-3}B_0^{-1/2}, 6\} \geq 54\sigma^2L_2^2/(L_1\epsilon_H^3 B_0^{1/2})$, which implies

$$\frac{3|\mathcal{I}_1|\sigma^2}{\rho L_1 B_0^{1/2}} \leq \frac{|\mathcal{I}_1|\epsilon_H^3}{18L_2^2}. \tag{46}$$

By (17) we have $B_0 > 96C_1\sigma^2\epsilon^{-2}$, which implies

$$\frac{3|\mathcal{I}_2^1|\sigma^2}{\rho L_1 B_0^{1/2}} \leq \frac{|\mathcal{I}_2^1|B_0^{1/2}\epsilon^2}{32C_1\rho L_1}. \tag{47}$$

Plugging (46) and (47) into (45) and rearranging the resulting inequality, then with probability $1 - |\mathcal{I}_1|\delta - 1/3 - |\mathcal{I}_2^1|\delta_0$, we have

$$\frac{|\mathcal{I}_1|\epsilon_H^3}{36L_2^2} + \frac{|\mathcal{I}_2^1|B_0^{1/2}\epsilon^2}{32C_1\rho L_1} \leq 3\Delta_F,$$

which immediately implies that

$$|\mathcal{I}_1| \leq 108\Delta_F L_2^2\epsilon_H^{-3} = O(\Delta_F L_2^2\epsilon_H^{-3}), \tag{48}$$

and

$$\begin{aligned}
|\mathcal{I}_2^1| &\leq 96C_1\rho\Delta_F L_1 B_0^{-1/2}\epsilon^{-2} \\
&= \max\{54\sigma^2L_1^{-1}L_2^2\epsilon_H^{-3}B_0^{-1/2}, 6\}\cdot 96C_1\Delta_F L_1 B_0^{-1/2}\epsilon^{-2} \\
&= \widetilde{O}(\Delta_F\sigma^{-1}L_1\epsilon^{-1}) + \widetilde{O}(\Delta_F L_2^2\epsilon_H^{-3}).
\end{aligned} \tag{49}$$

Thus we can calculate $U = |\mathcal{I}_1| + |\mathcal{I}_2| \leq 2|\mathcal{I}_1| + |\mathcal{I}_2^1| \leq 216\Delta_F L_2^2\epsilon_H^{-3} + 96C_1\rho\Delta_F L_1 B_0^{-1/2}\epsilon^{-2}$. Now we are ready to calculate the gradient complexity of Algorithm 5. By Lemma 35, we know that one single call of Neon2$^{\text{online}}$ needs $O\big(L_1^2/\epsilon_H^2\log^2(d/\delta)\big)$ stochastic gradient computations, and one single call of One-epoch-SNVRG needs $20B_0\log^3 B_0 = \widetilde{O}(\sigma^2/\epsilon^2)$ stochastic gradient computations. In addition, we need to compute $\mathbf{g}_u$ at each iteration of

Algorithm 5 (Line 3), which costs $B_0 = \widetilde{O}(\sigma^2/\epsilon^2)$ stochastic gradient computations. Thus, the expected total amount of stochastic gradient computations $\mathbb{E}T_{\text{total}}$ can be bounded as

$$\begin{aligned}\mathbb{E}T_{\text{total}} &= |\mathcal{I}_1| \cdot O(L_1^2/\epsilon_H^2 \log^2(d/\delta)) + |\mathcal{I}_2| \cdot \widetilde{O}(\sigma^2/\epsilon^2) + |\mathcal{I}| \cdot \widetilde{O}(\sigma^2/\epsilon^2) \\ &= |\mathcal{I}_1| \cdot \widetilde{O}(L_1^2/\epsilon_H^2) + (|\mathcal{I}_2^1| + |\mathcal{I}_2^2|) \cdot \widetilde{O}(\sigma^2/\epsilon^2) + |\mathcal{I}| \cdot \widetilde{O}(\sigma^2/\epsilon^2) \\ &= |\mathcal{I}_1| \cdot \widetilde{O}(L_1^2/\epsilon_H^2) + (|\mathcal{I}_2^1| + |\mathcal{I}_1|) \cdot \widetilde{O}(\sigma^2/\epsilon^2) \\ &= \widetilde{O}(\Delta_F L_1^2 L_2^2 \epsilon_H^{-5} + \Delta_F \sigma^2 L_2^2 \epsilon_H^{-3}\epsilon^{-2} + \Delta_F \sigma L_1 \epsilon^{-3}).\end{aligned}$$

Applying Markov inequality yields

$$T_{\text{total}} = \widetilde{O}(\Delta_F L_1^2 L_2^2 \epsilon_H^{-5} + \Delta_F \sigma^2 L_2^2 \epsilon_H^{-3}\epsilon^{-2} + \Delta_F \sigma L_1 \epsilon^{-3})$$

with probability at least 2/3. Furthermore, we have $|\mathcal{I}_1|\delta = |\mathcal{I}_1|/(3000\Delta_F L_2^2 \epsilon_H^{-3}) \leq 1/24$ and $|\mathcal{I}_2^1|\delta_0 = |\mathcal{I}_2^1|/(2500 C_1 \rho \Delta_F L_1 B_0^{-1/2}\epsilon^{-2}) < 1/24$. Therefore, with probability at least $1 - |\mathcal{I}_1|\delta - 1/3 - |\mathcal{I}_2^1|\delta_0 - 1/3 \geq 1/4$, Algorithm 5 can find an (ϵ, ϵ_H)-second order stationary point within

$$\widetilde{O}(\Delta_F L_1^2 L_2^2 \epsilon_H^{-5} + \Delta_F \sigma^2 L_2^2 \epsilon_H^{-3}\epsilon^{-2} + \Delta_F \sigma L_1 \epsilon^{-3})$$

stochastic gradient computations. ■

Appendix C. Proof of Main Theory with Third-order Smoothness

In this section, we prove the theoretical results of our proposed algorithms under third-order smoothness condition.

C.1. Proof of Theorem 23

The following lemma shows that the negative curvature descent step (Line 12) of Algorithm 4 achieves more function value decrease under third-order smoothness assumption. The proof can be found in Lemma 4.3 of Yu et al. (2018).

Lemma 37 (Yu et al. (2018)) *Suppose that $F = 1/n \sum_{i=1}^n f_i$, each f_i is L_1-smooth, L_2-Hessian Lipschitz continuous and F is L_3-third-order smooth. Let $\epsilon_H \in (0,1)$ and $\eta = \sqrt{3\epsilon_H/L_3}$. Suppose that $\lambda_{min}(\nabla^2 F(\mathbf{z}_{u-1})) < -\epsilon_H$ and that at the u-th iteration Algorithm 4 executes the Neon2$^{\text{finite}}$ algorithm (Line 7). Then with probability $1-\delta$ it holds that*

$$\mathbb{E}_\zeta\big[F(\mathbf{z}_u) - F(\mathbf{z}_{u-1})\big] \leq -\epsilon_H^2/(6L_3).$$

In addition, Neon2$^{\text{finite}}$ takes $O\big((n + n^{3/4}\sqrt{L_1/\epsilon_H})\log^2(d/\delta)\big)$ stochastic gradient computations.

Proof [Proof of Theorem 23] Denote $\mathcal{I} = \{1, \ldots, U\}$ as the index of iteration. Let $\mathcal{I} = \{1, \ldots, U\}$ be the index set of iteration. We use $\mathcal{I}_1$ and $\mathcal{I}_2$ to represent the index set of iterates where the $\mathbf{z}_u$ is obtained from Neon2$^{\text{finite}}$ and One-epoch-SNVRG. Since $U =$

$|\mathcal{I}_1| + |\mathcal{I}_2|$, we calculate $|\mathcal{I}_1|, |\mathcal{I}_2|$ separately. For $|\mathcal{I}_1|$, by Lemma 37, with probability at least $1-\delta$, we have

$$\mathbb{E}\big[F(\mathbf{z}_u) - F(\mathbf{z}_{u-1})\big] \le -\epsilon_H^2/(6L_3), \qquad \text{for } u \in \mathcal{I}_1. \tag{50}$$

Summing up (50) over $u \in \mathcal{I}_1$ and applying union bound, then with probability at least $1 - \delta \cdot |\mathcal{I}_1|$ we have

$$|\mathcal{I}_1| \cdot \epsilon_H^2/(6L_3) \le \sum_{u\in\mathcal{I}_1} \mathbb{E}\big[F(\mathbf{z}_{u-1}) - F(\mathbf{z}_u)\big] \le \sum_{u\in\mathcal{I}} \mathbb{E}\big[F(\mathbf{z}_{u-1}) - F(\mathbf{z}_u)\big] \le \Delta_F, \tag{51}$$

where the second inequality holds due to the fact that by Corollary 32 we have

$$0 \le \mathbb{E}\|\nabla F(\mathbf{z}_u)\|_2^2 \le \frac{CL_1}{n^{1/2}}\mathbb{E}\big[F(\mathbf{z}_{u-1}) - F(\mathbf{z}_u)\big], \qquad \text{for } u \in \mathcal{I}_2. \tag{52}$$

(51) directly implies

$$|\mathcal{I}_1| \le 6L_3\Delta_F/\epsilon_H^2.$$

For $|\mathcal{I}_2|$, we decompose $\mathcal{I}_2 = \mathcal{I}_2^1 \cup \mathcal{I}_2^2$, where $\mathcal{I}_2^1 = \{u \in \mathcal{I}_2 : \|\mathbf{g}_u\|_2 > \epsilon\}$ and $\mathcal{I}_2^2 = \{u \in \mathcal{I}_2 : \|\mathbf{g}_u\|_2 \le \epsilon\}$. If $u \in \mathcal{I}_2^2$, then at the $(u+1)$-th iteration, Algorithm 4 will execute Neon2$^{\text{finite}}$. Thus, we have $|\mathcal{I}_2^2| \le |\mathcal{I}_1|$. For $|\mathcal{I}_2^1|$, note that $\mathbf{x}_0 = \mathbf{z}_{u-1}$ and $\mathbf{x}_T = \mathbf{z}_u$ in Corollary 32 and summing up over $u \in \mathcal{I}_2^1$ yields

$$\begin{aligned}
\sum_{u\in\mathcal{I}_2^1} \mathbb{E}\|\nabla F(\mathbf{z}_u)\|_2^2 &\le \sum_{u\in\mathcal{I}_2^1} \frac{CL_1}{n^{1/2}}\mathbb{E}\big[F(\mathbf{z}_{u-1}) - F(\mathbf{z}_u)\big] \\
&\le \sum_{u\in\mathcal{I}} \frac{CL_1}{n^{1/2}}\mathbb{E}\big[F(\mathbf{z}_{u-1}) - F(\mathbf{z}_u)\big] \\
&\le \frac{CL_1}{n^{1/2}} \cdot \Delta_F,
\end{aligned} \tag{53}$$

where the second inequality follows from (51) and (52). Applying Markov's inequality, with probability at least $2/3$, we have

$$\sum_{u\in\mathcal{I}_2^1} \|\nabla F(\mathbf{z}_u)\|_2^2 \le \frac{3CL_1\Delta_F}{n^{1/2}}.$$

by definition for any $u \in \mathcal{I}_2^1$, we have $\|\nabla F(\mathbf{z}_u)\|_2 = \|\mathbf{g}_u\|_2 > \epsilon$. Then we have with probability at least $2/3$ that

$$|\mathcal{I}_2^1| \le \frac{3CL_1\Delta_F}{\epsilon^2 n^{1/2}}.$$

Total number of iteration is $U = |\mathcal{I}_1| + |\mathcal{I}_2| \le 2|\mathcal{I}_1| + |\mathcal{I}_2^1| \le 12L_3\Delta_F\epsilon_H^{-2} + 3CL_1\Delta_F\epsilon^{-2}n^{-1/2}$. We now calculate the gradient complexity of Algorithm 4. By Lemma 37 one single call of Neon2$^{\text{finite}}$ needs $O\big((n + n^{3/4}\sqrt{L_1/\epsilon_H})\log^2(d/\delta)\big)$ stochastic gradient computations and by

Corollary 32 one single call of One-epoch-SNVRG needs $20n\log^3 n$ stochastic gradient computations. Moreover, we need to compute $\mathbf{g}_u$ at each iteration, which takes $O(n)$ stochastic gradient computations. Thus, the expectation of the total amount of stochastic gradient computations $\mathbb{E}T_{\text{total}}$ can be bounded by

$$\begin{aligned}
&|\mathcal{I}_1|\cdot O\big((n+n^{3/4}\sqrt{L_1/\epsilon_H})\log^2(d/\delta)\big)+|\mathcal{I}_2|\cdot O(n\log^3 n)+|\mathcal{I}|\cdot O(n)\\
&=|\mathcal{I}_1|\cdot \widetilde{O}\big(n+n^{3/4}\sqrt{L_1/\epsilon_H}\big)+(|\mathcal{I}_2^1|+|\mathcal{I}_2^2|)\cdot\widetilde{O}(n)\\
&=|\mathcal{I}_1|\cdot \widetilde{O}\big(n+n^{3/4}\sqrt{L_1/\epsilon_H}\big)+(|\mathcal{I}_2^1|+|\mathcal{I}_1|)\cdot\widetilde{O}(n).
\end{aligned}\tag{54}$$

We further plug the upper bound of $|\mathcal{I}_1|$ and $|\mathcal{I}_2^1|$ into (54) and obtain

$$\begin{aligned}
\mathbb{E}T_{\text{total}}&=O(L_3\Delta_F\epsilon_H^{-2})\cdot\widetilde{O}\big(n+n^{3/4}\sqrt{L_1/\epsilon_H}\big)+O(L_1\Delta_F\epsilon^{-2}n^{-1/2})\widetilde{O}(n)\\
&=\widetilde{O}\big(\Delta_F nL_3\epsilon_H^{-2}+\Delta_F n^{3/4}L_1^{1/2}L_3\epsilon_H^{-5/2}+\Delta_F n^{1/2}L_1\epsilon^{-2}\big).
\end{aligned}$$

Using Markov inequality, with probability at least $2/3$, we have

$$T_{\text{total}}=\widetilde{O}\big(\Delta_F nL_3\epsilon_H^{-2}+\Delta_F n^{3/4}L_1^{1/2}L_3\epsilon_H^{-5/2}+\Delta_F n^{1/2}L_1\epsilon^{-2}\big).$$

Note that $|\mathcal{I}_1|\delta=|\mathcal{I}_1|/(72\cdot L_3\Delta_F\epsilon_H^{-2})\le 1/12$. By union bound, with probability at least $1-1/3-1/3-|\mathcal{I}_1|\delta\ge 1/4$, SNVRG+Neon2$^{\text{finite}}$ will find an (ϵ,ϵ_H)-second order stationary point within

$$\widetilde{O}\big(\Delta_F nL_3\epsilon_H^{-2}+\Delta_F n^{3/4}L_1^{1/2}L_3\epsilon_H^{-5/2}+\Delta_F n^{1/2}L_1\epsilon^{-2}\big)$$

stochastic gradient computations. ∎

C.2. Proof of Theorem 26

The following lemma shows that the negative curvature descent step (Line 13) of Algorithm 5 achieves more function value decrease under third-order smoothness assumption. The proof can be found in Lemma 4.6 of Yu et al. (2018).

Lemma 38 (Yu et al. (2018)) *Suppose that $F(\mathbf{x})=\mathbb{E}_{\xi\in\mathcal{D}}F(\mathbf{x};\xi)$, each $F(\mathbf{x};\xi)$ is L_1-smooth, L_2-Hessian Lipschitz continuous and $F(\mathbf{x})$ is L_3-third-order smooth. Let $\epsilon_H\in(0,1)$ and $\eta=\sqrt{3\epsilon_H/L_3}$. Suppose that $\lambda_{min}(\nabla^2F(\mathbf{z}_{u-1}))<-\epsilon_H$ and that at the u-th iteration Algorithm 5 executes the Neon2^{online} algorithm (Line 8). Then with probability $1-\delta$ it holds that*

$$\mathbb{E}_\zeta\big[F(\mathbf{z}_u)-F(\mathbf{z}_{u-1})\big]\le-\epsilon_H^2/(6L_3).$$

In addition, Neon2^{online} takes $O(L_1^2/\epsilon_H^2\log^2(d/\delta))$ stochastic gradient computations.

Proof [Proof of Theorem 26] Denote $\delta_0=1/(2500C_1\rho\Delta_F L_1B_0^{-1/2}\epsilon^{-2})$, then by (20) we have $B_0>32\sigma^2/\epsilon^2(1+\log^{1/2}(1/\delta_0))^2$. Let $\mathcal{I}=\{1,\ldots,U\}$ be the index set of all iterations. We use $\mathcal{I}_1$ and $\mathcal{I}_2$ to represent the index set of iterates where $\mathbf{z}_u$ is obtained from Neon2$^{\text{online}}$

and One-epoch-SNVRG respectively. Obviously $U = |\mathcal{I}_1| + |\mathcal{I}_2|$ and we need to upper bound $|\mathcal{I}_1|$ and $|\mathcal{I}_2|$. From Lemma 38, we have with probability at least $1-\delta$ that

$$\mathbb{E}\big[F(\mathbf{z}_u) - F(\mathbf{z}_{u-1})\big] \le -\epsilon_H^2/(6L_3), \qquad \text{for } u \in \mathcal{I}_1. \tag{55}$$

By Corollary 34, we have

$$\mathbb{E}\|\nabla F(\mathbf{z}_u)\|_2^2 \le C_1\bigg(\frac{\rho L_1}{B_0^{1/2}} \cdot \mathbb{E}\big[F(\mathbf{z}_{u-1}) - F(\mathbf{z}_u)\big] + \frac{\sigma^2}{B_0}\bigg), \qquad \text{for } u \in \mathcal{I}_2, \tag{56}$$

where $C_1 = 200$. We decompose $\mathcal{I}_2$ into two disjoint sets $\mathcal{I}_2 = \mathcal{I}_2^1 \cup \mathcal{I}_2^2$, where $\mathcal{I}_2^1 = \{u \in \mathcal{I}_2 : \|\mathbf{g}_u\|_2 > \epsilon/2\}$ and $\mathcal{I}_2^1 = \{u \in \mathcal{I}_2 : \|\mathbf{g}_u\|_2 \le \epsilon/2\}$. (56) leads to the following inequalities:

$$\mathbb{E}\big[F(\mathbf{z}_u) - F(\mathbf{z}_{u-1})\big] \le -\frac{B_0^{1/2}}{C_1\rho L_1}\mathbb{E}\|\nabla F(\mathbf{z}_u)\|_2^2 + \frac{\sigma^2}{\rho L_1 B_0^{1/2}}, \qquad \text{for } u \in \mathcal{I}_2^1, \tag{57}$$

$$\mathbb{E}\big[F(\mathbf{z}_u) - F(\mathbf{z}_{u-1})\big] \le \frac{\sigma^2}{\rho L_1 B_0^{1/2}}, \qquad \text{for } u \in \mathcal{I}_2^2. \tag{58}$$

Summing up (55) over $u \in \mathcal{I}_1$, (57) over $u \in \mathcal{I}_2^1$ and (58) over $u \in \mathcal{I}_2^2$, we have

$$\begin{aligned}
&\mathbb{E}\big[\sum_{u\in\mathcal{I}} F(\mathbf{z}_{u-1}) - F(\mathbf{z}_u)\big] \\
&\ge |\mathcal{I}_1| \cdot \frac{\epsilon_H^2}{6L_3} + \frac{B_0^{1/2}}{C_1\rho L_1}\sum_{u\in\mathcal{I}_2^1}\mathbb{E}\|\nabla F(\mathbf{z}_u)\|_2^2 - \sum_{u\in\mathcal{I}_2^1}\frac{\sigma^2}{\rho L_1 B_0^{1/2}} - \sum_{u\in\mathcal{I}_2^2}\frac{\sigma^2}{\rho L_1 B_0^{1/2}}.
\end{aligned} \tag{59}$$

For any $u \in \mathcal{I}_2^2$, we have $\|\mathbf{g}_u\|_2 \le \epsilon/2$, then algorithm will execute Neon2$^{\text{online}}$ at u-th iteration, which implies $|\mathcal{I}_2^2| \le |\mathcal{I}_1|$. Combining this with (59) and by the definition of Δ_F, with probability at least $1 - |\mathcal{I}_1|\delta$, we have

$$\frac{|\mathcal{I}_1|\epsilon_H^2}{6L_3} + \frac{B_0^{1/2}}{C_1\rho L_1}\sum_{u\in\mathcal{I}_2^1}\mathbb{E}\|\nabla F(\mathbf{z}_u)\|_2^2 \le \Delta_F + (|\mathcal{I}_1| + |\mathcal{I}_2^1|)\frac{\sigma^2}{\rho L_1 B_0^{1/2}}. \tag{60}$$

Applying Markov inequality, yields with probability at least $2/3$ that

$$\frac{|\mathcal{I}_1|\epsilon_H^2}{6L_3} + \frac{B_0^{1/2}}{C_1\rho L_1}\sum_{u\in\mathcal{I}_2^1}\|\nabla F(\mathbf{z}_u)\|_2^2 \le 3\bigg(\Delta_F + (|\mathcal{I}_1| + |\mathcal{I}_2^1|)\frac{\sigma^2}{\rho L_1 B_0^{1/2}}\bigg). \tag{61}$$

By Lemma 36, if $B_0 \ge 32\sigma^2/\epsilon^2(1 + \log^{1/2}(1/\delta_0))^2$, then for any $u \in \mathcal{I}_2^1$, with probability at least $1-\delta_0$, we have $\|\nabla F(\mathbf{z}_u)\|_2 > \epsilon/4$. Applying union bound, we have with probability at least $1 - |\mathcal{I}_1|\delta - 1/3 - |\mathcal{I}_2^1|\delta_0$ it holds that

$$\frac{|\mathcal{I}_1|\epsilon_H^2}{6L_3} + \frac{|\mathcal{I}_2^1|B_0^{1/2}\epsilon^2}{16C_1\rho L_1} \le 3\Delta_F + \frac{3|\mathcal{I}_2^1|\sigma^2}{\rho L_1 B_0^{1/2}} + \frac{3|\mathcal{I}_1|\sigma^2}{\rho L_1 B_0^{1/2}}. \tag{62}$$

By (20) we have $B_0 > 96C_1\sigma^2\epsilon^{-2}$, which indicates

$$\frac{3|\mathcal{I}_2^1|\sigma^2}{\rho L_1 B_0^{1/2}} \leq \frac{|\mathcal{I}_2^1| B_0^{1/2}\epsilon^2}{32C_1\rho L_1}. \tag{63}$$

By the choice of ρ we have $\rho = \max\{36\sigma^2 L_1^{-1}L_3\epsilon_H^{-2}B_0^{-1/2}, 6\} \geq 36\sigma^2 L_1^{-1}L_3\epsilon_H^{-2}B_0^{-1/2}$, which indicates

$$\frac{3|\mathcal{I}_1|\sigma^2}{\rho L_1 B_0^{1/2}} \leq \frac{|\mathcal{I}_1|\epsilon_H^2}{12L_3}. \tag{64}$$

Plugging (63), (64) into (62), then with probability $1 - |\mathcal{I}_1|\delta - 1/3 - |\mathcal{I}_2^1|\delta_0$, we have

$$\frac{|\mathcal{I}_1|\epsilon_H^2}{12L_3} + \frac{|\mathcal{I}_2^1| B_0^{1/2}\epsilon^2}{32C_1\rho L_1} \leq 3\Delta_F,$$

which immediately implies

$$|\mathcal{I}_1| \leq 36\Delta_F L_3 \epsilon_H^{-2} = O(\Delta_F L_3\epsilon_H^{-2}), \tag{65}$$

and

$$|\mathcal{I}_2^1| \leq 96C_1\rho\Delta_F L_1 B_0^{-1/2}\epsilon^{-2} = \widetilde{O}(\Delta_F\sigma^{-1}L_1\epsilon^{-1} + \Delta_F L_3\epsilon_H^{-2}). \tag{66}$$

Total number of iteration is $U = |\mathcal{I}_1| + |\mathcal{I}_2| \leq 2|\mathcal{I}_1| + |\mathcal{I}_2^1| \leq 72\Delta_F L_3\epsilon_H^{-2} + 96C_1\rho\Delta_F L_1 B_0^{-1/2}\epsilon^{-2}$. Now we calculate the gradient complexity of Algorithm 5. By Lemma 38 one single call of Neon2$^{\text{online}}$ needs $O\big(L_1^2/\epsilon_H^2 \log^2(d/\delta)\big)$ stochastic gradient computations and by Corollary 34 one single call of One-epoch-SNVRG needs $20B_0\log^3 B_0 = \widetilde{O}(\sigma^2/\epsilon^2)$ stochastic gradient computations. In addition, we need to compute $\mathbf{g}_u$ at each iteration, which takes $\widetilde{O}(\sigma^2/\epsilon^2)$ stochastic gradient computations. The expected total amount of stochastic gradient computations $\mathbb{E}T_{\text{total}}$ is

$$\begin{aligned}
\mathbb{E}T_{\text{total}} &= |\mathcal{I}_1|\cdot O(L_1^2/\epsilon_H^2\log^2(d/\delta)) + |\mathcal{I}_2|\cdot\widetilde{O}(\sigma^2/\epsilon^2) + |\mathcal{I}|\cdot\widetilde{O}(\sigma^2/\epsilon^2)\\
&= |\mathcal{I}_1|\cdot\widetilde{O}(L_1^2/\epsilon_H^2) + (|\mathcal{I}_2^1| + |\mathcal{I}_2^2|)\cdot\widetilde{O}(\sigma^2/\epsilon^2) + |\mathcal{I}|\cdot\widetilde{O}(\sigma^2/\epsilon^2)\\
&= |\mathcal{I}_1|\cdot\widetilde{O}(L_1^2/\epsilon_H^2) + (|\mathcal{I}_2^1| + |\mathcal{I}_1|)\cdot\widetilde{O}(\sigma^2/\epsilon^2)\\
&= \widetilde{O}(\Delta_F L_1^2 L_3\epsilon_H^{-4} + \Delta_F\sigma^2 L_3\epsilon_H^{-2}\epsilon^{-2} + \Delta_F\sigma L_1\epsilon^{-3}).
\end{aligned}$$

Applying Markov's inequality, with probability at least $2/3$, we have

$$T_{\text{total}} = \widetilde{O}(\Delta_F L_1^2 L_3\epsilon_H^{-4} + \Delta_F\sigma^2 L_3\epsilon_H^{-2}\epsilon^{-2} + \Delta_F\sigma L_1\epsilon^{-3}).$$

Note that $|\mathcal{I}_1|\delta = |\mathcal{I}_1|/(1000\Delta_F L_3\epsilon_H^{-2}) \leq 1/24$ and $|\mathcal{I}_2^1|\delta_0 = |\mathcal{I}_2^1|/(2500C_1\rho\Delta_F L_1 B_0^{-1/2}\epsilon^{-2}) < 1/24$. Therefore, with probability at least $1 - |\mathcal{I}_1|\delta - 1/3 - |\mathcal{I}_2^1|\delta_0 - 1/3 \geq 1/4$, Algorithm 5 can find an (ϵ, ϵ_H)-second-order stationary point with

$$\widetilde{O}(\Delta_F L_1^2 L_3\epsilon_H^{-4} + \Delta_F\sigma^2 L_3\epsilon_H^{-2}\epsilon^{-2} + \Delta_F\sigma L_1\epsilon^{-3})$$

stochastic gradient computations. ■

Appendix D. Proof of Supporting Lemmas

D.1. Proof of Lemma 28

We first prove our key lemma on One-epoch-SNVRG. In order to prove Lemma 28, we need the following supporting lemma, which shows that with any chosen epoch length T, the summation of expectation of the square of gradient norm $\sum_{j=0}^{T-1}\mathbb{E}\|\nabla F(\mathbf{x}_j)\|_2^2$ can be bounded.

Lemma 39 *Suppose we arbitrarily fix the amount of epochs $T > 1$ in Algorithm 1. In other words, we do not bother with Options I or II for the present. If the step size and batch size parameters in Algorithm 1 satisfy $M \ge 6L$ and $B_l \ge 6^{K-l+1}(\prod_{s=l}^{K} T_s)^2$ for any $1 \le l \le K$, then the iterates of Algorithm 1 satisfies*

$$\sum_{j=0}^{T-1}\mathbb{E}\|\nabla F(\mathbf{x}_j)\|_2^2 \le C\bigg(M\mathbb{E}\big[F(\mathbf{x}_0) - F(\mathbf{x}_T)\big] + \frac{2\sigma^2 T}{B_0}\cdot \mathbb{1}\{B_0 < n\}\bigg), \tag{67}$$

where $C = 100$.

Proof [Proof of Lemma 28] We can check that $2 \le B_0^{2^{-K}} < 4$, and we can check that the choice of $M, \{T_l\}, \{B_l\}$ in Lemma 28 satisfies the assumption of Lemma 39. Moreover, we have

$$\begin{aligned}
T &= \prod_{l=1}^{K} T_l \\
&> (B_0^{2^{-K}} - 1)\prod_{l=2}^{K}(B_0^{2^{l-K-2}} - 1) \\
&> \frac{1}{2}B_0^{2^{-K}}\cdot \prod_{l=2}^{K} B_0^{2^{l-K-2}}\cdot\bigg(1 - \bigg(\sum_{l=2}^{K}\frac{1}{B_0^{2^{l-K-2}}}\bigg)\bigg) \\
&\ge \frac{1}{2}B_0^{1/2}\bigg(1 - \bigg(\sum_{l=2}^{K}\frac{1}{2^{2^{l-2}}}\bigg)\bigg) \\
&> \frac{1}{10}B_0^{1/2},
\end{aligned} \tag{68}$$

where the first inequality holds due to the fact $\lfloor x\rfloor > x - 1$ for any $x > 1$, the second inequality holds since $2 \le B_0^{2^{-K}} < 4$ and the fact $\prod_{l=2}^{K}(x_l - 1) > \prod_{l=2}^{K} x_l(1 - \sum_{l=2}^{K} x_l^{-1})$ for any sequence $\{x_l\}_{l=2}^{K}$ satisfying $\forall 2 \le l \le K, x_l \ge 2$, the third inequality holds since $2^{2^K} \le B_0$, the last inequality holds due to the fact that $\sum_{l=2}^{K} 2^{-2^{l-2}} < 4/5$. We now submit (68) into (67), which immediately implies (23). Next we compute how many stochastic gradient computations we need in total after we run One-epoch-SNVRG once. According to the update of reference gradients in Algorithm 1, we only update $\mathbf{g}_t^{(0)}$ once at the beginning of Algorithm 1 (Line 23 is only reached when $r = 0$), which needs B_0 stochastic gradient computations. For $\mathbf{g}_t^{(l)}$, we only need to update it when $0 = (t \mod \prod_{j=l+1}^{K} T_j)$, and thus we need to sample $\mathbf{g}_t^{(l)}$ for $T/\prod_{j=l+1}^{K} T_j = \prod_{j=1}^{l} T_j$ times. We need $2B_l$ stochastic gradient computations for each sampling procedure (Line 20 in Algorithm 1). We use $\mathcal{T}$ to represent the total number of stochastic gradient computations, then based on above arguments we have

$$\mathcal{T} = B_0 + 2\sum_{l=1}^{K} B_l \cdot \prod_{j=1}^{l} T_j. \tag{69}$$

Now we calculate $\mathcal{T}$ under the parameter choice of Lemma 28. Note that we can easily verify the following inequalities:

$$\begin{aligned}
&\prod_{j=1}^{l} T_j \le B_0^{2^{-K}} \prod_{j=2}^{l} B_0^{2^{j-K-2}} = B_0^{\frac{2^l}{2^{K+1}}}, \\
&\bigg(\prod_{j=l}^{K} T_j\bigg)^2 \le \bigg(\prod_{j=l}^{K} B_0^{2^{j-K-2}}\bigg)^2 = B_0^{1-2^{K+1-l}}, \qquad\qquad \forall 2 \le l \le K, \\
&\bigg(\prod_{j=1}^{K} T_j\bigg)^2 \le \bigg(B_0^{2^{-K}} \cdot \prod_{j=2}^{K} B_0^{2^{j-K-2}}\bigg)^2 = B_0,
\end{aligned}$$

which implies that

$$\begin{aligned}
B_1 \cdot \prod_{j=1}^{1} T_j &= 6^K \bigg(\prod_{j=1}^{K} T_j\bigg)^2 T_1 \le 6^K B_0 \cdot 4, \\
B_l \cdot \prod_{j=1}^{l} T_j &= 6^{K-l+1} \bigg(\prod_{j=l}^{K} T_j\bigg)^2 \prod_{j=1}^{l} T_j \le 6^{K-l+1} B_0.
\end{aligned} \tag{70}$$

Submit (70) into (69) yields the following results:

$$\begin{aligned}
\mathcal{T} &= B_0 + 2\bigg(4 \times 6^K B_0 + \sum_{l=2}^{K} 6^{K-l+1} B_0\bigg) \\
&< B_0 + 9 \times 6^K B_0 \\
&\le B_0 + 9 \times 6^{\log\log B_0} B_0 \\
&< B_0 + 9 B_0 \log^3 B_0.
\end{aligned}$$

Therefore, the total gradient complexity $\mathcal{T}$ is bounded as follows.

$$\mathcal{T} = B_0 + 2\sum_{l=1}^{K} B_l \cdot \prod_{j=1}^{l} T_j \le B_0 + 9B_0 \log^3 B_0 \le 10 B_0 \log^3 B_0. \tag{71}$$

$\blacksquare$

D.2. Proof of Lemma 29

Now we prove Lemma 29 about the function value decrease of Algorithm 1 with Option II. Note that Lemma 39 shows that with any chosen epoch length T, the summation of expectation of the square of gradient norm $\sum_{j=0}^{T-1} \mathbb{E}\|\nabla F(\mathbf{x}_j)\|_2^2$ can be bounded. In order to prove the upper bound on $\mathbb{E}\|\nabla F(\mathbf{x}_T)\|_2^2$, we need the following technical lemma about geometric distribution.

Lemma 40 *Suppose that $G \sim Geom(p)$, where $\mathbb{P}(G = k) = p(1-p)^k, k \geq 0$. Let $a(j), b(j)$ be two series and $b(0) \geq 0$. If for any $k \geq 1$, it holds that $\sum_{j=0}^{k-1} a(j) \leq b(k)$, then we have*

$$\frac{1-p}{p}\mathbb{E}_G a(G) \leq \mathbb{E}_G b(G).$$

Proof [Proof of Lemma 29] We can easily check that the choice of $M, \{T_l\}, \{B_l\}$ in Lemma 29 satisfies the assumption of Lemma 39. By Algorithm 1, we have $T \sim \text{Geom}(p)$ where $p = 1/(1 + \prod_{j=1}^K T_j)$. Let

$$a(j) = \mathbb{E}\|\nabla F(\mathbf{x}_j)\|_2^2,\ b(j) = C\bigg(M\mathbb{E}\big[F(\mathbf{x}_0) - F(\mathbf{x}_j)\big] + \frac{\sigma^2 j}{B_0}\cdot \mathbb{1}\{B_0 < n\}\bigg).$$

Then by Lemma 39, for any $T \geq 1$, we have $\sum_{j=0}^{T-1} a(j) \leq b(T)$ and $b(0) = 0$. Thus, by Lemma 40, we have

$$\frac{1-p}{p}\mathbb{E}_T\mathbb{E}\|\nabla F(\mathbf{x}_T)\|_2^2 \leq C\bigg(M\mathbb{E}_T\mathbb{E}\big[F(\mathbf{x}_0) - F(\mathbf{x}_T)\big] + \frac{2\sigma^2\mathbb{E}_T T}{B_0}\cdot \mathbb{1}\{B_0 < n\}\bigg).$$

Since $\mathbb{E}_T T = (1-p)/p = \prod_{j=1}^K T_j > B_0^{1/2}/10$ due to (68) , we have

$$\begin{aligned}
\mathbb{E}\|\nabla F(\mathbf{x}_T)\|_2^2 &\leq C\bigg(\frac{M}{\prod_{j=1}^K T_j}\mathbb{E}\big[F(\mathbf{x}_0) - F(\mathbf{x}_T)\big] + \frac{2\sigma^2}{B_0}\cdot \mathbb{1}\{B_0 < n\}\bigg) \\
&\leq 10C\bigg(\frac{M}{B_0^{1/2}}\mathbb{E}\big[F(\mathbf{x}_0) - F(\mathbf{x}_T)\big] + \frac{2\sigma^2}{B_0}\cdot \mathbb{1}\{B_0 < n\}\bigg),
\end{aligned}$$

which immediately implies (32).

Finally we consider how many stochastic gradient computations for us to run One-epoch-SNVRG once. According to the update of reference gradients in Algorithm 1, for $\mathbf{g}_t^{(l)}$, we need to update it when $0 = (t \mod \prod_{j=l+1}^K T_j)$, and thus we need to sample $\mathbf{g}_t^{(l)}$ for $T/\prod_{j=l+1}^K T_j$ times. We need B_0 stochastic gradient computations to update $\mathbf{g}_t^{(0)}$ and $2B_l$ stochastic gradient computations for $\mathbf{g}_t^{(l)}$ (Lines 20 and 23 in Algorithm 1 respectively). If we use $\mathcal{T}$ to represent the total number of stochastic gradient computations, then based on above arguments, we have

$$\mathbb{E}\mathcal{T} \leq B_0 \cdot \frac{\mathbb{E}T}{\prod_{j=1}^K T_j} + 2\sum_{l=1}^K B_l \cdot \frac{\mathbb{E}T}{\prod_{j=l+1}^K T_j}$$

$$
\begin{aligned}
&= B_0 + 2\sum_{l=1}^{K} B_l \prod_{j=1}^{l} T_j \\
&\leq 10 B_0 \log^3 B_0,
\end{aligned}
$$

where the last inequality holds due to (71). ■

Appendix E. Proof of Key Lemma 39

In this section, we focus on proving Lemma 39, which holds for any fixed T and plays a pivotal role in the analyses of Algorithm 1 with both Option I and Option II. Let $M, \{T_i\}, \{B_i\}, B_0$ be the parameters as defined in Algorithm 1. We define filtration $\mathcal{F}_t = \sigma(\mathbf{x}_0, \ldots, \mathbf{x}_t)$. Let $\{\mathbf{x}_t^{(l)}\}, \{\mathbf{g}_t^{(l)}\}$ be the reference points and reference gradients in Algorithm 1. We define $\mathbf{v}_t^{(l)}$ as

$$
\mathbf{v}_t^{(l)} := \sum_{j=0}^{l} \mathbf{g}_t^{(j)}, \quad \text{for } 0 \leq l \leq K. \tag{72}
$$

We first present the following definition and two technical lemmas for the purpose of our analysis.

Definition 41 *We define constant series $\{c_j^{(s)}\}$ as the following. For each s, we define $c_{T_s}^{(s)}$ as*

$$
c_{T_s}^{(s)} = \frac{M}{6^{K-s+1} \prod_{l=s}^{K} T_l}. \tag{73}
$$

When $0 \leq j < T_s$, we define $c_j^{(s)}$ by induction:

$$
c_j^{(s)} = \bigg(1 + \frac{1}{T_s}\bigg) c_{j+1}^{(s)} + \frac{3L^2}{M} \cdot \frac{\prod_{l=s+1}^{K} T_l}{B_s}. \tag{74}
$$

Lemma 42 *For any p, s, where $1 \leq s \leq K$, $p \cdot \prod_{j=s}^{K} T_j < T$ and $q \prod_{j=1}^{K} T_j \leq p \cdot \prod_{j=s}^{K} T_j < (p+1) \cdot \prod_{j=s}^{K} T_j \leq (q+1) \prod_{j=1}^{K} T_j$, we define*

$$
start = p \cdot \prod_{j=s}^{K} T_j, \;\; end = \min\bigg\{start + \prod_{j=s}^{K} T_j, T\bigg\}
$$

for simplification. Then we have the following results:

$$
\begin{aligned}
&\mathbb{E}\bigg[\sum_{j=start}^{end-1} \frac{\|\nabla F(\mathbf{x}_j)\|_2^2}{100M} + F(\mathbf{x}_{end}) + c_{T_s}^{(s)} \cdot \|\mathbf{x}_{end} - \mathbf{x}_{start}\|_2^2 \big| \mathcal{F}_{start}\bigg] \\
&\leq F(\mathbf{x}_{start}) + \frac{2}{M} \cdot \mathbb{E}\big[\big\|\nabla F(\mathbf{x}_{start}) - \mathbf{v}_{start}\big\|_2^2 \big| \mathcal{F}_{start}\big] \cdot (end - start).
\end{aligned}
$$

Lemma 43 (Lei et al. (2017)) *Let* $\mathbf{a}_i$ *be vectors satisfying* $\sum_{i=1}^N \mathbf{a}_i = 0$. *Let* $\mathcal{J}$ *be a uniform random subset of* $\{1, \ldots, N\}$ *with size* m, *then*

$$\mathbb{E}\bigg\|\frac{1}{m}\sum_{j\in\mathcal{J}}\mathbf{a}_j\bigg\|_2^2 \leq \frac{\mathbb{1}(|\mathcal{J}|<N)}{mN}\sum_{j=1}^N\|\mathbf{a}_j\|_2^2.$$

Proof [Proof of Lemma 39] We have

$$\begin{aligned}
\sum_{j=0}^{T-1}\frac{\mathbb{E}\|\nabla F(\mathbf{x}_j)\|_2^2}{100M} + \mathbb{E}\big[F(\mathbf{x}_T)\big] &\leq \sum_{j=0}^{T-1}\frac{\mathbb{E}\|\nabla F(\mathbf{x}_j)\|_2^2}{100M} + \mathbb{E}\big[F(\mathbf{x}_T) + c_{T_1}^{(1)}\cdot\|\mathbf{x}_T-\mathbf{x}_0\|_2^2\big]\\
&\leq \mathbb{E}\big[F(\mathbf{x}_0)\big] + \frac{2}{M}\cdot\mathbb{E}\|\nabla F(\mathbf{x}_0)-\mathbf{g}_0\|_2^2\cdot T, \qquad (75)
\end{aligned}$$

where the second inequality comes from Lemma 42 with we take $s=1, p=0$. Moreover we have

$$\begin{aligned}
\mathbb{E}\|\nabla F(\mathbf{x}_0)-\mathbf{g}_0\|_2^2 &= \mathbb{E}\bigg\|\frac{1}{B_0}\sum_{i\in I}\big[\nabla f_i(\mathbf{x}_0)-\nabla F(\mathbf{x}_0)\big]\bigg\|_2^2\\
&\leq \mathbb{1}(B_0<n)\cdot\frac{1}{B_0}\cdot\frac{1}{n}\sum_{i=1}^n\big\|\nabla f_i(\mathbf{x}_0)-\nabla F(\mathbf{x}_0)\big\|_2^2 \qquad (76)\\
&\leq \mathbb{1}(B_0<n)\cdot\frac{\sigma^2}{B_0}, \qquad (77)
\end{aligned}$$

where (76) holds because of Lemma 43. Plug (77) into (75) and note that we have $M=6L$, and then we obtain

$$\sum_{j=0}^{T-1}\mathbb{E}\|\nabla F(\mathbf{x}_j)\|_2^2 \leq C\bigg(M\mathbb{E}\big[F(\mathbf{x}_0)-F(\mathbf{x}_T)\big] + \frac{2T\sigma^2}{B_0}\cdot\mathbb{1}(B_0<n)\bigg), \qquad (78)$$

where $C=100$, which complete the proof of Lemma 39. ■

Appendix F. Proof of Technical Lemmas

In this section, we provide the proofs of technical lemmas used in Appendix E.

F.1. Proof of Lemma 42

Let $M, \{T_l\}, \{B_l\}, B_0$ be the parameters defined in Algorithm 1 and $\{\mathbf{x}_t^{(l)}\}, \{\mathbf{g}_t^{(l)}\}$ be the reference points and reference gradients defined in Algorithm 1. Let $\mathbf{v}_t^{(l)}, \mathcal{F}_t$ be the variables and filtration defined in Appendix E and let $c_j^{(s)}$ be the constant series defined in Definition 41.

In order to prove Lemma 42, we will need the following supporting propositions and lemmas. We first state the proposition about the relationship among $\mathbf{x}_t^{(s)}, \mathbf{g}_t^{(s)}$ and $\mathbf{v}_t^{(s)}$:

Proposition 44 *Let $\mathbf{v}_t^{(l)}$ be defined as in (72). Let p, s satisfy $0 \le p \cdot \prod_{j=s+1}^K T_j < (p+1) \cdot \prod_{j=s+1}^K T_j < T$. For any t, t' satisfying $p \cdot \prod_{j=s+1}^K T_j \le t < t' < (p+1) \cdot \prod_{j=s+1}^K T_j$, it holds that*

$$\mathbf{x}_t^{(s)} = \mathbf{x}_{t'}^{(s)} = \mathbf{x}_{p \prod_{j=s+1}^K T_j}, \tag{79}$$

$$\mathbf{g}_t^{(s')} = \mathbf{g}_{t'}^{(s')}, \qquad \textit{for any } s' \textit{ that satisfies } 0 \le s' \le s, \tag{80}$$

$$\mathbf{v}_t^{(s)} = \mathbf{v}_{t'}^{(s)} = \mathbf{v}_{p \prod_{j=s+1}^K T_j}. \tag{81}$$

The following lemma spells out the relationship between $c_j^{(s-1)}$ and $c_{T_s}^{(s)}$. In a word, $c_j^{(s-1)}$ is about $1 + T_{s-1}$ times less than $c_{T_s}^{(s)}$:

Lemma 45 *If $B_s \ge 6^{K-s+1}(\prod_{l=s}^K T_l)^2, T_l \ge 1$ and $M \ge 6L$, then it holds that*

$$c_j^{(s-1)} \cdot (1 + T_{s-1}) < c_{T_s}^{(s)}, \qquad \textit{for } 2 \le s \le K, 0 \le j \le T_{s-1}, \tag{82}$$

and

$$c_j^{(K)} \cdot (1 + T_K) < M, \qquad \textit{for } 0 \le j \le T_K. \tag{83}$$

Next lemma is a special case of Lemma 42 with $s = K$:

Lemma 46 *Suppose p satisfies $q \prod_{i=1}^K T_i \le pT_K < (p+1)T_K \le (q+1) \prod_{i=1}^K T_i$ for some q and $pT_K < T$. For simplification, we denote*

$$\textit{start} = pT_K, \textit{end} = \min\{(p+1)T_K, T\}.$$

If $M > L$, then we have

$$\begin{aligned}
&\mathbb{E}\Bigg[F\big(\mathbf{x}_{\textit{end}}\big) + c_{T_K}^{(K)} \cdot \big\|\mathbf{x}_{\textit{end}} - \mathbf{x}_{\textit{start}}\big\|_2^2 + \sum_{j=\textit{start}}^{\textit{end}-1} \frac{\big\|\nabla F(\mathbf{x}_j)\big\|_2^2}{100M} \Bigg| \mathcal{F}_{\textit{start}}\Bigg] \\
&\le F(\mathbf{x}_{\textit{start}}) + \frac{2}{M} \cdot \mathbb{E}\big[\big\|\nabla F(\mathbf{x}_{\textit{start}}) - \mathbf{v}_{\textit{start}}\big\|_2^2 \big| \mathcal{F}_{\textit{start}}\big] \cdot (\textit{end} - \textit{start}).
\end{aligned}$$

The following lemma provides an upper bound of $\mathbb{E}\big[\big\|\nabla F(\mathbf{x}_t^{(l)}) - \mathbf{v}_t^{(l)}\big\|_2^2\big]$, which plays an important role in our proof of Lemma 42.

Lemma 47 *Let t^l be as defined in (10), then we have $\mathbf{x}_t^{(l)} = \mathbf{x}_{t^l}$, and*

$$\mathbb{E}\big[\big\|\nabla F(\mathbf{x}_t^{(l)}) - \mathbf{v}_t^{(l)}\big\|_2^2 \big| \mathcal{F}_{t^l}\big] \le \frac{L^2}{B_l}\big\|\mathbf{x}_t^{(l)} - \mathbf{x}_t^{(l-1)}\big\|_2^2 + \big\|\nabla F(\mathbf{x}_t^{(l-1)}) - \mathbf{v}_t^{(l-1)}\big\|_2^2.$$

Proof [Proof of Lemma 42] We use mathematical induction to prove that Lemma 42 holds for any $1 \le s \le K$. When $s = K$, we have the result hold because of Lemma 46. Suppose that for $s+1$, Lemma 42 holds for any p' which satisfies $p' \prod_{j=s+1}^K T_j < T$ and $q \prod_{j=1}^K T_j \le p' \prod_{j=s+1}^K T_j < (p'+1) \prod_{j=s+1}^K T_j \le (q+1) \prod_{j=1}^K T_j$. We need to prove Lemma 42 still holds for s and p, where p satisfies $p \prod_{j=s+1}^K T_j < T$ and $q \prod_{j=1}^K T_j \le p \prod_{j=s}^K T_j < (p+1) \prod_{j=s}^K T_j \le$

$(q+1)\prod_{j=1}^{K} T_j$. We choose $p' = pT_s + u$ which satisfies that $p'\prod_{j=s+1}^{K} T_j < T$, and we set indices start_u and end_u as

$$\mathrm{start}_u = p'\prod_{j=s+1}^{K} T_j, \qquad \mathrm{end}_u = \min\bigg\{\mathrm{start}_u + \prod_{j=s+1}^{K} T_j, T\bigg\}.$$

Then we have

$$\begin{aligned}
&\mathbb{E}\bigg[\sum_{j=\mathrm{start}_u}^{\mathrm{end}_u - 1} \frac{\|\nabla F(\mathbf{x}_j)\|_2^2}{100M} + F(\mathbf{x}_{\mathrm{end}_u}) + c_{T_{s+1}}^{(s+1)} \cdot \|\mathbf{x}_{\mathrm{end}_u} - \mathbf{x}_{\mathrm{start}_u}\|_2^2 \big| \mathcal{F}_{\mathrm{start}_u}\bigg] \\
&\le F(\mathbf{x}_{\mathrm{start}_u}) + \frac{2}{M}\cdot \mathbb{E}\big[\big\|\nabla F(\mathbf{x}_{\mathrm{start}_u}) - \mathbf{v}_{\mathrm{start}_u}\big\|_2^2\big|\mathcal{F}_{\mathrm{start}_u}\big]\cdot(\mathrm{end}_u - \mathrm{start}_u),
\end{aligned} \tag{84}$$

where the last inequality holds because of the induction hypothesis that Lemma 42 holds for $s+1$ and p'. Note that we have $\mathbf{x}_{\mathrm{start}_u} = \mathbf{x}_{\mathrm{start}_u}^{(s)}$ from Proposition 44, which implies

$$\begin{aligned}
\mathbb{E}\big[\big\|\nabla F(\mathbf{x}_{\mathrm{start}_u}) - \mathbf{v}_{\mathrm{start}_u}\big\|_2^2\big|\mathcal{F}_{\mathrm{start}_u}\big] &= \mathbb{E}\big[\big\|\nabla F(\mathbf{x}_{\mathrm{start}_u}^{(s)}) - \mathbf{v}_{\mathrm{start}_u}^{(s)}\big\|_2^2\big|\mathcal{F}_{\mathrm{start}_u}\big] \\
&\le \frac{L^2}{B_s}\big\|\mathbf{x}_{\mathrm{start}_u}^{(s)} - \mathbf{x}_{\mathrm{start}_u}^{(s-1)}\big\|_2^2 + \big\|\nabla F(\mathbf{x}_{\mathrm{start}_u}^{(s-1)}) - \mathbf{v}_{\mathrm{start}_u}^{(s-1)}\big\|_2^2 \qquad (85) \\
&= \frac{L^2}{B_s}\|\mathbf{x}_{\mathrm{start}_u} - \mathbf{x}_{\mathrm{start}}\|_2^2 + \big\|\nabla F(\mathbf{x}_{\mathrm{start}}) - \mathbf{v}_{\mathrm{start}}\big\|_2^2, \qquad (86)
\end{aligned}$$

where (85) holds because of Lemma 47 and (86) holds due to Proposition 44. Plugging (86) into (84) and taking expectation $\mathbb{E}[\cdot|\mathcal{F}_{\mathrm{start}}]$ for (84) will yield

$$\begin{aligned}
&\mathbb{E}\bigg[\sum_{j=\mathrm{start}_u}^{\mathrm{end}_u - 1} \frac{\|\nabla F(\mathbf{x}_j)\|_2^2}{100M} + F(\mathbf{x}_{\mathrm{end}_u}) + c_{T_{s+1}}^{(s+1)}\|\mathbf{x}_{\mathrm{end}_u} - \mathbf{x}_{\mathrm{start}_u}\|_2^2 \big| \mathcal{F}_{\mathrm{start}}\bigg] \\
&\le \mathbb{E}\bigg[F(\mathbf{x}_{\mathrm{start}_u}) + (\mathrm{end}_u - \mathrm{start}_u)\frac{2L^2}{MB_s}\|\mathbf{x}_{\mathrm{start}_u} - \mathbf{x}_{\mathrm{start}}\|_2^2 \\
&\qquad + \frac{2(\mathrm{end}_u - \mathrm{start}_u)}{M}\big\|\nabla F(\mathbf{x}_{\mathrm{start}}) - \mathbf{v}_{\mathrm{start}}\big\|_2^2\bigg|\mathcal{F}_{\mathrm{start}}\bigg] \\
&\le \mathbb{E}\bigg[F(\mathbf{x}_{\mathrm{start}_u}) + \bigg(\prod_{j=s+1}^{K} T_j\bigg)\frac{2L^2}{MB_s}\|\mathbf{x}_{\mathrm{start}_u} - \mathbf{x}_{\mathrm{start}}\|_2^2 \\
&\qquad + \frac{2(\mathrm{end}_u - \mathrm{start}_u)}{M}\big\|\nabla F(\mathbf{x}_{\mathrm{start}}) - \mathbf{v}_{\mathrm{start}}\big\|_2^2\bigg|\mathcal{F}_{\mathrm{start}}\bigg].
\end{aligned} \tag{87}$$

We now give a bound of $\|\mathbf{x}_{\mathrm{end}_u} - \mathbf{x}_{\mathrm{start}}\|_2^2$:

$$\begin{aligned}
&\|\mathbf{x}_{\mathrm{end}_u} - \mathbf{x}_{\mathrm{start}}\|_2^2 \\
&= \|\mathbf{x}_{\mathrm{start}_u} - \mathbf{x}_{\mathrm{start}}\|_2^2 + \|\mathbf{x}_{\mathrm{end}_u} - \mathbf{x}_{\mathrm{start}_u}\|_2^2 + 2\langle \mathbf{x}_{\mathrm{end}_u} - \mathbf{x}_{\mathrm{start}_u}, \mathbf{x}_{\mathrm{start}_u} - \mathbf{x}_{\mathrm{start}}\rangle \\
&\le \|\mathbf{x}_{\mathrm{start}_u} - \mathbf{x}_{\mathrm{start}}\|_2^2 + \|\mathbf{x}_{\mathrm{end}_u} - \mathbf{x}_{\mathrm{start}_u}\|_2^2 + \frac{1}{T_s}\cdot\|\mathbf{x}_{\mathrm{start}_u} - \mathbf{x}_{\mathrm{start}}\|_2^2 + T_s\cdot\|\mathbf{x}_{\mathrm{end}_u} - \mathbf{x}_{\mathrm{start}_u}\|_2^2
\end{aligned} \tag{88}$$

$$= \left(1 + \frac{1}{T_s}\right) \cdot \|\mathbf{x}_{\mathrm{start}_u} - \mathbf{x}_{\mathrm{start}}\|_2^2 + (1 + T_s) \cdot \|\mathbf{x}_{\mathrm{end}_u} - \mathbf{x}_{\mathrm{start}_u}\|_2^2, \tag{89}$$

where (88) holds because of Young's inequality. Taking expectation $\mathbb{E}[\cdot|\mathcal{F}_{\mathrm{start}}]$ over (89) and multiplying $c_{u+1}^{(s)}$ on both sides, we obtain

$$\begin{aligned} c_{u+1}^{(s)}\mathbb{E}\big[\|\mathbf{x}_{\mathrm{end}_u} - \mathbf{x}_{\mathrm{start}}\|_2^2\big|\mathcal{F}_{\mathrm{start}}\big] \le c_{u+1}^{(s)}\left(1 + \frac{1}{T_s}\right)&\mathbb{E}\big[\|\mathbf{x}_{\mathrm{start}_u} - \mathbf{x}_{\mathrm{start}}\|_2^2\big|\mathcal{F}_{\mathrm{start}}\big] \\ &+ c_{u+1}^{(s)}(1 + T_s)\mathbb{E}\big[\|\mathbf{x}_{\mathrm{end}_u} - \mathbf{x}_{\mathrm{start}_u}\|_2^2\big|\mathcal{F}_{\mathrm{start}}\big]. \end{aligned} \tag{90}$$

Adding up inequalities(90) and (87) together, we have

$$\begin{aligned} &\mathbb{E}\Bigg[\sum_{j=\mathrm{start}_u}^{\mathrm{end}_u - 1} \frac{\|\nabla F(\mathbf{x}_j)\|_2^2}{100M} + F(\mathbf{x}_{\mathrm{end}_u}) + c_{u+1}^{(s)}\|\mathbf{x}_{\mathrm{end}_u} - \mathbf{x}_{\mathrm{start}}\|_2^2 + c_{T_{s+1}}^{(s+1)}\|\mathbf{x}_{\mathrm{end}_u} - \mathbf{x}_{\mathrm{start}_u}\|_2^2\Big|\mathcal{F}_{\mathrm{start}}\Bigg] \\ &\le \mathbb{E}\Bigg[F(\mathbf{x}_{\mathrm{start}_u}) + \|\mathbf{x}_{\mathrm{start}_u} - \mathbf{x}_{\mathrm{start}}\|_2^2\Bigg[c_{u+1}^{(s)}\left(1 + \frac{1}{T_s}\right) + \frac{3L^2}{B_s M}\prod_{j=s+1}^{K} T_j\Bigg]\Bigg|\mathcal{F}_{\mathrm{start}}\Bigg] \\ &\quad + \frac{2}{M}\mathbb{E}\big[\big\|\nabla F(\mathbf{x}_{\mathrm{start}}) - \mathbf{v}_{\mathrm{start}}\big\|_2^2\big|\mathcal{F}_{\mathrm{start}}\big](\mathrm{end}_u - \mathrm{start}_u) \\ &\quad + c_{u+1}^{(s)}(1 + T_s)\mathbb{E}\big[\|\mathbf{x}_{\mathrm{end}_u} - \mathbf{x}_{\mathrm{start}_u}\|_2^2\big|\mathcal{F}_{\mathrm{start}}\big] \\ &< \mathbb{E}\big[F(\mathbf{x}_{\mathrm{start}_u}) + c_u^{(s)}\|\mathbf{x}_{\mathrm{start}_u} - \mathbf{x}_{\mathrm{start}}\|_2^2\big|\mathcal{F}_{\mathrm{start}}\big] \\ &\quad + \frac{2}{M}\mathbb{E}\big[\big\|\nabla F(\mathbf{x}_{\mathrm{start}}) - \mathbf{v}_{\mathrm{start}}\big\|_2^2\big|\mathcal{F}_{\mathrm{start}}\big](\mathrm{end}_u - \mathrm{start}_u) \\ &\quad + c_{T_{s+1}}^{(s+1)}\mathbb{E}\big[\|\mathbf{x}_{\mathrm{end}_u} - \mathbf{x}_{\mathrm{start}_u}\|_2^2\big|\mathcal{F}_{\mathrm{start}}\big], \end{aligned} \tag{91}$$

where the last inequality holds due to the fact that $c_u^{(s)} = c_{u+1}^{(s)}(1 + 1/T_s) + 3L^2/(B_s M) \cdot \prod_{j=s+1}^{K} T_j$ by Definition 41 and $c_{u+1}^{(s)} \cdot (1 + T_s) < c_{T_{s+1}}^{(s+1)}$ by Lemma 45. Cancelling out the term $c_{T_{s+1}}^{(s+1)}\mathbb{E}\big[\|\mathbf{x}_{\mathrm{end}_u} - \mathbf{x}_{\mathrm{start}_u}\|_2^2\big|\mathcal{F}_{\mathrm{start}}\big]$ from both sides of (91), we get

$$\begin{aligned} &\sum_{j=\mathrm{start}_u}^{\mathrm{end}_u - 1}\mathbb{E}\Bigg[\frac{\|\nabla F(\mathbf{x}_j)\|_2^2}{100M}\Bigg|\mathcal{F}_{\mathrm{start}}\Bigg] + \mathbb{E}\big[F(\mathbf{x}_{\mathrm{end}_u}) + c_{u+1}^{(s)} \cdot \|\mathbf{x}_{\mathrm{end}_u} - \mathbf{x}_{\mathrm{start}}\|_2^2\big|\mathcal{F}_{\mathrm{start}}\big] \\ &\le \mathbb{E}\big[F(\mathbf{x}_{\mathrm{start}_u}) + c_u^{(s)}\|\mathbf{x}_{\mathrm{start}_u} - \mathbf{x}_{\mathrm{start}}\|_2^2\big|\mathcal{F}_{\mathrm{start}}\big] \\ &\qquad + \frac{2}{M}\mathbb{E}\big[\big\|\nabla F(\mathbf{x}_{\mathrm{start}}) - \mathbf{v}_{\mathrm{start}}\big\|_2^2\big|\mathcal{F}_{\mathrm{start}}\big](\mathrm{end}_u - \mathrm{start}_u). \end{aligned} \tag{92}$$

We now try to telescope the above inequality. We first suppose that $u^* = \max\{0 \le u < T_s : \mathrm{start}_u < T\}$. Next we telescope (92) for $u = 0$ to u^*. Since we have $\mathrm{start}_u = \mathrm{end}_{u-1}, \mathrm{start}_0 = \mathrm{start}$ for $0 \le u \le u^*$, then we get

$$\begin{aligned} &\mathbb{E}\Bigg[\sum_{u=0}^{u^*}\sum_{j=\mathrm{start}_u}^{\mathrm{end}_u - 1}\frac{\|\nabla F(\mathbf{x}_j)\|_2^2}{100M} + F(\mathbf{x}_{\mathrm{end}_{u^*}}) + c_{u^*}^{(s)} \cdot \|\mathbf{x}_{\mathrm{end}_{u^*}} - \mathbf{x}_{\mathrm{start}}\|_2^2\Big|\mathcal{F}_{\mathrm{start}}\Bigg] \\ &\le F(\mathbf{x}_{\mathrm{start}}) + \frac{2T_s}{M} \cdot \mathbb{E}\big[\big\|\nabla F(\mathbf{x}_{\mathrm{start}}) - \mathbf{v}_{\mathrm{start}}\big\|_2^2\big|\mathcal{F}_{\mathrm{start}}\big] \cdot \sum_{u=0}^{u^*}(\mathrm{end}_u - \mathrm{start}_u). \end{aligned}$$

Since for $0 \le u \le u^*$, we have $\text{start}_u = \text{end}_{u-1}, \text{start}_0 = \text{start}$, $\text{end}_{u^*} = \text{end}$, and $c_{u^*}^{(s)} > c_{T_s}^{(s)}$, thus we have that

$$\mathbb{E}\bigg[\sum_{j=\text{start}}^{\text{end}-1} \frac{\|\nabla F(\mathbf{x}_j)\|_2^2}{100M} + F(\mathbf{x}_{\text{end}}) + c_{T_s}^{(s)} \cdot \|\mathbf{x}_{\text{end}} - \mathbf{x}_{\text{start}}\|_2^2 \big| \mathcal{F}_{\text{start}}\bigg]$$
$$\le F(\mathbf{x}_{\text{start}}) + \frac{2}{M} \cdot \mathbb{E}\big[\big\|\nabla F(\mathbf{x}_{\text{start}}) - \mathbf{v}_{\text{start}}\big\|_2^2 \big| \mathcal{F}_{\text{start}}\big] \cdot (\text{end} - \text{start}). \tag{93}$$

Therefore, we have proved that Lemma 42 still holds for s and p. Then by mathematical induction, we have for all $1 \le s \le K$ and p which satisfy $q\prod_{j=1}^K T_j \le p \cdot \prod_{j=s}^K T_j < (p+1) \cdot \prod_{j=s}^K T_j \le (q+1)\prod_{j=1}^K T_j$, Lemma 42 holds. ■

F.2. Proof of Lemma 43

The following proof is adapted from that of Lemma A.1 in Lei et al. (2017). We provide the proof here for the self-containedness of our paper.

Proof [Proof of Lemma 43] We only consider the case when $m < N$. Let $W_j = \mathbb{1}(j \in \mathcal{J})$, then we have

$$\mathbb{E}W_j^2 = \mathbb{E}W_j = \frac{m}{N}, \mathbb{E}W_j W_{j'} = \frac{m(m-1)}{N(N-1)}.$$

Thus we can rewrite the sample mean as

$$\frac{1}{m}\sum_{j \in \mathcal{J}} \mathbf{a}_j = \frac{1}{m}\sum_{i=1}^N W_i \mathbf{a}_i,$$

which immediately implies

$$\begin{aligned}
\mathbb{E}\bigg\|\frac{1}{m}\sum_{j\in\mathcal{J}}\mathbf{a}_j\bigg\|^2 &= \frac{1}{m^2}\bigg(\sum_{j=1}^N \mathbb{E}W_j^2\|\mathbf{a}_j\|_2^2 + \sum_{j\neq j'}\mathbb{E}W_jW_{j'}\langle\mathbf{a}_j, \mathbf{a}_{j'}\rangle\bigg) \\
&= \frac{1}{m^2}\bigg(\frac{m}{N}\sum_{j=1}^N\|\mathbf{a}_j\|_2^2 + \frac{m(m-1)}{N(N-1)}\sum_{j\neq j'}\langle\mathbf{a}_j, \mathbf{a}_{j'}\rangle\bigg) \\
&= \frac{1}{m^2}\bigg(\bigg(\frac{m}{N} - \frac{m(m-1)}{N(N-1)}\bigg)\sum_{j=1}^N\|\mathbf{a}_j\|_2^2 + \frac{m(m-1)}{N(N-1)}\bigg\|\sum_{j=1}^N\mathbf{a}_j\bigg\|_2^2\bigg) \\
&= \frac{1}{m^2}\bigg(\frac{m}{N} - \frac{m(m-1)}{N(N-1)}\bigg)\sum_{j=1}^N\|\mathbf{a}_j\|_2^2 \\
&\le \frac{1}{m}\cdot\frac{1}{N}\sum_{j=1}^N\|\mathbf{a}_j\|_2^2.
\end{aligned}$$

■

Appendix G. Proofs of Auxiliary Lemmas

In this section, we present the additional proofs of supporting lemmas used in Appendix F. Let $M, \{T_l\}, \{B_l\}$ and B_0 be the parameters defined in Algorithm 1. Let $\{\mathbf{x}_t^{(l)}\}, \{\mathbf{g}_t^{(l)}\}$ be the reference points and reference gradients used in Algorithm 1. Finally, $\mathbf{v}_t^{(l)}, \mathcal{F}_t$ are the variables and filtration defined in Appendix E and $c_j^{(s)}$ are the constant series defined in Definition 41.

G.1. Proof of Proposition 44

Proof [Proof of Proposition 44] By the definition of reference point $\mathbf{x}_t^{(s)}$ in (10), we can easily verify that (79) holds trivially.

Next we prove (80). Note that by (79) we have $\mathbf{x}_t^{(s)} = \mathbf{x}_{t'}^{(s)}$. For any $0 \le s' \le s$, it is also true that $\mathbf{x}_t^{(s')} = \mathbf{x}_{t'}^{(s')}$ by (10), which means $\mathbf{x}_t$ and $\mathbf{x}_{t'}$ share the same first $s+1$ reference points. Then by the update rule of $\mathbf{g}_t^{(s')}$ in Algorithm 1, we will maintain $\mathbf{g}_t^{(s')}$ unchanged from time step t to t'. In other worlds, we have $\mathbf{g}_t^{(s')} = \mathbf{g}_{t'}^{(s')}$ for all $0 \le s' \le s$.

We now prove the last claim (81). Based on (72) and (80), we have $\mathbf{v}_t^{(s)} = \sum_{s'=0}^{s} \mathbf{g}_t^{(s')} = \sum_{s'=0}^{s} \mathbf{g}_{p\cdot\prod_{j=s+1}^K T_j}^{(s')} = \mathbf{v}_{p\cdot\prod_{j=s+1}^K T_j}^{(s)}$. Since for any $s \le s'' \le K$, we have the following equations by the update in Algorithm 1 (Line 14).

$$
\begin{aligned}
\mathbf{x}_{p\cdot\prod_{j=s+1}^K T_j}^{(s'')} &= \mathbf{x}_{\lfloor p\cdot\prod_{j=s+1}^K T_j / \prod_{j=s''+1}^K T_j \rfloor \cdot \prod_{j=s''+1}^K T_j} \\
&= \mathbf{x}_{p\cdot\prod_{j=s+1}^K T_j / \prod_{j=s''+1}^K T_j \cdot \prod_{j=s''+1}^K T_j} \\
&= \mathbf{x}_{p\cdot\prod_{j=s+1}^K T_j}^{(s)}.
\end{aligned}
$$

Then for any $s < s'' \le K$, we have

$$
\mathbf{g}_{p\cdot\prod_{j=s+1}^K T_j}^{(s'')} = \frac{1}{B_{s''}} \sum_{i\in I} \Big[\nabla f_i\Big(\mathbf{x}_{p\cdot\prod_{j=s+1}^K T_j}^{(s'')}\Big) - \nabla f_i\Big(\mathbf{x}_{p\cdot\prod_{j=s+1}^K T_j}^{(s''-1)}\Big)\Big] = 0. \tag{94}
$$

Thus, we have

$$
\mathbf{v}_{p\cdot\prod_{j=s+1}^K T_j} = \sum_{s''=0}^{K} \mathbf{g}_{p\cdot\prod_{j=s+1}^K T_j}^{(s'')} = \sum_{s''=0}^{s} \mathbf{g}_{p\cdot\prod_{j=s+1}^K T_j}^{(s'')} = \sum_{s''=0}^{s} \mathbf{g}_t^{(s'')} = \mathbf{v}_t^{(s)}, \tag{95}
$$

where the first equality holds because of the definition of $\mathbf{v}_{p\cdot\prod_{j=s+1}^K T_j}$, the second equality holds due to (94) , the third equality holds due to (80) and the last equality holds due to (72). This completes the proof of (81). ■

G.2. Proof of Lemma 45

Proof [Proof of Lemma 45] For any fixed s, it can be seen that from the definition in (74), $c_j^{(s)}$ is monotonically decreasing with j. In order to prove (82), we only need to compare

$(1+T_{s-1}) \cdot c_0^{(s-1)}$ and $c_{T_s}^{(s)}$. Furthermore, by the definition of series $\{c_j^{(s)}\}$ in (74), it can be inducted that when $0 \le j \le T_{s-1}$,

$$c_j^{(s-1)} = \left(1 + \frac{1}{T_{s-1}}\right)^{T_{s-1}-j} \cdot c_{T_{s-1}}^{(s-1)} + \frac{(1+1/T_{s-1})^{T_{s-1}-j} - 1}{1/T_{s-1}} \cdot \frac{3L^2}{M} \cdot \frac{\prod_{l=s}^K T_l}{B_{s-1}}. \tag{96}$$

We take $j = 0$ in (96) and obtain

$$\begin{aligned} c_0^{(s-1)} &= \left(1 + \frac{1}{T_{s-1}}\right)^{T_{s-1}} \cdot c_{T_{s-1}}^{(s-1)} + \frac{(1+1/T_{s-1})^{T_{s-1}} - 1}{1/T_{s-1}} \cdot \frac{3L^2}{M} \cdot \frac{\prod_{l=s}^K T_l}{B_{s-1}} \\ &< 2.8 \times c_{T_{s-1}}^{(s-1)} + \frac{6L^2}{M} \cdot \frac{\prod_{l=s-1}^K T_l}{B_{s-1}} \qquad (97) \\ &\le \frac{2.8M + 6L^2/M}{6^{K-s+2} \cdot \prod_{l=s-1}^K T_l} \qquad (98) \\ &< \frac{3M}{6^{K-s+2} \cdot \prod_{l=s-1}^K T_l}, \qquad (99) \end{aligned}$$

where (97) holds because $(1+1/n)^n < 2.8$ for any $n \ge 1$, (98) holds due to the definition of $c_{T_{s-1}}^{(s-1)}$ in (73) and $B_{s-1} \ge 6^{K-s+2}(\prod_{l=s-1}^K T_l)^2$ and (99) holds because $M \ge 6L$. Recall that $c_j^{(s)}$ is monotonically decreasing with j and the inequality in (99). Thus for all $2 \le s \le K$ and $0 \le j \le T_{s-1}$, we have

$$\begin{aligned} (1+T_{s-1}) \cdot c_j^{(s-1)} &\le (1+T_{s-1}) \cdot c_0^{(s-1)} \\ &\le (1+T_{s-1}) \cdot \frac{3M}{6^{K-s+2} \cdot \prod_{l=s-1}^K T_l} \\ &< \frac{6M}{6^{K-s+2} \cdot \prod_{l=s}^K T_l} \\ &= c_{T_s}^{(s)}, \end{aligned} \tag{100}$$

where the third inequality holds because $(1+T_{s-1})/T_{s-1} \le 2$ when $T_{s-1} \ge 1$ and the last equation comes from the definition of $c_{T_s}^s$ in (73). This completes the proof of (82).

Using similar techniques, we can obtain the upper bound for c_0^K which is similar to inequality (99) with $s-1$ replaced by K. Therefore, we have

$$(1+T_K) \cdot c_j^{(K)} \le (1+T_K) \cdot c_0^{(K)} < \frac{6M}{6^{K-K+1} \cdot \prod_{l=K}^K T_l} \le M,$$

which completes the proof of (83). ■

G.3. Proof of Lemma 46

Now we prove Lemma 46, which is a special case of Lemma 42 when we choose $s = K$.

Proof [Proof of Lemma 46] To simplify notations, we use $\mathbb{E}[\cdot]$ to denote the conditional expectation $\mathbb{E}[\cdot|\mathcal{F}_{p \cdot T_K}]$ in the rest of this proof. For $pT_K \le pT_K + j < \min\{(p+1)T_K, T\}$, we denote $\mathbf{h}_{p\cdot T_K+j} = -(10M)^{-1}\cdot \mathbf{v}_{p\cdot T_K+j}$. According to the update in Algorithm 1 (Line 9), we have

$$\mathbf{x}_{p\cdot T_K+j+1} = \mathbf{x}_{p\cdot T_K+j} + \mathbf{h}_{p\cdot T_K+j}, \tag{101}$$

which immediately implies

$$\begin{aligned}
&F(\mathbf{x}_{p\cdot T_K+j+1})\\
&= F(\mathbf{x}_{p\cdot T_K+j} + \mathbf{h}_{p\cdot T_K+j})\\
&\le F(\mathbf{x}_{p\cdot T_K+j}) + \langle \nabla F(\mathbf{x}_{p\cdot T_K+j}), \mathbf{h}_{p\cdot T_K+j}\rangle + \frac{L}{2}\|\mathbf{h}_{p\cdot T_K+j}\|_2^2 \qquad (102)\\
&= \big[\langle \mathbf{v}_{p\cdot T_K+j}, \mathbf{h}_{p\cdot T_K+j}\rangle + 5M\|\mathbf{h}_{p\cdot T_K+j}\|_2^2\big] + F(\mathbf{x}_{p\cdot T_K+j})\\
&\qquad + \langle \nabla F(\mathbf{x}_{p\cdot T_K+j}) - \mathbf{v}_{p\cdot T_K+j}, \mathbf{h}_{p\cdot T_K+j}\rangle + \bigg(\frac{L}{2} - 5M\bigg)\|\mathbf{h}_{p\cdot T_K+j}\|_2^2\\
&\le F(\mathbf{x}_{p\cdot T_K+j}) + \langle \nabla F(\mathbf{x}_{p\cdot T_K+j}) - \mathbf{v}_{p\cdot T_K+j}, \mathbf{h}_{p\cdot T_K+j}\rangle + (L-5M)\|\mathbf{h}_{p\cdot T_K+j}\|_2^2, \qquad (103)
\end{aligned}$$

where (102) is due to the L-smoothness of F and (103) holds because $\langle \mathbf{v}_{p\cdot T_K+j}, \mathbf{h}_{p\cdot T_K+j}\rangle + 5M\|\mathbf{h}_{p\cdot T_K+j}\|_2^2 = -5M\|\mathbf{h}_{p\cdot T_K+j}\|_2^2 \le 0$. Further by Young's inequality, we obtain

$$\begin{aligned}
F(\mathbf{x}_{p\cdot T_K+j+1}) &\le F(\mathbf{x}_{p\cdot T_K+j}) + \frac{1}{2M}\|\nabla F(\mathbf{x}_{p\cdot T_K+j}) - \mathbf{v}_{p\cdot T_K+j}\|_2^2 + \bigg(\frac{M}{2} + L - 5M\bigg)\|\mathbf{h}_{p\cdot T_K+j}\|_2^2\\
&\le F(\mathbf{x}_{p\cdot T_K+j}) + \frac{1}{M}\|\nabla F(\mathbf{x}_{p\cdot T_K+j}) - \mathbf{v}_{p\cdot T_K+j}\|_2^2 - 3M\|\mathbf{h}_{p\cdot T_K+j}\|_2^2, \qquad (104)
\end{aligned}$$

where the second inequality holds because $M > L$. Now we bound the term $c_{j+1}^{(K)}\|\mathbf{x}_{p\cdot T_K+j+1} - \mathbf{x}_{p\cdot T_K}\|_2^2$. By (101) we have

$$\begin{aligned}
&c_{j+1}^{(K)}\|\mathbf{x}_{p\cdot T_K+j+1} - \mathbf{x}_{p\cdot T_K}\|_2^2\\
&= c_{j+1}^{(K)}\|\mathbf{x}_{p\cdot T_K+j} - \mathbf{x}_{p\cdot T_K} + \mathbf{h}_{p\cdot T_K+j}\|_2^2\\
&= c_{j+1}^{(K)}\big[\|\mathbf{x}_{p\cdot T_K+j} - \mathbf{x}_{p\cdot T_K}\|_2^2 + \|\mathbf{h}_{p\cdot T_K+j}\|_2^2 + 2\langle \mathbf{x}_{p\cdot T_K+j} - \mathbf{x}_{p\cdot T_K}, \mathbf{h}_{p\cdot T_K+j}\rangle\big].
\end{aligned}$$

Applying Young's inequality yields

$$\begin{aligned}
&c_{j+1}^{(K)}\|\mathbf{x}_{p\cdot T_K+j+1} - \mathbf{x}_{p\cdot T_K}\|_2^2\\
&\le c_{j+1}^{(K)}\bigg[\|\mathbf{x}_{p\cdot T_K+j} - \mathbf{x}_{p\cdot T_K}\|_2^2 + \|\mathbf{h}_{p\cdot T_K+j}\|_2^2\\
&\qquad + \frac{1}{T_K}\|\mathbf{x}_{p\cdot T_K+j} - \mathbf{x}_{p\cdot T_K}\|_2^2 + T_K\|\mathbf{h}_{p\cdot T_K+j}\|_2^2\bigg]\\
&= c_{j+1}^{(K)}\bigg[\bigg(1 + \frac{1}{T_K}\bigg)\|\mathbf{x}_{p\cdot T_K+j} - \mathbf{x}_{p\cdot T_K}\|_2^2 + (1+T_K)\|\mathbf{h}_{p\cdot T_K+j}\|_2^2\bigg], \qquad (105)
\end{aligned}$$

Adding up inequalities (105) and (104), we get

$$F(\mathbf{x}_{p\cdot T_K+j+1}) + c_{j+1}^{(K)}\|\mathbf{x}_{p\cdot T_K+j+1} - \mathbf{x}_{p\cdot T_K}\|_2^2$$

$$
\begin{aligned}
&\leq F(\mathbf{x}_{p\cdot T_K+j}) + \frac{1}{M}\|\nabla F(\mathbf{x}_{p\cdot T_K+j}) - \mathbf{v}_{p\cdot T_K+j}\|_2^2 - \big[3M - c_{j+1}^{(K)}(1+T_K)\big]\|\mathbf{h}_{p\cdot T_K+j}\|_2^2 \\
&\qquad + c_{j+1}^{(K)}\bigg(1+\frac{1}{T_K}\bigg)\|\mathbf{x}_{p\cdot T_K+j} - \mathbf{x}_{p\cdot T_K}\|_2^2 \\
&\leq F(\mathbf{x}_{p\cdot T_K+j}) + \frac{1}{M}\|\nabla F(\mathbf{x}_{p\cdot T_K+j}) - \mathbf{v}_{p\cdot T_K+j}\|_2^2 - 2M\|\mathbf{h}_{p\cdot T_K+j}\|_2^2 \\
&\qquad + c_{j+1}^{(K)}\bigg(1+\frac{1}{T_K}\bigg)\|\mathbf{x}_{p\cdot T_K+j} - \mathbf{x}_{p\cdot T_K}\|_2^2, \qquad (106)
\end{aligned}
$$

where the last inequality holds due to the fact that $c_{j+1}^{(K)}(1+T_K) < M$ by Lemma 45. Next we bound $\|\nabla F(\mathbf{x}_{p\cdot T_K+j})\|_2^2$ with $\|\mathbf{h}_{p\cdot T_K+j}\|_2^2$. Note that by (101)

$$
\begin{aligned}
\|\nabla F(\mathbf{x}_{p\cdot T_K+j})\|_2^2 &= \big\|\big[\nabla F(\mathbf{x}_{p\cdot T_K+j}) - \mathbf{v}_{p\cdot T_K+j}\big] - 10M\mathbf{h}_{p\cdot T_K+j}\big\|_2^2 \\
&\leq 2\big(\|\nabla F(\mathbf{x}_{p\cdot T_K+j}) - \mathbf{v}_{p\cdot T_K+j}\|_2^2 + 100M^2\|\mathbf{h}_{p\cdot T_K+j}\|_2^2\big),
\end{aligned}
$$

which immediately implies

$$
-2M\|\mathbf{h}_{p\cdot T_K+j}\|_2^2 \leq \frac{2}{100M}\big(\|\nabla F(\mathbf{x}_{p\cdot T_K+j}) - \mathbf{v}_{p\cdot T_K+j}\|_2^2 - \frac{1}{100M}\|\nabla F(\mathbf{x}_{p\cdot T_K+j})\|_2^2. \qquad (107)
$$

Plugging (107) into (106), we have

$$
\begin{aligned}
&F(\mathbf{x}_{p\cdot T_K+j+1}) + c_{j+1}^{(K)}\|\mathbf{x}_{p\cdot T_K+j+1} - \mathbf{x}_{p\cdot T_K}\|_2^2 \\
&\leq F(\mathbf{x}_{p\cdot T_K+j}) + \frac{1}{M}\|\nabla F(\mathbf{x}_{p\cdot T_K+j}) - \mathbf{v}_{p\cdot T_K+j}\|_2^2 + \frac{1}{50M}\cdot\|\nabla F(\mathbf{x}_{p\cdot T_K+j}) - \mathbf{v}_{p\cdot T_K+j}\|_2^2 \\
&\qquad - \frac{1}{100M}\|\nabla F(\mathbf{x}_{p\cdot T_K+j})\|_2^2 + c_{j+1}^{(K)}\bigg(1+\frac{1}{T_K}\bigg)\|\mathbf{x}_{p\cdot T_K+j} - \mathbf{x}_{p\cdot T_K}\|_2^2 \\
&\leq F(\mathbf{x}_{p\cdot T_K+j}) + \frac{2}{M}\|\nabla F(\mathbf{x}_{p\cdot T_K+j}) - \mathbf{v}_{p\cdot T_K+j}\|_2^2 - \frac{1}{100M}\|\nabla F(\mathbf{x}_{p\cdot T_K+j})\|_2^2 \\
&\qquad + c_{j+1}^{(K)}\bigg(1+\frac{1}{T_K}\bigg)\|\mathbf{x}_{p\cdot T_K+j} - \mathbf{x}_{p\cdot T_K}\|_2^2. \qquad (108)
\end{aligned}
$$

Next we bound $\|\nabla F(\mathbf{x}_{p\cdot T_K+j}) - \mathbf{v}_{p\cdot T_K+j}\|_2^2$. First, by Lemma 47 we have

$$
\mathbb{E}\Big\|\nabla F(\mathbf{x}_{p\cdot T_K+j}^{(K)}) - \mathbf{v}_{p\cdot T_K+j}^{(K)}\Big\|_2^2 \leq \frac{L^2}{B_K}\mathbb{E}\Big\|\mathbf{x}_{p\cdot T_K+j}^{(K)} - \mathbf{x}_{p\cdot T_K+j}^{(K-1)}\Big\|_2^2 + \mathbb{E}\Big\|\nabla F(\mathbf{x}_{p\cdot T_K+j}^{(K-1)}) - \mathbf{v}_{p\cdot T_K+j}^{(K-1)}\Big\|_2^2.
$$

Since $\mathbf{x}_{p\cdot T_K+j}^{(K)} = \mathbf{x}_{p\cdot T_K+j}, \mathbf{v}_{p\cdot T_K+j}^{(K)} = \mathbf{v}_{p\cdot T_K+j}$, $\mathbf{x}_{p\cdot T_K+j}^{(K-1)} = \mathbf{x}_{p\cdot T_K}$ and $\mathbf{v}_{p\cdot T_K+j}^{(K-1)} = \mathbf{v}_{p\cdot T_K}$, we have

$$
\mathbb{E}\|\nabla F(\mathbf{x}_{p\cdot T_K+j}) - \mathbf{v}_{p\cdot T_K+j}\|_2^2 \leq \frac{L^2}{B_K}\mathbb{E}\|\mathbf{x}_{p\cdot T_K+j} - \mathbf{x}_{p\cdot T_K}\|_2^2 + \mathbb{E}\|\nabla F(\mathbf{x}_{p\cdot T_K}) - \mathbf{v}_{p\cdot T_K}\|_2^2. \qquad (109)
$$

Taking expectation $\mathbb{E}[\cdot]$ with (108) and plugging (109) into (108) , we obtain

$$
\mathbb{E}\bigg[F(\mathbf{x}_{p\cdot T_K+j+1}) + c_{j+1}^{(K)}\|\mathbf{x}_{p\cdot T_K+j+1} - \mathbf{x}_{p\cdot T_K}\|_2^2 + \frac{1}{100M}\|\nabla F(\mathbf{x}_{p\cdot T_K+j})\|_2^2\bigg]
$$

$$
\begin{aligned}
&\leq \mathbb{E}\bigg[F(\mathbf{x}_{p\cdot T_K+j}) + \bigg(c_{j+1}^{(K)}\bigg(1+\frac{1}{T_K}\bigg) + \frac{3L^2}{B_K M}\bigg)\|\mathbf{x}_{p\cdot T_K+j} - \mathbf{x}_{p\cdot T_K}\|_2^2 \\
&\qquad + \frac{2}{M}\|\nabla F(\mathbf{x}_{p\cdot T_K}) - \mathbf{v}_{p\cdot T_K}\|_2^2\bigg] \\
&= \mathbb{E}\bigg[F(\mathbf{x}_{p\cdot T_K+j}) + c_j^{(K)}\|\mathbf{x}_{p\cdot T_K+j} - \mathbf{x}_{p\cdot T_K}\|_2^2 + \frac{2}{M}\cdot\|\nabla F(\mathbf{x}_{p\cdot T_K}) - \mathbf{v}_{p\cdot T_K}\|_2^2\bigg], \qquad (110)
\end{aligned}
$$

where (110) holds because we have $c_j^{(K)} = c_{j+1}^{(K)}(1+1/T_K) + 3L^2/(B_K M)$ by Definition 41. Telescoping (110) for $j = 0$ to $\mathrm{end} - \mathrm{start} - 1$, we have

$$
\begin{aligned}
&\mathbb{E}\big[F\big(\mathbf{x}_{\mathrm{end}}\big) + c_{T_K}^{(K)}\cdot\|\mathbf{x}_{\mathrm{end}} - \mathbf{x}_{\mathrm{start}}\|_2^2\big] + \frac{1}{100M}\sum_{j=\mathrm{start}}^{\mathrm{end}-1}\mathbb{E}\|\nabla F(\mathbf{x}_j)\|_2^2 \\
&\leq \mathbb{E}\big[F\big(\mathbf{x}_{\mathrm{end}}\big) + c_{\mathrm{end}-\mathrm{start}}^{(K)}\cdot\|\mathbf{x}_{\mathrm{end}} - \mathbf{x}_{\mathrm{start}}\|_2^2\big] + \frac{1}{100M}\sum_{j=\mathrm{start}}^{\mathrm{end}-1}\mathbb{E}\|\nabla F(\mathbf{x}_j)\|_2^2 \\
&\leq F(\mathbf{x}_{\mathrm{start}}) + \frac{2(\mathrm{end}-\mathrm{start})}{M}\cdot\mathbb{E}\|\nabla F(\mathbf{x}_{\mathrm{start}}) - \mathbf{v}_{\mathrm{start}}\|_2^2,
\end{aligned}
$$

which completes the proof. ■

G.4. Proof of Lemma 47

Proof [Proof of Lemma 47] If $t^l = t^{l-1}$, we have $\mathbf{x}_t^{(l)} = \mathbf{x}_t^{(l-1)}$ and $\mathbf{v}_t^{(l)} = \mathbf{v}_t^{(l-1)}$. In this case the statement in Lemma 47 holds trivially. Therefore, we assume $t^l \neq t^{l-1}$ in the following proof. Note that

$$
\begin{aligned}
&\mathbb{E}\big[\big\|\nabla F(\mathbf{x}_t^{(l)}) - \mathbf{v}_t^{(l)}\big\|_2^2|\mathcal{F}_{t^l}\big] \\
&= \mathbb{E}\big[\big\|\nabla F(\mathbf{x}_t^{(l)}) - \mathbf{v}_t^{(l)} - \mathbb{E}\big[\nabla F(\mathbf{x}_t^{(l)}) - \mathbf{v}_t^{(l)}\big]\big\|_2^2|\mathcal{F}_{t^l}\big] + \big\|\mathbb{E}\big[\nabla F(\mathbf{x}_t^{(l)}) - \mathbf{v}_t^{(l)}|\mathcal{F}_{t^l}\big]\big\|_2^2 \\
&= \underbrace{\mathbb{E}\Bigg[\Bigg\|\nabla F(\mathbf{x}_t^{(l)}) - \sum_{j=0}^{l}\mathbf{g}_t^{(j)} - \mathbb{E}\Bigg[\nabla F(\mathbf{x}_t^{(l)}) - \sum_{j=0}^{l}\mathbf{g}_t^{(j)}\Bigg]\Bigg\|_2^2\Bigg|\mathcal{F}_{t^l}\Bigg]}_{J_1} + \underbrace{\Bigg\|\mathbb{E}\Bigg[\nabla F(\mathbf{x}_t^{(l)}) - \sum_{j=0}^{l}\mathbf{g}_t^{(j)}\Bigg|\mathcal{F}_{t^l}\Bigg]\Bigg\|_2^2}_{J_2},
\end{aligned}
\qquad (111)
$$

where in the second equation we used the definition $\mathbf{v}_t^{(l)} = \sum_{i=0}^{l}\mathbf{g}_t^{(i)}$ in (72). We first upper bound term J_1. According to the update rule in Algorithm 1 (Line 20-23), when $j < l$, $\mathbf{g}_t^{(j)}$ will not be updated at the t^l-th iteration. Thus we have $\mathbb{E}[\mathbf{g}_t^{(j)}|\mathcal{F}_{t^l}] = \mathbf{g}_t^{(j)}$ for all $j < l$. In addition, by the definition of $\mathcal{F}_{t^l}$, we have $\mathbb{E}[\nabla F(\mathbf{x}_t^{(l)})|\mathcal{F}_{t^l}] = \nabla F(\mathbf{x}_t^{(l)})$. Then we have the following equation

$$
J_1 = \mathbb{E}\big[\big\|\mathbf{g}_t^{(l)} - \mathbb{E}\big[\mathbf{g}_t^{(l)}|\mathcal{F}_{t^l}\big]\big\|_2^2|\mathcal{F}_{t^l}\big]. \qquad (112)
$$

We further have

$$
\mathbf{g}_t^{(l)} = \frac{1}{B_l}\sum_{i\in I}\big[\nabla f_i(\mathbf{x}_t^{(l)}) - \nabla f_i(\mathbf{x}_t^{(l-1)})\big], \quad \mathbb{E}\big[\mathbf{g}_t^{(l)}\big|\mathcal{F}_{t^l}\big] = \nabla F(\mathbf{x}_t^{(l)}) - \nabla F(\mathbf{x}_t^{(l-1)}).
$$

Therefore, we can apply Lemma 43 to (112) and obtain

$$\begin{aligned}
J_1 &\le \frac{1}{B_l}\cdot\frac{1}{n}\sum_{i=1}^{n}\big\|\nabla f_i(\mathbf{x}_t^{(l)}) - \nabla f_i(\mathbf{x}_t^{(l-1)}) - \big[\nabla F(\mathbf{x}_t^{(l)}) - \nabla F(\mathbf{x}_t^{(l-1)})\big]\big\|_2^2 \\
&\le \frac{1}{B_l n}\sum_{i=1}^{n}\big\|\nabla f_i(\mathbf{x}_t^{(l)}) - \nabla f_i(\mathbf{x}_t^{(l-1)})\big\|_2^2 \\
&\le \frac{L^2}{B_l}\big\|\mathbf{x}_t^{(l)} - \mathbf{x}_t^{(l-1)}\big\|_2^2, \qquad (113)
\end{aligned}$$

where the second inequality is due to the fact that $\mathbb{E}[\|\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}]\|_2^2] \le \mathbb{E}\|\boldsymbol{X}\|_2^2$ for any random vector $\boldsymbol{X}$ and the last inequality holds due to the fact that F has averaged L-Lipschitz gradient.

Next we turn to bound term J_2. Note that

$$\mathbb{E}\big[\mathbf{g}_t^{(l)}\big|\mathcal{F}_{t^l}\big] = \mathbb{E}\bigg[\frac{1}{B_l}\sum_{i\in I}\big[\nabla f_i(\mathbf{x}_t^{(l)}) - \nabla f_i(\mathbf{x}_t^{(l-1)})\big]\bigg|\mathcal{F}_{t^l}\bigg] = \nabla F(\mathbf{x}_t^{(l)}) - \nabla F(\mathbf{x}_t^{(l-1)}),$$

which immediately implies

$$\begin{aligned}
\mathbb{E}\bigg[\nabla F(\mathbf{x}_t^{(l)}) - \sum_{j=0}^{l}\mathbf{g}_t^{(j)}\bigg|\mathcal{F}_{t^l}\bigg] &= \mathbb{E}\bigg[\nabla F(\mathbf{x}_t^{(l)}) - \nabla F(\mathbf{x}_t^{(l)}) + \nabla F(\mathbf{x}_t^{(l-1)}) - \sum_{j=0}^{l-1}\mathbf{g}_t^{(j)}\bigg|\mathcal{F}_{t^l}\bigg] \\
&= \mathbb{E}\big[\nabla F(\mathbf{x}_t^{(l-1)}) - \mathbf{v}_t^{(l-1)}\big|\mathcal{F}_{t^l}\big] \\
&= \nabla F(\mathbf{x}_t^{(l-1)}) - \mathbf{v}_t^{(l-1)},
\end{aligned}$$

where the last equation is due to the definition of $\mathcal{F}_t$. Plugging J_1 and J_2 into (111) yields the following result:

$$\mathbb{E}\big[\big\|\nabla F(\mathbf{x}_t^{(l)}) - \mathbf{v}_t^{(l)}\big\|_2^2\big|\mathcal{F}_{t^l}\big] \le \frac{L^2}{B_l}\big\|\mathbf{x}_t^{(l)} - \mathbf{x}_t^{(l-1)}\big\|_2^2 + \big\|\nabla F(\mathbf{x}_t^{(l-1)}) - \mathbf{v}_t^{(l-1)}\big\|_2^2,$$

which completes the proof. ■

Appendix H. More Details of the Proposed Algorithms

In this section, we give additional details about the proposed algorithms. In particular, we will present an equivalent version of Algorithm 1, which shows an alternative view of interpreting it. We will also present the detailed Neon algorithm for the self-containedness.

H.1. An Equivalent Version of Algorithm 1

Recall the One-epoch-SNVRG algorithm in Algorithm 1. Here we present an equivalent version of Algorithm 1 using nested loops, which is displayed in Algorithm 6 and is more aligned with the illustration in Figure 2(b). Note that the notation used in Algorithm 6 is slightly different from that in Algorithm 1 to avoid confusion.

Algorithm 6 One-epoch-SNVRG($F, \mathbf{x}_0, K, M, \{T_i\}, \{B_i\}, B$)

Input: Function F, starting point $\mathbf{x}_0$, loop number K, step size parameter M, loop parameters $T_i, i \in [K]$, batch parameters $B_i, i \in [K]$, base batch $B > 0$.
Output: $[\mathbf{x}_{\mathrm{out}}, \mathbf{x}_{\mathrm{end}}]$
$T \leftarrow \prod_{l=1}^{K} T_l$
Uniformly generate index set $I \subset [n]$ without replacement
$\mathbf{g}_{[t_0]}^{(0)} \leftarrow \frac{1}{B} \sum_{i \in I} \nabla f_{i_d}(\mathbf{x}_0)$
$\mathbf{x}_{[0]}^{(l)} \leftarrow \mathbf{x}_0, \quad 0 \le l \le K,$
for $t_1 = 0, \ldots, T_1 - 1$ **do**
  Uniformly generate index set $I \subset [n]$ without replacement, $|I| = B_1$
  $\mathbf{g}_{[t_1]}^{(1)} \leftarrow \frac{1}{B_1} \sum_{i \in I} \big[\nabla f_i(\mathbf{x}_{[t_1]}^{(1)}) - \nabla f_i(\mathbf{x}_{[0]}^{(0)})\big]$
  $\ldots$
  for $t_l = 0, \ldots, T_l - 1$ **do**
    Uniformly generate index set $I \subset [n]$ without replacement, $|I| = B_l$
    $\mathbf{g}_{[t_l]}^{(l)} \leftarrow \frac{1}{B_l} \sum_{i \in I} \big[\nabla f_i(\mathbf{x}_{[t_l]}^{(l)}) - \nabla f_i(\mathbf{x}_{[t_{l-1}]}^{(l-1)})\big]$
    $\ldots$
    for $t_K = 0, \ldots, T_K - 1$ **do**
      Uniformly generate index set $I \subset [n]$ without replacement, $|I| = B_K$
      $\mathbf{g}_{[t_K]}^{(K)} \leftarrow \frac{1}{B_K} \sum_{i \in I} \big[\nabla f_i(\mathbf{x}_{[t_K]}^{(K)}) - \nabla f_i(\mathbf{x}_{[t_{K-1}]}^{(K-1)})\big]$
      Denote $t = \sum_{j=1}^{K} t_j \prod_{l=j+1}^{K} T_l$, then let $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - 1/(10M) \cdot \sum_{l=0}^{K} \mathbf{g}_{[t_l]}^{(l)}$
      $\mathbf{x}_{[t_K+1]}^{(K)} \leftarrow \mathbf{x}_{t+1}$
    end for
    $\ldots$
    $\mathbf{x}_{[t_l+1]}^{(l)} \leftarrow \mathbf{x}_{[T_{l+1}]}^{(l+1)}$
  end for
  $\ldots$
  $\mathbf{x}_{[t_1+1]}^{(1)} \leftarrow \mathbf{x}_{[T_2]}^{(2)}$
end for
$\mathbf{x}_{\mathrm{out}} \leftarrow$ a uniformly random choice from $\{\mathbf{x}_0, ..., \mathbf{x}_{T-1}\}$
return $[\mathbf{x}_{\mathrm{out}}, \mathbf{x}_T]$

H.2. The Procedure of Neon2

Recall that we use Neon2 (Allen-Zhu and Li, 2018) in Algorithms 4 and 5 as a subroutine to find the negative curvature direction for escaping saddle points. For the self-containedness of this paper, we present the procedure of Neon2 in Algorithm 7. The key idea of NEON/NEON$^+$ (Xu et al., 2018b) and Neon2 (Allen-Zhu and Li, 2018) is to find a negative curvature direction around the stationary point $\mathbf{z}$ (or $\mathbf{x}_0$) using the update in Line 6 of Algorithm 7, which can be seen as an approximation of the power method. If no such direction is found by this procedure, Algorithm 7 returns $\mathbf{v} = \perp$. Xu et al. (2018b); Allen-Zhu and Li (2018) proved that with a constant probability this procedure will output a negative curvature direction that decreases the function value sufficiently (see Lemmas 33 and 35 for the details). The outer for loop in Algorithm 7 is used to boost the probability to

Algorithm 7 Neon2$(F, \mathbf{z}, L_1, L_2, \delta, \epsilon_H)$

Input: initial point $\mathbf{x}_0 = \mathbf{z}$, step size $\eta = \epsilon_H/(CL_1^2 \log(100d))$, noise variance $\sigma = \eta^2\epsilon_H^3/(L_2(100d)^{3C})$, $C > 0$ is a constant
for $j = 1, \ldots, \log(1/\delta)$ **do**
  $\xi \leftarrow$ random Gaussian vector
  $\mathbf{x}_1 = \mathbf{x}_0 + \xi$
  for $t = 1, \ldots, T$ **do**
    $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta(\nabla f_i(\mathbf{x}_t) - \nabla f_i(\mathbf{x}_0))$, where i is randomly drawn from $[n]$
    if $\|\mathbf{x}_{t+1} - \mathbf{x}_0\| \geq (100d)^C\sigma$ **then**
      $\widehat{\mathbf{v}}_j = (\mathbf{x}_{t+1} - \mathbf{x}_0)/\|\mathbf{x}_{t+1} - \mathbf{x}_0\|_2$
      break
    else if $t = T$ **then**
      $\widehat{\mathbf{v}}_j = \perp$
    end if
  end for
  if $\widehat{\mathbf{v}}_j \neq \perp$ **then**
    $m = CL_1^2\log(1/\delta)/\epsilon_H^2$, $\mathbf{v}' = C\epsilon_H\mathbf{v}/L_2$
    Draw $i_1, \ldots, i_m$ from $[n]$
    $z_j = 1/(m\|\mathbf{v}'\|_2^2)\sum_{l=1}^m\langle\mathbf{v}', \nabla f_{i_l}(\mathbf{z}+\mathbf{v}') - \nabla f_{i_l}(\mathbf{z})\rangle$
    if $z_j \leq -3\epsilon_H/4$ **then**
      return $\mathbf{v} = \widehat{\mathbf{v}}_j$
    end if
  end if
end for
return $\mathbf{v} = \perp$

$1-\delta$ for any $\delta \in (0,1)$. Note that the step size η depends on Hessian smoothness parameter L_2. To estimate L_2, we could use line search technique (Nesterov and Polyak, 2006) or calculate the difference between two Taylor expansions (Weiser et al., 2007).

References

Naman Agarwal, Zeyuan Allen-Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. Finding approximate local minima faster than gradient descent. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1195–1199. ACM, 2017.

Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1200–1205. ACM, 2017.

Zeyuan Allen-Zhu. Natasha 2: Faster non-convex optimization than sgd. In *Advances in Neural Information Processing Systems*, pages 2676–2687, 2018a.

Zeyuan Allen-Zhu. Katyusha x: Simple momentum method for stochastic sum-of-nonconvex optimization. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 179–185. PMLR, 2018b.

Zeyuan Allen-Zhu and Elad Hazan. Variance reduction for faster non-convex optimization. In *International Conference on Machine Learning*, pages 699–707, 2016.

Zeyuan Allen-Zhu and Yuanzhi Li. Neon2: Finding local minima via first-order oracles. In *Advances in Neural Information Processing Systems*, pages 3720–3730, 2018.

Animashree Anandkumar and Rong Ge. Efficient approaches for escaping higher order saddle points in non-convex optimization. In *Conference on Learning Theory*, pages 81–102, 2016.

Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Global optimality of local search for low rank matrix recovery. In *Advances in Neural Information Processing Systems*, pages 3873–3881, 2016.

Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. "Convex until proven guilty": Dimension-free acceleration of gradient descent on non-convex functions. In *International Conference on Machine Learning*, pages 654–663, 2017.

Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. Accelerated methods for nonconvex optimization. *SIAM Journal on Optimization*, 28(2):1751–1772, 2018.

Xi Chen, Simon S Du, and Xin T Tong. On stationary-point hitting time and ergodicity of stochastic gradient langevin dynamics. *Journal of Machine Learning Research*, 21(68): 1–41, 2020.

Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

Frank E Curtis, Daniel P Robinson, and Mohammadreza Samadi. A trust region algorithm with a worst-case iteration complexity of $O(\epsilon^{-3/2})$ for nonconvex optimization. *Mathematical Programming*, 162(1-2):1–32, 2017.

Hadi Daneshmand, Jonas Kohler, Aurelien Lucchi, and Thomas Hofmann. Escaping saddles with stochastic gradients. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1155–1164. PMLR, 2018.

Yann Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Mathematics*, 111(6 Pt 1):2475–2485, 2014.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 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, 2014b.

Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal nonconvex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems*, pages 686–696, 2018.

Cong Fang, Zhouchen Lin, and Tong Zhang. Sharp analysis for nonconvex sgd escaping from saddle points. In *Conference on Learning Theory*, pages 1192–1234, 2019.

Dan Garber and Elad Hazan. Fast and simple pca via convex optimization. *arXiv preprint arXiv:1509.05647*, 2015.

Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle pointsonline stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, pages 797–842, 2015.

Rong Ge, Jason D Lee, and Tengyu Ma. Matrix completion has no spurious local minimum. In *Advances in Neural Information Processing Systems*, pages 2973–2981, 2016.

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 and Guanghui Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1-2):59–99, 2016.

Saeed Ghadimi, Guanghui Lan, and Hongchao Zhang. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155(1-2):267–305, 2016.

Reza Harikandeh, Mohamed Osama Ahmed, Alim Virani, Mark Schmidt, Jakub Konečnỳ, and Scott Sallinen. Stopwasting my gradients: Practical svrg. In *Advances in Neural Information Processing Systems*, pages 2251–2259, 2015.

Christopher J Hillar and Lek-Heng Lim. Most tensor problems are np-hard. *Journal of the ACM (JACM)*, 60(6):45, 2013.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *International Conference on Machine Learning*, pages 1724–1732, 2017.

Chi Jin, Praneeth Netrapalli, and Michael I. Jordan. Accelerated gradient descent escapes saddle points faster than gradient descent. In *Proceedings of the 31st Conference On Learning Theory*, volume 75, pages 1042–1085. PMLR, 2018.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Jonas Moritz Kohler and Aurelien Lucchi. Sub-sampled cubic regularization for non-convex optimization. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1895–1904. PMLR, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *Conference on learning theory*, pages 1246–1257, 2016.

Jason D Lee, Ioannis Panageas, Georgios Piliouras, Max Simchowitz, Michael I Jordan, and Benjamin Recht. First-order methods almost always avoid strict saddle points. *Mathematical programming*, 176(1-2):311–337, 2019.

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

Kfir Y Levy. The power of normalization: Faster evasion of saddle points. *arXiv preprint arXiv:1611.04831*, 2016.

Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, pages 3384–3392, 2015.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Yurii Nesterov and Boris T Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011.

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

Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Stochastic recursive gradient algorithm for nonconvex optimization. *arXiv preprint arXiv:1705.07261*, 2017b.

Lam M Nguyen, Marten van Dijk, Dzung T Phan, Phuong Ha Nguyen, Tsui-Wei Weng, and Jayant R Kalagnanam. Finite-sum smooth optimization with sarah. *arXiv preprint arXiv:1901.07648*, 2019.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

Boris Teodorovich Polyak. Gradient methods for minimizing functionals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 3(4):643–653, 1963.

Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

Maxim Raginsky, Alexander Rakhlin, and Matus Telgarsky. Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis. In *Conference on Learning Theory*, pages 1674–1703, 2017.

Sashank Reddi, Manzil Zaheer, Suvrit Sra, Barnabas Poczos, Francis Bach, Ruslan Salakhutdinov, and Alex Smola. A generic approach for escaping saddle points. In *International Conference on Artificial Intelligence and Statistics*, pages 1233–1242, 2018.

Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International Conference on Machine Learning*, pages 314–323, 2016a.

Sashank J Reddi, Suvrit Sra, Barnabás Póczos, and Alex Smola. Fast incremental method for smooth nonconvex optimization. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 1971–1977. IEEE, 2016b.

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

Nicolas L Roux, Mark Schmidt, and Francis R Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in Neural Information Processing Systems*, pages 2663–2671, 2012.

Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2002.

Shai Shalev-Shwartz. Sdca without duality, regularization, and individual convexity. In *International Conference on Machine Learning*, pages 747–754, 2016.

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.

Quoc Tran-Dinh, Nhan H Pham, Dzung T Phan, and Lam M Nguyen. A hybrid stochastic optimization framework for stochastic composite nonconvex optimization. *arXiv preprint arXiv:1907.03793*, 2019.

Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

Zhe Wang, Kaiyi Ji, Yi Zhou, Yingbin Liang, and Vahid Tarokh. Spiderboost and momentum: Faster variance reduction algorithms. In *Advances in Neural Information Processing Systems*, pages 2403–2413, 2019.

Martin Weiser, Peter Deuflhard, and Bodo Erdmann. Affine conjugate adaptive newton methods for nonlinear elastomechanics. *Optimisation Methods and Software*, 22(3):413–431, 2007.

Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Pan Xu, Jinghui Chen, Difan Zou, and Quanquan Gu. Global convergence of langevin dynamics based algorithms for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 3122–3133, 2018a.

Peng Xu, Fred Roosta, and Michael W Mahoney. Newton-type methods for non-convex optimization under inexact hessian information. *Mathematical Programming*, pages 1–36, 2019.

Yi Xu, Jing Rong, and Tianbao Yang. First-order stochastic algorithms for escaping from saddle points in almost linear time. In *Advances in Neural Information Processing Systems*, pages 5531–5541, 2018b.

Yaodong Yu, Difan Zou, and Quanquan Gu. Saving gradient and negative curvature computations: Finding local minima more efficiently. *arXiv preprint arXiv:1712.03950*, 2017.

Yaodong Yu, Pan Xu, and Quanquan Gu. Third-order smoothness helps: Faster stochastic optimization algorithms for finding local minima. In *Advances in Neural Information Processing Systems*, pages 4526–4536, 2018.

Xiao Zhang, Lingxiao Wang, Yaodong Yu, and Quanquan Gu. A primal-dual analysis of global optimality in nonconvex low-rank matrix recovery. In *International conference on machine learning*, 2018.

Yuchen Zhang, Percy Liang, and Moses Charikar. A hitting time analysis of stochastic gradient langevin dynamics. In *Conference on Learning Theory*, pages 1980–2022, 2017.

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

Dongruo Zhou, Pan Xu, and Quanquan Gu. Stochastic variance-reduced cubic regularized Newton methods. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5990–5999. PMLR, 2018a.

Dongruo Zhou, Pan Xu, and Quanquan Gu. Stochastic nested variance reduced gradient descent for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 3922–3933, 2018b.