Increasing Batch Size Improves Convergence of Stochastic Gradient Descent with Momentum

Keisuke Kamo* CE245016@MEIJI.AC.JP
Hideaki Iiduka* IIDUKA@CS.MEIJI.AC.JP
Department of Computer Science, Meiji University, Japan

Editors: Hung-yi Lee and Tongliang Liu

Abstract

Stochastic gradient descent with momentum (SGDM), in which a momentum term is added to SGD, has been well studied in both theory and practice. The theoretical studies show that the settings of the learning rate and momentum weight affect the convergence of SGDM. Meanwhile, the practical studies have shown that the batch-size setting strongly affects the performance of SGDM. In this paper, we focus on mini-batch SGDM with a constant learning rate and constant momentum weight, which is frequently used to train deep neural networks. We show theoretically that using a constant batch size does not always minimize the expectation of the full gradient norm of the empirical loss in training a deep neural network, whereas using an increasing batch size definitely minimizes it; that is, an increasing batch size improves the convergence of mini-batch SGDM. We also provide numerical results supporting our analyses, indicating specifically that mini-batch SGDM with an increasing batch size converges to stationary points faster than with a constant batch size, while also reducing computational cost. Python implementations of the optimizers used in the numerical experiments are available at https://github.com/iiduka-researches/NSHB_increasing_batchsize_acml25/.

Keywords: convergence rate; increasing batch size; nonconvex optimization; stochastic gradient descent with momentum

1. Introduction

Stochastic gradient descent (SGD) and its variants, such as SGD with momentum (SGDM) and adaptive methods, are useful optimizers for minimizing the empirical loss defined by the mean of nonconvex loss functions in training a deep neural network (DNN). In the present paper, we focus on SGDM optimizers, in which a momentum term is added to SGD. Various types of SGDM have been proposed, including stochastic heavy ball (SHB) (Polyak, 1964), normalized-SHB (NSHB) (Gupal and Bazhenov, 1972), Nesterov's accelerated gradient method (Nesterov, 1983; Sutskever et al., 2013), synthesized Nesterov variants (Lessard et al., 2016), Triple Momentum (Van Scoy et al., 2018), Robust Momentum (Cyrus et al., 2018), PID control-based methods (An et al., 2018), stochastic unified momentum (SUM) (Yan et al., 2018), accelerated SGD (Jain et al., 2018; Kidambi et al., 2018; Varre and Flammarion, 2022; Li et al., 2024), quasi-hyperbolic momentum (QHM) (Ma and Yarats, 2019), and proximal-type SHB (PSHB) (Mai and Johansson, 2020).

Since the empirical loss is nonconvex with respect to a parameter $\boldsymbol{\theta} \in \mathbb{R}^d$ of a DNN, we are interested in nonconvex optimization for SGDM. Let $\boldsymbol{\theta}_t \in \mathbb{R}^d$ be the t-th approximation

* Equal contribution

of SGDM to minimize the nonconvex empirical loss function $f\colon \mathbb{R}^d \to \mathbb{R}$. SGDM is defined as $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \boldsymbol{m}_t$, where $\eta_t > 0$ is a learning rate and $\boldsymbol{m}_t$ is a momentum buffer. For example, SGDM with $\boldsymbol{m}_t := \beta \boldsymbol{m}_{t-1} + \nabla f_{B_t}(\boldsymbol{\theta}_t)$ is SHB, where $\nabla f_{B_t}\colon \mathbb{R}^d \to \mathbb{R}^d$ denotes the stochastic gradient of f and $\beta \in [0,1)$ is a momentum weight. SGDM with $\boldsymbol{m}_t := \beta \boldsymbol{m}_{t-1} + (1-\beta)\nabla f_{B_t}(\boldsymbol{\theta}_t)$ is NSHB. Since SHB with $\beta = 0$ (NSHB with $\beta = 0$) coincides with SGD, which defined by $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla f_{B_t}(\boldsymbol{\theta}_t)$, SGDM is defined as SGD with an added momentum term (e.g., $\beta \boldsymbol{m}_{t-1}$ in the case of SHB).

Table 1: Convergence of SGDM optimizers to minimize L-smooth f over the number of steps T. "Noise" in the Gradient column means that the optimizer uses noisy observations, i.e., $\boldsymbol{g}(\boldsymbol{\theta}) = \nabla f(\boldsymbol{\theta}) + (\text{Noise})$, of the full gradient $\nabla f(\boldsymbol{\theta})$, where σ^2 is an upper bound of Noise, while "Increasing (resp. Constant) Mini-batch" in the Gradient column means that the optimizer uses a mini-batch gradient $\nabla f_{B_t}(\boldsymbol{\theta}) = \frac{1}{b_t}\sum_{i=1}^{b_t} \nabla f_{\xi_{t,i}}(\boldsymbol{\theta})$ with a batch size b_t such that $b_t \leq b_{t+1}$ (resp. $b_t = b$). "Bounded Gradient" in the Additional Assumption column means that there exists $G > 0$ such that, for all $t \in \mathbb{N}$, $\|\nabla f(\boldsymbol{\theta}_t)\| \leq G$, where $(\boldsymbol{\theta}_t)_{t=0}^{T-1}$ is the sequence generated by the optimizer. "Polyak-Łojasiewicz" in the Additional Assumption column means that there exists $\rho > 0$ such that, for all $t \in \mathbb{N}$, $\|\nabla f(\boldsymbol{\theta}_t)\|^2 \geq 2\rho(f(\boldsymbol{\theta}_t) - f^\star)$, where $f^\star$ is the optimal value of f over $\mathbb{R}^d$. Here, we let $\mathbb{E}\|\nabla f_T\| := \min_{t\in[0:T-1]} \mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|]$. Results (1)–(7) were presented in (1) (Yan et al., 2018, Theorem 1), (2) (Gitman et al., 2019, Theorem 1), (3) (Gitman et al., 2019, Theorem 2), (4) (Mai and Johansson, 2020, Theorem 1), (5) (Yu et al., 2019, Corollary 1), (6) (Liu et al., 2020, Theorem 1), and (7) (Liang et al., 2023, Theorem 4.1).

Optimizer	Gradient	Additional Assumption	Learning Rate η_t	Weight β_t	Convergence Analysis
(1) SUM	Noise	Bounded Gradient	$\eta = O(\frac{1}{\sqrt{T}})$	$\beta_t = \beta$	$\mathbb{E}\|\nabla f_T\| = O(\frac{1}{T^{1/4}})$
(2) QHM	Noise	Bounded Gradient	$\eta_t \to 0$	$\beta_t \to 0$	$\exists(\boldsymbol{\theta}_{t_i}) : \nabla f(\boldsymbol{\theta}_{t_i}) \to 0$
(3) QHM	Noise	Bounded Gradient	$\eta_t \to 0$	$\beta_t \to 1$	$\exists(\boldsymbol{\theta}_{t_i}) : \nabla f(\boldsymbol{\theta}_{t_i}) \to 0$
(4) PSHB	Noise	Bounded Gradient	$\eta = O(\frac{1}{\sqrt{T}})$	$\beta_t = \beta$	$\mathbb{E}\|\nabla f_T\| = O(\frac{1}{T^{1/4}})$
(5) SHB	Noise	———	$\eta = O(\frac{1}{\sqrt{T}})$	$\beta_t = \beta$	$\mathbb{E}\|\nabla f_T\| = O(\frac{1}{T^{1/4}})$
(6) NSHB	Noise	———	$\eta = O(\frac{1}{L})$	$\beta_t = \beta$	$\mathbb{E}\|\nabla f_T\| = O(\sqrt{\frac{1}{T} + \sigma^2})$
(7) SUM	Noise	Polyak-Łojasiewicz	$\eta_t \to 0$	$\beta_t = \beta$	$\mathbb{E}[f(\boldsymbol{\theta}_t)] \to f^\star$
NSHB [Theorem 3.1]	Constant Mini-batch	———	$\eta = O(\frac{1}{L})$	$\beta_t = \beta$	$\mathbb{E}\|\nabla f_T\| = O(\sqrt{\frac{1}{T} + \frac{\sigma^2}{b}})$
SHB [Theorem A.1]]	Constant Mini-batch	———	$\eta = O(\frac{1}{L})$	$\beta_t = \beta$	$\mathbb{E}\|\nabla f_T\| = O(\sqrt{\frac{1}{T} + \frac{\sigma^2}{b}})$
NSHB [Theorem 3.2]	Increasing Mini-batch	———	$\eta = O(\frac{1}{L})$	$\beta_t = \beta$	$\mathbb{E}\|\nabla f_T\| = O(\frac{1}{T^{1/2}})$
SHB [Theorem A.2]]	Increasing Mini-batch	———	$\eta = O(\frac{1}{L})$	$\beta_t = \beta$	$\mathbb{E}\|\nabla f_T\| = O(\frac{1}{T^{1/2}})$

Table 1 summarizes the convergence analyses of SGDM for nonconvex optimization. For example, NSHB ((6) in Table 1) using a constant learning rate $\eta_t = \eta > 0$ and a constant momentum weight $\beta_t = \beta$ satisfies $\min_{t\in[0:T-1]} \mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|] = O(\sqrt{\frac{1}{T} + \sigma^2})$ (Liu et al., 2020, Theorem 1), where T is the number of steps, $\nabla f\colon \mathbb{R}^d \to \mathbb{R}^d$ is the gradient of f, σ^2 is the upper bound of the variance of the stochastic gradient of f, and $\mathbb{E}[X]$ denotes the expectation of a random variable X. In comparison, QHM ((2) in Table 1), which is a generalization of NSHB, using a decaying learning rate η_t and a decaying momentum weight β_t satisfies $\liminf_{t \to +\infty} \|\nabla f(\boldsymbol{\theta}_t)\| = 0$ (Gitman et al., 2019, Theorem 1). As can be seen from these convergence analysis results, the performance of SGDM in finding a stationary point $\boldsymbol{\theta}^\star$ of f (i.e., $\nabla f(\boldsymbol{\theta}^\star) = \mathbf{0}$) depends on the settings of the learning rate η_t and the momentum weight β_t.

Moreover, we would like to emphasize that the setting of the batch size b_t affects the performance of SGDM. Previous results presented in (Shallue et al., 2019; Zhang et al., 2019) numerically showed that, for deep learning optimizers, the number of steps needed to train a DNN is halved for each doubling of the batch size. In (Smith et al., 2018), it was numerically shown that using an enormous batch size leads to a reduction in the number of parameter updates and the model training time. In addition, related research has explored plain SGD, highlighting the broader interest in batch-size strategies beyond SGDM. For instance, the two-scale adaptive (TSA) method (Gao et al., 2022) co-adapts the batch size and step size to obtain exact convergence and favorable sample complexity. The related work has also investigated batch-size growth for Riemannian SGD (Sakai and Iiduka, 2025). While these studies share the idea of using an increasing batch size, their analyses are limited to plain SGD. Therefore, in this paper, we theoretically investigate how the setting of the batch size affects the convergence of SGDM under constant hyperparameters.

1.1. Contribution

In this paper, we focus on mini-batch SGDM with a constant learning rate $\eta > 0$ and constant momentum weight $\beta \in [0, 1)$, which is frequently used to train DNNs.

1. The first theoretical contribution of the paper is to show that an upper bound of $\min_{t \in [0:T-1]} \mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|]$ for mini-batch SGDM using a *constant* batch size b is

$$O\left(\sqrt{\frac{f(\boldsymbol{\theta}_0) - f^\star}{\eta T} + \frac{L\eta\sigma^2}{b}}\right),$$

which implies that mini-batch SGDM does not always minimize the expectation of the full gradient norm of the empirical loss in training a DNN (Table 1; Theorems 3.1 and A.1).

The bias term $\frac{f(\boldsymbol{\theta}_0) - f^\star}{\eta T}$ converges to 0 when $T \to +\infty$. However, the variance term $\frac{L\eta\sigma^2}{b}$ remains a constant positive real number regardless of how large T is. In contrast, using a large batch size b makes the variance term $\frac{L\eta\sigma^2}{b}$ small. Hence, we can expect that the upper bound of $\min_{t \in [0:T-1]} \mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|]$ for mini-batch SGDM with *increasing* batch size converges to 0.

2. The second theoretical contribution is to show that an upper bound of $\min_{t \in [0:T-1]} \mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|]$ for mini-batch SGDM with *increasing* batch size b_t such that b_t is multiplied by $\delta > 1$ every E epochs is

$$O\left(\sqrt{\frac{f(\boldsymbol{\theta}_0) - f^\star}{\eta T} + \frac{L\eta\sigma^2\delta}{(\beta^2\delta - 1)b_0 T}}\right), \tag{1}$$

which implies that mini-batch SGDM minimizes the expectation of the full gradient norm of the empirical loss in the sense of an $O(\frac{1}{\sqrt{T}})$ rate of convergence (Table 1; Theorems 3.2 and A.2).

The previous results reported in (Byrd et al., 2012; Balles et al., 2016; De et al., 2017; Smith et al., 2018; Goyal et al., 2018; Shallue et al., 2019; Zhang et al., 2019) indicated that increasing batch sizes are useful for training DNNs with deep-learning optimizers. However, the existing analyses of SGDM have indicated that knowing only the theoretical performance of mini-batch SGDM with an increasing batch size may be insufficient (Table 1). The paper shows theoretically that SGDM with an increasing batch size converges to stationary points of the empirical loss (Theorems 3.2 and A.2). The previous results in (Yan et al., 2018, Theorem 1), (Mai and Johansson, 2020, Theorem 1), and (Yu et al., 2019, Corollary 1) (Table 1(1), (4), and (5)) showed that SGDM with a constant learning rate $\eta = O(\frac{1}{\sqrt{T}})$ and a constant momentum weight β has convergence rate $O(\frac{1}{T^{1/4}})$. Our results (Theorems 3.2 and A.2) guarantee that, if the batch size increases, then SGDM satisfies $\min_{t \in [0:T-1]} \mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|] = O(\frac{1}{T^{1/2}})$, which is an improvement on the previous convergence rate $O(\frac{1}{T^{1/4}})$.

The result in (1) indicates that the performance of mini-batch SGDM with an increasing batch size b_t depends on δ. Let η and β be fixed (e.g., $\eta = 0.1$ and $\beta = 0.9$). Then, (1) indicates that the larger δ is, the smaller the variance term $\frac{L\eta\sigma^2\delta}{(\beta^2\delta-1)b_0 T}$ becomes (since $\frac{\delta}{\beta^2\delta-1} = \frac{1}{(0.9)^2 - 1/\delta}$ becomes small as δ becomes large). We are interested in verifying whether this theoretical result holds in practice. Our numerical results in Section 4 support the theoretical findings, suggesting that an increasing batch size leads to faster convergence.

We trained ResNet-18 on the CIFAR-100 and Tiny-ImageNet datasets by using not only NSHB and SHB but also baseline optimizers: SGD, Adam (Kingma and Ba, 2015), AdamW (Loshchilov and Hutter, 2019), and RMSprop (Tieleman and Hinton, 2012). The results on Tiny-ImageNet are provided in Appendix A.4. A particularly interesting result in Section 4 is that an increasing batch size benefits Adam in the sense of minimizing $\min_{t \in [0:T-1]} \|\nabla f(\boldsymbol{\theta}_t)\|$ fastest. Hence, in the future, we would like to verify whether Adam with an increasing batch size theoretically has a better convergence rate than SGDM.

3. The third contribution of this paper is to show that an increasing batch size significantly reduces the *computational cost* required to achieve optimization and generalization criteria.

In this paper, we define the computational cost as the stochastic first-order oracle (SFO) complexity, in accordance with prior work that formalized this concept (Sato and Iiduka, 2023; Imaizumi and Iiduka, 2024), where SFO is interpreted as the total number of stochastic gradient evaluations. This perspective resonates with the extensive empirical investigation made by Shallue et al. (Shallue et al., 2019), which systematically examined how the batch size affects training efficiency across diverse neural network settings. Let T be the number of training steps and b the batch size. Then, the total SFO complexity is given by Tb when using a fixed batch size. Our numerical results in Section 4.2 demonstrate that, under realistic GPU memory constraints, SGDM with an increasing batch size requires significantly fewer gradient computations to achieve competitive optimization and generalization performance compared with using a fixed batch size. These results indicate that an increasing batch size is not only theoretically appealing but also practically effective in reducing the computational burden in deep-learning training.

2. Mini-batch SGDM for Empirical Risk Minimization

2.1. Empirical risk minimization

Let $\boldsymbol{\theta} \in \mathbb{R}^d$ be a parameter of a DNN, where $\mathbb{R}^d$ is d-dimensional Euclidean space with inner product $\langle \cdot, \cdot \rangle$ and induced norm $\|\cdot\|$. Let $\mathbb{R}_+ := \{x \in \mathbb{R} \colon x \geq 0\}$ and $\mathbb{R}_{++} := \{x \in \mathbb{R} \colon x > 0\}$. Let $\mathbb{N}$ be the set of natural numbers. Let $S = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$ be the training set, where the data point $\boldsymbol{x}_i$ is associated with a label $\boldsymbol{y}_i$ and $n \in \mathbb{N}$ is the number of training samples. Let $f_i(\cdot) := f(\cdot; (\boldsymbol{x}_i, \boldsymbol{y}_i)) \colon \mathbb{R}^d \to \mathbb{R}_+$ be the loss function corresponding to the i-th labeled training data $(\boldsymbol{x}_i, \boldsymbol{y}_i)$. Empirical risk minimization (ERM) minimizes the empirical loss defined for all $\boldsymbol{\theta} \in \mathbb{R}^d$ as $f(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i \in [n]} f(\boldsymbol{\theta}; (\boldsymbol{x}_i, \boldsymbol{y}_i)) = \frac{1}{n} \sum_{i \in [n]} f_i(\boldsymbol{\theta})$, where $[n] := \{1, 2, \cdots, n\}$.

We assume that the loss functions f_i $(i \in [n])$ satisfy the following conditions.

Assumption 1. *Let n be the number of training samples and let $L_i > 0$ $(i \in [n])$.*

(A1) $f_i \colon \mathbb{R}^d \to \mathbb{R}$ $(i \in [n])$ is differentiable and L_i-smooth (i.e., there exists $L_i > 0$ such that, for all $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d$, $\|\nabla f_i(\boldsymbol{\theta}_1) - \nabla f_i(\boldsymbol{\theta}_2)\| \leq L_i \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|$), $L := \frac{1}{n} \sum_{i \in [n]} L_i$, and $f^\star$ is the minimum value of f over $\mathbb{R}^d$.

(A2) Let ξ be a random variable independent of $\boldsymbol{\theta} \in \mathbb{R}^d$. $\nabla f_\xi \colon \mathbb{R}^d \to \mathbb{R}^d$ is the stochastic gradient of ∇f such that (i) for all $\boldsymbol{\theta} \in \mathbb{R}^d$, $\mathbb{E}_\xi[\nabla f_\xi(\boldsymbol{\theta})] = \nabla f(\boldsymbol{\theta})$ and (ii) there exists $\sigma \geq 0$ such that, for all $\boldsymbol{\theta} \in \mathbb{R}^d$, $\mathbb{V}_\xi[\nabla f_\xi(\boldsymbol{\theta})] = \mathbb{E}_\xi[\|\nabla f_\xi(\boldsymbol{\theta}) - \nabla f(\boldsymbol{\theta})\|^2] \leq \sigma^2$, where $\mathbb{E}_\xi[\cdot]$ denotes the expectation with respect to ξ.

(A3) Let $b \in \mathbb{N}$ such that $b \leq n$ and let $\boldsymbol{\xi} = (\xi_1, \xi_2, \cdots, \xi_b)^\top$ comprise b independent and identically distributed variables that are independent of $\boldsymbol{\theta} \in \mathbb{R}^d$. The full gradient $\nabla f(\boldsymbol{\theta})$ is taken to be the mini-batch gradient at $\boldsymbol{\theta}$ defined by $\nabla f_B(\boldsymbol{\theta}) := \frac{1}{b} \sum_{i=1}^{b} \nabla f_{\xi_i}(\boldsymbol{\theta})$.

2.2. Mini-batch NSHB and mini-batch SHB

Let $\boldsymbol{\theta}_t \in \mathbb{R}^d$ be the t-th approximated parameter of DNN. Then, mini-batch NSHB uses b_t loss functions $f_{\xi_{t,1}}, \cdots, f_{\xi_{t,b_t}}$ randomly chosen from $\{f_1, \cdots, f_n\}$ at each step t, where $\boldsymbol{\xi}_t = (\xi_{t,1}, \cdots, \xi_{t,b_t})^\top$ is independent of $\boldsymbol{\theta}_t$ and b_t is a batch size satisfying $b_t \leq n$. The Mini-batch NSHB optimizer is listed in Algorithm 1.

The simplest optimizer for adding a momentum term (denoted by $\beta \boldsymbol{m}_{t-1}$) to SGD is the stochastic heavy ball (SHB) method (Polyak, 1964), which is provided in PyTorch (Paszke et al., 2019). SHB is defined as follows:

$$\boldsymbol{m}_t = \beta \boldsymbol{m}_{t-1} + \nabla f_{B_t}(\boldsymbol{\theta}_t), \ \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \boldsymbol{m}_t, \tag{2}$$

where $\beta \in [0, 1)$ and $\alpha > 0$. SHB defined by (2) with $\beta = 0$ coincides with SGD. SHB defined by (2) has the form $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \nabla f_{B_t}(\boldsymbol{\theta}_t) + \beta(\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1})$. Meanwhile, Algorithm 1 is called the normalized-SHB (NSHB) optimizer (Gupal and Bazhenov, 1972) and has the form $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta(1 - \beta) \nabla f_{B_t}(\boldsymbol{\theta}_t) + \beta(\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1})$. Hence, NSHB (Algorithm 1) with $\eta = \frac{\alpha}{1-\beta}$ coincides with SHB defined by (2).

Algorithm 1 Mini-batch NSHB optimizer

Require: $\boldsymbol{\theta}_0, \boldsymbol{m}_{-1} := \mathbf{0}$ (initial point), $b_t > 0$ (batch size), $\eta > 0$ (learning rate), $\beta \in [0,1)$ (momentum weight), $T \geq 1$ (steps)

Ensure: $(\boldsymbol{\theta}_t) \subset \mathbb{R}^d$

for $t = 0, 1, \dots, T-1$ **do**

  $\nabla f_{B_t}(\boldsymbol{\theta}_t) := \frac{1}{b_t} \sum_{i=1}^{b_t} \nabla f_{\xi_{t,i}}(\boldsymbol{\theta}_t)$

  $\boldsymbol{m}_t := \beta \boldsymbol{m}_{t-1} + (1-\beta) \nabla f_{B_t}(\boldsymbol{\theta}_t)$

  $\boldsymbol{\theta}_{t+1} := \boldsymbol{\theta}_t - \eta \boldsymbol{m}_t$

end for

3. Mini-batch SGDM with Constant and Increasing Batch Sizes

3.1. Constant batch size scheduler

The following indicates that an upper bound of $\min_{t \in [0:T-1]} \mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|]$ of mini-batch NSHB using a constant batch size

$$[\text{Constant BS}] \ b_t = b \ (t \in \mathbb{N}) \tag{3}$$

does not always converge to 0 (a proof of Theorem 3.1 is given in Appendix A.3).

Theorem 3.1 (Upper bound of $\min_t \mathbb{E}\|\nabla f(\boldsymbol{\theta}_t)\|$ of mini-batch NSHB with Constant BS)**.** *Suppose that Assumption 1 holds and consider the sequence $(\boldsymbol{\theta}_t)$ generated by Algorithm 1 with a momentum weight $\beta \in (0,1)$, a constant learning rate $\eta > 0$ such that $\eta \leq \frac{1-\beta}{2\sqrt{2}\sqrt{\beta+\beta^2}L}$, and Constant BS defined by (3), where $L := \frac{1}{n}\sum_{i \in [n]} L_i$ and $f^\star$ is the minimum value of f over $\mathbb{R}^d$ (see (A1)). Then, for all $T \geq 1$,*

$$\min_{t \in [0:T-1]} \mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|^2] \leq \frac{2(f(\boldsymbol{\theta}_0) - f^\star)}{\eta T} + \frac{L\eta\sigma^2}{b}\left\{\frac{3\beta^2+\beta}{2(1+\beta)} + 1\right\},$$

that is,

$$\min_{t \in [0:T-1]} \mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|] = O\left(\sqrt{\frac{1}{T} + \frac{\sigma^2}{b}}\right).$$

From the discussion in Section 2.2, we find that NSHB (Algorithm 1) with $\eta = \frac{\alpha}{1-\beta}$ coincides with SHB defined by (2). The theoretical results for SHB can also be derived accordingly. Appendices A.2 and A.3 contain a detailed analysis of SHB.

3.2. Increasing batch size scheduler

We consider an increasing batch size b_t such that

$$b_t \leq b_{t+1} \ (t \in \mathbb{N}).$$

An example of b_t (Smith et al., 2018; Umeda and Iiduka, 2025) is, for all $m \in [0:M]$ and all $t \in S_m = \mathbb{N} \cap [\sum_{k=0}^{m-1} K_k E_k, \sum_{k=0}^{m} K_k E_k)$ ($S_0 := \mathbb{N} \cap [0, K_0 E_0)$),

$$[\text{Exponential Growth BS}] \ b_t = \delta^{m\left\lceil \frac{t}{\sum_{k=0}^{m} K_k E_k} \right\rceil} b_0, \tag{4}$$

where $\delta > 1$, and E_m and K_m are the numbers of, respectively, epochs and steps per epoch when the batch size is $\delta^m b_0$. For example, the exponential growth batch size defined by (4) with $\delta = 2$ makes the batch size double every E_m epochs. We may modify the parameters a and δ to a_t and δ_t monotone increasing with t. The total number of steps for the batch size to increase M times is $T = \sum_{m=0}^{M} K_m E_m$.

The following is a convergence analysis of Algorithm 1 with increasing batch sizes.

Theorem 3.2 (Convergence of mini-batch NSHB with Exponential Growth BS)**.** *Suppose that Assumption 1 holds and consider the sequence $(\boldsymbol{\theta}_t)$ generated by Algorithm 1 with a momentum weight $\beta \in (0,1)$, a constant learning rate $\eta > 0$ such that*

$$\eta \leq \max\left\{ \frac{1-\beta}{2\sqrt{2}\sqrt{\beta+\beta^2}L}, \frac{(1-\beta)^2}{(5\beta^2 - 6\beta + 5)L} \right\}, \tag{5}$$

and Exponential Growth BS as defined by (4) with $\delta > 1$ and $\beta^2\delta > 1$. Then, for all $T \geq 1$,

$$\min_{t\in[0:T-1]} \mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|^2] \leq \frac{2(f(\boldsymbol{\theta}_0) - f^\star)}{\eta T} + \frac{2L\eta\sigma^2 K_{\max} E_{\max}\delta}{(\beta^2\delta - 1)b_0 T}\left(\frac{\beta^2}{1-\beta^2} - \frac{1}{\delta - 1}\right),$$

where $K_{\max} := \max\{K_m \colon m \in [0:M]\}$ and $E_{\max} := \max\{E_m \colon m \in [0:M]\}$, that is,

$$\min_{t\in[0:T-1]} \mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|] = O\left(\frac{1}{\sqrt{T}}\right).$$

From the discussion in Section 2.2 indicating that NSHB (Algorithm 1) with $\eta = \frac{\alpha}{1-\beta}$ coincides with SHB defined by (2), Theorem 3.2 leads to the following convergence rate of SHB defined by (2) with an increasing batch size.

Here, we sketch a proof of Theorem 3.2 (a detailed proof is given in Appendix A.2).

1. First, we show that $\frac{\sigma^2}{b_t}$ is an upper bound of the variance of $\nabla f_{B_t}(\boldsymbol{\theta}_t)$ (Proposition A.1) and that $(1-\beta)^2\sigma^2 \sum_{i=0}^{t} \frac{\beta^{2(t-i)}}{b_i}$ is an upper bound of the variance of $\boldsymbol{m}_t = (1-\beta)\sum_{i=0}^{t} \beta^{t-i}\nabla f_{B_i}(\boldsymbol{\theta}_i)$ (Lemma A.1) by using the idea underlying the proof of (Liu et al., 2020, Lemma 1).

2. Next, we show that an auxiliary point $\boldsymbol{z}_t = \frac{1}{1-\beta}\boldsymbol{\theta}_t - \frac{\beta}{1-\beta}\boldsymbol{\theta}_{t-1}$ ($t \geq 1$), which is used to analyze SGDM (Yan et al., 2018; Yu et al., 2019; Liu et al., 2020), satisfies $\mathbb{E}_{\boldsymbol{\xi}_t}[f(\boldsymbol{z}_{t+1})] \leq f(\boldsymbol{z}_t) - \eta \underbrace{\mathbb{E}_{\boldsymbol{\xi}_t}[\langle \nabla f(\boldsymbol{z}_t), \nabla f_{B_t}(\boldsymbol{\theta}_t)\rangle]}_{X_t} + \frac{L\eta^2}{2} \underbrace{\mathbb{E}_{\boldsymbol{\xi}_t}[\|\nabla f_{B_t}(\boldsymbol{\theta}_t)\|^2]}_{Y_t}$ by using the descent lemma (see (7)). Then, using the Cauchy–Schwarz inequality, Young's inequality, and the upper bound of the variance of $\boldsymbol{m}_t$ (Lemma A.1), we arrive at an an upper bound on $-\eta\mathbb{E}[X_t]$ (see (12)). As well, we find an upper bound on $\mathbb{E}[Y_t]$ by using the upper bound $\frac{\sigma^2}{b_t}$ of the variance of $\nabla f_{B_t}(\boldsymbol{\theta}_t)$ (see Lemma A.2 for details on the upper bounds of $-\eta\mathbb{E}[X_t]$ and $\mathbb{E}[Y_t]$).

3. After that, we define the Lyapunov function L_t by $L_t = f(\boldsymbol{z}_t) - f^\star + \sum_{i=1}^{t-1} c_i \|\boldsymbol{\theta}_{t+1-i} - \boldsymbol{\theta}_{t-i}\|^2$, where c_i is defined as in Lemma A.3. Using the above upper bounds of $-\eta\mathbb{E}[X_t]$ and $\mathbb{E}[Y_t]$, we find that $\mathbb{E}[L_{t+1} - L_t] \leq -D\mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|^2] + U_t$ (Lemma A.3), where $D \in \mathbb{R}$ depends on η, β, and c_1, and $U_t > 0$ depends on σ^2, b_t, and c_1.

4. Then, setting η such that it satisfies (5) (see also Appendix A.7) leads to the finding that $D \geq \frac{\eta}{2} > 0$ and $U_t \leq L\eta^2\sigma^2 \sum_{i=0}^{t} \frac{\beta^{2(t-i)}}{b_i}$ (see (19) and (20)). As a result, we have that
$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|^2] \leq \frac{2L_0}{\eta T} + \frac{2L\eta\sigma^2}{T}\sum_{t=0}^{T-1}\sum_{i=0}^{t}\frac{\beta^{2(t-i)}}{b_i}$$
(see Lemma A.4). Finally, using(4) leads to the assertion of Theorem 3.2.

3.3. Setting of hyperparameter δ in Exponential Growth BS (4)

Let η and β be fixed in Algorithm 1 (e.g., $\eta = 0.1$ and $\beta = 0.9$). Then, Theorems 3.2 and A.2 indicate that $O(\sqrt{\frac{f(\boldsymbol{\theta}_0)-f^\star}{\eta T} + \frac{L\eta\sigma^2\delta}{(\beta^2\delta-1)b_0 T}})$ is an upper bound of $\min_{t\in[0:T-1]}\mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|]$ for each of mini-batch NSHB and mini-batch SHB with exponential growth BS (4), which implies that the larger δ is, the smaller the variance term $\frac{L\eta\sigma^2\delta}{(\beta^2\delta-1)b_0 T}$ becomes (since $\frac{\delta}{\beta^2\delta-1} = \frac{1}{(0.9)^2-1/\delta}$ becomes small as δ becomes large). In Section 4, we verify whether this theoretical result holds in practice.

3.4. Comparison of our convergence results with previous ones

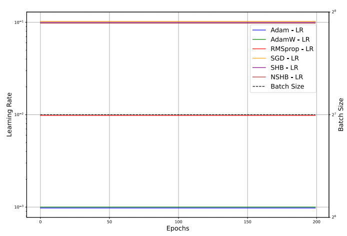
Let us compare Theorems 3.1, 3.2, A.1 and A.2 with the previous results listed in Table 1. Theorem 1 in Liu et al. (2020) ((6) in Table 1) indicated that NSHB using a constant learning rate $\eta = O(\frac{1}{L})$ and a constant momentum weight β satisfies $\min_{t\in[0:T-1]}\mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|] = O(\sqrt{\frac{1}{T}+\sigma^2})$. Since the upper bound $O(\sqrt{\frac{1}{T}+\sigma^2})$ converges to $O(\sigma) > 0$ when $T \to +\infty$, NSHB in this case does not always converge to stationary points of f. The result in Liu et al. (2020) coincides with Theorem 3.1 indicating that NSHB has $\min_{t\in[0:T-1]}\mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|] = O(\sqrt{\frac{1}{T}+\frac{\sigma^2}{b}})$ in the sense that NSHB using a constant learning rate and momentum weight does not converge to stationary points of f[1]. Corollary 1 in Yu et al. (2019) ((5) in Table 1) indicated that SHB using constant $\eta = O(\frac{1}{\sqrt{T}})$ and constant momentum weight β satisfies $\min_{t\in[0:T-1]}\mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|] = O(\frac{1}{T^{1/4}})$. $\eta = O(\frac{1}{\sqrt{T}})$ is needed in order set the number of steps T before implementing SHB. Since T is fixed, we cannot diverge it; that is, the upper bound $O(\frac{1}{T^{1/4}})$ for SHB is a fixed positive constant and does not converge to 0. Meanwhile, from Theorem 3.1, it follows that SHB with a constant learning rate $\eta = O(\frac{1}{L})$ and a constant momentum weight also satisfies $\min_{t\in[0:T-1]}\mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|] = O(\sqrt{\frac{1}{T}+\frac{\sigma^2}{b}})$. Hence, Theorem 3.1 coincides with the result in Corollary 1 in Yu et al. (2019) in the sense that SHB with a constant learning rate does not always converge to stationary points of f.

Theorems 1 and 2 in Gitman et al. (2019) ((2) and (3) in Table 1) indicated that QHM, which is a generalization of NSHB, using a decaying learning rate η_t and a decaying momentum weight β_t or an increasing momentum weight β_t satisfies $\liminf_{t\to+\infty}\|\nabla f(\boldsymbol{\theta}_t)\| = 0$. Our results in the form of Theorems 3.2 and A.2 guarantee the convergence of NSHB and SHB with constant learning rate $\eta = O(\frac{1}{L})$, constant momentum weight β, and an increasing batch size b_t in the sense of $\min_{t\in[0:T-1]}\mathbb{E}[\|\nabla f(\boldsymbol{\theta}_t)\|] = O(\frac{1}{\sqrt{T}})$.

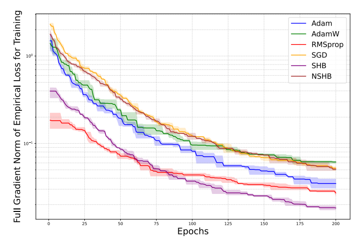
1. (Liu et al., 2020, Theorem 3) proved convergence of multistage SGDM. However, since the proof of (Liu et al., 2020, (60), Pages 35 and 36) might not hold for $\beta_i < 1$, the theorem does not apply here.

4. Numerical Results

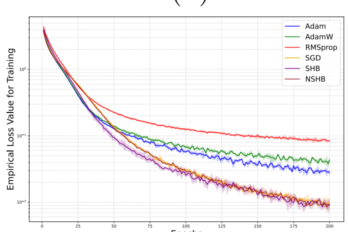
We examined training ResNet-18 on the CIFAR-100 and Tiny-ImageNet datasets using not only NSHB and SHB but also baseline optimizers: SGD, Adam, AdamW, and RMSprop with constant and increasing batch sizes. The results on Tiny-ImageNet are provided in Appendix A.4. We used a computer equipped with NVIDIA A100 80GB and Dual Intel Xeon Silver 4316 2.30GHz, 40 Cores (20 cores per CPU, 2 CPUs). The software environment was Python 3.8.2, PyTorch 2.2.2+cu118, and CUDA 12.2. We set the total number of epochs to $E = 200$ and the constant momentum weight as the default values in PyTorch. The learning rate for Adam and AdamW was set to 10^{-3}, for RMSprop to 10^{-2}, and for SGD, SHB, and NSHB to 10^{-1}; see also Figure 1(a). We should note that the learning rates used in our experiments appear to be theoretically justified within the range implied by Theorem 3.2 (see Appendix A.7 for further details). All results are averaged over three independent trials, and the mean, maximum, and minimum at each epoch are shown.



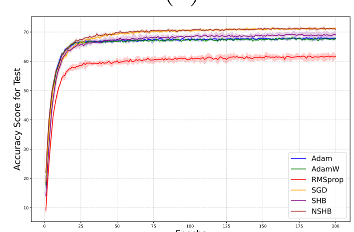
Learning rate and batch size schedules
(a)



Full gradient norm versus epochs
(b)

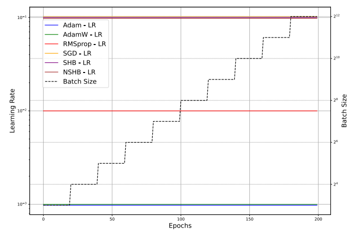


Empirical loss versus epochs
(c)

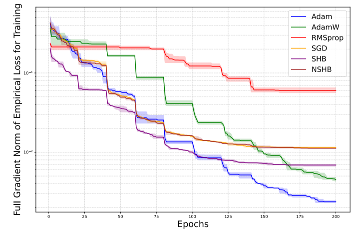


Test accuracy score versus epochs
(d)

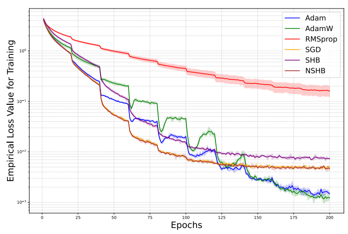
Figure 1: (a) Schedules for each optimizer with constant learning rates and a constant batch size, (b) Full gradient norm of empirical loss for training, (c) Empirical loss value for training, and (d) Accuracy score for test to train ResNet-18 on CIFAR-100 dataset.



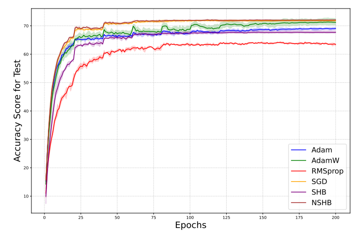
Learning rate and batch size schedules
(a)



Full gradient norm versus epochs
(b)

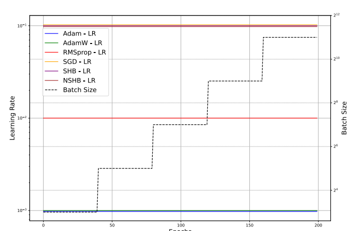


Empirical loss versus epochs
(c)

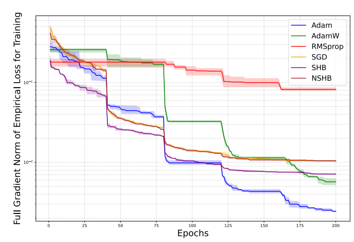


Test accuracy score versus epochs
(d)

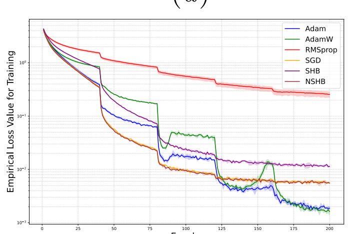
Figure 2: (a) Schedules for each optimizer with constant learning rate and a batch size doubling every 20 epochs, (b) Full gradient norm of empirical loss for training, (c) Empirical loss value for training, and (d) Accuracy score for test to train ResNet-18 on CIFAR-100 dataset.



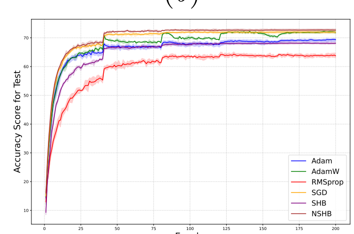
Learning rate and batch size schedules
(a)



Full gradient norm versus epochs
(b)



Empirical loss versus epochs
(c)



Test accuracy score versus epochs
(d)

Figure 3: (a) Schedules for each optimizer with constant learning rates and a batch size quadrupling every 40 epochs, (b) Full gradient norm of empirical loss for training, (c) Empirical loss value for training, and (d) Accuracy score for test to train ResNet-18 on CIFAR-100 dataset.

Let us first consider the learning rate and batch size scheduler in Figure 1(a) with a constant batch size ($b = 2^7$). Figure 1(b) compares the full gradient norm $\min_{e \in [E]} \|\nabla f(\boldsymbol{\theta}_e)\|$ for training for each optimizer and indicates that SHB decreased the full gradient norm quickly. Figures 1(c) and (d) compare the empirical loss $f(\boldsymbol{\theta}_e)$ and the test accuracy score. These figures indicate that SGD, SHB, and NSHB minimized f quickly and had test accuracies of approximately 70 %. Next, let us compare Figure 1 with Figure 2 for when the scheduler uses the same learning rates as in Figure 1(a) and a batch size doubling every 20 epochs with the initial batch size set at $b_0 = 2^3$. Figures 2(b) and (c) both show that using a doubly increasing batch size results in a faster decrease in $\min_{e \in [E]} \|\nabla f(\boldsymbol{\theta}_e)\|$ and $f(\boldsymbol{\theta}_e)$, compared with using a constant batch size as in Figures 1(b) and (c). The numerical results in Figures 1(b) and 2(b) are supported theoretically by Theorems 3.1, 3.2, A.1 and A.2 indicating that NSHB and SHB with increasing batch sizes minimize the gradient norm of f faster than with constant batch sizes. In Figures 1(d) and 2(d), it can be seen that using a doubly increasing batch size leads to improved test accuracy for all optimizers except SHB, compared with using a constant batch size. Earlier, we observed that, with a constant batch size, convergence is slower and accuracy improves more gradually. On the other hand, these results suggest that using an increasing batch size leads to faster convergence and more efficient training. Additionally, the optimizer's performance is better overall when using an increasing batch size.

Now, let us compare Figure 2 ($\delta = 2$) with Figure 3 ($\delta = 4$) when the scheduler uses the same learning rates as in Figure 1(a) and a batch size quadrupling every 40 epochs with the initial batch size set at $b_0 = 2^3$. From Figures 3(b) and (c), it can be observed that the larger the batch size is, the faster the decrease of the full gradient norm $\|\nabla f(\boldsymbol{\theta}_e)\|$ and the empirical loss $f(\boldsymbol{\theta}_e)$ become. Specifically, the quadruply increasing batch size ($\delta = 4$; Figure 3) decreases the full gradient norm $\|\nabla f(\boldsymbol{\theta}_e)\|$ and the empirical loss $f(\boldsymbol{\theta}_e)$ more rapidly than the doubly increasing batch size ($\delta = 2$; Figure 2). Figures 2(d) and 3(d) indicate that SGD and NSHB had test accuracies greater than 70 %, which implies that, for SGD and NSHB, using an increasing batch size would improve generalization more than using a constant batch size (Figure 1(d)).

4.1. Discussion and future work

Fast convergence of Adam: A particularly interesting result in Figures 2–3 is that an increasing batch size is applicable for Adam in the sense of it helping to minimize the full gradient norm of f fastest. Hence, we can expect that Adam with an increasing batch size has a convergence rate better than the $O(\frac{1}{\sqrt{T}})$ convergence rate of NSHB and SHB in Theorems 3.2 and A.2. In the future, we should verify that this result holds theoretically.

Full gradient norm and training loss versus test accuracy: As promised in Theorems 3.1 and 3.2, NSHB with increasing batch sizes ($\delta = 2, 4$) minimized the full gradient norm of f faster than with a constant batch size (Figures 1(b), 2(b), and 3(b)). As a result, NSHB with an increasing batch size ($\delta = 2, 4$) minimized the training loss f (Figures 1(c), 2(c), and 3(c)) and had higher test accuracies than with a constant batch size (Figures 1(d), 2(d), and 3(d)). Moreover, Figures 1–3 indicate that AdamW had almost the same trend. Although Adam and AdamW with increasing batch sizes both minimized f quickly, their test accuracies were different (Figure 3(d)). Here, we have the following insights:

(1) An increasing batch size quickly minimizes the full gradient norm of the training loss in both theory and practice. In particular, SGDM with an increasing batch size converges to stationary points of the training loss, as promised in our theoretical results.
(2) Optimal increasing-batch size-schedulers with which optimizers have high test accuracies should be discussed. Specifically, we need to find the optimal E_m and δ with which SGDM and adaptive methods (e.g., Adam and AdamW) can improve generalization.

4.2. Computational cost evaluation

We evaluated the efficiency of fixed and increasing batch-size schedules in terms of the number of stochastic gradient computations required to achieve specific training goals. To quantify this, we define the SFO complexity, which corresponds to the total number of gradient evaluations. If the batch size is b and the number of training steps is T, then the SFO complexity is given by Tb.

To simulate realistic GPU memory constraints, we capped the maximum batch size at 1024. All experiments were conducted using the CIFAR-100 dataset with the NSHB optimizer, under the same settings as in the other numerical experiments.

We compared the SFO complexity required to (i) reach a gradient norm threshold (e.g., $\|\nabla f(\boldsymbol{\theta}_t)\| < 0.05$) and (ii) achieve 70% test accuracy (see also Appendices A.5 and A.6).

Experimental settings. We considered both fixed and increasing batch size schedules in the gradient norm and test accuracy evaluations. For the gradient norm evaluation, the fixed setting used $b = 8$ and $b = 128$, while the increasing setting started with $b = 8$ (doubling every 20 epochs) and with $b = 128$ (doubling every 50 epochs). For the test accuracy evaluation, the fixed setting also used $b = 8$ and $b = 128$, while the increasing setting started with $b = 8$ (doubling every 20 epochs) and with $b = 128$ (doubling every 25 epochs).

Results.

Table 2: SFO complexity to reach gradient norm threshold ($b = 8$)

Method	$\|\nabla f(\boldsymbol{\theta}_t)\| < 0.1$	$\|\nabla f(\boldsymbol{\theta}_t)\| < 0.05$
Fixed batch size ($b = 8$)	2,750,000	5,250,000
Increasing batch size (initial $b = 8$)	2,050,016	2,500,160

Table 3: SFO complexity to reach gradient norm threshold ($b = 128$)

Method	$\|\nabla f(\boldsymbol{\theta}_t)\| < 0.1$	$\|\nabla f(\boldsymbol{\theta}_t)\| < 0.06$
Fixed batch size ($b = 128$)	5,755,520	9,809,408
Increasing batch size (initial $b = 128$)	2,903,808	5,061,376

Table 4: SFO complexity to reach 70% test accuracy

Method	SFO
Fixed batch size ($b = 8$)	5,250,000
Increasing batch size (initial $b = 8$)	2,050,016
Fixed batch size ($b = 128$)	2,502,400
Increasing batch size (initial $b = 128$)	1,301,376

Discussion. These results show that increasing the batch size significantly reduces the total number of stochastic gradient computations needed to achieve optimization and generalization goals, especially under realistic memory constraints. Our experiments confirm that using an increasing batch size reduces gradient evaluations compared with using a fixed batch size, even when both achieve similar test accuracy and optimization performance. This highlights that larger batch sizes are not just a theoretical convenience but offer clear practical benefits. They provide an efficient way to reduce training costs without compromising generalization, especially in large-scale deep learning under memory and compute constraints.

5. Conclusion

This paper presented convergence analyses of mini-batch SGDM with a constant learning rate and momentum weight. We showed that, unlike prior studies that assume a decaying learning rate to ensure convergence, increasing the batch size under a constant learning rate and momentum not only guarantees convergence but also achieves faster convergence. Numerical experiments supported our theory, demonstrating faster convergence, higher test accuracy, and reduced computational costs compared with a constant batch size. Moreover, our results suggested that increasing batch size can also benefit adaptive methods such as Adam and AdamW. Future work includes extending our analysis to larger-scale datasets and deeper architectures, as well as generalizing the framework beyond the exponential growth schedule to cover polynomial growth and adaptive schemes. These directions of study will further clarify the role of batch size in modern optimization and strengthen the connection between theory and practice.

Acknowledgements

We are sincerely grateful to Program Chairs, Area Chairs, and the four anonymous reviewers for helping us improve the original manuscript. This work was supported by the Japan Society for the Promotion of Science (JSPS) KAKENHI Grant Number 24K14846 awarded to Hideaki Iiduka.

References

Wangpeng An, Haoqian Wang, Qingyun Sun, Jun Xu, Qionghai Dai, and Lei Zhang. A PID controller approach for stochastic optimization of deep networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8522–8531, 2018.

Lukas Balles, Javier Romero, and Philipp Hennig. Coupling adaptive batch sizes with learning rates, 2016. *Thirty-Third Conference on Uncertainty in Artificial Intelligence*, 2017.

Amir Beck. *First-Order Methods in Optimization.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017.

Richard H. Byrd, Gillian M. Chin, Jorge Nocedal, and Yuchen Wu. Sample size selection in optimization methods for machine learning. *Mathematical Programming*, 134(1):127–155, 2012.

Saman Cyrus, Bin Hu, Bryan Van Scoy, and Laurent Lessard. A robust accelerated optimization algorithm for strongly convex functions. In *2018 Annual American Control Conference (ACC)*, pages 1376–1381, 2018.

Soham De, Abhay Yadav, David Jacobs, and Tom Goldstein. Automated inference with adaptive batches. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *PMLR*, pages 1504–1513. PMLR, 2017.

Zhan Gao, Alec Koppel, and Alejandro Ribeiro. Balancing rates and variance via adaptive batch-size for stochastic optimization problems. *IEEE Transactions on Signal Processing*, 70:3693–3708, 2022.

Igor Gitman, Hunter Lang, Pengchuan Zhang, and Lin Xiao. Understanding the role of momentum in stochastic gradient methods. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour, 2018.

A. Gupal and L. T. Bazhenov. A stochastic analog of the conjugate gradient method. *Cybernetics*, 8(1):138–140, 1972.

Kento Imaizumi and Hideaki Iiduka. Iteration and stochastic first-order oracle complexities of stochastic gradient descent using constant and decaying learning rates. *Optimization*, page 1–24, 2024.

Prateek Jain, Sham M. Kakade, Rahul Kidambi, Praneeth Netrapalli, and Aaron Sidford. Accelerating stochastic gradient descent for least squares regression. In *Conference On Learning Theory, COLT 2018*, volume 75 of *PMLR*, pages 545–604. PMLR, 2018.

Rahul Kidambi, Praneeth Netrapalli, Prateek Jain, and Sham M. Kakade. On the insufficiency of existing momentum schemes for stochastic optimization. In *6th International Conference on Learning Representations*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1): 57–95, 2016.

Xuheng Li, Yihe Deng, Jingfeng Wu, Dongruo Zhou, and Quanquan Gu. Risk bounds of accelerated SGD for overparameterized linear regression. In *The Twelfth International Conference on Learning Representations*, 2024.

Yuqing Liang, Jinlan Liu, and Dongpo Xu. Stochastic momentum methods for non-convex learning without bounded assumptions. *Neural Networks*, 165:830–845, 2023.

Yanli Liu, Yuan Gao, and Wotao Yin. An improved analysis of stochastic gradient descent with momentum. In *Advances in Neural Information Processing Systems*, volume 33, pages 18261–18271. Curran Associates, Inc., 2020.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations*, 2019.

Jerry Ma and Denis Yarats. Quasi-hyperbolic momentum and Adam for deep learning. In *International Conference on Learning Representations*, 2019.

Vien Mai and Mikael Johansson. Convergence of a stochastic gradient method with momentum for non-smooth non-convex optimization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *PMLR*, pages 6630–6639. PMLR, 2020.

Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. *Doklady AN USSR*, 269:543–547, 1983.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4:1–17, 1964.

Hiroyuki Sakai and Hideaki Iiduka. A general framework of Riemannian adaptive optimization methods with a convergence analysis. *Transactions on Machine Learning Research*, 2025.

Naoki Sato and Hideaki Iiduka. Existence and estimation of critical batch size for training generative adversarial networks with two time-scale update rule. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *PMLR*, pages 30080–30104. PMLR, 2023.

Christopher J. Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E. Dahl. Measuring the effects of data parallelism on neural network training. *Journal of Machine Learning Research*, 20:1–49, 2019.

Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1139–1147, 2013.

Tijmen Tieleman and Geoffrey Hinton. RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4:26–31, 2012.

Hikaru Umeda and Hideaki Iiduka. Increasing both batch size and learning rate accelerates stochastic gradient descent. *Transactions on Machine Learning Research*, 2025.

Bryan Van Scoy, Randy A. Freeman, and Kevin M. Lynch. The fastest known globally convergent first-order method for minimizing strongly convex functions. *IEEE Control Systems Letters*, 2(1):49–54, 2018.

Aditya Varre and Nicolas Flammarion. Accelerated SGD for non-strongly-convex least squares. In Po-Ling Loh and Maxim Raginsky, editors, *Proceedings of Thirty Fifth Conference on Learning Theory*, volume 178 of *PMLR*, pages 2062–2126. PMLR, 2022.

Yan Yan, Tianbao Yang, Zhe Li, Qihang Lin, and Yi Yang. A unified analysis of stochastic momentum methods for deep learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2955–2961. International Joint Conferences on Artificial Intelligence Organization, 2018.

Hao Yu, Rong Jin, and Sen Yang. On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *PMLR*, pages 7184–7193. PMLR, 2019.

Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George E. Dahl, Christopher J. Shallue, and Roger Grosse. Which algorithmic choices matter at which batch sizes? Insights from a noisy quadratic model. In *Advances in Neural Information Processing Systems*, volume 32, 2019.