SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

November 2024, Vol. 67, Iss. 11, 212101:1–212101:15
https://doi.org/10.1007/s11432-022-3892-8

Stochastic normalized gradient descent with momentum for large-batch training

Shen-Yi ZHAO†, Chang-Wei SHI†, Yin-Peng XIE & Wu-Jun LI*

National Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210023, *China*

Received 27 June 2022/Revised 3 November 2022/Accepted 13 April 2023/Published online 23 October 2024

Abstract Stochastic gradient descent (SGD) and its variants have been the dominating optimization methods in machine learning. Compared with SGD with small-batch training, SGD with large-batch training can better utilize the computational power of current multi-core systems such as graphics processing units (GPUs) and can reduce the number of communication rounds in distributed training settings. Thus, SGD with large-batch training has attracted considerable attention. However, existing empirical results showed that large-batch training typically leads to a drop in generalization accuracy. Hence, how to guarantee the generalization ability in large-batch training becomes a challenging task. In this paper, we propose a simple yet effective method, called stochastic normalized gradient descent with momentum (SNGM), for large-batch training. We prove that with the same number of gradient computations, SNGM can adopt a larger batch size than momentum SGD (MSGD), which is one of the most widely used variants of SGD, to converge to an ϵ-stationary point. Empirical results on deep learning verify that when adopting the same large batch size, SNGM can achieve better test accuracy than MSGD and other state-of-the-art large-batch training methods.

Keywords non-convex problems, large-batch training, stochastic normalized gradient descent, momentum

1 Introduction

In machine learning, we often need to solve the following empirical risk minimization problem:

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

where $\boldsymbol{w} \in \mathbb{R}^d$ denotes the model parameter, n denotes the number of training samples, and $f_i(\boldsymbol{w})$ denotes the loss on the i-th training sample. The problem in (1) can be used to formulate a broad family of machine learning models, such as logistic regression and deep learning models.

Stochastic gradient descent (SGD) [1, 2] and its variants [3, 4] have been the dominating optimization methods for solving (1). SGD and its variants are iterative methods. In the t-th iteration, these methods randomly choose a subset (also called a mini-batch) $\mathcal{I}_t \subset \{1, 2, \ldots, n\}$ and compute the stochastic mini-batch gradient $\sum_{i\in\mathcal{I}_t} \nabla f_i(\boldsymbol{w}_t)/B$ for updating the model parameter, where $B = |\mathcal{I}_t|$ is the batch size. Existing studies [5, 6] have proven that with a batch size of B, SGD and its momentum variant, called momentum SGD (MSGD), achieve a $\mathcal{O}(1/\sqrt{TB})$ convergence rate for smooth non-convex problems, where T is the total number of model parameter updates.

With the population of multi-core systems and the easy implementation of data parallelism, many distributed variants of SGD have been proposed, including parallel SGD [7], decentralized SGD [8, 9], local SGD [10, 11], and local momentum SGD [6, 12]. Theoretical results show that all these methods can achieve a $\mathcal{O}(1/\sqrt{TKb})$ convergence rate for smooth non-convex problems. Here, b is the batch size of each worker, and K is the number of workers. By setting $Kb = B$, we can find that the convergence rate of these distributed methods is consistent with that of sequential (non-distributed) methods. In distributed

* Corresponding author (email: liwujun@nju.edu.cn)
† Zhao S-Y and Shi C-W have the same contribution to this work.

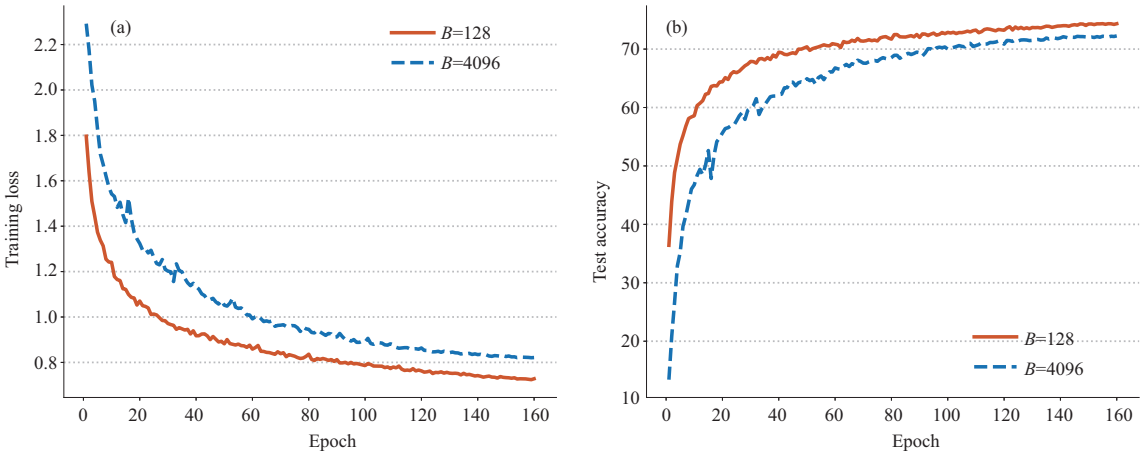

Figure 1 (Color online) Training loss and test accuracy of training a non-convex model (a network with two convolutional layers) on the CIFAR-10 dataset. (a) Training loss; (b) test accuracy.

settings, a smaller number of model parameter updates T imply lower synchronization and communication costs. Hence, a small T can further speed up the training process. Based on the $\mathcal{O}(1/\sqrt{TKb})$ convergence rate, we can find that if we adopt a larger b, T will be smaller. Hence, large-batch training can reduce the number of communication rounds in distributed training. Another benefit of adopting large-batch training is better utilizing the computational power of current multi-core systems like graphics processing units (GPUs) [13]. Hence, large-batch training has recently attracted considerable attention in machine learning.

Unfortunately, empirical results [14,15] showed that existing SGD methods with a large batch size will lead to a drop in the generalization accuracy of deep learning models. Figure 1 shows a comparison of training loss and test accuracy between MSGD with a small batch size and MSGD with a large batch size. We can find that large-batch training indeed degrades the test accuracy and increases the training loss. Several researchers attempted to explain this phenomenon [15,16]. For example, researchers in [15] observed that SGD with a small batch size typically makes the model parameter converge to a flattened minimum, while SGD with a large batch size typically makes the model parameter fall into the region of a sharp minimum. Further, a flattened minimum can generally achieve better generalization ability than a sharp minimum [15]. Hence, how to guarantee the generalization ability in large-batch training has become a challenging task [13,15–17].

Many methods have been proposed for improving the performance of SGD with large batch sizes. Refs. [18,19] proposed several tricks, such as warm-up and learning rate scaling schemes, to bridge the generalization gap under large-batch training settings. Researchers in [16] argued that SGD with a large batch size needs to increase the number of iterations. Further, authors in [13] observed that gradients at different layers of deep neural networks vary widely in the norm and proposed the layer-wise adaptive rate scaling (LARS) method. A similar method that updates the model parameter in a layer-wise way was proposed in [20]. The work in [21] proposed CLARS as a variant of LARS. Some studies [22,23] directly adopted the layer-wise adaptive rate scaling strategy of LARS to enhance the model's generalization ability in large-batch training scenarios. Apart from these empirical findings, there have been some theoretical studies on large-batch training. For example, the convergence analyses of LARS have been reported in [24]. The work in [25] analyzed the inconsistency bias in decentralized momentum SGD and proposed DecentLaM for decentralized large-batch training. Furthermore, researchers in [17] argued that the extrapolation technique is suitable for large-batch training and proposed EXTRAP-SGD. However, experimental implementations of these methods still require additional training tricks, such as warm-up, which may make the results inconsistent with the theory. If we avoid these tricks, these methods may suffer from severe performance degradation. For LARS and its variants, the proposal of the layer-wise update strategy is primarily based on empirical observations. Its reasonability and necessity remain doubtful from an optimization perspective.

In this paper, we first review the convergence property of MSGD, one of the most widely used variants of SGD, and analyze the failure of MSGD in large-batch training from an optimization perspective. Then, we propose a novel method, called stochastic normalized gradient descent with momentum (SNGM), for

large-batch training. The main contributions of this paper are outlined as follows.

• SNGM is as simple as MSGD. The only difference from MSGD is that SNGM adopts a normalized gradient to update the momentum.

• We explore the relationship between the smoothness constant L and the batch size in MSGD and find that given a fixed number of gradient computations $\mathcal{C}$, the optimal batch size has an upper bound $\mathcal{O}(\sqrt{\mathcal{C}}/L)$ for MSGD.

• The batch size of SNGM is not constrained by the smoothness constant. With the same number of gradient computations, SNGM can adopt a larger batch size than MSGD to converge to an ϵ-stationary point.

• Compared with LARS, SNGM does not adopt the layer-wise update strategy. And our theory shows that layer-wise updates slow down the convergence rate.

• Empirical results on deep learning show that with the same large batch size, SNGM can achieve better test accuracy than MSGD and other state-of-the-art large-batch training methods.

2 Preliminaries

In this paper, let $\|\cdot\|$ denote the Euclidean norm, and $\boldsymbol{w}^*$ denote one of the optimal solutions of (1), i.e., $\boldsymbol{w}^* \in \arg\min_{\boldsymbol{w}} F(\boldsymbol{w})$. We call $\boldsymbol{w}$ an ϵ-stationary point of $F(\cdot)$ if $\|\nabla F(\boldsymbol{w})\| \leqslant \epsilon$. The computation complexity of an algorithm is the total number of its gradient computations. Furthermore, we provide the following assumptions and definitions.

Assumption 1 (σ-bounded variance)**.** Let $\sigma > 0$. For any $\boldsymbol{w}$, $\mathbb{E}\|\nabla f_i(\boldsymbol{w}) - \nabla F(\boldsymbol{w})\|^2 \leqslant \sigma^2$.

The bounded variance assumption is widely used in the analysis of stochastic optimization for non-convex problems [5, 6, 10, 11].

Definition 1 (Smoothness)**.** A function $h(\cdot)$ is L-smooth ($L > 0$) if for any $\boldsymbol{u}, \boldsymbol{w}$, $h(\boldsymbol{u}) \leqslant h(\boldsymbol{w}) + \nabla h(\boldsymbol{w})^{\mathrm{T}}(\boldsymbol{u} - \boldsymbol{w}) + \frac{L}{2}\|\boldsymbol{u} - \boldsymbol{w}\|^2$. L is called smoothness constant in this paper.

Definition 2 (Relaxed smoothness [26])**.** A function $h(\cdot)$ is (L, λ)-smooth ($L \geqslant 0$, $\lambda \geqslant 0$) if $h(\cdot)$ is twice differentiable and for any $\boldsymbol{w}$, $\|H_h(\boldsymbol{w})\| \leqslant L + \lambda\|\nabla h(\boldsymbol{w})\|$, where $H_h(\boldsymbol{w})$ denotes the Hessian matrix of $h(\boldsymbol{w})$.

From the above definitions, we can find that if a function $h(\cdot)$ is $(L, 0)$-smooth, then it is a classical L-smooth function [27]. We have the following property of relaxed smoothness.

Lemma 1. If $h(\cdot)$ is (L, λ)-smooth, then for any $\boldsymbol{u}, \boldsymbol{w}, \alpha$ such that $\|\boldsymbol{u} - \boldsymbol{w}\| \leqslant \alpha$, we have

$$\|\nabla h(\boldsymbol{u})\| \leqslant (L\alpha + \|\nabla h(\boldsymbol{w})\|)\mathrm{e}^{\lambda\alpha}.$$

The proof of Lemma 1 is given in Appendix A.

3 Relationship between smoothness constant and batch size in MSGD

In this section, we analyze the convergence property of MSGD to determine the relationship between smoothness constant and batch size, which provides insightful hints for designing our new method.

MSGD can be written as follows:

$$\boldsymbol{v}_{t+1} = \beta\boldsymbol{v}_t + \boldsymbol{g}_t, \tag{2}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\boldsymbol{v}_{t+1}, \tag{3}$$

where $\boldsymbol{g}_t = \sum_{i\in\mathcal{I}_t} \nabla f_i(\boldsymbol{w}_t)/B$ is a stochastic mini-batch gradient with the batch size being B, and $\boldsymbol{v}_{t+1}$ is the Polyak's momentum [28].

We aim to determine how large the batch size can be without performance loss. The convergence rate of MSGD with a batch size B for L-smooth functions can be derived from the work in [6]. When $\eta \leqslant (1-\beta)^2/((1+\beta)L)$, we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\boldsymbol{w}_t)\|^2 \leqslant \frac{2(1-\beta)[F(\boldsymbol{w}_0) - F(\boldsymbol{w}^*)]}{\eta T} + \frac{L\eta\sigma^2}{(1-\beta)^2 B} + \frac{4L^2\eta^2\sigma^2}{(1-\beta)^2}$$

$$= \mathcal{O}\left(\frac{B}{\eta\mathcal{C}}\right) + \mathcal{O}\left(\frac{\eta}{B}\right) + \mathcal{O}(\eta^2), \tag{4}$$

where $\mathcal{C} = TB$ denotes the total number of gradient computations. According to Corollary 1 in [6], we set $\eta = \sqrt{B}/\sqrt{T} = B/\sqrt{\mathcal{C}}$ and obtain that

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\boldsymbol{w}_t)\| \leqslant \sqrt{\mathcal{O}\left(\frac{1}{\sqrt{\mathcal{C}}}\right) + \mathcal{O}\left(\frac{B^2}{\mathcal{C}}\right)}. \tag{5}$$

Since the condition $\eta \leqslant (1-\beta)^2/((1+\beta)L)$ is required for (4) in [6], we first obtain that $B \leqslant \mathcal{O}(\sqrt{\mathcal{C}}/L)$. Furthermore, according to the right term of (5), to achieve an ϵ-stationary point, we have to set $\mathcal{C} \geqslant \mathcal{O}(1/\epsilon^4)$ and $B \leqslant \mathcal{O}(\sqrt{\mathcal{C}}\epsilon)$. Hence in MSGD, we have to set the batch size satisfying

$$B \leqslant \mathcal{O}\left(\min\left\{\frac{\sqrt{\mathcal{C}}}{L}, \mathcal{C}^{1/4}\right\}\right). \tag{6}$$

We can find that a larger L leads to a smaller batch size in MSGD. If B does not satisfy (6), MSGD will get higher computation complexity.

In fact, to the best of our knowledge, we can observe three conditions on which all existing convergence analyses [5–8, 10] of SGD and its variants rely for guaranteeing the $\mathcal{O}(1/\epsilon^4)$ computation complexity for both convex and non-convex problems:

- The objective function is L-smooth;
- The learning rate η is less than $\mathcal{O}(1/L)$;
- The batch size B is proportional to the learning rate η.

A direct corollary is that the batch size is constrained by the smoothness constant L, i.e., $B \leqslant \mathcal{O}(1/L)$. Hence, we cannot increase the batch size casually in these SGD-based methods. Otherwise, it may slow down the convergence rate, and we need to compute more gradients, which is consistent with the results obtained in [16].

We also observe that although EXTRAP-SGD [17] adopts the extrapolation technique, which simulates the variance of a small-batch gradient in large-batch training, its learning rate is constrained by the smoothness constant. In particular, the average square of the gradient norm has an upper bound $\mathcal{O}(B/(\eta\mathcal{C})) + \mathcal{O}((\eta+\eta^2)/B)$ with $\eta \leqslant \mathcal{O}(1/L)$. Hence, the batch size of EXTRAP-SGD remains constrained by the smoothness constant.

4 Stochastic normalized gradient descent with momentum

In this section, we introduce our novel method, called SNGM, and present it in Algorithm 1. In the t-th iteration, SNGM runs the following update:

$$\boldsymbol{u}_{t+1} = \beta\boldsymbol{u}_t + \frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|}, \tag{7}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\boldsymbol{u}_{t+1}, \tag{8}$$

where $\boldsymbol{g}_t = \sum_{i\in\mathcal{I}_t}\nabla f_i(\boldsymbol{w}_t)/B$ is a stochastic mini-batch gradient with the batch size being B. If we let $\boldsymbol{w}_{-1} = \boldsymbol{w}_0$, since $-\eta\boldsymbol{u}_t = \boldsymbol{w}_t - \boldsymbol{w}_{t-1}$ holds for $t = 0, 1, \ldots, T$, SNGM can also be written as follows:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\boldsymbol{u}_{t+1} = \boldsymbol{w}_t - \eta\left(\beta\boldsymbol{u}_t + \frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|}\right) = \boldsymbol{w}_t - \eta\frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|} + \beta(\boldsymbol{w}_t - \boldsymbol{w}_{t-1}). \tag{9}$$

When $\beta = 0$, SNGM will degenerate to stochastic normalized gradient descent (SNGD) [4, 29]. In the following content, we will compare SNGM with MSGD and LARS [24], the two most related work in the literature on large-batch training.

4.1 Comparison with MSGD

$\boldsymbol{u}_t$ is a variant of Polyak's momentum. However, different from Polyak's MSGD which adopts $\boldsymbol{g}_t$ directly for updating $\boldsymbol{u}_{t+1}$, SNGM adopts the normalized gradient $\boldsymbol{g}_t/\|\boldsymbol{g}_t\|$ for updating $\boldsymbol{u}_{t+1}$. We have the following lemma about $\boldsymbol{u}_t$.

Algorithm 1 SNGM

Initialization: $\boldsymbol{u}_0 = \boldsymbol{0}, \boldsymbol{w}_0$, $\eta > 0, \beta \in [0, 1), B > 0, T > 0$;
for $t = 0, 1, \ldots, T-1$ **do**
  Randomly choose B samples, denoted by $\mathcal{I}_t$;
  $\boldsymbol{g}_t = \sum_{i \in \mathcal{I}_t} \nabla f_i(\boldsymbol{w}_t)/B$;
  $\boldsymbol{u}_{t+1} = \beta \boldsymbol{u}_t + \boldsymbol{g}_t / \|\boldsymbol{g}_t\|$;
  $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{u}_{t+1}$;
end for

Lemma 2. Let $\{\boldsymbol{u}_t\}$ be the sequence produced by (7). We have $\forall t \geqslant 0$, $\|\boldsymbol{u}_t\| \leqslant 1/(1-\beta)$.

The proof of Lemma 2 is given in Appendix B.

We can observe that the momentum in SNGM is naturally bounded whether $\boldsymbol{g}_t$ is large or small, while MSGD often needs the bounded gradient assumption [30]. In Subsection 5.1, we will provide a theoretical comparison of the convergence and feasible batch size between MSGD and SNGM.

4.2 Comparison with LARS

LARS also adopts normalized gradient for large-batch training. Following the analysis in [24], we set $\beta = 0$[1)]. In particular, let $\boldsymbol{g}_t = (\boldsymbol{g}_t^{(1)}, \boldsymbol{g}_t^{(2)}, \ldots, \boldsymbol{g}_t^{(S)})$, in which $\boldsymbol{g}_t^{(s)}$ is the gradient of the s-th block in $\boldsymbol{g}_t$. LARS updates the parameter as follows:

$$\boldsymbol{w}_{t+1}^{(s)} = \boldsymbol{w}_t^{(s)} - \frac{\eta \phi(\|\boldsymbol{w}_t^{(s)}\|)}{\|\boldsymbol{g}_t^{(s)}\|} \boldsymbol{g}_t^s,$$

where $\phi(\cdot) : \mathbb{R} \to \mathbb{R}$ is a scale function. Both $\phi(z) = z$ and $\phi(z) = \min\{\max\{z, \gamma_l\}, \gamma_u\}$ ($\gamma_u \geqslant \gamma_l > 0$) are recommended in [24]. When $\phi(z) = z$, $\phi(\|\boldsymbol{w}_t^{(s)}\|)/\|\boldsymbol{g}_t^{(s)}\|$ can be treated as an estimation for $1/L$, where L is the smoothness constant of $F(\boldsymbol{w})$ [24].

First, we find that in some cases, $\phi(z) = z$ may destroy the convergence. In particular, we consider the following minimization problem:

$$\min_x F(x) = 0.5(x+1)^2, \tag{10}$$

in which the optimal solution is $x^* = -1$. In this one-dimension problem, the gradient is $g = x + 1$, and $g/\|g\| = \mathrm{sgn}(g)$, where $\mathrm{sgn}(\cdot)$ is the signum function. When LARS is used for solving (10), the iteration can be written as follows:

$$x \leftarrow x - \eta |x| \mathrm{sgn}(x+1).$$

When the initialization $x_0 > 0$, it is easy to verify that with a small learning rate $\eta < 1$, the above update can be written as $x \leftarrow (1-\eta)x$. Hence, LARS will make x converge to 0, rather than the optimal solution -1.

When $\phi(z)$ satisfies that $0 < \gamma_l \leqslant \phi(z) \leqslant \gamma_u < \infty$ for all z (e.g., $\phi(z) = \min\{\max\{z, \gamma_l\}, \gamma_u\}$), LARS can be treated as a block-wise variant of SNGM with a finely-tuned learning rate, i.e.,

$$\boldsymbol{w}_{t+1}^{(s)} = \boldsymbol{w}_t^{(s)} - \frac{\eta_t}{\|\boldsymbol{g}_t^{(s)}\|} \boldsymbol{g}_t^s,$$

where η_t satisfies $\eta\gamma_l \leqslant \eta_t \leqslant \eta\gamma_u$. Although it has been observed that the gradient norms in each block are quite different [24], we will show that such a block-wise update strategy will slightly slow down the convergence in Section 6.

5 Convergence analysis

In this section, we prove the convergence rate of SNGM for both smooth and relaxed smooth objective functions. First, we introduce the auxiliary variable as follows.

Lemma 3. Let $\boldsymbol{z}_t = \frac{1}{1-\beta}\boldsymbol{w}_t - \frac{\beta}{1-\beta}\boldsymbol{w}_{t-1}$; then we have $\boldsymbol{z}_{t+1} = \boldsymbol{z}_t - \frac{\eta}{1-\beta}\frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|}$.

The proof of Lemma 3 is given in Appendix C.

1) We find that there are two different versions of LARS. The first one [13] normalizes gradient and the other one [24] normalizes momentum. When $\beta = 0$ and the weight decay is zero, the two versions are the same.

Table 1 Comparison between MSGD and SNGM for an L-smooth objective function[a)]

	Learning rate	Batch size	Upper bound of $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\boldsymbol{w}_t)\|$
MSGD	$\frac{B}{\sqrt{\mathcal{C}}}$	$\min\{\frac{\sqrt{\mathcal{C}}}{L}, \mathcal{C}^{1/4}\}$	$\sqrt{\mathcal{O}(\frac{1}{\sqrt{\mathcal{C}}})+\mathcal{O}(\frac{B^2}{\mathcal{C}})}$
SNGM	$\frac{\sqrt{B}}{\sqrt{\mathcal{C}}}$	$\sqrt{\mathcal{C}}$	$\mathcal{O}(\frac{1}{\mathcal{C}^{1/4}})$

a) $\mathcal{C}$ denotes the computation complexity (total number of gradient computations).

5.1 Smooth objective function

For a smooth objective function, we have the following convergence result of SNGM.

Theorem 1. Let $F(\boldsymbol{w})$ be an L-smooth function ($L > 0$). The sequence $\{\boldsymbol{w}_t\}$ is produced by Algorithm 1. Then for any $\eta > 0, B > 0$, we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\boldsymbol{w}_t)\| \leqslant \frac{(1-\beta)[F(\boldsymbol{w}_0)-F(\boldsymbol{w}^*)]}{\eta T}+\frac{L}{2}\kappa\eta+\frac{2\sigma}{\sqrt{B}}, \tag{11}$$

where $\kappa = \frac{1+\beta}{(1-\beta)^2}$.

The proof of Theorem 1 is given in Appendix D.

We can observe that different from (4), which needs $\eta \leqslant \mathcal{O}(1/L)$, Eq. (11) is true for any positive learning rate. According to Theorem 1, we obtain the following computation complexity of SNGM.

Corollary 1. Let $F(\boldsymbol{w})$ be an L-smooth function ($L > 0$). The sequence $\{\boldsymbol{w}_t\}$ is produced by Algorithm 1. Given any total number of gradient computations $\mathcal{C} > 0$, let $T = \lceil \mathcal{C}/B \rceil$, $B = \sqrt{\mathcal{C}}$, and $\eta = \sqrt{B/\mathcal{C}}$. Then we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\boldsymbol{w}_t)\| \leqslant \frac{(1-\beta)[F(\boldsymbol{w}_0)-F(\boldsymbol{w}^*)]}{\mathcal{C}^{1/4}}+\frac{L(1+\beta)}{2(1-\beta)^2\mathcal{C}^{1/4}}+\frac{2\sigma}{\mathcal{C}^{1/4}}.$$

Hence, the computation complexity for achieving an ϵ-stationary point is $\mathcal{O}(1/\epsilon^4)$.

Remark 1. We make a comparison between MSGD and SNGM in Table 1. We can observe that the batch size of SNGM can be set as $\sqrt{\mathcal{C}}$, which does not rely on the smoothness constant L, and the $\mathcal{O}(1/\epsilon^4)$ computation complexity is still guaranteed. Nevertheless, the batch size of MSGD cannot be larger than $\mathcal{O}(\sqrt{\mathcal{C}}/L)$ to achieve the computation complexity of $\mathcal{O}(1/\epsilon^4)$. Hence, SNGM can adopt a larger batch size than MSGD, particularly when L is large.

5.2 Relaxed smooth objective function

Recently, the authors in [26] observed the relaxed smooth property in deep neural networks. According to Definition 2, the relaxed smooth property is more general than the L-smooth property. For a relaxed smooth objective function, we have the following convergence result of SNGM.

Theorem 2. Let $F(\boldsymbol{w})$ be an (L,λ)-smooth function ($L \geqslant 0, \lambda \geqslant 0$). The sequence $\{\boldsymbol{w}_t\}$ is produced by Algorithm 1 with the learning rate η and batch size B. Then we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\boldsymbol{w}_t)\| \leqslant \frac{2(1-\beta)[F(\boldsymbol{w}_0)-F(\boldsymbol{w}^*)]}{\eta T}+8L\kappa\eta+\frac{4\sigma}{\sqrt{B}}, \tag{12}$$

where $\kappa = \frac{1+\beta}{(1-\beta)^2}$ and $8\eta\kappa\lambda \leqslant 1$.

The proof of Theorem 2 is given in Appendix E.

According to Theorem 2, we obtain the computation complexity of SNGM.

Corollary 2. Let $F(\boldsymbol{w})$ be an (L,λ)-smooth function ($L \geqslant 0, \lambda \geqslant 0$). The sequence $\{\boldsymbol{w}_t\}$ is produced by Algorithm 1. Given any total number of gradient computations $\mathcal{C} > 0$, let $T = \lceil \mathcal{C}/B \rceil$, $B = \sqrt{\mathcal{C}}$, $\eta = \sqrt[4]{1/\mathcal{C}}$, and $8\eta\kappa\lambda \leqslant 1$. Then we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\boldsymbol{w}_t)\| \leqslant \frac{2(1-\beta)[F(\boldsymbol{w}_0)-F(\boldsymbol{w}^*)]}{\mathcal{C}^{1/4}}+\frac{8L(1+\beta)}{(1-\beta)^2\mathcal{C}^{1/4}}+\frac{4\sigma}{\mathcal{C}^{1/4}}.$$

Hence, the computation complexity for achieving an ϵ-stationary point is $\mathcal{O}(1/\epsilon^4)$.

Remark 2. Theorem 2 and Corollary 2 extend the convergence analyses in Theorem 1 and Corollary 1 for a smooth objective function to a relaxed smooth objective function, which is a more general scenario. For a relaxed smooth objective function, SNGM with batch size $B = \sqrt{\mathcal{C}}$ can still guarantee a $\mathcal{O}(1/\epsilon^4)$ computation complexity.

6 Block-wise update slows down convergence rate

LARS [13, 24] adopts a block-wise strategy for updating, which is different from that in our SNGM. In this section, we prove that block-wise updates will actually slow down the convergence rate.

We start from the block-wise variant of SNGM. Let

$$\boldsymbol{w}_t = (\boldsymbol{w}_t^{(1)}, \boldsymbol{w}_t^{(2)}, \ldots, \boldsymbol{w}_t^{(S)}),$$
$$\boldsymbol{g}_t = (\boldsymbol{g}_t^{(1)}, \boldsymbol{g}_t^{(2)}, \ldots, \boldsymbol{g}_t^{(S)}).$$

The block-wise variant of SNGM can be written as follows:

$$\boldsymbol{u}_{t+1}^{(s)} = \beta \boldsymbol{u}_t^{(s)} + \frac{\boldsymbol{g}_t^{(s)}}{\|\boldsymbol{g}_t^{(s)}\|}, \tag{13}$$
$$\boldsymbol{w}_{t+1}^{(s)} = \boldsymbol{w}_t^{(s)} - \eta \boldsymbol{u}_{t+1}^{(s)}, \tag{14}$$

where $\boldsymbol{u}_t^{(s)}$, $\boldsymbol{w}_t^{(s)}$, and $\boldsymbol{g}_t^{(s)} \in \mathbb{R}^{d_s}$ denote the s-th block of $\boldsymbol{u}_t, \boldsymbol{w}_t$, and $\boldsymbol{g}_t$, respectively, $\sum_{s=1}^S d_s = d$. Let $\boldsymbol{z}_t = \frac{1}{1-\beta}\boldsymbol{w}_t - \frac{\beta}{1-\beta}\boldsymbol{w}_{t-1}$ and $\boldsymbol{z}_t = (\boldsymbol{z}_t^{(1)}, \boldsymbol{z}_t^{(2)}, \ldots, \boldsymbol{z}_t^{(S)})$. Lemma 3 can be easily rewritten as its block-wise form:

$$\boldsymbol{z}_{t+1}^{(s)} = \boldsymbol{z}_t^{(s)} - \frac{\eta}{1-\beta}\frac{\boldsymbol{g}_t^{(s)}}{\|\boldsymbol{g}_t^{(s)}\|}, \tag{15}$$

where $s = 1, 2, \ldots, S$. We have the following convergence result about block-wise SNGM.

Theorem 3. Let $F(\boldsymbol{w})$ be an (L, λ)-smooth function ($L \geqslant 0, \lambda \geqslant 0$). The sequence $\{\boldsymbol{w}_t\}$ is produced by (13) and (14). We define a constant $\rho(S)$ as follows:

$$\rho(S) = \sup_{\{\boldsymbol{g}_t\}} \frac{\|\boldsymbol{g}_t\|}{\sum_{s=1}^S \|\boldsymbol{g}_t^{(s)}\|}. \tag{16}$$

Then we have

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla F(\boldsymbol{w}_t)\| \leqslant \frac{2\rho(S)(1-\beta)[F(\boldsymbol{w}_0) - F(\boldsymbol{w}^*)]}{\eta T} + 8L\kappa\rho(S)S\eta + \frac{2\sigma(1+\rho(S)\sqrt{S})}{\sqrt{B}}, \tag{17}$$

where $\kappa = \frac{1+\beta}{(1-\beta)^2}$ and $8\eta\kappa\lambda\rho(S)S \leqslant 1$.

The proof of Theorem 3 is given in Appendix F.

Remark 3. The analysis in Theorem 3 can be easily extended to LARS with the condition $0 < \gamma_l \leqslant \phi(z) \leqslant \gamma_u < \infty$ holding for all z. Let $\phi(\|\boldsymbol{w}_t^{(s)}\|) = \phi_t^{(s)}, \eta_t^{(s)} = \eta\phi_t^{(s)}$ for short. Then we have

$$\boldsymbol{z}_{t+1}^{(s)} = \boldsymbol{z}_t^{(s)} - \frac{\eta}{1-\beta}\frac{\phi_t^{(s)}\boldsymbol{g}_t^{(s)}}{\|\boldsymbol{g}_t^{(s)}\|} = \boldsymbol{z}_t^{(s)} - \frac{\eta_t^{(s)}}{1-\beta}\frac{\boldsymbol{g}_t^{(s)}}{\|\boldsymbol{g}_t^{(s)}\|}.$$

Please note that in (F1)–(F6), we do not take the expections. Hence, we can replace η with $\eta_t^{(s)}$ in (F1). Since $\eta\gamma_l \leqslant \eta_t^{(s)} \leqslant \eta\gamma_u$, we can get a similar convergence rate as that in Theorem 3 by relaxing $\eta_t^{(s)}$ to $\eta\gamma_u$ and relaxing $-\eta_t^{(s)}$ to $-\eta\gamma_l$.

We can observe that Eq. (17) is almost the same as (12) except for the constant $\rho(S)$. In fact, let $\boldsymbol{n}_t = (\boldsymbol{g}_t^{(1)}/\|\boldsymbol{g}_t^{(1)}\|, \boldsymbol{g}_t^{(2)}/\|\boldsymbol{g}_t^{(2)}\|, \ldots, \boldsymbol{g}_t^{(S)}/\|\boldsymbol{g}_t^{(S)}\|)$, which is the update vector in block-wise SNGM. Then

$$\cos(\boldsymbol{g}_t, \boldsymbol{n}_t) = \frac{\boldsymbol{g}_t^{\mathrm{T}}\boldsymbol{n}_t^{\mathrm{T}}}{\|\boldsymbol{g}_t\|\|\boldsymbol{n}_t\|} = \frac{\sum_{s=1}^S \|\boldsymbol{g}_t^{(s)}\|}{\sqrt{S}\|\boldsymbol{g}_t\|} \geqslant \frac{1}{\sqrt{S}\rho(S)}.$$

In large-batch training, the gradient variance σ^2/B is small, and subsequently, the gap $\|\boldsymbol{g}_t - \nabla F(\boldsymbol{w}_t)\|$ is small; i.e., $-\boldsymbol{g}_t$ denotes a descent direction with a high probability. Thus, the included angle between $-\boldsymbol{g}_t$ and $-\boldsymbol{n}_t$ is intuitively expected to be small, i.e., $S = 1$ or $\rho(S) \approx 1/\sqrt{S}$ for $S > 1$, and subsequently, $-\boldsymbol{n}_t$ is also a descent direction. Since $\|\boldsymbol{g}_t\|^2 = \sum_{s=1}^{S} \|\boldsymbol{g}_t^{(s)}\|^2$, $\cos(\boldsymbol{g}_t, \boldsymbol{n}_t) \approx 1$ holds only when these $\|\boldsymbol{g}_t^{(s)}\|$, $s = 1, 2, \ldots, S$ are similar to each other. Unfortunately, empirical results in [13] showed that in multi-layer neural networks, the gradient norms of different layers are quite different. Thus, $-\boldsymbol{n}_t$ will deviate from the expected descent direction with a high probability. Hence, block-wise updates may intuitively slow down the convergence rate.

More specifically, for any fixed $\boldsymbol{w}_0$, B, and T, let $x = 2(1-\beta)[F(\boldsymbol{w}_0) - F(\boldsymbol{w}^*)]/T$, $y = 8L\kappa$, $z = 4\sigma/\sqrt{B}$. Let $P(\eta)$ and $Q(\eta, S)$ denote the right term of (12) and (17) for short, respectively. In practice, given fixed iteration numbers and batch size, we often tune the learning rate to achieve a good convergence result. Hence, we will compare $\min P(\eta)$ and $\min Q(\eta, S)$. For $P(\eta)$, we have $P(\eta) = \frac{x}{\eta} + \eta y + z \geqslant 2\sqrt{xy} + z$, and the equality holds if and only if $\eta = \sqrt{x/y}$. For $Q(\eta, S)$, we have $Q(\eta, S) = \frac{\rho(S)x}{\eta} + \eta\rho(S)Sy + 0.5(1+\rho(S)\sqrt{S})z \geqslant 2\rho(S)\sqrt{S}\sqrt{xy} + 0.5(1+\rho(S)\sqrt{S})z$, and the equality holds if and only if $\eta = \sqrt{x/(Sy)}$. According to the property of norm and Cauchy-Schwarz inequality, it is easy to verify that $\frac{1}{\sqrt{S}} \leqslant \rho(S) \leqslant 1$. Hence, we obtain

$$\min_{\eta, S} Q(\eta, S) \geqslant \min_{\eta} P(\eta), \tag{18}$$

and the equality holds if $S = 1$ or $\rho(S) = 1/\sqrt{S}$. This result implies that block-wise SNGM cannot achieve a better convergence rate than SNGM. Owing to the difference of the gradient norms in each layer during the deep model training process, it is hard for the equality in (18) to hold in practice. Thus, the block-wise updates will actually slow down the convergence rate.

7 Experiments

All experiments are performed using the PyTorch platform on a server with eight NVIDIA Tesla V100 GPU cards. We consider three common deep learning tasks: image classification, natural language processing (NLP), and click-through rate (CTR) prediction for large-batch training evaluation.

7.1 Image classification

We compare SNGM with four baselines: MSGD, LARS [24], EXTRAP-SGD [17], and CLARS [21]. For LARS, EXTRAP-SGD, and CLARS, we adopt the open source code[2)3)4)] shared by the authors. Following the work in [31], the weight decay is set as 0.0001 and the momentum coefficient is set as 0.9.

First, we use the dataset CIFAR-10 and the model ResNet20 [31] to evaluate SNGM. We train the model with eight GPUs. Each GPU will compute a gradient with the batch size being $B/8$. If $B/8 \geqslant 128$, we will use the gradient accumulation [32] with the batch size being 128. We train the model with 160 epochs (i.e., pass through the dataset 160 times). The cosine annealing learning rate [33] (without restarts) is adopted for the five methods. In the m-th epoch, the learning rate is $\eta_m = \eta_0 \times 0.5(1 + \cos(m\pi/160))$, $m = 0, 1, \ldots, 159$. We do not use any training tricks like warm-up [18]. Hence, the only hyperparameter for tuning in each of the five methods is the learning rate. EXTRAP-SGD has an extra inner learning rate for tuning.

Table 2 presents the test accuracy results of the five methods with a small batch size of 128 and different large batch sizes $\{1024, 2048, 4096, 8192\}$. We can observe that for almost all batch sizes, the methods that adopt normalized gradients, including LARS, CLARS, and SNGM, achieve better performance than others. Compared with LARS and CLARS, SNGM achieves better test accuracy for different batch sizes. Figure 2 shows the learning curves of the five methods. We can observe that in the small-batch training, SNGM and other large-batch training methods achieve similar performance in terms of training loss and test accuracy as MSGD. In large-batch training, SNGM achieves better training loss and test accuracy than the four baselines. Furthermore, it achieves faster convergence rates than LARS for the small and large batch sizes, which is consistent with our convergence analysis for the block-wise update strategy.

2) https://github.com/NUS-HPC-AI-Lab/LARS-ImageNet-PyTorch.
3) http://proceedings.mlr.press/v119/lin20b.html.
4) https://github.com/slowbull/largebatch.

Table 2 Test accuracy results (%) on the ResNet20/CIFAR-10 training task with different batch sizes[a)]

	$B = 128$	$B = 1024$	$B = 2048$	$B = 4096$	$B = 8192$
MSGD	92.42	90.62	89.78	87.81	83.93
LARS [24]	92.47	91.90	91.05	90.42	87.07
EXTRP-SGD [17]	91.90	90.41	89.14	88.35	85.78
CLARS [21]	91.40	91.22	91.09	89.99	87.06
SNGM	**92.52**	**92.19**	**91.80**	**91.03**	**89.23**

a) The best results achieved among the compared methods are in bold.

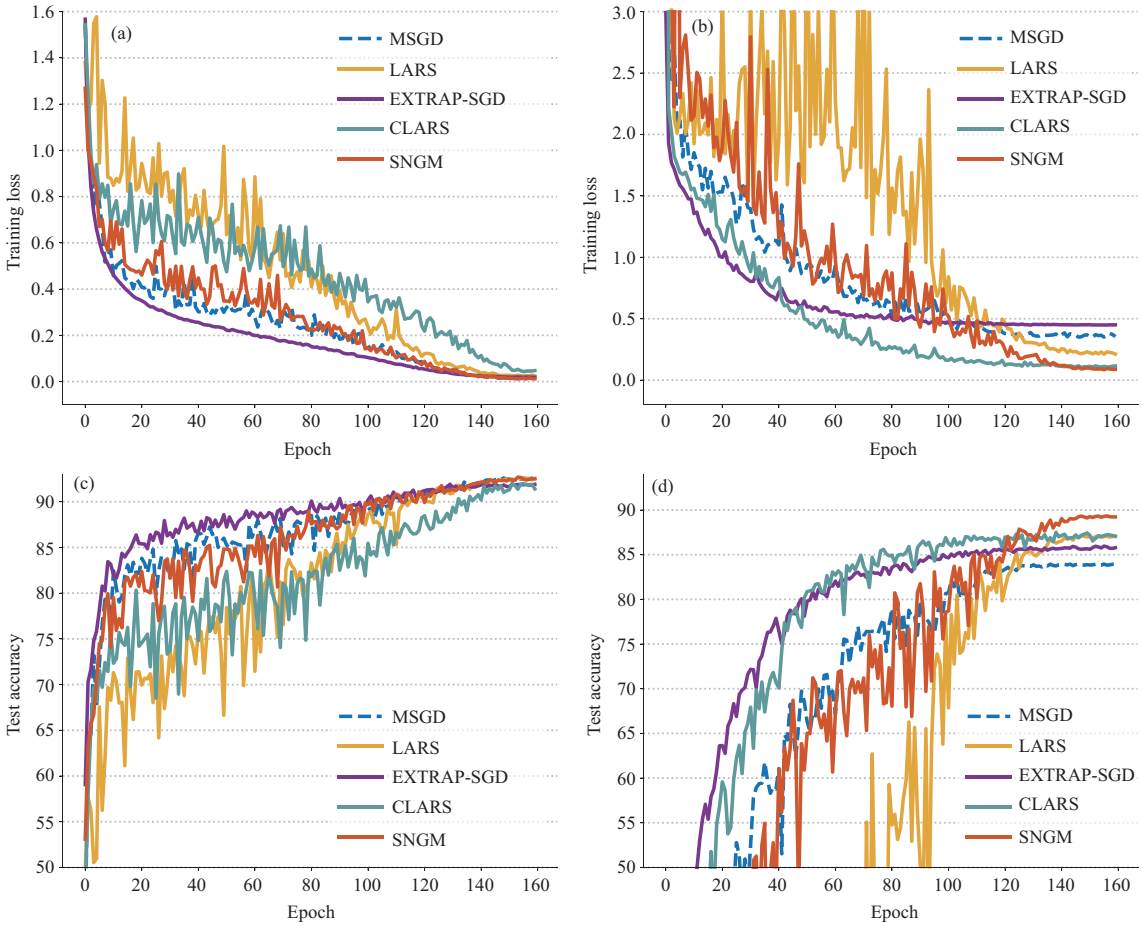

Figure 2 (Color online) Training loss and test accuracy on CIFAR-10. (a) Training loss when $B = 128$; (b) training loss when $B = 8192$; (c) test accuracy when $B = 128$; (d) test accuracy when $B = 8192$.

Table 3 Training time (s) per epoch of SNGM on the ResNet20/CIFAR-10 training task

	$B = 128$	$B = 1024$	$B = 2048$	$B = 4096$	$B = 8192$
Time/epoch	13.95	2.15	1.78	1.6	1.49

Table 3 shows the training time per epoch of SNGM with different batch sizes. When $B = 128$, SNGM has to execute communication frequently and each GPU only computes a mini-batch gradient with the size of 16, which cannot fully utilize the computation power. Hence, compared with other results, SNGM requires more training time for the batch size of 128. Furthermore, we can observe that the training time decreases with the increasing batch size.

Please note that EXTRAP-SGD has two learning rates for tuning and needs to compute two mini-batch gradients in each iteration. EXTRAP-SGD requires more time than other methods to tune hyperparameters and train models. Similarly, CLARS needs to compute extra mini-batch gradients to estimate the layer-wise learning rate for each iteration, which requires more training time and computing resources. Therefore, we will not compare SNGM with EXTRAP-SGD and CLARS in the following experiments.

Table 4 Test accuracy results (%) on ImageNet with different batch sizes[a)]

	$B = 1024$	$B = 32768$
LARS [24]	**76.77**	75.77
SNGM	76.72	**76.16**

a) The best results achieved among the compared methods are in bold.

Table 5 Test accuracy results (%) of the ViT fine-tuning task with different batch sizes[a)]

		$B = 128$	$B = 1024$	$B = 4096$	$B = 8192$	$B = 16384$
CIFAR-10	MSGD	98.96	99.02	98.93	98.81	98.40
	LARS [24]	**99.08**	98.97	98.94	98.85	98.36
	SNGM	99.00	**99.12**	**98.98**	**99.00**	**98.82**
CIFAR-100	MSGD	92.91	**92.99**	92.86	91.57	85.84
	LARS [24]	92.59	92.57	91.80	92.17	90.36
	SNGM	**93.06**	**92.99**	**92.93**	**92.36**	**90.48**

a) The best results achieved among the compared methods are in bold.

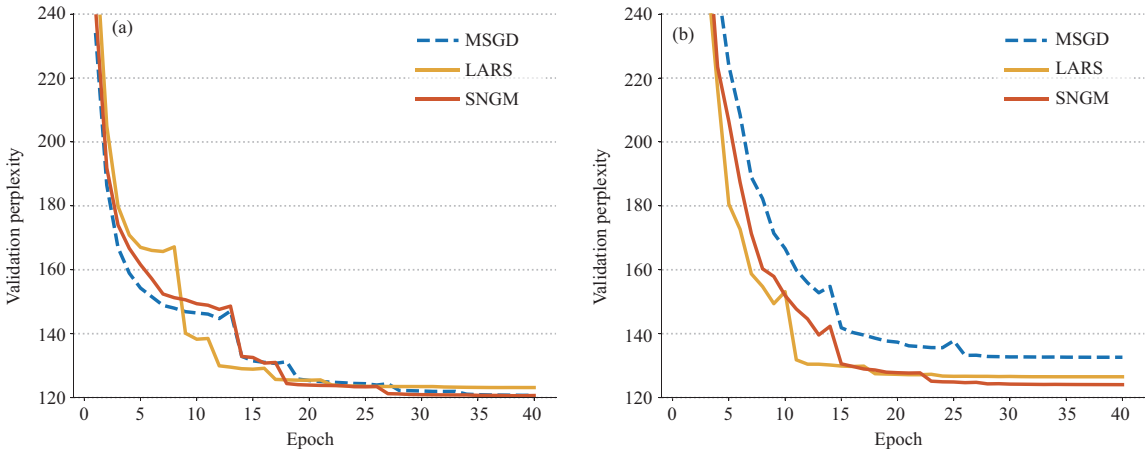

Figure 3 (Color online) Validation perplexity on Wikitext-2. (a) $B = 20$; (b) $B = 2000$.

To further verify the superiority of SNGM with respect to LARS, we also evaluate them on a larger dataset ImageNet [34] and a larger model ResNet50 [31]. We train the model with 90 epochs. As recommended in [13], we use warm-up and polynomial learning rate strategy. Table 4 shows the test accuracy of the methods with a small batch size of 1024 and a large batch size of 32768. Experimental results show that SNGM and LARS achieve almost similar performance in small-batch training. In large-batch training, SNGM outperforms LARS, which is consistent with the results of the ResNet20/CIFAR-10 training task.

Recently, the Transformer architecture has achieved excellent performance on computer vision tasks. Therefore, we also evaluate SNGM based on the ViT fine-tuning task. We use a pre-trained ViT[5)] [35] model and fine-tune it on the CIFAR-10/CIFAR-100 datasets. The experiments are implemented based on the Transformers[6)] framework. We fine-tune the model with 20 epochs. We do not use training tricks such as warm-up [18]. We adopt the linear learning rate decay strategy as default in the Transformers framework. Table 5 shows the test accuracy results of the methods with different batch sizes. SNGM achieves the best performance for almost all batch size settings.

7.2 NLP

We evaluate SNGM by training a two-layer LSTM [36] on the Wikitext-2 [37] dataset. The hidden dimension, embedding size, and dropout ratio are 200, 200, and 0.2, respectively. Gradient clipping is used for MSGD. We train the model with 40 epochs.

Figure 3 shows the validation perplexity of the three methods with a small batch size of 20 and a large batch size of 2000. In small-batch training, SNGM and LARS achieve validation perplexity comparable

5) https://huggingface.co/google/vit-base-patch16-224-in21k.
6) https://github.com/huggingface/transformers.

Table 6 Test perplexity results on Wikitext-2[a)]

	$B = 20$	$B = 1000$	$B = 2000$
MSGD	**113.26**	114.34	118.50
LARS	115.71	116.29	119.35
SNGM	113.74	**112.90**	**115.65**

a) The best results achieved among the compared methods are in bold.

Table 7 Training time (s) per epoch of SNGM on the LSTM/Wikitext-2 training task

B	20	1000	2000
Time/epoch	69.8	24.44	15.22

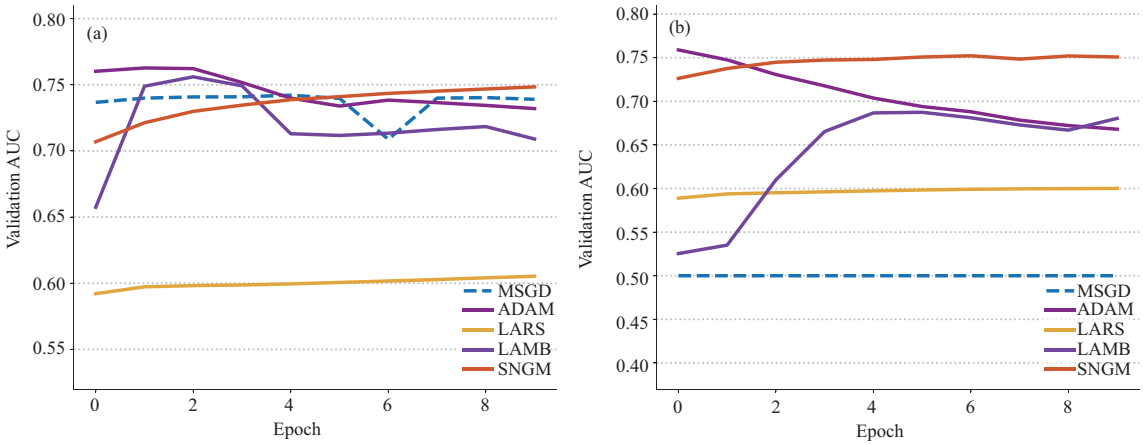

Figure 4 (Color online) Validation AUC on Criteo. (a) $B = 1024$; (b) $B = 8192$.

to that of MSGD. Meanwhile, in large-batch training, SNGM achieves better performance than MSGD and LARS.

Table 6 shows the test perplexity of the three methods with different batch sizes. We can observe that for small batch size, SNGM achieves test perplexity comparable to that of MSGD, and for large batch size, SNGM is better than MSGD. Similar to the results of image classification, SNGM outperforms LARS for different batch sizes.

Table 7 shows the training time per epoch of SNGM with different batch sizes. We can observe that larger batch sizes can reduce the training time, which is similar to the results of image classification tasks.

7.3 Click-through rate prediction

We further conduct CTR prediction experiments to evaluate SNGM. We train DeepFM [38] on a CTR prediction dataset containing ten million samples that are sampled from the Criteo dataset[7)]. We set aside 20% of the samples as the test set and divide the remaining samples into training and validation sets with a ratio of 4 : 1. We compare SNGM with four baselines: MSGD, ADAM [39], LARS [24], and LAMB [24]. LAMB is a layer-wise adaptive large-batch optimization method based on ADAM, while LARS is based on MSGD. The experiments are implemented based on the DeepCTR[8)] framework. The momentum coefficient is set as 0.9 and the weight decay is set as 0.001. The initial learning rate is selected from $\{0.001, 0.01, 0.1\}$ according to the performance on the validation set. We do not adopt any learning rate decay or warm-up strategies. The model is trained with 10 epochs.

Figure 4 shows the validation AUC of the five methods with a small batch size of 1024 and a large batch size of 8192. Table 8 shows the test AUC. SNGM achieves better performance than the other methods, especially in large-batch training.

7) https://ailab.criteo.com/download-criteo-1tb-click-logs-dataset/.
8) https://github.com/shenweichen/DeepCTR-Torch.

Table 8 Test AUC results on Criteo[a)]

	MSGD	ADAM	LARS	LAMB	SNGM
$B = 1024$	0.7397	0.7311	0.6065	0.7066	**0.7489**
$B = 8192$	0.5	0.6666	0.6006	0.6811	**0.7514**

a) The best results achieved among the compared methods are in bold.

8 Conclusion

In this paper, we propose SNGM for large-batch training. We theoretically show that the batch size of MSGD is constrained by the smoothness constant while that of SNGM does not rely on the smoothness constant. Hence, with the same number of gradient computations, SNGM can adopt a larger batch size than MSGD to converge to the ϵ-stationary point. Empirical results on deep learning further verify that SNGM can achieve better test accuracy than MSGD and other state-of-the-art large-batch training methods.

Acknowledgements This work was supported by National Key R&D Program of China (Grant No. 2020YFA0713901), National Natural Science Foundation of China (Grant Nos. 61921006, 62192783), and Fundamental Research Funds for the Central Universities (Grant No. 020214380108).

References

1 Robbins H, Monro S. A stochastic approximation method. Ann Math Statist, 1951, 22: 400–407
2 Ding F, Yang H Z, Liu F. Performance analysis of stochastic gradient algorithms under weak conditions. Sci China Ser F-Inf Sci, 2008, 51: 1269–1280
3 Chen C Y, Wang W L, Zhang Y Z, et al. A convergence analysis for a class of practical variance-reduction stochastic gradient MCMC. Sci China Inf Sci, 2019, 62: 012101
4 Zhao S-Y, Xie Y-P, Li W-J. On the convergence and improvement of stochastic normalized gradient descent. Sci China Inf Sci, 2021, 64: 132103
5 Li M, Zhang T, Chen Y Q, et al. Efficient mini-batch training for stochastic optimization. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, 2014. 661–670
6 Yu H, Jin R, Yang S. On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization. In: Proceedings of the International Conference on Machine Learning, Long Beach, 2019. 7184–7193
7 Li M, Andersen D G, Smola A J, et al. Communication efficient distributed machine learning with the parameter server. In: Proceedings of the Advances in Neural Information Processing Systems, Montréal, 2014. 19–27
8 Lian X R, Zhang C, Zhang H, et al. Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent. In: Proceedings of the Advances in Neural Information Processing Systems, Long Beach, 2017. 5330–5340
9 Lin T, Karimireddy S P, Stich S U, et al. Quasi-global momentum: accelerating decentralized deep learning on heterogeneous data. In: Proceedings of the International Conference on Machine Learning, 2021. 6654–6665
10 Yu H, Yang S, Zhu S H. Parallel restarted SGD with faster convergence and less communication: demystifying why model averaging works for deep learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, Hawaii, 2019. 5693–5700
11 Lin T, Stich S U, Patel K K, et al. Don't use large mini-batches, use local SGD. In: Proceedings of the International Conference on Learning Representations, Addis Ababa, 2020
12 Gao H C, Xu A, Huang H. On the convergence of communication-efficient local SGD for federated learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2021. 7510–7518
13 You Y, Gitman I, Ginsburg B. Scaling SGD batch size to 32K for ImageNet training. 2017. ArXiv:1708.03888
14 LeCun Y A, Bottou L, Orr G B, et al. Neural Networks: Tricks of the Trade. Berlin: Springer Science & Business Media, 2012. 9–48
15 Keskar N S, Mudigere D, Nocedal J, et al. On large-batch training for deep learning: generalization gap and sharp minima. In: Proceedings of the International Conference on Learning Representations, Toulon, 2017
16 Hoffer E, Hubara I, Soudry D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In: Proceedings of the Advances in Neural Information Processing Systems, Long Beach, 2017. 1731–1741
17 Lin T, Kong L J, Stich S U, et al. Extrapolation for large-batch training in deep learning. In: Proceedings of the International Conference on Machine Learning, 2020. 6094–6104
18 Goyal P, Dollár P, Girshick R, et al. Accurate, large minibatch SGD: training ImageNet in 1 hour. 2017. ArXiv:1706.02677
19 You Y, Hseu J, Ying C, et al. Large-batch training for LSTM and beyond. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, 2019. 1–16
20 Ginsburg B, Castonguay P, Hrinchuk O, et al. Stochastic gradient methods with layer-wise adaptive moments for training of deep networks. 2019. ArXiv:1905.11286
21 Huo Z Y, Gu B, Huang H. Large batch optimization for deep learning using new complete layer-wise adaptive rate scaling. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2021. 7883–7890
22 Liu Y, Mai S Q, Chen X N, et al. Towards efficient and scalable sharpness-aware minimization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Louisiana, 2022. 12360–12370
23 Liu R, Mozafari B. Communication-efficient distributed learning for large batch optimization. In: Proceedings of the International Conference on Machine Learning, Baltimore, 2022. 13925–13946
24 You Y, Li J, Reddi S, et al. Large batch optimization for deep learning: training BERT in 76 minutes. In: Proceedings of the International Conference on Learning Representations, Addis Ababa, 2020
25 Yuan K, Chen Y M, Huang X M, et al. DecentLaM: decentralized momentum SGD for large-batch deep training. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, Montréal, 2021. 3009–3019
26 Zhang J Z, He T X, Sra S, et al. Why gradient clipping accelerates training: a theoretical justification for adaptivity. In: Proceedings of the International Conference on Learning Representations, Addis Ababa, 2020

27 Nesterov Y E. Introductory Lectures on Convex Optimization: A Basic Course. Berlin: Springer Science & Business Media, 2004
28 Polyak B T. Some methods of speeding up the convergence of iteration methods. USSR Comput Math Math Phys, 1964, 4: 1–17
29 Hazan E, Levy K, Shalev-Shwartz S. Beyond convexity: stochastic quasi-convex optimization. In: Proceedings of the Advances in Neural Information Processing Systems, Montréal, 2015. 1594–1602
30 Yan Y, Yang T B, Li Z, et al. A unified analysis of stochastic momentum methods for deep learning. In: Proceedings of the International Joint Conference on Artificial Intelligence, Stockholm, 2018. 2955–2961
31 He K M, Zhang X Y, Ren S Q, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, 2016. 770–778
32 Ott M, Edunov S, Grangier D, et al. Scaling neural machine translation. In: Proceedings of the Conference on Machine Translation, Brussels, 2018. 1–9
33 Loshchilov I, Hutter F. SGDR: stochastic gradient descent with warm restarts. In: Proceedings of the International Conference on Learning Representations, Toulon, 2017
34 Deng J, Dong W, Socher R, et al. ImageNet: a large-scale hierarchical image database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, 2009. 248–255
35 Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16 × 16 words: transformers for image recognition at scale. In: Proceedings of the International Conference on Learning Representations, 2021
36 Merity S, Keskar N S, Socher R. Regularizing and optimizing LSTM language models. In: Proceedings of the International Conference on Learning Representations, Vancouver, 2018
37 Merity S, Xiong C M, Bradbury J, et al. Pointer sentinel mixture models. In: Proceedings of the International Conference on Learning Representations, Toulon, 2017
38 Guo H F, Tang R M, Ye Y M, et al. DeepFM: a factorization-machine based neural network for CTR prediction. In: Proceedings of the International Joint Conference on Artificial Intelligence, Melbourne, 2017. 1725–1731
39 Kingma D P, Ba J. ADAM: a method for stochastic optimization. In: Proceedings of the International Conference on Learning Representations, San Diego, 2015

Appendix A Proof of Lemma 1

For any $\boldsymbol{u}, \boldsymbol{w}$, let $r(x) = x(\boldsymbol{u} - \boldsymbol{w}) + \boldsymbol{w}$, $p(x) = \|\nabla h(r(x))\|, x \in [0, 1]$. Then we have

$$\begin{aligned} p(x) = \|\nabla h(r(x))\| &= \left\| \int_0^x H_h(r(y)) r'(y) \mathrm{d}y + \nabla h(r(0)) \right\| = \left\| \int_0^x H_h(r(y))(\boldsymbol{u} - \boldsymbol{w}) \mathrm{d}y + \nabla h(\boldsymbol{w}) \right\| \\ &\leqslant \|\boldsymbol{u} - \boldsymbol{w}\| \int_0^x \|H_h(r(y))\| \mathrm{d}y + \|\nabla h(\boldsymbol{w})\| \leqslant \alpha \int_0^x (L + \lambda \|\nabla h(r(y))\|) \mathrm{d}y + \|\nabla h(\boldsymbol{w})\| \\ &\leqslant L\alpha + \|\nabla h(\boldsymbol{w})\| + \lambda\alpha \int_0^x p(y) \mathrm{d}y. \end{aligned}$$

According to Gronwall's inequality, we obtain $p(x) \leqslant (L\alpha + \|\nabla h(\boldsymbol{w})\|)\mathrm{e}^{\lambda\alpha}$. Then we have

$$\|\nabla h(\boldsymbol{u})\| = \|\nabla h(r(1))\| = p(1) \leqslant (L\alpha + \|\nabla h(\boldsymbol{w})\|)\mathrm{e}^{\lambda\alpha}.$$

Appendix B Proof of Lemma 2

According to (7), we have

$$\|\boldsymbol{u}_{t+1}\| \leqslant \beta\|\boldsymbol{u}_t\| + 1 \leqslant \beta^2\|\boldsymbol{u}_{t-1}\| + \beta + 1 \leqslant \beta^{t+1}\|\boldsymbol{u}_0\| + \beta^t + \beta^{t-1} + \cdots + 1 \leqslant \frac{1}{1-\beta}.$$

Appendix C Proof of Lemma 3

According to (9), we have

$$\begin{aligned} \boldsymbol{z}_{t+1} &= \frac{1}{1-\beta}\boldsymbol{w}_{t+1} - \frac{\beta}{1-\beta}\boldsymbol{w}_t = \frac{1}{1-\beta}\left(\boldsymbol{w}_t - \eta\frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|} + \beta(\boldsymbol{w}_t - \boldsymbol{w}_{t-1})\right) - \frac{\beta}{1-\beta}\boldsymbol{w}_t \\ &= \frac{1}{1-\beta}\boldsymbol{w}_t - \frac{\beta}{1-\beta}\boldsymbol{w}_{t-1} - \frac{\eta}{1-\beta}\frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|} = \boldsymbol{z}_t - \frac{\eta}{1-\beta}\frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|}. \end{aligned}$$

Appendix D Proof of Theorem 1

According to Lemma 3, we have $\boldsymbol{z}_{t+1} = \boldsymbol{z}_t - \frac{\eta}{1-\beta}\frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|}$. Using the smooth property, we obtain

$$\begin{aligned} F(\boldsymbol{z}_{t+1}) &\leqslant F(\boldsymbol{z}_t) - \frac{\eta}{1-\beta}\nabla F(\boldsymbol{z}_t)^{\mathrm{T}}\frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|} + \frac{L\eta^2}{2(1-\beta)^2} \\ &= F(\boldsymbol{z}_t) - \frac{\eta}{1-\beta}\left[(\nabla F(\boldsymbol{z}_t) - \nabla F(\boldsymbol{w}_t))^{\mathrm{T}}\frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|} + (\nabla F(\boldsymbol{w}_t) - \boldsymbol{g}_t)^{\mathrm{T}}\frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|}\right] - \frac{\eta}{1-\beta}\|\boldsymbol{g}_t\| + \frac{L\eta^2}{2(1-\beta)^2} \\ &\leqslant F(\boldsymbol{z}_t) + \frac{\eta L}{1-\beta}\|\boldsymbol{z}_t - \boldsymbol{w}_t\| + \frac{\eta}{1-\beta}\|\nabla F(\boldsymbol{w}_t) - \boldsymbol{g}_t\| - \frac{\eta}{1-\beta}\|\boldsymbol{g}_t\| + \frac{L\eta^2}{2(1-\beta)^2}. \end{aligned}$$

Since $\boldsymbol{w}_{t+1} - \boldsymbol{w}_t = \beta(\boldsymbol{w}_t - \boldsymbol{w}_{t-1}) - \eta\boldsymbol{g}_t/\|\boldsymbol{g}_t\|$, we obtain

$$\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\| \leqslant \beta\|\boldsymbol{w}_t - \boldsymbol{w}_{t-1}\| + \eta \leqslant \frac{\eta}{1-\beta}.$$

Hence, $\|\boldsymbol{w}_t - \boldsymbol{w}_{t-1}\| \leqslant \eta/(1-\beta)$ and

$$\|\boldsymbol{z}_t - \boldsymbol{w}_t\| = \frac{\beta}{1-\beta}\|\boldsymbol{w}_t - \boldsymbol{w}_{t-1}\| \leqslant \frac{\beta\eta}{(1-\beta)^2}.$$

Combining the above inequalities, we obtain

$$\|\boldsymbol{g}_t\| \leqslant \frac{(1-\beta)[F(\boldsymbol{z}_t) - F(\boldsymbol{z}_{t+1})]}{\eta} + \frac{L\eta}{2(1-\beta)} + \frac{L\beta\eta}{(1-\beta)^2} + \|\nabla F(\boldsymbol{w}_t) - \boldsymbol{g}_t\|.$$

Since $\|\nabla F(\boldsymbol{w}_t)\| \leqslant \|\nabla F(\boldsymbol{w}_t) - \boldsymbol{g}_t\| + \|\boldsymbol{g}_t\|$, we obtain

$$\|\nabla F(\boldsymbol{w}_t)\| \leqslant \frac{(1-\beta)[F(\boldsymbol{z}_t) - F(\boldsymbol{z}_{t+1})]}{\eta} + \frac{L\eta}{2(1-\beta)} + \frac{L\beta\eta}{(1-\beta)^2} + 2\|\nabla F(\boldsymbol{w}_t) - \boldsymbol{g}_t\|.$$

Using the fact that $\mathbb{E}\|\nabla F(\boldsymbol{w}_t) - \boldsymbol{g}_t\| \leqslant \sigma/\sqrt{B}$ and summing up the above inequality from $t=0$ to $T-1$, we obtain

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\boldsymbol{w}_t)\| \leqslant \frac{(1-\beta)[F(\boldsymbol{w}_0) - F(\boldsymbol{w}^*)]}{\eta T} + \frac{L}{2}\kappa\eta + \frac{2\sigma}{\sqrt{B}}.$$

Appendix E Proof of Theorem 2

According to Lemma 3, we have $\boldsymbol{z}_{t+1} = \boldsymbol{z}_t - \frac{\eta}{1-\beta}\frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|}$. Using the Taylor theorem, there exists ξ_t such that

$$\begin{aligned} F(\boldsymbol{z}_{t+1}) &\leqslant F(\boldsymbol{z}_t) - \frac{\eta}{1-\beta}\nabla F(\boldsymbol{z}_t)^{\mathrm{T}}\frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|} + \frac{\eta^2\|H_F(\xi_t)\|}{2(1-\beta)^2} \\ &= F(\boldsymbol{z}_t) - \frac{\eta}{1-\beta}(\nabla F(\boldsymbol{z}_t) - \nabla F(\boldsymbol{w}_t) + \nabla F(\boldsymbol{w}_t) - \boldsymbol{g}_t)^{\mathrm{T}}\frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|} - \frac{\eta}{1-\beta}\|\boldsymbol{g}_t\| + \frac{\eta^2\|H_F(\xi_t)\|}{2(1-\beta)^2}. \end{aligned}$$

Let $\psi_t(\boldsymbol{w}) = (\nabla F(\boldsymbol{w}) - \nabla F(\boldsymbol{w}_t))^{\mathrm{T}}\frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|}$. Using the Taylor theorem, there exists ζ_t such that

$$|\psi_t(\boldsymbol{z}_t)| = |\psi_t(\boldsymbol{w}_t) + \nabla\psi_t(\zeta_t)(\boldsymbol{z}_t - \boldsymbol{w}_t)| = |\nabla\psi_t(\zeta_t)(\boldsymbol{z}_t - \boldsymbol{w}_t)| \leqslant \|H_F(\zeta_t)\|\|\boldsymbol{z}_t - \boldsymbol{w}_t\|.$$

Combining the above inequalities, we obtain

$$\|\boldsymbol{g}_t\| \leqslant \frac{(1-\beta)[F(\boldsymbol{z}_t) - F(\boldsymbol{z}_{t+1})]}{\eta} + \frac{\eta\|H_F(\xi_t)\|}{2(1-\beta)} + \|H_F(\zeta_t)\|\|\boldsymbol{z}_t - \boldsymbol{w}_t\| + \|\nabla F(\boldsymbol{w}_t) - \boldsymbol{g}_t\|.$$

Since $\boldsymbol{w}_{t+1} - \boldsymbol{w}_t = \beta(\boldsymbol{w}_t - \boldsymbol{w}_{t-1}) - \eta\frac{\boldsymbol{g}_t}{\|\boldsymbol{g}_t\|}$, we obtain

$$\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\| \leqslant \beta\|\boldsymbol{w}_t - \boldsymbol{w}_{t-1}\| + \eta \leqslant \frac{\eta}{1-\beta}.$$

Hence, $\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\| \leqslant \frac{\eta}{1-\beta}$ and

$$\|\boldsymbol{z}_t - \boldsymbol{w}_t\| = \frac{\beta}{1-\beta}\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\| \leqslant \frac{\beta\eta}{(1-\beta)^2}.$$

Combining the above inequalities, we obtain

$$\|\boldsymbol{g}_t\| \leqslant \frac{(1-\beta)[F(\boldsymbol{z}_t) - F(\boldsymbol{z}_{t+1})]}{\eta} + \frac{\eta\|H_F(\xi_t)\|}{2(1-\beta)} + \frac{\beta\eta\|H_F(\zeta_t)\|}{(1-\beta)^2} + \|\nabla F(\boldsymbol{w}_t) - \boldsymbol{g}_t\|.$$

Since $\|\nabla F(\boldsymbol{w}_t)\| \leqslant \|\nabla F(\boldsymbol{w}_t) - \boldsymbol{g}_t\| + \|\boldsymbol{g}_t\|$, we obtain

$$\|\nabla F(\boldsymbol{w}_t)\| \leqslant \frac{(1-\beta)[F(\boldsymbol{z}_t) - F(\boldsymbol{z}_{t+1})]}{\eta} + \frac{\eta\|H_F(\xi_t)\|}{2(1-\beta)} + \frac{\beta\eta\|H_F(\zeta_t)\|}{(1-\beta)^2} + 2\|\nabla F(\boldsymbol{w}_t) - \boldsymbol{g}_t\|.$$

Next, we bound the two Hessian matrices. For convenience, let $\kappa = \frac{1+\beta}{(1-\beta)^2}$. Since $\|\boldsymbol{z}_t - \boldsymbol{w}_t\| \leqslant \frac{\beta\eta}{(1-\beta)^2}$ and

$$\|\boldsymbol{z}_{t+1} - \boldsymbol{w}_t\| \leqslant \|\boldsymbol{z}_{t+1} - \boldsymbol{z}_t\| + \|\boldsymbol{z}_t - \boldsymbol{w}_t\| \leqslant \eta\left(\frac{1}{1-\beta} + \frac{\beta}{(1-\beta)^2}\right) \leqslant \kappa\eta,$$

combining with Definition 2, Lemma 1, and $\lambda\kappa\eta \leqslant 1$, we obtain

$$\begin{aligned} \|H_F(\zeta_t)\| &\leqslant L + (L + \lambda\|\nabla F(\boldsymbol{w}_t)\|)\mathrm{e}, \\ \|H_F(\xi_t)\| &\leqslant L + (L + \lambda\|\nabla F(\boldsymbol{w}_t)\|)\mathrm{e}. \end{aligned}$$

Then we have

$$\begin{aligned} \|\nabla F(\boldsymbol{w}_t)\| &\leqslant \frac{(1-\beta)[F(\boldsymbol{z}_t) - F(\boldsymbol{z}_{t+1})]}{\eta} + \left[\frac{\eta}{2(1-\beta)} + \frac{\beta\eta}{(1-\beta)^2}\right][L + (L + \lambda\|\nabla F(\boldsymbol{w}_t)\|)\mathrm{e}] + 2\|\nabla F(\boldsymbol{w}_t) - \boldsymbol{g}_t\| \\ &\leqslant \frac{(1-\beta)[F(\boldsymbol{z}_t) - F(\boldsymbol{z}_{t+1})]}{\eta} + 4\kappa\eta[L + \lambda\|\nabla F(\boldsymbol{w}_t)\|] + 2\|\nabla F(\boldsymbol{w}_t) - \boldsymbol{g}_t\|. \end{aligned}$$

Since $4\lambda\kappa\eta \leqslant \frac{1}{2}$, we obtain

$$\|\nabla F(\boldsymbol{w}_t)\| \leqslant \frac{2(1-\beta)[F(\boldsymbol{z}_t) - F(\boldsymbol{z}_{t+1})]}{\eta} + 8L\kappa\eta + 4\|\nabla F(\boldsymbol{w}_t) - \boldsymbol{g}_t\|.$$

Summing up the above inequality from $t=0$ to $T-1$ and using the fact that $\mathbb{E}\|\nabla F(\boldsymbol{w}_t) - \boldsymbol{g}_t\| \leqslant \frac{\sigma}{\sqrt{B}}$, we obtain

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\boldsymbol{w}_t)\| \leqslant \frac{2(1-\beta)[F(\boldsymbol{w}_0) - F(\boldsymbol{w}^*)]}{\eta T} + 8L\kappa\eta + 4\frac{\sigma}{\sqrt{B}}.$$

Appendix F Proof of Theorem 3

Similar to the derivation in (9), the update rule of the block-wise variant of SNGM in (13) and (14) can be reorganized as

$$\boldsymbol{w}_{t+1}^{(s)}-\boldsymbol{w}_t^{(s)}=\beta(\boldsymbol{w}_t^{(s)}-\boldsymbol{w}_{t-1}^{(s)})-\eta\frac{\boldsymbol{g}_t^{(s)}}{\|\boldsymbol{g}_t^{(s)}\|},$$

where $s=1,2,\ldots,S$. According to (15), if let $\boldsymbol{z}_t=(\boldsymbol{z}_t^{(1)},\boldsymbol{z}_t^{(2)},\ldots,\boldsymbol{z}_t^{(S)})$, $\boldsymbol{n}_t=(\frac{\boldsymbol{g}_t^{(1)}}{\|\boldsymbol{g}_t^{(1)}\|},\frac{\boldsymbol{g}_t^{(2)}}{\|\boldsymbol{g}_t^{(2)}\|},\ldots,\frac{\boldsymbol{g}_t^{(S)}}{\|\boldsymbol{g}_t^{(S)}\|})$, $\psi_t(\boldsymbol{w})=(\nabla F(\boldsymbol{w})-\nabla F(\boldsymbol{w}_t))^{\mathrm{T}}\boldsymbol{n}_t$, we have

$$\begin{aligned}F(\boldsymbol{z}_{t+1})&\leqslant F(\boldsymbol{z}_t)-\frac{\eta}{1-\beta}\sum_{s=1}^{S}\nabla F^{(s)}(\boldsymbol{z}_t)^{\mathrm{T}}\frac{\boldsymbol{g}_t^{(s)}}{\|\boldsymbol{g}_t^{(s)}\|}+\frac{S\eta^2}{2(1-\beta)^2}\|H_F(\xi_t)\|\\&=F(\boldsymbol{z}_t)-\frac{\eta}{1-\beta}\sum_{s=1}^{S}(\nabla F^{(s)}(\boldsymbol{z}_t)-\nabla F^{(s)}(\boldsymbol{w}_t)+\nabla F^{(s)}(\boldsymbol{w}_t)-\boldsymbol{g}_t^{(s)})^{\mathrm{T}}\frac{\boldsymbol{g}_t^{(s)}}{\|\boldsymbol{g}_t^{(s)}\|}\\&\quad-\frac{\eta}{1-\beta}\sum_{s=1}^{S}\|\boldsymbol{g}_t^{(s)}\|+\frac{S\eta^2}{2(1-\beta)^2}\|H_F(\xi_t)\|,\end{aligned}\tag{F1}$$

$$|\psi_t(\boldsymbol{z}_t)|=|\psi_t(\boldsymbol{w}_t)+\nabla\psi_t(\zeta_t)(\boldsymbol{z}_t-\boldsymbol{w}_t)|=|\nabla\psi_t(\zeta_t)(\boldsymbol{z}_t-\boldsymbol{w}_t)|\leqslant\sqrt{S}\|H_F(\zeta_t)\|\|\boldsymbol{z}_t-\boldsymbol{w}_t\|,\tag{F2}$$

and

$$\left|\sum_{s=1}^{S}(\nabla F^{(s)}(\boldsymbol{w}_t)-\boldsymbol{g}_t^{(s)})^{\mathrm{T}}\frac{\boldsymbol{g}_t^{(s)}}{\|\boldsymbol{g}_t^{(s)}\|}\right|=|(\nabla F(\boldsymbol{w}_t)-\boldsymbol{g}_t)^{\mathrm{T}}\boldsymbol{n}_t|\leqslant\sqrt{S}\|\nabla F(\boldsymbol{w}_t)-\boldsymbol{g}_t\|.\tag{F3}$$

Combining (F1), (F2), and (F3), we obtain

$$\begin{aligned}\sum_{s=1}^{S}\|\boldsymbol{g}_t^{(s)}\|&\leqslant\frac{(1-\beta)[F(\boldsymbol{z}_t)-F(\boldsymbol{z}_{t+1})]}{\eta}+\frac{S\eta}{2(1-\beta)}\|H_F(\xi_t)\|\\&\quad+\sqrt{S}(\|H_F(\zeta_t)\|\|\boldsymbol{z}_t-\boldsymbol{w}_t\|+\|\nabla F(\boldsymbol{w}_t)-\boldsymbol{g}_t\|).\end{aligned}\tag{F4}$$

Since $\boldsymbol{w}_{t+1}-\boldsymbol{w}_t=\beta(\boldsymbol{w}_t-\boldsymbol{w}_{t-1})-\eta\boldsymbol{n}_t$, we obtain

$$\|\boldsymbol{w}_{t+1}-\boldsymbol{w}_t\|\leqslant\beta\|\boldsymbol{w}_t-\boldsymbol{w}_{t-1}\|+\sqrt{S}\eta\leqslant\frac{\sqrt{S}\eta}{1-\beta}.$$

Hence,

$$\|\boldsymbol{z}_t-\boldsymbol{w}_t\|\leqslant\frac{\beta}{1-\beta}\|\boldsymbol{w}_t-\boldsymbol{w}_{t-1}\|\leqslant\frac{\sqrt{S}\eta\beta}{(1-\beta)^2}.\tag{F5}$$

Combining (F4) and (F5), we obtain

$$\sum_{s=1}^{S}\|\boldsymbol{g}_t^{(s)}\|\leqslant\frac{(1-\beta)[F(\boldsymbol{z}_t)-F(\boldsymbol{z}_{t+1})]}{\eta}+\frac{\eta S}{2(1-\beta)}\|H_F(\xi_t)\|+\frac{\beta S\eta}{(1-\beta)^2}\|H_F(\zeta_t)\|+\sqrt{S}\|\nabla F(\boldsymbol{w}_t)-\boldsymbol{g}_t\|.$$

Since $\|\nabla F(\boldsymbol{w}_t)\|\leqslant\|\nabla F(\boldsymbol{w}_t)-\boldsymbol{g}_t\|+\|\boldsymbol{g}_t\|\leqslant\|\nabla F(\boldsymbol{w}_t)-\boldsymbol{g}_t\|+\sum_{s=1}^{S}\rho(S)\|\boldsymbol{g}_t^{(s)}\|$, we obtain

$$\|\nabla F(\boldsymbol{w}_t)\|\leqslant\frac{\rho(S)(1-\beta)[F(\boldsymbol{z}_t)-F(\boldsymbol{z}_{t+1})]}{\eta}+\frac{\rho(S)\eta S}{2(1-\beta)}\|H_F(\xi_t)\|+\frac{\rho(S)\beta S\eta}{(1-\beta)^2}\|H_F(\zeta_t)\|+(1+\rho(S)\sqrt{S})\|\nabla F(\boldsymbol{w}_t)-\boldsymbol{g}_t\|.$$

Next, we bound the two Hessian matrices. Since $\|\boldsymbol{z}_t-\boldsymbol{w}_t\|\leqslant\frac{\sqrt{S}\beta\eta}{(1-\beta)^2}$ and

$$\|\boldsymbol{z}_{t+1}-\boldsymbol{w}_t\|\leqslant\|\boldsymbol{z}_{t+1}-\boldsymbol{z}_t\|+\|\boldsymbol{z}_t-\boldsymbol{w}_t\|\leqslant\sqrt{S}\eta\left(\frac{1}{1-\beta}+\frac{\beta}{(1-\beta)^2}\right)\leqslant\sqrt{S}\eta\kappa,$$

combining with Definition 2, Lemma 1, and $\sqrt{S}\eta\kappa\lambda\leqslant1$, we obtain

$$\begin{aligned}\|H_F(\zeta_t)\|&\leqslant L+(L+\lambda\|\nabla F(\boldsymbol{w}_t)\|)\mathrm{e},\\\|H_F(\xi_t)\|&\leqslant L+(L+\lambda\|\nabla F(\boldsymbol{w}_t)\|)\mathrm{e}.\end{aligned}$$

Then we obtain

$$\begin{aligned}\|\nabla F(\boldsymbol{w}_t)\|&\leqslant\frac{\rho(S)(1-\beta)[F(\boldsymbol{z}_t)-F(\boldsymbol{z}_{t+1})]}{\eta}+\left[\frac{\rho(S)\eta S}{2(1-\beta)}+\frac{\rho(S)\beta S\eta}{(1-\beta)^2}\right][L+(L+\lambda\|\nabla F(\boldsymbol{w}_t)\|)\mathrm{e}]+(1+\rho(S)\sqrt{S})\|\nabla F(\boldsymbol{w}_t)-\boldsymbol{g}_t\|\\&\leqslant\frac{\rho(S)(1-\beta)[F(\boldsymbol{z}_t)-F(\boldsymbol{z}_{t+1})]}{\eta}+4\kappa\rho(S)S\eta(L+\lambda\|\nabla F(\boldsymbol{w}_t)\|)+(1+\rho(S)\sqrt{S})\|\nabla F(\boldsymbol{w}_t)-\boldsymbol{g}_t\|.\end{aligned}$$

Since $4\kappa\rho(S)S\eta\lambda\leqslant\frac{1}{2}$, we obtain

$$\|\nabla F(\boldsymbol{w}_t)\|\leqslant\frac{2\rho(S)(1-\beta)[F(\boldsymbol{z}_t)-F(\boldsymbol{z}_{t+1})]}{\eta}+8L\kappa\rho(S)S\eta+2(1+\rho(S)\sqrt{S})\|\nabla F(\boldsymbol{w}_t)-\boldsymbol{g}_t\|.\tag{F6}$$

Summing up the above inequality from $t=0$ to $T-1$, we obtain

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\boldsymbol{w}_t)\|\leqslant\frac{2\rho(S)(1-\beta)[F(\boldsymbol{w}_0)-F(\boldsymbol{w}^*)]}{\eta T}+8L\kappa\rho(S)S\eta+\frac{2(1+\rho(S)\sqrt{S})\sigma}{\sqrt{B}}.$$