Faster Double Adaptive Gradient Methods

Feihu Huang[1,2*], **Yuning Luo**[1]

[1]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
[2]MIIT Key Laboratory of Pattern Analysis and Machine Intelligence, Nanjing, China
huangfeihu2018@gmail.com

Abstract

In this paper, we propose a class of faster double adaptive gradient methods to solve nonconvex finite-sum optimization problems possibly with nonsmooth regularization by simultaneously using adaptive learning rate and adaptive mini-batch size. Specifically, we first propose a double adaptive stochastic gradient method (i.e., 2AdaSGD), and prove that our 2AdaSGD obtains a low stochastic first-order oracle (SFO) complexity for finding a stationary solution under the population smoothness condition. Furthermore, we propose a variance reduced double adaptive stochastic gradient method (i.e., 2AdaSPIDER), and prove that our 2AdaSPIDER obtains an optimal SFO complexity under the average smoothness condition, which is lower than the SFO complexity of the existing double adaptive gradient algorithms. In particular, we introduce a new stochastic gradient mapping to adaptively adjust mini-batch size in our stochastic gradient methods. We conduct some numerical experiments to verify efficiency of our proposed methods.

Introduction

In the era of big data and large model, efficient stochastic optimization algorithms have received widespread attention in machine learning community (Bottou, Curtis, and Nocedal 2018). In this paper, we consider studying efficient stochastic algorithms to solve the following nonconvex optimization problem possibly with nonsmooth regularization,

$$\min_{x\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} f_i(x;\xi^i) + h(x), \tag{1}$$

where function $f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x;\xi^i)$ is smooth but possibly nonconvex, and function $h(x)$ is convex and possibly nonsmooth. Here the samples $\{\xi_i\}_{i=1}^n$ are drawn from an unknown data distribution, and $x \in \mathbb{R}^d$ denotes the parameter vector of model. When n is large, recently the large-scale problem (1) is widely applied in many machine learning tasks such as training deep learning models (You et al. 2019). Stochastic gradient descent (SGD) (Robbins and Monro 1951; Ghadimi and Lan 2013) and its variants (Allen-Zhu and Hazan 2016; Allen-Zhu 2018; Reddi et al. 2016; J Reddi et al. 2016; Fang et al. 2018; Wang et al. 2019; Zhou, Xu, and Gu 2020) have been extensively used to solve these large-scale problems. In fact, the performances of these large-scale models often rely on the hyper-parameters of their training process.

It is well known that one of most critical hyper-parameters is learning rate in optimization algorithms. Recently, many adaptive gradient methods (Duchi, Hazan, and Singer 2011; Kingma and Ba 2014; Zhuang et al. 2020; Xie et al. 2024) with adaptive learning rates have been developed. For example, Adagrad (Duchi, Hazan, and Singer 2011) is the first adaptive gradient method by using the global adaptive learning rate. Adam (Kingma and Ba 2014) algorithm is a momentum-based adaptive gradient method by using the coordinate-wise learning rate, which is a popular optimization operator for training deep learning models. Subsequently, some variants of Adam algorithm (Reddi, Kale, and Kumar 2019; Chen et al. 2019) have been proposed to deal with its divergence under the nonconvex setting. More recently, some accelerated adaptive gradient methods have been proposed based on the variance reduced techniques. Specifically, (Cutkosky and Orabona 2019) proposed an efficient adaptive method (i.e., STORM) via momentum-based variance reduced technique. Subsequently, (Huang, Li, and Huang 2021; Kavis et al. 2022) proposed some variance reduced adaptive methods. Meanwhile, (Yun, Lozano, and Yang 2021) proposed an effective adaptive proximal gradient method for the nonconvex problem with nonsmooth regularization.

Another important hyper-parameter is batch size that also heavily affects generalization performance of the large-scale models (Lau, Liu, and Kolar 2024). Thus, the adaptive batch size is also a good choice like adaptive learning rate (Smith et al. 2018; McCandlish et al. 2018). Recently, some adaptive gradient methods (Devarakonda, Naumov, and Garland 2017; De et al. 2017; Smith et al. 2018; Zhou, Yuan, and Feng 2018; Ji et al. 2020) have been proposed by using adaptive batch size. For example, (De et al. 2017) adapted the adaptive batch size by satisfying certain properties of gradient and variance. (Zhou, Yuan, and Feng 2018) used the adaptive batch size by exponential and polynomial increasing batch size. (Ji et al. 2020) applied the adaptive batch size to the variance reduced methods based on history gradients.

More recently, (Lau, Liu, and Kolar 2024) have been

*Corresponding Author

Algorithm	Reference	ALR	ABS	SFO	NSR
Adam	(Kingma and Ba 2014; Zhang et al. 2022)	$\surd$		$O(\epsilon^{-2})$	
Adam-type	(Chen et al. 2019; Reddi, Kale, and Kumar 2019)	$\surd$		$O(\epsilon^{-2})$	
Adagrad	(Ward, Wu, and Bottou 2020)	$\surd$		$O(\epsilon^{-2})$	
AdaBelief	(Zhuang et al. 2020)	$\surd$		$O(\epsilon^{-2})$	
STORM	(Cutkosky and Orabona 2019)	$\surd$		$O(\epsilon^{-\frac{3}{2}})$	
Super-Adam	(Huang, Li, and Huang 2021)	$\surd$		$O(\epsilon^{-\frac{3}{2}})$	
AbaSPIDER	(Ji et al. 2020)		$\surd$	$O\big(\sqrt{n}\epsilon^{-1} \wedge \epsilon^{-\frac{3}{2}}\big)$	
AdaSPIDER	(Kavis et al. 2022)	$\surd$		$O(n + \sqrt{n}\epsilon^{-1})$	
PROXGEN	(Yun, Lozano, and Yang 2021)	$\surd$		$O(\epsilon^{-2})$	$\surd$
AdAdaGrad	(Lau, Liu, and Kolar 2024)	$\surd$	$\surd$	$O(\epsilon^{-2})$	
2AdaSGD	ours	$\surd$	$\surd$	$O\big(n\epsilon^{-1} \wedge \epsilon^{-2}\big)$	$\surd$
2AdaSPIDER	ours	$\surd$	$\surd$	$O\big(\sqrt{n}\epsilon^{-1} \wedge \epsilon^{-\frac{3}{2}}\big)$	$\surd$

Table 1: Comparison of our methods and the existing adaptive gradient methods for nonconvex optimization. ALR denotes adaptive learning rate. ABS denotes adaptive mini-batch size. SFO denotes stochastic first-order oracle (SFO) complexity of the proposed methods for finding an ϵ-stationary solution of nonconvex optimization (i.e., $\mathbb{E}\|\nabla f(x)\|^2 \leq \epsilon$ or its equivalent form). NSR denotes that the proposed method could work well for the nonsmooth regularization.

begun to study the adaptive gradient method (i.e, AdAdaGrad) by simultaneously using adaptive learning rate and adaptive batch size. However, the proposed AdAdaGrad method (Lau, Liu, and Kolar 2024) suffers from high computation and sample complexity for finding stationary solution (please see Table 1). Thus, to fill this gap, we propose a class of faster double adaptive gradient methods by simultaneously using adaptive learning rate and adaptive batch size. Our main contributions are given as follows:

1) We propose a double adaptive stochastic gradient method (i.e., 2AdaSGD) to solve Problem (1) by introducing stochastic gradient mappings. Moreover, we also propose a variance reduced double adaptive stochastic gradient method (i.e., 2AdaSPIDER) based on the variance reduced technique of SARAH/SPIDER (Nguyen et al. 2017; Fang et al. 2018; Wang et al. 2019).
2) We provide a solid convergence analysis for our methods. Under some mild conditions, we prove that our 2AdaSGD obtains a low stochastic first-order oracle (SFO) complexity of $O\big(n\epsilon^{-1} \wedge \epsilon^{-2}\big)$, and the worst-case SFO complexity of our 2AdaSPIDER method is $O\big(\sqrt{n}\epsilon^{-1} \wedge \epsilon^{-\frac{3}{2}}\big)$, which matches the lower bound complexity for finding an ϵ-stationary solution of nonconvex finite-sum optimization (Fang et al. 2018).
3) We conduct some numerical experiments to verify the efficiency of our proposed methods.

Related Work

In the section, we review some typical adaptive learning rate methods and adaptive batch size methods, respectively.

Adaptive Learning Rate Methods

It is well known that learning rate is one of most critical hyper-parameters in optimization algorithms, which affects performances of the large-scale models. Recently, the adaptive gradient methods (Duchi, Hazan, and Singer 2011; Kingma and Ba 2014; Loshchilov and Hutter 2017; Zhuang et al. 2020; Xie et al. 2024) by using adaptive learning rates have been widely studied in machine learning community. For example, Adam (Kingma and Ba 2014) is one of popular adaptive gradient methods by using a coordinate-wise adaptive learning rate and using simultaneously momentum technique to accelerate algorithm. Thus, the Adam is the default optimization tool for training large-scale model. Subsequently, some variants of Adam algorithm (Reddi, Kale, and Kumar 2019; Chen et al. 2019; Guo et al. 2021) have been proposed to obtain a convergence guarantee under the non-convex setting. By using coordinate-wise adaptive learning rate, Adam algorithm generally has a bad generalization performance in training machine learning models. To improve the generalization performance of Adam, recently some adaptive gradient methods such as AdamW (Loshchilov and Hutter 2017), Padam (Chen et al. 2018) and AdaBelief (Zhuang et al. 2020) have been proposed. Recently, some accelerated adaptive gradient methods (Cutkosky and Orabona 2019; Huang, Li, and Huang 2021; Kavis et al. 2022) have been proposed based on the variance-reduced techniques.

Adaptive Batch Size Methods

In fact, batch size also is one of most critical hyper-parameters in optimization algorithms. Recently, some adaptive batch size methods (Devarakonda, Naumov, and Garland 2017; De et al. 2017; Smith et al. 2018; Zhou, Yuan, and Feng 2018; Ji et al. 2020) have been studied in machine learning community. Specifically, (Zhou, Yuan, and Feng 2018) presented the adaptive gradient method by exponentially or polynomially increasing batch size. (De et al. 2017) used a class of adaptive batch sizes satisfying certain properties of gradient and variance. (Ji et al. 2020) proposed the adaptive variance reduced methods by using the adaptive batch-size based on norm of the old stochastic gradients. More recently, (Lau, Liu, and Kolar 2024) proposed

Algorithm 1: Double Adaptive SGD (2AdaSGD) Algorithm

Input: T, $m \in \mathbb{N}^+$, $\alpha_1 > 0$, $\alpha_2 > 0$, $\eta > 0$ and $\epsilon > 0$;
initialize: $x_0 \in \mathbb{R}^d$ and $\|g_{-1}\| = \cdots = \|g_{-m}\| = \gamma > 0$;
for $t = 0, 1, \ldots, T-1$ **do**
  Compute $\beta_t = \frac{1}{m}\sum_{i=1}^m \|g_{t-i}\|^2$;
  Randomly sample $\mathcal{B}_t$ from $[n]$ without replacement, where $b_t = \min\left\{\lfloor \alpha_1\sigma^2\beta_t^{-1}\rfloor, \lfloor\alpha_2\sigma^2\epsilon^{-1}\rfloor, n\right\}$;
  $v_t = \nabla f_{\mathcal{B}_t}(x_t) = \frac{1}{b_t}\sum_{i\in\mathcal{B}_t}\nabla f_i(x_t;\xi_t^i)$;
  Generate the adaptive matrix $A_t \in \mathbb{R}^{d\times d}$;
  One example of A_t by using update rule: given $a_0 = 0$, $0 < \varrho < 1$, $\rho > 0$, and then compute $a_t = \varrho a_{t-1} + (1-\varrho)\big(\nabla f_{\mathcal{B}_t}(x_t)\big)^2$, $A_t = \text{diag}(\sqrt{a_t} + \rho I_d)$;
  Update $x_{t+1} = \arg\min_{x\in\mathbb{R}^d}\left\{\langle v_t, x\rangle + \frac{1}{2\eta}(x - x_t)^T A_t(x - x_t) + h(x)\right\}$;
  Set $g_t = \frac{x_t - x_{t+1}}{\eta}$;
end for
Output: x_ζ chosen uniformly random from $\{x_t\}_{t=1}^T$.

Algorithm 2: Double Adaptive SPIDER (2AdaSPIDER) Algorithm

Input: S, $m \in \mathbb{N}^+$, $\alpha_1 > 0$, $\alpha_2 > 0$, $\eta > 0$, $b > 0$ and $\epsilon > 0$;
initialize: $x_0 = \tilde{x}^0 \in \mathbb{R}^d$ and $\beta_1 > 0$;
for $s = 1, 2, \ldots, S$ **do**
  $x_0^s = \tilde{x}^{s-1}$;
  Randomly sample $\mathcal{B}_s$ from $[n]$ without replacement, where $B_s = \min\left\{\lfloor \alpha_1\sigma^2\beta_s^{-1}\rfloor, \lfloor\alpha_2\sigma^2\epsilon^{-1}\rfloor, n\right\}$;
  $v_0^s = \nabla f_{\mathcal{B}_s}(\tilde{x}^{s-1}) = \frac{1}{B_s}\sum_{i\in\mathcal{B}_s}\nabla f_i(\tilde{x}^{s-1};\xi_i^s)$;
  Generate the adaptive matrix $A^s \in \mathbb{R}^{d\times d}$;
  One example of A^s by using update rule: given $a^0 = 0$, $0 < \varrho < 1$, $\rho > 0$, and then compute $a^s = \varrho a^{s-1} + (1-\varrho)\big(\nabla f_{\mathcal{B}_s}(x_0^s)\big)^2$, $A^s = \text{diag}(\sqrt{a^s} + \rho I_d)$;
  Update $x_1^s = \arg\min_{x\in\mathbb{R}^d}\left\{\langle v_0^s, x\rangle + \frac{1}{2\eta}(x - x_0^s)^T A^s(x - x_0^s) + h(x)\right\}$;
  Compute $\beta_{s+1} = \|g_0^s\|^2/m$ with $g_0^s = \frac{x_0^s - x_1^s}{\eta}$;
  for $t = 1, 2, \ldots, m-1$ **do**
    Randomly sample $\mathcal{I}_t$ from $[n]$ without replacement and $|\mathcal{I}_t| = b$;
    Compute $v_t^s = \nabla f_{\mathcal{I}_t}(x_t^s) - \nabla f_{\mathcal{I}_t}(x_{t-1}^s) + v_{t-1}^s$;
    Generate the adaptive matrix $A_t^s \in \mathbb{R}^{d\times d}$;
    One example of A_t^s by using update rule: given $a_0^s = 0$, $0 < \varrho < 1$, $\rho > 0$, and then compute $a_t^s = \varrho a_{t-1}^s + (1-\varrho)\big(\nabla f_{\mathcal{I}_t}(x_t^s)\big)^2$, $A_t^s = \text{diag}(\sqrt{a_t^s} + \rho I_d)$;
    Update $x_{t+1}^s = \arg\min_{x\in\mathbb{R}^d}\left\{\langle v_t^s, x\rangle + \frac{1}{2\eta}(x - x_t^s)^T A_t^s(x - x_t^s) + h(x)\right\}$;
    Compute $\beta_{s+1} \leftarrow \beta_{s+1} + \|g_t^s\|^2/m$ with $g_t^s = \frac{x_t^s - x_{t+1}^s}{\eta}$;
  end for
end for
Output: x_ζ chosen uniformly random from $\{\{x_t^s\}_{t=1}^m\}_{s=1}^S$.

the adaptive gradient method by using adaptive batch-size derived from the existing adaptive sampling methods.

Notations

$\lfloor\cdot\rfloor$ denotes the floor function. Let $a \wedge b = \min(a, b)$. Given two vectors x and y, x^r $(r > 0)$ denotes the element-wise power operation, x/y denotes the element-wise division and $\max(x, y)$ denotes the element-wise maximum. I_d denotes a d-dimensional identity matrix. $\|\cdot\|$ denotes the ℓ_2 norm for vectors and spectral norm for matrices. $\langle x, y\rangle$ denotes the inner product of two vectors x and y. Index function $\mathcal{X}(a) = 1$ if the event a occurs and 0 otherwise.

Preliminaries

In this section, we provide some mild conditions for the above problem (1) and the following our methods.

Assumption 1. *Variance of unbiased stochastic gradient is bounded,* i.e., *there exists a constant $\sigma > 0$ such that for all $i = 1, 2, \cdots, n$,*

$$\mathbb{E}[\nabla f(x;\xi^i)] = \nabla f(x),\ \mathbb{E}\|\nabla f(x;\xi^i) - \nabla f(x)\|^2 \le \sigma^2.$$

Assumption 2. *Function $F(x) = f(x) + h(x)$ is bounded from below in $x \in \mathbb{R}^d$,* i.e., *$F^* = \inf_{x\in\mathbb{R}^d} f(x) + h(x)$.*

Assumption 3. *Function $f(x) = \frac{1}{n}\sum_{i=1}^n f_i(x;\xi^i)$ is L-smooth, i.e.,*

$$\|\nabla f(x_1) - \nabla f(x_2)\| \le L\|x_1 - x_2\|,\ \forall x_1, x_2 \in \mathbb{R}^d.$$

Assumption 4. *Each component function $f(x;\xi^i)$ is L-smooth for all $i = 1, 2, \cdots, n$, i.e.,*

$$\|\nabla f(x_1;\xi^i) - \nabla f(x_2;\xi^i)\| \le L\|x_1 - x_2\|,\ \forall x_1, x_2 \in \mathbb{R}^d.$$

Assumption 1 is commonly applied in the existing stochastic optimization (Ghadimi, Lan, and Zhang 2016; Cutkosky and Orabona 2019). Assumption 2 guarantees feasibility of Problem (1). Assumption 3 is widely used in stochastic optimization algorithms (Chen et al. 2019; Zhuang et al. 2020). Assumption 4 is widely used in the variance-reduced algorithms (Fang et al. 2018; Cutkosky and Orabona 2019). According to Assumption 4, we have $\|\nabla f(x) - \nabla f(y)\| = \|\mathbb{E}[\nabla f(x;\xi^i) - \nabla f(y;\xi^i)]\| \le \mathbb{E}\|\nabla f(x;\xi^i) - \nabla f(y;\xi^i)\| \le L\|x - y\|$ for all $x, y \in \mathbb{R}^d$. Thus, Assumption 3 is milder than Assumption 4.

Assumption 5. *Assume the adaptive matrix A_t for all $t \ge 1$ satisfies $A_t \succeq \rho I_d \succ 0$, and $\rho > 0$ denotes a lower bound of the smallest eigenvalue of A_t for all $t \ge 1$.*

Assumption 5 ensures that the adaptive matrices $\{A_t\}_{t\ge1}$ are positive definite, in which their smallest eigenvalues have a lower bound $\rho > 0$, as in (Huang, Li, and Huang 2021; Yun, Lozano, and Yang 2021).

Faster Double Adaptive Gradient Methods

In this section, we propose an efficient double adaptive SGD (i.e.,2AdaSGD) algorithm by simultaneously using adaptive learning rate and adaptive batch size. Then we further propose a variance reduced double adaptive SGD (i.e., 2AdaSPIDER) algorithm based on the variance reduced technique of SPIDER (Nguyen et al. 2017; Fang et al. 2018; Wang et al. 2019).

2AdaSGD Algorithm

Algorithm 1 shows an algorithmic framework of our 2AdaSGD method. In Algorithm 1, we set mini-batch size

$$b_t = \min\big\{\lfloor\alpha_1\sigma^2\beta_t^{-1}\rfloor, \lfloor\alpha_2\sigma^2\epsilon^{-1}\rfloor, n\big\},$$

which depends on average norm of stochastic gradient mappings $\beta_t = \frac{1}{m}\sum_{i=1}^m \|g_{t-i}\|^2$ and the variance of stochastic gradient σ^2. Thus, these stochastic gradient mappings $\{g_{t-i}\}_{i=1}^m$ adaptively adjust mini-batch size in our 2AdaSGD algorithm. Specifically, when the regularization term $h(x) = 0$ and the adaptive matrix $A_t = I_d$, $g_t = \nabla f_{\mathcal{B}_t}(x_t)$ is the stochastic gradient as in (Ji et al. 2020); otherwise, $g_t = \frac{x_t - x_{t+1}}{\eta}$ is the stochastic gradient mapping.

In our Algorithm 1, we also use the adaptive matrix A_t to update variable x. Here the adaptive matrix A_t generally is diagonal, and it represents many adaptive learning rates. For example, we can use the Adam-like adaptive learning rate:

$$a_t = \varrho a_{t-1} + (1-\varrho)\big(\nabla f_{\mathcal{B}_t}(x_t)\big)^2,\ A_t = \text{diag}(\sqrt{a_t} + \rho I_d),$$

where $\varrho \in (0,1)$ and $\rho > 0$ denotes a small positive number. We can also use the AdamW-like adaptive learning rate:

$$\begin{aligned} a_t &= \varrho a_{t-1} + (1-\varrho)\big(\nabla f_{\mathcal{B}_t}(x_t) + \lambda x_t\big)^2,\\ A_t &= \text{diag}(\sqrt{a_t} + \rho I_d), \end{aligned}$$

where $\lambda > 0$ denotes the regularization parameter.

At the line 8 of Algorithm 1, when $h(x) = 0$ and $A_t = \text{diag}(\hat{a}_t)$ with $\hat{a}_t = (\hat{a}_{1,t}, \hat{a}_{2,t}, \cdots, \hat{a}_{d,t}) \in \mathbb{R}^d$, we have

$$\begin{aligned} x_{t+1} &= \arg\min_{x\in\mathbb{R}^d}\big\{\langle v_t, x\rangle + \frac{1}{2\eta}(x-x_t)^T A_t(x-x_t) + h(x)\big\}\\ &= \arg\min_{x\in\mathbb{R}^d}\big\{\langle v_t, x\rangle + \frac{1}{2\eta}(x-x_t)^T \text{diag}(\hat{a}_t)(x-x_t)\big\}\\ &= x_t - \frac{\eta}{\hat{a}_t}\cdot v_t, \end{aligned} \tag{2}$$

where $\cdot$ denotes the coordinate-wise product. In other words, we have $x_{i,t+1} = x_{i,t} - \frac{\eta}{\hat{a}_{i,t}} v_{i,t}$ for all $i \in \{1, 2, \cdots, d\}$.

When $h(x) = \lambda||x||_1$ and $A_t = \text{diag}(\hat{a}_t)$ with $\hat{a}_t = (\hat{a}_{1,t}, \hat{a}_{2,t}, \cdots, \hat{a}_{d,t}) \in \mathbb{R}^d$, we have

$$\begin{aligned} x_{t+1} &= \arg\min_{x\in\mathbb{R}^d}\big\{\langle v_t, x\rangle + \frac{1}{2\eta}(x-x_t)^T A_t(x-x_t) + h(x)\big\}\\ &= \arg\min_{x\in\mathbb{R}^d}\big\{\langle v_t, x\rangle + \frac{1}{2\eta}(x-x_t)^T A_t(x-x_t) + \lambda||x||_1\big\}\\ &= \mathcal{S}\big(x_t - \frac{\eta}{\hat{a}_t}\cdot v_t, \frac{\eta\lambda}{\hat{a}_t}\big), \end{aligned} \tag{3}$$

where $\mathcal{S}(a,\lambda) = \text{sign}(a)\max(|a| - \lambda, 0)$ denotes the soft threshold operator. In other words, we have $x_{i,t+1} = \mathcal{S}\big(x_{i,t} - \frac{\eta}{\hat{a}_{i,t}} v_{i,t}, \frac{\eta\lambda}{\hat{a}_{i,t}}\big)$ for all $i \in \{1, 2, \cdots, d\}$.

2AdaSPIDER Algorithm

Algorithm 2 provides an algorithmic framework of our 2AdaSPIDER method. Algorithm 2 basically follows the above Algorithm 1. In fact, Algorithm 2 uses the same adaptive batch size and adaptive learning rate as in Algorithm 1. While our 2AdaSPIDER algorithm uses the variance reduced stochastic gradient estimator (Nguyen et al. 2017; Fang et al. 2018; Wang et al. 2019): at the outer loop

$$v_0^s = \nabla f_{\mathcal{B}_s}(\tilde{x}^{s-1}) = \frac{1}{B_s}\sum_{i\in\mathcal{B}_s}\nabla f_i(\tilde{x}^{s-1}; \xi_i^s), \tag{4}$$

and at the inner loop

$$v_t^s = \nabla f_{\mathcal{I}_t}(x_t^s) - \nabla f_{\mathcal{I}_t}(x_{t-1}^s) + v_{t-1}^s. \tag{5}$$

Convergence Analysis

In the section, we study the convergence properties of our methods under some mild assumptions. All related proofs are provided in the Appendix. Here we first define a useful gradient mapping as in (Ghadimi, Lan, and Zhang 2016): $\mathcal{G}_t = \frac{1}{\eta}(x_t - \hat{x}_{t+1})$, and the iteration $\hat{x}_{t+1}$ is generated from

$$\begin{aligned} \hat{x}_{t+1} = \arg\min_{x\in\mathbb{R}^d}\Big\{&\langle\nabla f(x_t), x\rangle + \frac{1}{2\eta}(x-x_t)^T A_t(x-x_t)\\ &+ h(x)\Big\}, \end{aligned} \tag{6}$$

where $\eta > 0$. Clearly, when $h(x) = 0$ and $A_t = I_d$, we have $\mathcal{G}_t = \nabla f(x_t)$.

Theorem 1. *Under the above Assumptions 1, 2, 3 and 5, suppose the sequence $\{x_t\}_{t=0}^{T-1}$ be generated from Algorithm 1, and the output x_ζ is uniformly at random chosen from them. Set $\theta_1 = \frac{3\rho}{4} - \frac{\eta L}{2} - \frac{1}{\rho\alpha_1}$ and $\theta_2 = \frac{2}{\rho^2\alpha_1} + 2$, we have*

$$\begin{aligned} \mathbb{E}\|\mathcal{G}_\zeta\|^2 \le{}& \frac{\theta_2(F(x_0) - F^*)}{\theta_1\eta T} + \frac{\theta_2 m\gamma^2}{T\theta_1\rho\alpha_1} + \frac{\theta_2\epsilon}{\theta_1\rho\alpha_2}\\ &+ \frac{2m\gamma^2}{T\rho^2\alpha_1} + \frac{2\epsilon}{\rho^2\alpha_2}. \end{aligned} \tag{7}$$

Remark 1. *Let $\alpha_1 = \alpha_2 = \frac{4}{\rho^2}$ and $0 < \eta \le \frac{\rho}{2L}$, we have $\theta_1 \ge \frac{\rho}{4}$ and $\theta_2 = 10$. Let $m = O(1) < n$ and $\gamma = \frac{1}{\sqrt{m}}$. Then let $T = O(\epsilon^{-1})$, we have $\mathbb{E}\|\mathcal{G}_\zeta\|^2 \le O(\epsilon^{-1})$. The SFO complexity of our 2AdaSGD method is*

$$\begin{aligned} \sum_{t=0}^T b_t &\le \sum_{t=0}^T \min\Big\{\alpha_1\sigma^2\beta_t^{-1}, \alpha_2\sigma^2\epsilon^{-1}, n\Big\}\\ &= \sum_{t=0}^T \min\Big\{\frac{4\sigma^2}{\frac{\rho^2}{m}\sum_{i=1}^m\|g_{t-i}\|^2}, \frac{4\sigma^2}{\rho^2}\epsilon^{-1}, n\Big\}\\ &\le T\min\Big\{\frac{4\sigma^2}{\rho^2}\epsilon^{-1}, n\Big\} = O\big(n\epsilon^{-1}\wedge\epsilon^{-2}\big). \end{aligned} \tag{8}$$

Theorem 2. *Under the above Assumptions 1, 2, 4 and 5, suppose the sequence $\{\{x_t^s\}_{t=0}^{m-1}\}_{s=1}^S$ be generated from Algorithm 2, and the output x_ζ is uniformly at random chosen*

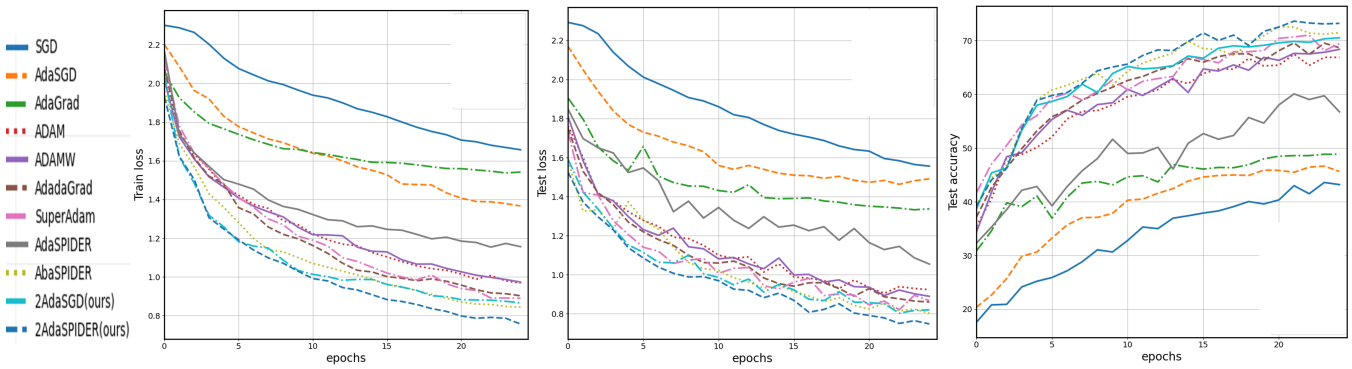

Figure 1: Image Classification on CIFAR-10 dataset without Regularization.

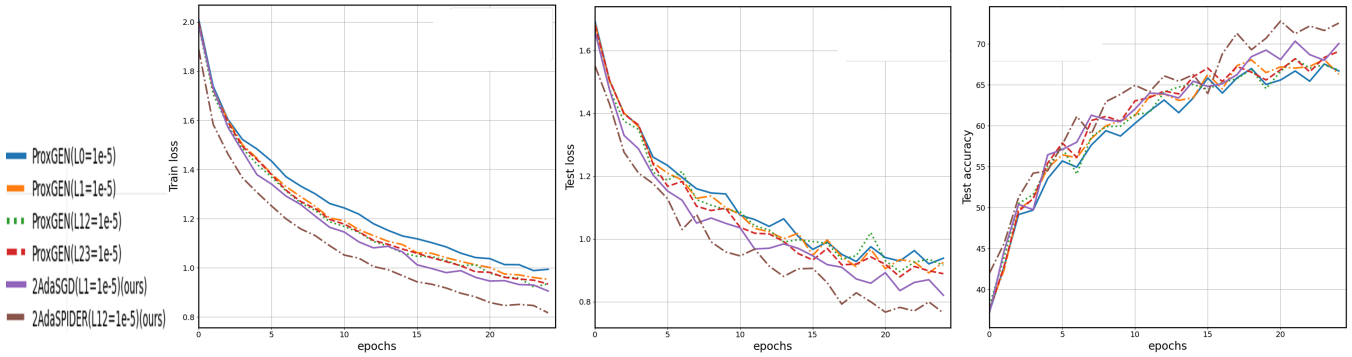

Figure 2: Image Classification on CIFAR-10 dataset with Regularization.

from them. Set $\alpha = \min(\alpha_1/2, \alpha_1/2)$, $\theta_1 = \frac{3\rho}{4} - \frac{\eta L}{2} - \frac{\eta^2 L^2 m}{b\rho} - \frac{1}{2\alpha\rho} > 0$ *and* $\theta_2 = \frac{2m\eta^2 L^2}{\rho^2 b} + \frac{1}{\rho^2\alpha} + 2$, *we have*

$$\mathbb{E}\|\mathcal{G}_\zeta\|^2 \leq \frac{\theta_2(F(x_0) - F^*)}{\theta_1 \eta T} + \frac{\theta_2 \epsilon}{\theta_1 \alpha \rho} + \frac{2\epsilon}{\rho^2 \alpha}, \qquad (9)$$

where $T = Sm$ *and* $x_0^1 = x_0$.

Remark 2. *Let* $\alpha_1 = \alpha_2 = \frac{4}{\rho^2}$ *and* $0 < \eta \leq \frac{\rho}{2L}$, *we have* $\theta_1 \geq \frac{\rho}{8}$ *and* $\theta_2 = \frac{9}{4} + \frac{2\eta^2 L^2}{\rho^2}$. *Further let* $\eta = \frac{\rho}{2L}$, *we have* $\theta_1 = \frac{\rho}{8}$ *and* $\theta_2 = \frac{11}{4}$. *Then let* $b \leq (n \wedge \epsilon^{-1})^{\frac{1}{2}}$, $m = (n \wedge \epsilon^{-1})b^{-1}$, $S = \frac{\epsilon^{-1}}{(n\wedge\epsilon^{-1})^{\frac{1}{2}}}$ *and* $T = O(\epsilon^{-1})$, *we have* $\mathbb{E}\|\mathcal{G}_\zeta\|^2 \leq O(\epsilon^{-1})$. *The SFO complexity of our 2AdaSPIDER is*

$$\begin{aligned}
&\sum_{s=1}^{S} B_t + Smb \\
&\leq \sum_{s=1}^{S} \min\left\{\alpha_1\sigma^2\beta_s^{-1}, \alpha_2\sigma^2\epsilon^{-1}, n\right\} + Smb \\
&\leq \sum_{s=1}^{S} \min\left\{\frac{4\sigma^2}{\frac{\rho^2}{m}\sum_{t=1}^{m}\|g_{t-1}^{s-1}\|^2}, \frac{4\sigma^2}{\rho^2}\epsilon^{-1}, n\right\} \\
&\quad + S(n \wedge \epsilon^{-1}) \\
&\leq O\big(\sqrt{n}\epsilon^{-1} \wedge \epsilon^{-\frac{3}{2}}\big). \qquad (10)
\end{aligned}$$

Numerical Experiments

In this section, we conduct some experiments on image classification and language modeling tasks to verify efficiency of our proposed methods. In the following experiments, we compare our methods with the existing methods provided in Table 1. Since the Super-Adam (Huang, Li, and Huang 2021) extends the STORM (Cutkosky and Orabona 2019), we exclude the STORM in the comparisons. All experiments are run over a machine with Intel(R) Xeon(R) Platinum 8352V CPU and 1 Nvidia RTX 4090 GPU.

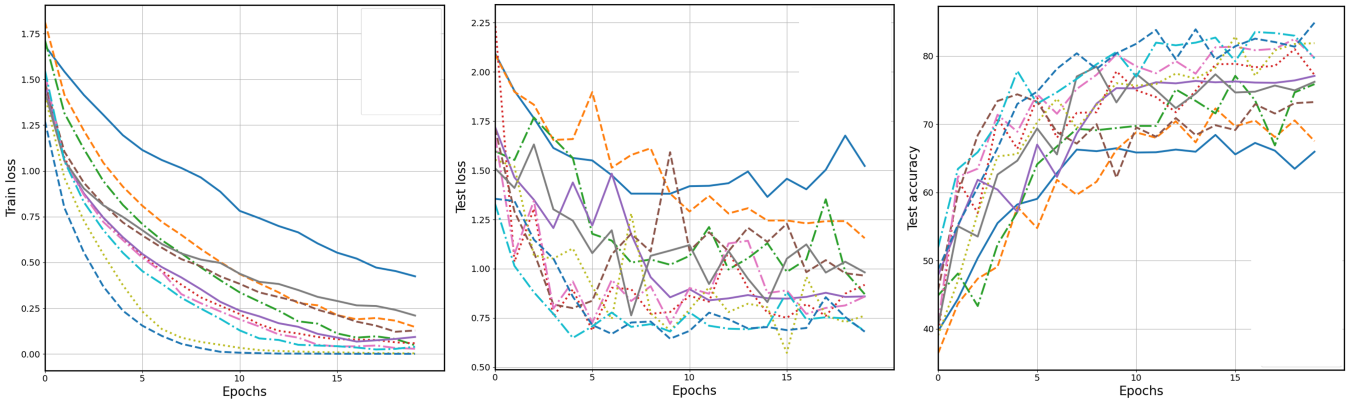

Figure 3: Image Classification on ImageNet dataset without Regularization.

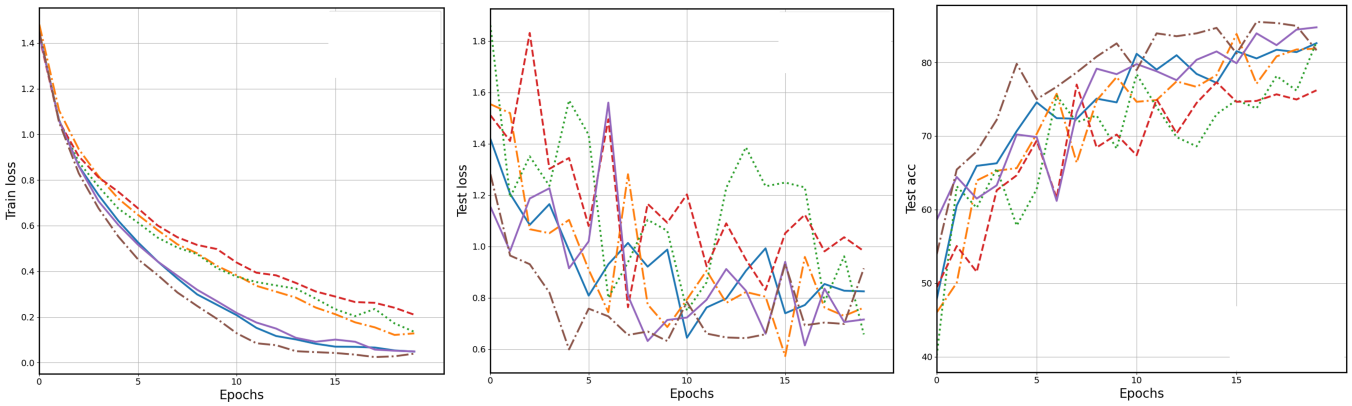

Figure 4: Image Classification on ImageNet dataset with Regularization.

Inputs (d channels)
Conv $d \to 32$, Batchnorm , ReLU Conv $32 \to 64$, Batchnorm , ReLU Conv $64 \to 64$, Batchnorm , ReLU
Max Pool
Linear $1024 \to 512$, ReLU Dropout (0.5) Linear $512 \to C$
Outputs

Table 2: The CNN used in our experiment. C is the number of classes, and d is the number of channels for inputs.

Image Classification Task

In the experiment, we conduct image classification task on CIFAR-10 (Krizhevsky, Hinton et al. 2009) and Imagenet (Deng et al. 2009) datasets, respectively. Given the training sample data $\{x_i, y_i\}_{i=1}^n$ where x_i denotes the features of data and y_i is the label, here we perform training neural networks by solving the following problem:

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} L\big(\Phi(x_i; w), y_i\big) + h(w), \tag{11}$$

where $\Phi(\cdot; w)$ denote the neural network model, and $L(\cdot, \cdot)$ denotes the loss function, and $h(w)$ denotes a regularization. Here w is the parameters in the neural network model, we will learn these parameters $w \in \mathbb{R}^d$. In the experiment, we use the cross-entropy loss to $L(\cdot, \cdot)$. Specifically, we train a 3-layer Convolutional Neural Network (CNN) on the CIFAR-10 dataset and train the ResNet18 (He et al. 2016) on the Imagenet dataset. Here the neural network architecture of the 3-layer CNN is provided in Table 2. For the learning rates and other hyper-parameters, we do grid search and report the best one for each optimizer. We set $\gamma = 10^{-3}$, $m = 50$ in our 2AdaSGD algorithm, and set $\gamma = 10^{-2}$, $b = 64$ in our 2AdaSPIDER algorithm. In other algorithms, we set the basic learning rate as 0.001, and the basic batch-size as 64.

When $h(w) = 0$, i.e., without regularization, from the Figures 1 and 3, our methods basically have better performances than the other methods. At the same time, when

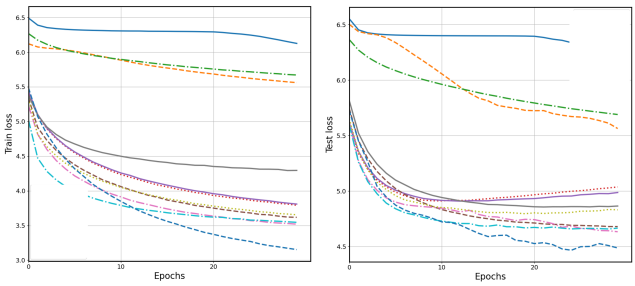

Figure 5: Language Modeling on LSTM without Regularization.

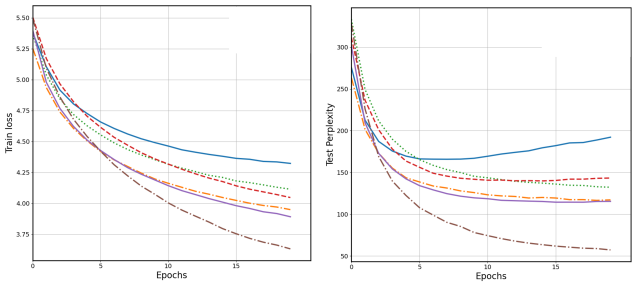

Figure 6: Language Modeling on LSTM with Regularization.

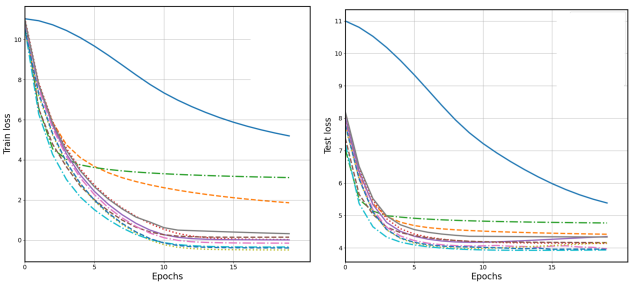

Figure 7: Language Modeling on Transformer without Regularization.

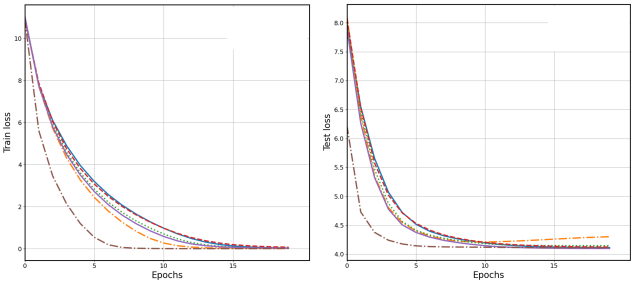

Figure 8: Language Modeling on Transformer with Regularization.

$h(w) = \|w\|_1$, i.e., with L_1-regularization, from the Figures 2 and 4, our methods also basically have better performances than the other methods.

Language Modeling Task

In the experiment, we conduct language modeling task on the Penn-Treebank (Marcus, Santorini, and Marcinkiewicz 1993) and WikiText2 (Merity et al. 2016) datasets, respectively. Specifically, we will train a 2-layer LSTM (Hochreiter and Schmidhuber 1997) on the Penn-Treebank dataset and train a 2-layer Transformer (Vaswani 2017) on the WikiText2 dataset. Given n samples $\{x^i\}_{i=1}^n$, we will learn the language model by solving the following problem:

$$\min_{\theta\in\mathbb{R}^d} -\frac{1}{n}\sum_{i=1}^{n}\sum_{t=1}^{l_i}\log\left(P(x_t^i|x_{1:t-1}^i;\theta)\right) + h(\theta), \qquad (12)$$

where sample x^i includes l_i tokens for all $i \in (1, 2, \cdots, n)$, and $h(\theta)$ denotes a regularization. Here $P(x_t^i|x_{1:t-1}^i;\theta)$ denotes the probability function of the token x_t^i given the tokens $x_{1:t-1}^i$, and $\theta \in \mathbb{R}^d$ is the parameters of the language model. For the learning rates and other hyper-parameters, we do grid search and report the best one for each optimizer. We set $\gamma = 10^{-3}$, $m = 50$ in our 2AdaSGD algorithm, and set $\gamma = 10^{-2}$, $b = 20$ in our 2AdaSPIDER algorithm. In other algorithms, we set the basic learning rate as 0.001, and the basic batch-size as 20.

When $h(\theta) = 0$, i.e., without regularization, from the Figures 5 and 7, our methods basically have better performances than the other methods. At the same time, when $h(\theta) = \|\theta\|_1$, i.e., with L_1-regularization, from the Figures 6 and 8, our methods also basically have better performances than the other methods.

Conclusion

In the paper, we studied double adaptive gradient methods for nonconvex optimization possibly with nonsmooth regularization, and proposed a class of double adaptive stochastic gradient methods by simultaneously using adaptive mini-batch size and adaptive learning rate. Moreover, we studied convergence properties of our methods, and proved that the worst-case SFO complexity of our 2AdaSGD method is $O\big(n\epsilon^{-1} \wedge \epsilon^{-2}\big)$ for finding an ϵ-stationary solution, and the worst-case SFO complexity of our 2AdaSPIDER method is the optimal SFO complexity of $O\big(\sqrt{n}\epsilon^{-1} \wedge \epsilon^{-\frac{3}{2}}\big)$. To the best of our knowledge, our methods are the first double adaptive stochastic gradient methods for solving nonconvex optimization with nonsmooth regularization.

Acknowledgments

This paper was partially supported by NSFC under Grant No. 62376125. It was also partially supported by the Fundamental Research Funds for the Central Universities NO.NJ2023032.

References

Allen-Zhu, Z. 2018. Natasha 2: Faster non-convex optimization than SGD. *Advances in neural information processing systems*, 31.

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

Bottou, L.; Curtis, F. E.; and Nocedal, J. 2018. Optimization methods for large-scale machine learning. *SIAM review*, 60(2): 223–311.

Chen, J.; Zhou, D.; Tang, Y.; Yang, Z.; Cao, Y.; and Gu, Q. 2018. Closing the generalization gap of adaptive gradient methods in training deep neural networks. *arXiv preprint arXiv:1806.06763*.

Chen, X.; Liu, S.; Sun, R.; and Hong, M. 2019. On the convergence of a class of Adam-type algorithms for non-convex optimization. In *7th International Conference on Learning Representations, ICLR 2019*.

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

De, S.; Yadav, A.; Jacobs, D.; and Goldstein, T. 2017. Automated inference with adaptive batches. In *Artificial Intelligence and Statistics*, 1504–1513. PMLR.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Devarakonda, A.; Naumov, M.; and Garland, M. 2017. Adabatch: Adaptive batch sizes for training deep neural networks. *arXiv preprint arXiv:1712.02029*.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

Fang, C.; Li, C. J.; Lin, Z.; and Zhang, T. 2018. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. *Advances in neural information processing systems*, 31.

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

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

Guo, Z.; Xu, Y.; Yin, W.; Jin, R.; and Yang, T. 2021. A novel convergence analysis for algorithms of the adam family and beyond. *arXiv preprint arXiv:2104.14840*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Huang, F.; Li, J.; and Huang, H. 2021. Super-adam: faster and universal framework of adaptive gradients. *Advances in Neural Information Processing Systems*, 34: 9074–9085.

J Reddi, S.; Sra, S.; Poczos, B.; and Smola, A. J. 2016. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. *Advances in neural information processing systems*, 29.

Ji, K.; Wang, Z.; Weng, B.; Zhou, Y.; Zhang, W.; and Liang, Y. 2020. History-gradient aided batch size adaptation for variance reduced algorithms. In *International Conference on Machine Learning*, 4762–4772. PMLR.

Kavis, A.; Skoulakis, S.; Antonakopoulos, K.; Dadi, L. T.; and Cevher, V. 2022. Adaptive stochastic variance reduction for non-convex finite-sum minimization. *Advances in Neural Information Processing Systems*, 35: 23524–23538.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Lau, T. T.-K.; Liu, H.; and Kolar, M. 2024. AdAdaGrad: Adaptive Batch Size Schemes for Adaptive Gradient Methods. *arXiv preprint arXiv:2402.11215*.

Loshchilov, I.; and Hutter, F. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Marcus, M.; Santorini, B.; and Marcinkiewicz, M. A. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2): 313–330.

McCandlish, S.; Kaplan, J.; Amodei, D.; and Team, O. D. 2018. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Nguyen, L. M.; Liu, J.; Scheinberg, K.; and Takáč, M. 2017. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *International conference on machine learning*, 2613–2621. PMLR.

Reddi, S. J.; Hefny, A.; Sra, S.; Poczos, B.; and Smola, A. 2016. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, 314–323. PMLR.

Reddi, S. J.; Kale, S.; and Kumar, S. 2019. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*.

Robbins, H.; and Monro, S. 1951. A stochastic approximation method. *The annals of mathematical statistics*, 400–407.

Smith, S. L.; Kindermans, P.-J.; Ying, C.; and Le, Q. V. 2018. Don't Decay the Learning Rate, Increase the Batch Size. In *International Conference on Learning Representations*.

Vaswani, A. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Wang, Z.; Ji, K.; Zhou, Y.; Liang, Y.; and Tarokh, V. 2019. Spiderboost and momentum: Faster variance reduction algorithms. *Advances in Neural Information Processing Systems*, 32.

Ward, R.; Wu, X.; and Bottou, L. 2020. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. *Journal of Machine Learning Research*, 21(219): 1–30.

Xie, X.; Zhou, P.; Li, H.; Lin, Z.; and Yan, S. 2024. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

You, Y.; Li, J.; Reddi, S.; Hseu, J.; Kumar, S.; Bhojanapalli, S.; Song, X.; Demmel, J.; Keutzer, K.; and Hsieh, C.-J. 2019. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*.

Yun, J.; Lozano, A. C.; and Yang, E. 2021. Adaptive proximal gradient methods for structured neural networks. *Advances in Neural Information Processing Systems*, 34: 24365–24378.

Zhang, Y.; Chen, C.; Shi, N.; Sun, R.; and Luo, Z.-Q. 2022. Adam can converge without any modification on update rules. *Advances in neural information processing systems*, 35: 28386–28399.

Zhou, D.; Xu, P.; and Gu, Q. 2020. Stochastic nested variance reduction for nonconvex optimization. *Journal of machine learning research*, 21(103): 1–63.

Zhou, P.; Yuan, X.; and Feng, J. 2018. New insight into hybrid stochastic gradient descent: Beyond with-replacement sampling and convexity. *Advances in Neural Information Processing Systems*, 31.

Zhuang, J.; Tang, T.; Ding, Y.; Tatikonda, S. C.; Dvornek, N.; Papademetris, X.; and Duncan, J. 2020. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Advances in neural information processing systems*, 33: 18795–18806.