Both Asymptotic and Non-Asymptotic Convergence of Quasi-Hyperbolic Momentum using Increasing Batch Size

Kento Imaizumi KENTOIMAIZUMI@GMAIL.COM
Hideaki Iiduka IIDUKA@CS.MEIJI.AC.JP
Meiji University, Japan

Editors: Hung-yi Lee and Tongliang Liu

Abstract

Momentum methods were originally introduced for their superiority to stochastic gradient descent (SGD) in deterministic settings with convex objective functions. However, despite their widespread application to deep neural networks — a representative case of stochastic nonconvex optimization — the theoretical justification for their effectiveness in such settings remains limited. Quasi-hyperbolic momentum (QHM) is an algorithm that generalizes various momentum methods and has been studied to better understand the class of momentum-based algorithms as a whole. In this paper, we provide both asymptotic and non-asymptotic convergence results for mini-batch QHM with an increasing batch size. We show that achieving asymptotic convergence requires either a decaying learning rate or an increasing batch size. Since a decaying learning rate adversely affects non-asymptotic convergence, we demonstrate that using mini-batch QHM with an increasing batch size — without decaying the learning rate — can be a more effective strategy. Our experiments show that even a finite increase in batch size can provide benefits for training neural networks. The code is available at https://github.com/iiduka-researches/qhm_acml25.

Keywords: nonconvex optimization; batch size; learning rate; momentum; asymptotic convergence; non-asymptotic convergence

1. Introduction

Stochastic gradient descent (SGD) (Robbins and Monro, 1951) is the simplest gradient method for minimizing empirical risk minimization problems, and many variants have been aimed at improving its theoretical and empirical performance, including momentum methods and adaptive methods such as Adam (Kingma and Ba, 2015), AdamW (Loshchilov and Hutter, 2019), and RMSProp (Tieleman and Hinton, 2012). Momentum methods are the most common variant; they accelerate convergence to optimal points by adding a momentum term, which is a combination of the current stochastic gradient and the past momentum term. The initial goal of momentum methods was to speed up the convergence of convex optimization in the deterministic setting. In particular, the heavy-ball (HB) method (Polyak, 1964) and Nesterov's accelerated gradient (NAG) (Nesterov, 1983) were shown to accelerate linear convergence over the standard gradient method under such conditions. On the other hand, several numerical experiments (Krizhevsky et al., 2012; Hinton et al., 2012; He et al., 2015; Redmon et al., 2016; Dai et al., 2016; Habibian et al., 2021) have employed stochastic variants — stochastic HB (SHB) and stochastic NAG (SNAG) — in deep learning for stochastic nonconvex optimization (Sutskever et al., 2013). Thus, there is a growing need for theoretical results demonstrating the usefulness of SHB and SNAG in environments that are the exact opposite of convex and deterministic settings.

(Yan et al., 2018) proved non-asymptotic convergence of stochastic unified momentum (SUM) (with an earlier preprint (Yang et al., 2016)), including SGD, SHB, and SNAG, for nonconvex optimization in the stochastic setting. Subsequently, (Yu et al., 2019) applied momentum to a distributed nonconvex setting. However, they imposed the condition $\alpha = \mathcal{O}\left(\frac{1}{\sqrt{K}}\right)$, where α is a constant learning rate and K is the total number of steps, for which it is difficult to ensure asymptotic convergence and apply in practice. (Liu et al., 2020) gave an upper bound on the norm of the gradient of the objective function for NSHB, i.e., $\min_{k\in[0:K-1]} \mathbb{E}[\|\nabla f(\boldsymbol{x}_k)\|^2] = \mathcal{O}\left(\frac{1}{K} + \sigma^2\right)$, in a nonconvex and stochastic environment without assuming boundedness of the gradient, where $\nabla f : \mathbb{R}^d \to \mathbb{R}^d$ is the gradient of f, $\{\boldsymbol{x}_k\}_{k\in\mathbb{N}_0}$ is the sequence generated by NSHB, 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. This analysis involved two major challenges. First, the learning rate α and momentum weight β are constant, and this limits the practical learning rates of schedulers, such as cosine-annealing learning rate (Loshchilov and Hutter, 2017) (Their proof is claimed to cover step decay learning rates, but it no longer applies if the learning rate depends on the iteration k). Second, due to the noise term, their analysis does not guarantee asymptotic convergence, i.e. $\liminf_{k\to\infty} \mathbb{E}[\|\nabla f(\boldsymbol{x}_k)\|] = \mathcal{O}(\sigma)$.

In this paper, we focus on quasi-hyperbolic momentum (QHM) (Ma and Yarats, 2019) (Section 2.2, Algorithm 1). Similar to SUM, QHM becomes various optimizers depending on how γ_k is set up, such as SGD ($\gamma_k = 0$), Normalized-SHB (NSHB) (Gupal and Bazhenov, 1972)($\gamma_k = 1$), and SNAG (α_k, β_k, and γ_k are constant, and $\gamma_k = \beta_k$) (Gitman et al., 2019). SHB and NSHB are interchangeable by adjusting the learning rate scheduler and momentum scheduler (see Appendix A.2). Hence, we believe that an analysis of QHM will contribute to our understanding of momentum methods. (Gitman et al., 2019) proved asymptotic convergence of QHM under the condition of decreasing α_k and $\beta_k\gamma_k$ or under decreasing α_k and γ_k and increasing β_k.

The performance of optimizers used in a stochastic setting undoubtedly depends heavily on the batch size, and momentum methods are no exception (Bollapragada et al., 2024; Sun et al., 2021). Here, there are two main approaches to analyzing the batch size: small-batch learning and large-batch learning. In small-batch learning, the primary motivation is to identify the critical batch size that minimizes the number of stochastic first-order oracle (SFO) calls, which represent the computational cost of stochastic gradient evaluations (Ghadimi et al., 2016; Sato and Iiduka, 2023; Imaizumi and Iiduka, 2024). This theory is based on the result that, although a larger batch size reduces the variance of stochastic gradients — which in turn allows the noise term, a key challenge in non-asymptotic learning, to decrease linearly — the number of steps required to reach an ϵ–approximation does not continue to decrease linearly once the batch size exceeds a certain critical value (Shallue et al., 2019). (Hoffer et al., 2017) indicates that the performance degradation of large-batch training is caused by an insufficient number of steps. In large-batch learning, the primary motivation is to speed up the training process by using the optimizer for large-batch training such as layer-wise adaptive rate scaling (LARS) (You et al., 2017) and the layer-wise adaptive moments optimizer for batch training (LAMB) (You et al., 2020).

The above methods use a constant batch size b that is independent of the iteration number k. (Smith et al., 2018) analyzed the existing results and found that decreasing

the learning rate and increasing the batch size are equivalent in terms of regularization; they argued for increasing the batch size rather than decreasing the learning rate from the perspective of faster training. (Keskar et al., 2017) pointed out two interesting insights about small batches and large batches from the perspective of sharpness of the loss function: (i) When using a large batch, the optimization tends to converge to sharp local minima, which leads to a significant drop in generalization performance. On the other hand, when using a small batch, the optimization avoids sharp local minima and tends to converge to flat local minima. (ii) Training with a small batch in the early stages and switching to a large batch in the later stages leads to convergence to flatter local minima compared with using only a small batch. Although both of these studies claim the benefit of an increasing batch size, they lack theoretical support.

1.1. Contribution

In this paper, we focus on mini-batch QHM (QHM using a batch size) with various learning rate and momentum weight schedulers and prove both asymptotic and non-asymptotic convergence of nonconvex optimization in the stochastic setting.

- To compare constant and increasing batch-size settings, we provide asymptotic and non-asymptotic convergence analyses of mini-batch QHM, NSHB, and SGD in stochastic nonconvex optimization.

Specifically, we find that mini-batch QHM, NSHB, and SGD achieve asymptotic convergence by setting $\sum \alpha_k \to +\infty$, $\beta_k \gamma_k \to 0$, and $\frac{\alpha_k^2}{b_k} \to 0$, where b_k is the batch size. This enables the use of practical learning-rate schedules such as constant learning rate, cosine annealing, and polynomial decay, which would otherwise require a decaying learning rate under a constant batch size. In the analysis of non-asymptotic convergence, mini-batch QHM, NSHB, and SGD using a constant learning rate, step decay momentum weights, and constant batch size are shown to have a convergence rate of $\mathcal{O}\left(\frac{1}{K} + \frac{\sigma^2}{b}\right)$. These results suggest that the inability to achieve asymptotic convergence is due to the noise term. While using a large batch size can reduce the upper bound, it does not guarantee convergence. On the other hand, in the case of using an increasing batch size (De et al., 2017; Smith et al., 2018; Li et al., 2024; Umeda and Iiduka, 2025; Sato and Iiduka, 2025), the convergence rate improves to $\mathcal{O}\left(\frac{1}{K}\right)$, overcoming the challenge of the noise term.

- While a decaying learning rate has the advantage of achieving asymptotic convergence when using a constant batch size, it also becomes a cause of performance degradation in non-asymptotic convergence.

In the case of using an increasing batch size, mini-batch QHM, NSHB, and SGD using a decaying learning rate and step decay momentum weights have a convergence rate of $\mathcal{O}\left(\frac{1}{\sqrt{K}}\right)$, i.e., slower convergence compared with using practical learning rates, including constant learning rates. We also provide asymptotic and non-asymptotic convergence results under different conditions to allow γ_k to be fixed to an arbitrary constant, although this requires a decaying learning rate. The case of a constant batch size requires the condition $\beta_k \to 1$, which further requires the learning rate to decay faster than the increase in β_k,

leading to degraded performance. In the case of an increasing batch size, the condition is relaxed so that β_k can be fixed, but since a decaying learning rate is still used, performance remains suboptimal.

- We empirically compared cases of using an increasing batch size and using a small batch size or large batch size in training deep neural networks with mini-batch QHM, NSHB, and SGD under practical learning rates, including a decaying learning rate. The results support the theoretical claims.

Our theoretical condition $b_k \to +\infty$ is a rather strong assumption. However, empirical results show that we can still benefit from increasing batch sizes, even if the upper bound grows.

2. Preliminaries

2.1. Notation

Let $\mathbb{R}^d$ be a d-dimensional Euclidean space with inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \boldsymbol{x}^\top \boldsymbol{y}$ $(\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d)$ inducing the norm $\|\boldsymbol{x}\| := \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$, and let $\mathbb{N}$ and $\mathbb{N}_0$ be the set of natural numbers and the set of natural numbers including zero, respectively. Define $[n] := \{1, 2, \ldots, n\}$ and $[0 : n] := \{0, 1, \ldots, n\}$ for $n \geq 1$. Let $\{x_k\}_{k \in \mathbb{N}}$ and $\{y_k\}_{k \in \mathbb{N}}$ be positive real sequences and let $x(\epsilon), y(\epsilon) > 0$, where $\epsilon > 0$. $\mathcal{O}$ denotes Landau's symbol; i.e., $y_k = \mathcal{O}(x_k)$ if there exist $c > 0$ and $k_0 \in \mathbb{N}$ such that $y_k \leq c x_k$ for all $k \geq k_0$, and $y(\epsilon) = \mathcal{O}(x(\epsilon))$ if there exists $c > 0$ such that $y(\epsilon) \leq c x(\epsilon)$. Given a parameter $\boldsymbol{x} \in \mathbb{R}^d$ and a data point z in a data domain Z, a machine-learning model provides a prediction whose quality can be measured by a differentiable nonconvex loss function $\ell(\boldsymbol{x}; z)$. We aim to minimize the empirical loss defined for all $\boldsymbol{x} \in \mathbb{R}^d$ by $f(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^n \ell(\boldsymbol{x}; z_i) = \frac{1}{n} \sum_{i=1}^n f_i(\boldsymbol{x})$, where $S = \{z_1, z_2, \ldots, z_n\}$ denotes the training set and $f_i(\cdot) := \ell(\cdot; z_i)$ denotes the loss function corresponding to the i-th training data z_i.

2.2. Conditions and Algorithm

The following are standard conditions.

(C1) $f := \frac{1}{n} \sum_{i=1}^n f_i \colon \mathbb{R}^d \to \mathbb{R}$ is L–smooth, i.e. $\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq L \|\boldsymbol{x} - \boldsymbol{y}\|$. f is bounded below from $f_\star \in \mathbb{R}$.

(C2) Let $(\boldsymbol{x}_k)_{k \in \mathbb{N}} \subset \mathbb{R}^d$ be the sequence generated by mini-batch QHM. For each iteration k, $\mathbb{E}_{\xi_k}[\nabla f_{\xi_k}(\boldsymbol{x}_k)] = \nabla f(\boldsymbol{x}_k)$, where $\xi_0, \xi_1, \ldots$ are independent samples and the random variable ξ_k is independent of $(\boldsymbol{x}_l)_{l=0}^k$. There exists a nonnegative constant σ^2 such that $\mathbb{V}_{\xi_k}[\nabla f_{\xi_k}(\boldsymbol{x}_k)] \leq \sigma^2$, where $\mathbb{V}[X]$ denotes the variance of a random variable X.

(C3) For each iteration k, mini-batch QMH samples a batch B_k of size b_k independently of k and estimates the full gradient ∇f as $\nabla f_{B_k}(\boldsymbol{x}_k) := \frac{1}{b_k} \sum_{i \in [b_k]} \nabla f_{\xi_{k,i}}(\boldsymbol{x}_k)$, where $\xi_{k,i}$ is a random variable generated by the i-th sampling in the k-th iteration.

Algorithm 1 is the mini-batch QHM optimizer under (C1) – (C3).

Algorithm 1 Mini-batch Quasi-Hyperbolic Momentum (mini-batch QHM)

Require: $\boldsymbol{x}_0 \in \mathbb{R}^d$ (initial point), $\alpha_k \in [0, +\infty)$ (learning rate), $\beta_k \in [0, 1), \gamma_k \in [0, 1]$ (momentum weights), $b_k \in \mathbb{N}$ (batch size), $K \in \mathbb{N}$ (steps)

Ensure: $\boldsymbol{x}_K$

1: **for** $k = 0, 1, \ldots, K-1$ **do**
2: $\quad \nabla f_{B_k}(\boldsymbol{x}_k) := \frac{1}{b_k} \sum_{i \in [b_k]} \nabla f_{\xi_{k,i}}(\boldsymbol{x}_k)$
3: $\quad \boldsymbol{d}_k := (1 - \beta_k) \nabla f_{B_k}(\boldsymbol{x}_k) + \beta_k \boldsymbol{d}_{k-1}$
4: $\quad \boldsymbol{m}_k := (1 - \gamma_k) \nabla f_{B_k}(\boldsymbol{x}_k) + \gamma_k \boldsymbol{d}_k$
5: $\quad \boldsymbol{x}_{k+1} := \boldsymbol{x}_k - \alpha_k \boldsymbol{m}_k$
6: **end for**

2.3. Batch Size Schedulers

Let $M \in \mathbb{N}$ be the total number of epochs, and let $E \in \mathbb{N}$ be the number of interval epochs. The batch size is updated per epoch, not per step. So, the batch size schedulers below are indexed by the number of epochs $m \in [0 : M-1]$:

$$\text{[Constant BS] } b'_m = b, \tag{1}$$

$$\text{[Exponential BS] } b'_m = b_0 \delta^{\lfloor \frac{m}{E} \rfloor}, \tag{2}$$

where $b, b_0 \in \mathbb{N}$ and $\delta > 1$. b'_m is related to the batch size indexed by the step k, as follows:

$$\{b_k\}_{k \in \mathbb{N}_0} = \{\underbrace{b'_0, \ldots, b'_0}_{T_0}, \underbrace{b'_1, \ldots, b'_1}_{T_1}, \ldots, \underbrace{b'_{M-1}, \ldots, b'_{M-1}}_{T_{M-1}}, \ldots\},$$

where $T_m = \lceil \frac{n}{b'_m} \rceil$ is the number of steps per epoch m.

2.4. Learning Rate Schedulers

We will examine the following learning rate schedules that are updated each epoch $m \in [0 : M-1]$:

$$\text{[Constant LR] } {\alpha'}_m = \alpha_{\max}, \tag{3}$$

$$\text{[Sqrt-Decaying LR] } {\alpha'}_m = \frac{\alpha_{\max}}{\sqrt{m+1}}, \tag{4}$$

$$\text{[Decaying LR] } {\alpha'}_m = \frac{\alpha_{\max}}{m+1}, \tag{5}$$

$$\text{[Cosine-annealing LR] } {\alpha'}_m = \alpha_{\min} + \frac{\alpha_{\max} - \alpha_{\min}}{2} \left(1 + \cos \frac{m\pi}{M-1}\right), \tag{6}$$

$$\text{[Polynomial Decay LR] } {\alpha'}_m = \alpha_{\min} + (\alpha_{\max} - \alpha_{\min}) \left(1 - \frac{m}{M}\right)^p, \tag{7}$$

where $p > 0$, and $0 \leq \alpha_{\min} \leq \alpha_{\max}$. In the case of [Cosine LR], ${\alpha'}_m = \alpha_{\max}$ if $M = 1$, whereas in the case of [Cosine LR] or [Polynomial LR], the total number of steps $K \in \mathbb{N}$ satisfies $K = \sum_{m=0}^{M-1} T_m$. ${\alpha'}_m$ is related to the learning rate indexed by the step k in the same way as the batch size.

2.5. Momentum Weight Schedulers

We will examine the following schedules for the momentum weights γ_k and β_k in Algorithm 1 that are updated per epoch:

$$\text{[Constant Beta]}\ \beta'_m = \beta_{\max}, \tag{8}$$

$$\text{[Step Decay Beta]}\ \beta'_m = \beta_{\max}\zeta^{\lfloor \frac{m}{E} \rfloor}, \tag{9}$$

$$\text{[Increasing Beta]}\ \beta'_m = 1 - \frac{1-\beta_{\min}}{(m+1)^{3/4}}, \tag{10}$$

$$\text{[Constant Gamma]}\ \gamma'_m = \gamma_{\max}, \tag{11}$$

$$\text{[Step Decay Gamma]}\ \gamma'_m = \gamma_{\max}\lambda^{\lfloor \frac{m}{E} \rfloor}. \tag{12}$$

Here, $\lambda, \zeta \in (0,1)$, $\beta_{\min}, \beta_{\max} \in [0,1)$, and $\gamma_{\max} \in [0,1]$. β'_m and γ'_m are related to the momentum weights indexed by the step k in the same way as the batch size and learning rate.

The reason for updating the scheduler per epoch rather than per step is that the principle of "constant and drop", where the learning rate is initially set large and then dropped every few epochs, is a well-known way of adjusting the learning rate. Algorithms that apply the "constant and drop" idea are referred to as multistage algorithms (Liu et al., 2020; Sun et al., 2021).

3. Asymptotic and Non-Asymptotic Convergence of Mini-batch Quasi-Hyperbolic Momentum

In this section, we show both asymptotic and non-asymptotic convergence of mini-batch QHM with various momentum weights schedulers and argue that an increasing batch size schedule should be used rather than a constant batch size.

In order to prove the following theorems, we will make an additional assumption that is the same as the one made in (Yan et al., 2018) and (Gitman et al., 2019).

Assumption 1 (Boundedness of the gradient) *Let $G > 0$ satisfy, for all $i \in [n]$ and $\boldsymbol{x} \in \mathbb{R}^d$, $\|\nabla f_i(\boldsymbol{x})\| \leq G$.*

The convergence analysis without assuming a bounded gradient not only complicates the proof but also imposes specific conditions on the learning rate and momentum weight. (Liu et al., 2020) indicated non-asymptotic convergence of NSHB without assuming a bounded gradient. However, when defining the auxiliary sequence, which is a key component of their proof (for more details, see Eq. (6) in (Liu et al., 2020)), it is necessary to assume that the learning rate and momentum weight are constant. Although they claim that a step-decay learning rate can also be used, the auxiliary sequence cannot be defined unless the learning rate is independent of the iteration number k. Moreover, we show that Assumption 1 holds whenever the loss function f_i is L_i–smooth and has both upper and lower bounds (see Proposition 9). In addition, our numerical experiments confirm that f_i is bounded in practice (see Appendix A.8).

Theorem 2 (Upper bound of the squared norm of the full gradient using decreasing $\beta_k\gamma_k$) *Suppose that conditions (C1)–(C3) and Assumption 1 hold, and let $\alpha_k \in [\alpha_{\min}, \alpha_{\max}] \subset [0, \frac{2}{L})$, $\overline{\gamma\beta} := \sup_k \gamma_k\beta_k < 1$. Then, the sequence $\{\boldsymbol{x}_k\}_{k\in\mathbb{N}_0}$ generated by Algorithm 1 has the following properties.*

(i) [Convergence Rate] *For all $K \in \mathbb{N}$,*

$$\min_{k\in[0:K-1]} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] \leq \frac{2(f(\boldsymbol{x}_0)-f_\star)}{(1-\overline{\gamma\beta})(2-\alpha_{\max}L)} \underbrace{\frac{1}{\sum_{k=0}^{K-1}\alpha_k}}_{A_K} + \frac{L\sigma^2}{(1-\overline{\gamma\beta})(2-\alpha_{\max}L)} \underbrace{\frac{\sum_{k=0}^{K-1}\alpha_k^2/b_k}{\sum_{k=0}^{K-1}\alpha_k}}_{B_K}$$
$$+ \frac{2\alpha_{\max}(1+\alpha_{\max}L)G^2}{(1-\overline{\gamma\beta})(2-\alpha_{\max}L)} \underbrace{\frac{\sum_{k=0}^{K-1}\gamma_k\beta_k}{\sum_{k=0}^{K-1}\alpha_k}}_{C_K}.$$

(ii) [Convergence] *Additionally, let $\{\alpha_k\}$, $\{\beta_k\}$, $\{\gamma_k\}$, and $\{b_k\}$ satisfy the following conditions:*

$$\sum_{k=0}^{+\infty}\alpha_k = +\infty, \quad \sum_{k=0}^{+\infty}\gamma_k\beta_k < +\infty, \quad \sum_{k=0}^{+\infty}\frac{\alpha_k^2}{b_k} < +\infty.$$

Then,

$$\liminf_{k\to\infty} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|\right] = 0.$$

Theorem 2 asserts that, for mini-batch QHM with decreasing $\gamma_k\beta_k$, achieving non-asymptotic convergence requires either a decreasing learning rate or increasing batch size, and it is based on the result of (Smith et al., 2018) on empirical risk minimization problems. In the case of using [Cosine LR] or [Polynomial LR], α_k is only defined up to $K = \sum_{m=0}^{M-1} T_m$. Hence, we employ the following learning-rate schedulers for Theorem 2(ii):

$$\alpha_k = \begin{cases} \text{[Cosine LR (6)] or [Polynomial Decay (7)]} & \left(k \leq \sum_{m=0}^{M-1} T_m\right) \\ \text{[Sqrt-Decaying LR (4)]} & \left(k > \sum_{m=0}^{M-1} T_m\right). \end{cases}$$

Note that this is introduced for the theoretical analysis and not in the numerical experiments or Theorem 2(i).

The following Corollary 3 considers non-asymptotic convergence in the case of using [Step Decay Beta (9)] and [Step Decay Gamma (12)] in accordance with Theorem 2, and it asserts that mini-batch QHM using [Constant BS (1)] and a learning rate with upper and lower bounds, such as [Constant LR (3)], [Cosine LR (6)], and [Polynomial LR (7)], does not converge to a local minimum because B_K does not converge to 0. While using [Sqrt-Decaying LR (4)] can address the issue of the noise term even with [Constant BS (1)], it slows convergence.

Corollary 3 (Upper bound of A_K, B_K, and C_K in Theorem 2) *When using [Step Decay Beta (9)] and [Step Decay Gamma (12)], A_K, B_K, and C_K in Theorem 2 satisfy,*

for all $K \in \mathbb{N}$,

$$A_K \leq \begin{cases} \frac{1}{\alpha_{\max} K} & [\textit{Constant LR (3)}] \\ \frac{1}{2\alpha_{\max}\left(\sqrt{K+1}-1\right)} & [\textit{Sqrt-Decaying LR (4)}] \\ \frac{2}{(\alpha_{\min}+\alpha_{\max})K} & [\textit{Cosine LR (6)}] \\ \frac{p+1}{(\alpha_{\max}+p\alpha_{\min})K} & [\textit{Polynomial LR (7)}], \end{cases}$$

$$B_K \leq \begin{cases} \frac{\alpha_{\max}}{b} & [\textit{Constant LR (3)}] \\ \frac{\alpha_{\max} T_0(1+\log(K+1))}{2b(\sqrt{K+1}-1)} & [\textit{Sqrt-Decaying LR (4)}] \\ \frac{\alpha_{\max}+\alpha_{\min}}{b} + \frac{T_0(\alpha_{\max}-\alpha_{\min})^2}{2b(\alpha_{\max}+\alpha_{\min})K} & [\textit{Cosine LR (6)}] \\ \frac{(p+1)\alpha_{\max}^2+2p\alpha_{\max}\alpha_{\min}+2p^2\alpha_{\min}^2}{b(2p+1)(\alpha_{\max}+p\alpha_{\min})} + \frac{(p+1)(\alpha_{\max}^2-\alpha_{\min}^2)T_0}{(\alpha_{\max}+p\alpha_{\min})K} & [\textit{Polynomial LR (7)}] \end{cases} [\textit{Constant BS (1)}],$$

$$B_K \leq \begin{cases} \frac{\alpha_{\max} T_0 E\delta}{b_0(\delta-1)K} & [\textit{Constant LR (3)}] \\ \frac{\alpha_{\max} T_0 E\delta}{2b_0(\delta-1)(\sqrt{K+1}-1)} & [\textit{Sqrt-Decaying LR (4)}] \\ \frac{2\alpha_{\max}^2 T_0 E\delta}{b_0(\delta-1)(\alpha_{\max}+\alpha_{\min})K} & [\textit{Cosine LR(6)}] \\ \frac{\alpha_{\max}^2(p+1)T_0 E\delta}{b_0(\alpha_{\max}+p\alpha_{\min})(\delta-1)K} & [\textit{Polynomial LR(7)}] \end{cases} [\textit{Exponential BS (2)}],$$

$$C_K \leq \begin{cases} \frac{\gamma_{\max}\beta_{\max} T_0 E}{\alpha_{\max}(1-\lambda\zeta)K} & [\textit{Constant LR (3)}] \\ \frac{\gamma_{\max}\beta_{\max} T_0 E}{2\alpha_{\max}(1-\lambda\zeta)(\sqrt{K+1}-1)} & [\textit{Sqrt-Decaying LR (4)}] \\ \frac{\gamma_{\max}\beta_{\max} T_0 E}{(1-\lambda\zeta)(\alpha_{\max}+\alpha_{\min})K} & [\textit{Cosine LR (6)}] \\ \frac{\gamma_{\max}\beta_{\max}(p+1)T_0 E}{(\alpha_{\max}+p\alpha_{\min})(1-\lambda\zeta)K} & [\textit{Polynomial LR (7)}], \end{cases}$$

That is, Algorithm 1 has the following convergence rates: in the case of using [Constant BS (1)],

$$\min_{k\in[0:K-1]} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] \leq \begin{cases} \mathcal{O}\left(\frac{1}{K}+\frac{\sigma^2}{b}\right) & [\textit{Constant LR(3)}][\textit{Cosine LR(6)}][\textit{Polynomial LR(7)}] \\ \mathcal{O}\left(\frac{\log K}{\sqrt{K}}\right) & [\textit{Sqrt-Decaying LR(4)}], \end{cases}$$

and in the case of using [Exponential BS (2)],

$$\min_{k\in[0:K-1]} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] \leq \begin{cases} \mathcal{O}\left(\frac{1}{K}\right) & [\textit{Constant LR(3)}][\textit{Cosine LR(6)}][\textit{Polynomial LR(7)}] \\ \mathcal{O}\left(\frac{1}{\sqrt{K}}\right) & [\textit{Sqrt-Decaying LR(4)}]. \end{cases}$$

Theorem 2 has limited application to mini-batch NSHB under the conditions $\gamma_k = 1$ and either decreasing β_k or decreasing γ_k. In particular, (Sutskever et al., 2013) uses SNAG with an increasing β_k. Thus, Theorem 4 assumes an increasing β_k. See Appendix A.11 for a comparison with adaptive batch size methods.

Theorem 4 (Upper bound of the squared norm of the full gradient using increasing β_k) *Suppose that conditions (C1)–(C3) and Assumption 1 hold and that α_k and β_k satisfy, for all $k \in \mathbb{N}_0$, $\beta_k\left(\frac{\alpha_k}{2}+1\right) \leq 1$. Then, the sequence $\{\boldsymbol{x}_k\}_{k\in\mathbb{N}_0}$ generated by Algorithm 1 has the following properties:*

(i) [Convergence Rate] *For all $K \in \mathbb{N}$,*

$$\min_{k\in[0:K-1]} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] \le 2(f(\boldsymbol{x}_0)-f_\star)\underbrace{\frac{1}{\sum_{k=0}^{K-1}\alpha_k}}_{A_K} + 4\sigma^2 \underbrace{\sum_{k=0}^{K-1}\frac{1-\beta_k}{b_k}\frac{1}{\sum_{k=0}^{K-1}\alpha_k}}_{B'_k}$$
$$+ 5L^2G^2\underbrace{\frac{\sum_{k=0}^{K-1}\alpha_k^2}{\sum_{k=0}^{K-1}\alpha_k}}_{C'_K} + 2L^2G^2\underbrace{\sum_{k=0}^{K-1}\frac{\alpha_k^2}{1-\beta_k}\frac{1}{\sum_{k=0}^{K-1}\alpha_k}}_{D'_K}.$$

(ii) [Convergence] *Additionally, let* $\{\alpha_k\}$ *and* $\{\beta_k\}$ *satisfy the following conditions:*

$$\sum_{k=0}^{+\infty}\alpha_k = +\infty, \quad \sum_{k=0}^{+\infty}\frac{\alpha_k^2}{1-\beta_k} < +\infty, \quad \sum_{k=0}^{+\infty}\frac{1-\beta_k}{b_k} < +\infty.$$

Then,

$$\liminf_{k\to\infty}\mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|\right] = 0.$$

Theorem 4 indicates that, for mini-batch QHM with decreasing α_k, achieving non-asymptotic convergence requires either increasing β_k and decreasing α_k^2 faster than β_k is increased or increasing the batch size and that it does not depend on γ_k. Hence, we can take γ_k to be a constant.

The condition $\beta_k(\frac{\alpha_k}{2}+1)\le 1$ is practical because $\beta_{\max}\le 0.9523\cdots$ should be chosen if $\alpha_{\max}=0.1$ is assumed and $\alpha_{\max}\le 0.1$ should be chosen if $\beta_{\min}=0.5$. As in Corollary 3, Corollary 5 below considers non-asymptotic convergence in the case of using [Decaying LR (5)] or [Sqrt-Decaying LR (4)] in accordance with Theorem 4; it asserts that mini-batch QHM using [Constant BS (1)] and [Constant Beta (8)] does not converge to a local minimum because B_K does not converge to 0.

Corollary 5 (Upper bound of B'_K, C'_K, and D'_K in Theorem 4) *When using [Decaying LR (5)] or [Sqrt-Decaying LR (4)], B'_K, C'_K, and D'_K in Theorem 4 satisfy, for all $K\in\mathbb{N}$,*

$$A'_K \le \begin{cases} \frac{1}{2\alpha_{\max}(\sqrt{K+1}-1)} & [\textit{Sqrt-Decaying LR (4)}]\wedge[\textit{Constant Beta (8)}] \\ \frac{1}{\alpha_{\max}\log(K+1)} & [\textit{Decaying LR (5)}]\wedge[\textit{Increasing Beta (10)}], \end{cases}$$

$$B'_K \le \begin{cases} \begin{cases} \frac{K(1-\beta_{\max})}{b(\sqrt{K+1}-1)} & [\textit{Sqrt-Decaying LR (4)}]\wedge[\textit{Constant Beta (8)}] \\ \frac{T_0\left(1+4(1-\beta_{\min})(K+1)^{1/4}\right)}{b\alpha_{\max}\log(K+1)} & [\textit{Decaying LR (5)}]\wedge[\textit{Increasing Beta (10)}] \end{cases} & [\textit{Constant BS (1)}] \\ \begin{cases} \frac{(1-\beta_{\max})T_0E\delta}{2b_0\alpha_{\max}(\delta-1)(\sqrt{K+1}-1)} & [\textit{Sqrt-Decaying LR (4)}]\wedge[\textit{Constant Beta (8)}] \\ \frac{T_0E\delta}{b_0\alpha_{\max}(\delta-1)\log(K+1)} & [\textit{Decaying LR (5)}]\wedge[\textit{Increasing Beta (10)}] \end{cases} & [\textit{Exponential BS (2)}], \end{cases}$$

$$C'_K \le \begin{cases} \frac{\alpha_{\max}T_0(1+\log(K+1))}{2(\sqrt{K+1}-1)} & [\textit{Sqrt-Decaying LR (4)}] \\ \frac{2\alpha_{\max}T_0}{\log(K+1)} & [\textit{Decaying LR (5)}], \end{cases}$$

$$D'_K \leq \begin{cases} \dfrac{\alpha_{\max}}{(1-\beta_{\max})(\sqrt{K+1}-1)} & [\textit{Sqrt-Decaying LR (4)}] \wedge [\textit{Constant Beta (8)}] \\ \dfrac{2\alpha_{\max} T_0}{(1-\beta_{\min})\log(K+1)} & [\textit{Decaying LR (5)}] \wedge [\textit{Increasing Beta (10)}]. \end{cases}$$

That is, Algorithm 1 has the following convergence rates: in the case of using [Constant BS (1)],

$$\min_{k\in[0:K-1]} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] \leq \begin{cases} \mathcal{O}\left(\sqrt{K}\right) & [\textit{Sqrt-Decaying LR (4)}] \wedge [\textit{Constant Beta (8)}] \\ \mathcal{O}\left(\frac{1}{\log K}\right) & [\textit{Decaying LR (5)}] \wedge [\textit{Increasing Beta (10)}], \end{cases}$$

and in the case of [Exponential BS (2)],

$$\min_{k\in[0:K-1]} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] \leq \begin{cases} \mathcal{O}\left(\frac{\log K}{\sqrt{K}}\right) & [\textit{Sqrt-Decaying LR (4)}] \wedge [\textit{Constant Beta (8)}] \\ \mathcal{O}\left(\frac{1}{\log K}\right) & [\textit{Decaying LR (5)}] \wedge [\textit{Increasing Beta (10)}]. \end{cases}$$

Because [Decaying LR (5)] decreases more rapidly than [Sqrt-Decaying LR (4)], its non-asymptotic convergence performance is further degraded. While the use of [Sqrt-Decaying LR (4)] and [Constant Beta (8)] leads to divergence in the case of a fixed batch size, in practice, the performance of the setting described in Theorem 2, employing [Sqrt-Decaying LR (4)], [Constant Beta (8)], and [Constant Gamma (11)], is characterized by $\mathcal{O}(\frac{\log K}{\sqrt{K}})$.

4. Numerical Results

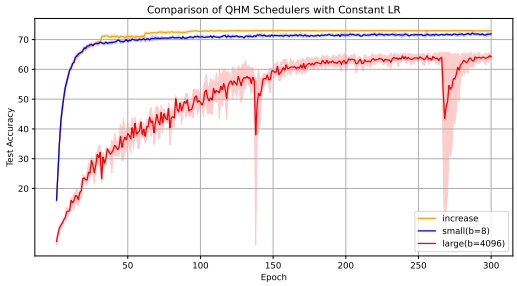
We experimented with mini-batch QHM and NSHB using various lr and momentum weight schedulers to show that they benefit from using an increasing batch size. Specifically, we examined the test accuracy and the minimum of the full gradient norm of the empirical loss in training ResNet–18 (He et al., 2015) on the CIFAR–100 dataset (Krizhevsky, 2009) by using a small batch size ($b = 2^8$), large batch size ($b = 2^{12}$), and increasing batch size. In addition, we performed similar experiments using Vision Transformer-Tiny (ViT-Tiny) as the model. The details can be found in Appendix A.12. Our experiments demonstrated that the scheduler with the increasing batch size is preferable, as neither the small nor large batch size proved effective. The experimental environment was an NVIDIA A100 PCIe 80GB GPU. The software environment was Python 3.11.11, Pytorch 2.2.2, and CUDA 12.2. The code is available at https://github.com/iiduka-researches/qhm_acml25.

We set the total number of epochs to $M = 300$, the interval epochs to $E = 30$, the common ratios in (9) and (12) to $\lambda = \zeta = 2$, the minimum learning rates in (6) and (7) to $\alpha_{\min} = 0$, and the power in (7) to $p = 2$. The other parameters in the lr and momentum weight schedulers were determined by a grid search to be $\alpha_{\max} \in \{0.05, 0.1, 0.25, 0.5\}$, $\beta_{\max}, \beta_{\min} \in \{0.3, 0.5, 0.9\}$, and $\gamma_{\max} \in \{0.2, 0.4, 0.7\}$.

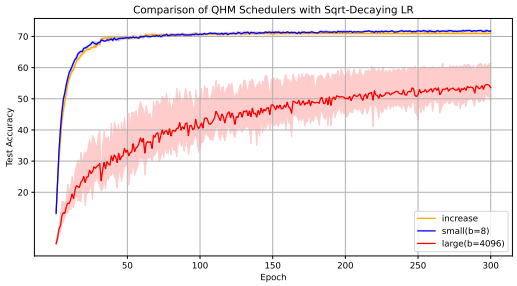
The numerical results reported below are average results over three runs with different random seeds for the initial values. The range between the maximum and minimum values is illustrated using a shaded region.

First, we verified that using increasing batch sizes leads to better performance in mini-batch QHM and NSHB in comparison with using constant batch sizes. In particular, we

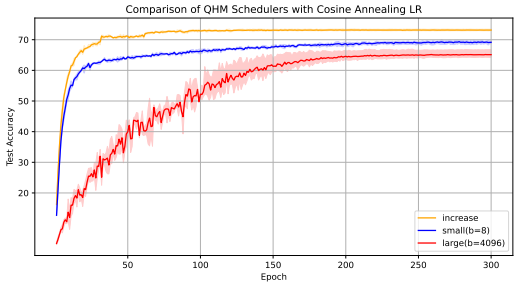
used a batch size doubling (i.e. $\delta = 2$ in (2)) every 30 epochs from an initial batch size $b_0 = 2^3$. In the comparison of increasing batch sizes and constant batch sizes, one must be careful that the total number of steps K differs and that M does not differ. In other words, as the batch size increases, the number of steps required to complete M epochs decreases, which consequently leads to larger gradient norms. Figures 1 and 2 indicate that the mini-batch QHM and NSHB with increasing batch size outperformed those using the small or large constant batch sizes in terms of test accuracy and minimum value of the full gradient norm, depending on the learning rate scheduler. In particular, it can be observed that the performance improvement is attributed to the sharp changes in test accuracy and minimum value of the full gradient norm caused by increasing the batch size at 30 and 60 epochs. In the latter stages of training, the gradual performance improvement observed with the small batch size is hypothesized to result from the larger number of steps compared to the increasing batch size. This indicates that the slightly better results of the small batch size relative to those of the increasing batch size (e.g., Figure 1(b) and Figure 2(a) and (c)) can be explained by the fact that the increasing batch size achieves comparable performance with fewer steps. Consequently, these results indicate that, rather than fixing the batch size to a small or large value from the beginning, gradually increasing it from a small to a large value is beneficial for training neural networks For the comparison with SFO, see Appendix A.9). This finding also applies to mini-batch QHM and NSHB. Moreover, while the theoretical results required the batch size to diverge, our experiments demonstrate that even a finite increase in batch size can yield similar benefits.



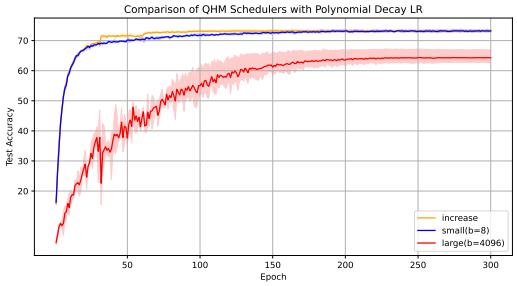
(a) Constant LR



(b) Sqrt-Decaying LR

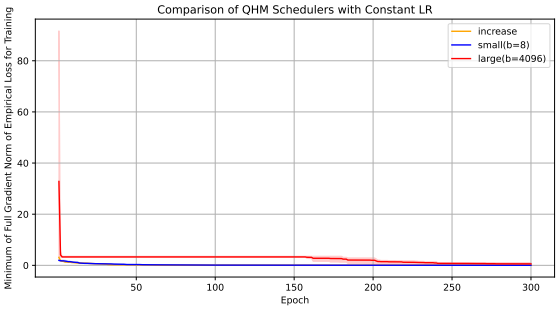


(c) Cosine Annealing LR

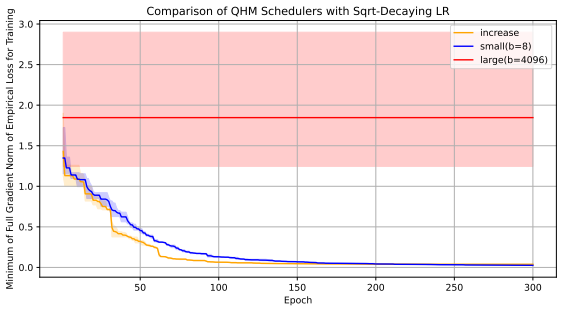


(d) Polynomial Decay LR

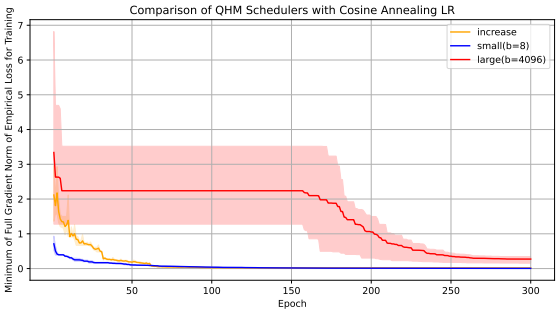
Figure 1: Test accuracy score versus number of epochs for comparison of increasing and constant batch sizes in using mini-batch QHM with various lr-schedulers and step-decay momentum weights to train ResNet-18 on the CIFAR-100 dataset.



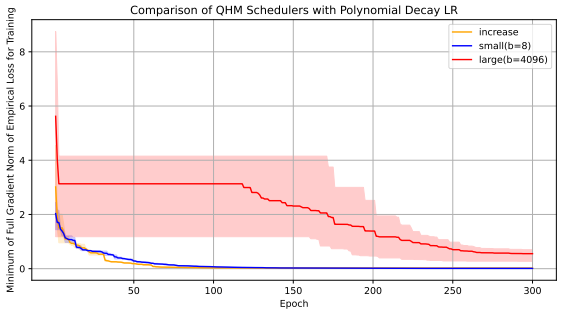
(*a*) Constant LR



(*b*) Sqrt-Decaying LR

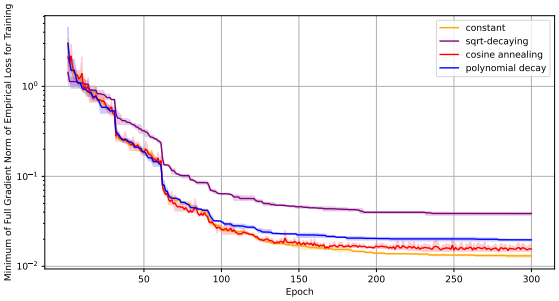


(*c*) Cosine Annealing LR

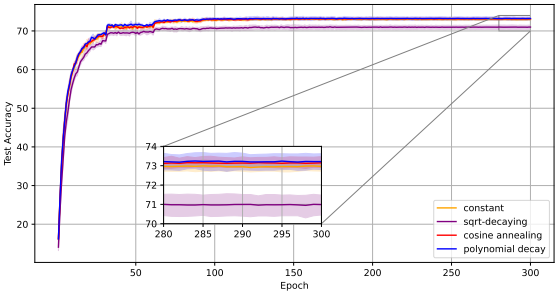


(*d*) Polynomial Decay LR

Figure 2: Minimum of full gradient norm of empirical loss versus number of epochs for comparison of increasing and constant batch sizes in using mini-batch QHM with various lr-schedulers and step-decay momentum weights to train ResNet-18 on the CIFAR-100 dataset.

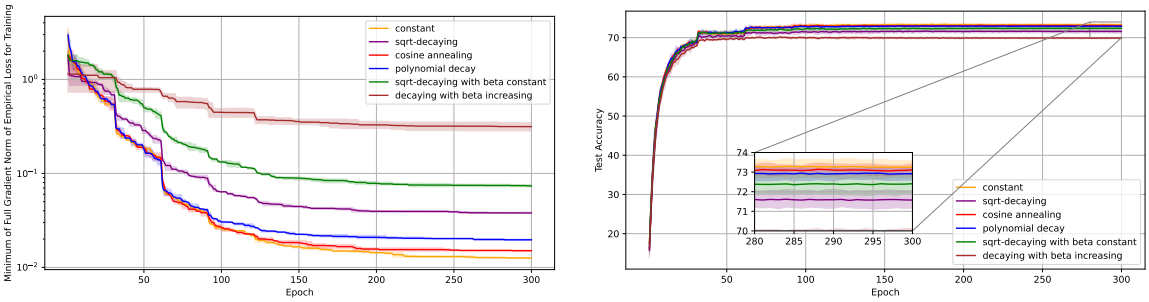


(*a*) Minimum of full gradient norm of empirical loss versus number of epochs



(*b*) Test accuracy score versus number of epochs

Figure 3: Comparison of lr–scheduler in using mini-batch QHM with Step Decay Beta and Gamma to train ResNet-18 on the CIFAR-100 dataset.



(*a*) Minimum of full gradient norm of empirical loss versus number of epochs

(*b*) Test accuracy score versus number of epochs

Figure 4: Comparison of lr and beta–scheduler in using mini-batch NSHB with Constant Gamma ($\gamma_k = 1$) to train ResNet-18 on the CIFAR-100 dataset.

Next, we confirmed that [Sqrt-Decaying LR (4)] and [Decaying LR (5)] negatively affect training compared with the other lr-schedulers. Figures 3 and 4 compare mini-batch QHM and NSHB with increasing batch sizes under various learning rates. Both the test accuracy and the minimum gradient norm indicate that the performance of methods using [Sqrt-Decaying LR (4)] and [Decaying LR (5)] is worse, while [Constant LR (3)], [Cosine LR (6)], and [Polynomial LR (7)] achieved comparable performance to what is predicted by Theorem 2. This holds true regardless of whether mini-batch QHM or mini-batch NSHB is used. In particular, in the case of mini-batch NSHB, although we changed the learning rate decay and the beta scheduler, the performance remained poor. Furthermore, we observed that the overall performances of mini-batch QHM and NSHB are similar. This is because, although mini-batch QHM can reduce the upper bound in Theorem 2 more quickly, they have essentially the same performance in terms of non-asymptotic convergence.

5. Conclusion

This paper proved both asymptotic and non-asymptotic convergence of mini-batch QHM, NSHB, and SGD using learning rate and momentum weight schedulers with an increasing batch size. Since an increasing batch size relaxes the conditions required for asymptotic convergence, it allows us to avoid using a decaying learning rate, which is often detrimental to non-asymptotic convergence performance. While theoretically letting the batch size diverge is a very strong assumption, our numerical experiments demonstrate that even a finite increase toward this limit can still provide benefits. One limitation of this study is the limited number of models and datasets in the experiments. Hence, we should conduct similar experiments with more models and datasets to support our theoretical results.

6. Acknowledgements

We are sincerely grateful to Program Co-Chairs, Area Chairs, and Anonymous Reviewers for helping us improve the original manuscript. This research is partly supported by the computational resources of the DGX A100 named TAIHO at Meiji University. This work

was supported by the Japan Society for the Promotion of Science (JSPS) KAKENHI Grant Number 24K14846 awarded to Hideaki Iiduka.

References

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

Raghu Bollapragada, Tyler Chen, and Rachel Ward. On the fast convergence of minibatch heavy ball momentum. *IMA Journal of Numerical Analysis*, 2024.

Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2016.

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*, Proceedings of Machine Learning Research. PMLR, 2017.

Saeed Ghadimi, Guanghui Lan, and Hongchao Zhang. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 2016.

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*. Curran Associates, Inc., 2019.

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

Amirhossein Habibian, Davide Abati, Taco Cohen, and Babak Ehteshami Bejnordi. Skip-convolutions for efficient video processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 2012.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, 2017.

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

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

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

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012.

Shuaipeng Li, Penghao Zhao, Hailin Zhang, Xingwu Sun, Hao Wu, Dian Jiao, Weiyan Wang, Chengjun Liu, Zheng Fang, Jinbao Xue, Yangyu Tao, Bin Cui, and Di Wang. Surge phenomenon in optimal learning rate and batch size scaling. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2024.

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

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

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.

Yurii Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Proceedings of the USSR Academy of Sciences*, 1983.

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

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 1951.

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*, Proceedings of Machine Learning Research. PMLR, 2023.

Naoki Sato and Hideaki Iiduka. Explicit and implicit graduated optimization in deep neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

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*, 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.

Jianhui Sun, Ying Yang, Guangxu Xun, and Aidong Zhang. A stagewise hyperparameter scheduler to improve generalization. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21. Association for Computing Machinery, 2021.

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*, Proceedings of Machine Learning Research. PMLR, 2013.

Tijmen Tieleman and Geoffrey Hinton. RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 2012.

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

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*. International Joint Conferences on Artificial Intelligence Organization, 2018.

Tianbao Yang, Qihang Lin, and Zhe Li. Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization, 2016.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks, 2017.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In *International Conference on Learning Representations*, 2020.

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*, Proceedings of Machine Learning Research. PMLR, 2019.