Scaled Conjugate Gradient Method for Nonconvex Optimization in Deep Neural Networks

Naoki Sato   NAOKI310303@GMAIL.COM
Computer Science Course,
Graduate School of Science and Technology,
Meiji University, Kanagawa 214-8571, Japan

Koshiro Izumi   KOSHI-IZUMI@SECOM.CO.JP
Visual Solutions Department 1,
Technology Development Division,
SECOM Co., Ltd., Tokyo 181-8528, Japan

Hideaki Iiduka   IIDUKA@CS.MEIJI.AC.JP
Department of Computer Science,
Meiji University, Kanagawa 214-8571, Japan

Editor: Prateek Jain

Abstract

A scaled conjugate gradient method that accelerates existing adaptive methods utilizing stochastic gradients is proposed for solving nonconvex optimization problems with deep neural networks. It is shown theoretically that, whether with a constant or diminishing learning rate, the proposed method can obtain a stationary point of the problem. Additionally, its rate of convergence with a diminishing learning rate is verified to be superior to that of the conjugate gradient method. The proposed method is shown to minimize training loss functions faster than the existing adaptive methods in practical applications of image and text classification. Furthermore, in the training of generative adversarial networks, one version of the proposed method achieved the lowest Fréchet inception distance score among those of the adaptive methods.

Keywords: adaptive method, conjugate gradient method, deep neural network, generative adversarial network, nonconvex optimization, scaled conjugate gradient method

1. Introduction

Nonconvex optimization is needed for neural networks and learning systems. Various methods that use deep neural networks have been reported. In particular, stochastic gradient descent (SGD) (Robbins and Monro, 1951) is a simple deep-learning optimizer that in theory can be applied to nonconvex optimization in deep neural networks (Fehrman et al., 2020; Scaman and Malherbe, 2020; Chen et al., 2020; Loizou et al., 2021).

Momentum methods (Polyak, 1964; Nesterov, 1983) and adaptive methods are powerful deep-learning optimizers for accelerating SGD. The adaptive methods include, for example, adaptive gradient (Adagrad) (Duchi et al., 2011), root mean square propagation (RMSprop) (Tieleman and Hinton, 2012), adaptive moment estimation (Adam) (Kingma and Ba, 2015), adaptive mean square gradient (AMSGrad) (Reddi et al., 2018), and Adam with decoupled

weight decay (AdamW) (Loshchilov and Hutter, 2019). In (Chen et al., 2019), generalized Adam (GAdam), which includes Adagrad, RMSprop, and AMSGrad, was proposed for nonconvex optimization. Adapting step sizes based on belief in observed gradients (AdaBelief) (Zhuang et al., 2020) is a recent deep-learning optimizer for nonconvex optimization. Convergence rates for SGD (Scaman and Malherbe, 2020) and the adaptive methods (Chen et al., 2019; Zhuang et al., 2020) reported in recent papers are summarized in Table 1. Note that "LR" in Table 1 means learning rate.

Table 1: Comparison of convergence rates for deep-learning optimizers used in nonconvex optimization

Algorithm	Nonconvex optimization	
	Constant LR	Diminishing LR
SGD (Scaman and Malherbe, 2020)	$\mathcal{O}\left(\frac{1}{N}\right) + C$	$\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$
GAdam (Chen et al., 2019)	———	$\mathcal{O}\left(\frac{\log N}{\sqrt{N}}\right)$
AdaBelief (Zhuang et al., 2020)	———	$\mathcal{O}\left(\frac{\log N}{\sqrt{N}}\right)$
CG (Iiduka and Kobayashi, 2020)	———	$\mathcal{O}\left(\sqrt{\frac{\log N}{N}}\right)$
SCG (Proposed)	$\mathcal{O}\left(\frac{1}{N}\right) + C_1\alpha + C_2\beta + C_3\delta$	$\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$

The constant learning rate rule requires that $\alpha_n = \alpha > 0$, $\beta_n = \beta \geq 0$, and $\delta_n = \delta \geq 0$, whereas the diminishing learning rate rule requires that $\alpha_n = \mathcal{O}(1/\sqrt{n})$, $\beta_n = \beta^n$, and $\delta_n = \delta^n$. C, C_1, C_2, and C_3 are positive constants independent of the learning rate and number of iterations N. See Section 2 for an explanation of the mathematical notation. The proposed method (SCG) can cover all the algorithms in Table 1 (see Section 1.3 for details). Hence, it enables algorithms, such as GAdam, AdaBelief, and the CG method, with diminishing learning rates achieve an $\mathcal{O}(1/\sqrt{N})$ convergence rate (Theorem 3), which is an improvement on the results in (Chen et al., 2019; Zhuang et al., 2020; Iiduka and Kobayashi, 2020).

1.1 Motivation

1.1.1 SCALED CONJUGATE GRADIENT (SCG) METHOD

The above-mentioned adaptive methods use the stochastic gradient of an observed loss function at each iteration. Meanwhile, conjugate gradient (CG) methods can be applied to large-scale nonconvex optimization (see Chapter 5.2 in (Nocedal and Wright, 2006)) by using the following conjugate gradient direction, which is defined from the current gradient $\mathsf{G}(\boldsymbol{x}_n, \xi_n)$ and the past direction $\mathbf{G}_{n-1}$:

$$\mathbf{G}_n := \mathsf{G}(\boldsymbol{x}_n, \xi_n) - \delta_n \mathbf{G}_{n-1}, \tag{1}$$

where $\delta_n > 0$ is a parameter. A CG method making use of this (Iiduka and Kobayashi, 2020) was presented for solving nonconvex optimization problems with deep neural networks. Table 1 shows that CG has a better convergence rate than those of GAdam and AdaBelief.

There are useful methods for general nonconvex optimization based on CG methods, such as *scaled CG (SCG)* and three-term CG (Møller, 1993; Zhang et al., 2006; Cheng, 2007; Andrei, 2007; Nakamura et al., 2013; Narushima and Yabe, 2014). The SCG method generates the following search direction using a *scaled* current gradient and the past direction:

$$\mathbf{G}_n := (1 + \gamma_n)\mathsf{G}(\boldsymbol{x}_n, \xi_n) - \delta_n\mathbf{G}_{n-1}, \tag{2}$$

where $\gamma_n \geq 0$ is a parameter. The SCG direction (2) with $\gamma_n = 0$ coincides with the CG direction (1). Adding a scaled parameter γ_n allows us to accelerate CG methods for nonconvex optimization (Nakamura et al., 2013; Narushima and Yabe, 2014). Accordingly, the first motivation of this paper is to show whether the SCG method for solving nonconvex optimization problems with deep neural networks, Algorithm 1 below, accelerates powerful deep-learning optimizers, such as RMSprop, Adagrad, Adam, and AdamW.

Algorithm 1 Scaled conjugate gradient method

Require: $(\alpha_n)_{n\in\mathbb{N}} \subset (0, 1), (\beta_n)_{n\in\mathbb{N}} \subset [0, 1), (\gamma_n)_{n\in\mathbb{N}} \subset [0, +\infty), (\delta_n)_{n\in\mathbb{N}} \subset [0, 1/2], \zeta \in [0, 1)$

1: $n \leftarrow 0, \boldsymbol{x}_0, \mathbf{G}_{-1}, \boldsymbol{m}_{-1} \in \mathbb{R}^d, \mathsf{H}_0 \in \mathbb{S}_{++}^d \cap \mathbb{D}^d$
2: **loop**
3:    $\mathbf{G}_n := (1 + \gamma_n)\mathsf{G}(\boldsymbol{x}_n, \xi_n) - \delta_n\mathbf{G}_{n-1}$
4:    $\boldsymbol{m}_n := \beta_n\boldsymbol{m}_{n-1} + (1 - \beta_n)\mathbf{G}_n$
5:    $\hat{\boldsymbol{m}}_n = (1 - \zeta^{n+1})^{-1}\boldsymbol{m}_n$
6:    $\mathsf{H}_n \in \mathbb{S}_{++}^d \cap \mathbb{D}^d$ (see (8) and (9) for examples of H_n)
7:    Find $\mathbf{d}_n \in \mathbb{R}^d$ that solves $\mathsf{H}_n\mathbf{d} = -\hat{\boldsymbol{m}}_n$
8:    $\boldsymbol{x}_{n+1} := \boldsymbol{x}_n + \alpha_n\mathbf{d}_n$
9:    $n \leftarrow n + 1$
10: **end loop**

The existing SCG methods in (Møller, 1993; Andrei, 2007; Nakamura et al., 2013; Narushima and Yabe, 2014) use the full gradient ∇f of the loss function f and, in particular, the SCG method in (Møller, 1993) can solve minimization problems in neural networks. The novelty of Algorithm 1 compared with the existing SCG methods is its use of the stochastic gradient $\mathsf{G}(\boldsymbol{x}_n, \xi_n)$ to make Algorithm 1 implementable for deep neural networks.

The second motivation related to the results in Table 1 is to clarify whether the SCG method (Algorithm 1) can in theory be applied to nonconvex optimization in deep neural networks. A particularly interesting concern is whether, under a diminishing learning rate rule, the SCG method converges faster than the CG method (Iiduka and Kobayashi, 2020). Moreover, we would like to clarify whether, under a constant learning rate rule, the SCG method can in theory approximate a desirable solution to a nonconvex optimization problem.

The third motivation is to provide evidence that the SCG method performs better than the existing methods, such as RMSprop, Adagrad, Adam, AMSGrad, and AdamW, on image and text classification tasks. This is because the usefulness of the SCG method should be verified from the viewpoint of not only theory but also practice.

1.2 Contribution

The present work makes theoretical and practical contributions. The theoretical contribution is to show that the SCG method, whether using a constant or diminishing learning rate, can find a stationary point of an optimization problem (Theorems 1 and 3). Using constant learning rates allows the SCG method to have approximately an $\mathcal{O}(1/N)$ convergence rate, where N denotes the number of iterations (see Table 1). For diminishing learning rates, it is shown that the rate of convergence of the SCG method is $\mathcal{O}(1/\sqrt{N})$, which is better than the $\mathcal{O}(\sqrt{\log N/N})$ rate of convergence of the CG method (Iiduka and Kobayashi, 2020) (see Table 1).

The practical contribution is to provide numerical examples in which the SCG method performs better than the existing adaptive methods at image classification, text classification, and image generation (Section 4). In particular, it is shown that, for training ResNet-18 on the CIFAR-100 and CIFAR-10 datasets, the SCG method with constant learning rates minimize the training loss functions faster than other methods. For the text classification task, the SCG method and RMSProp minimize the training loss functions faster than other methods. Since the SCG method uses the scaled conjugate gradient direction to accelerate the existing adaptive methods, it is considered that the method used in the experiments could optimize the training loss functions. In addition, among the adaptive methods, the SCG method achieves the lowest FID score in training several GANs.

1.3 Related work

Algorithm 1 when $\gamma_n = 0$, i.e., the non-scaled CG method, coincides with the CG method (Iiduka and Kobayashi, 2020) using $\mathbf{G}_n := \mathsf{G}(\boldsymbol{x}_n, \xi_n) - \delta_n \mathbf{G}_{n-1}$. The CG method with diminishing learning rates has $\mathcal{O}(\sqrt{\log N/N})$ convergence. Meanwhile, a convergence rate analysis of the CG method with constant learning rates has not yet been performed (see the "CG" column of Table 1). However, SCG with constant learning rates has $\mathcal{O}(1/N) + C_1\alpha + C_2\beta + C_3\delta$ convergence (see the "SCG" column of Table 1). Since SCG is CG when $\gamma_n = 0$, the results of the present study also indicate that CG with constant learning rates has $\mathcal{O}(1/N) + C_1\alpha + C_2\beta + C_3\delta$ convergence.

Algorithm 1 when $\gamma_n = 0$ and $\delta_n = 0$ is a unification (Iiduka, 2022) of the adaptive methods using $\mathbf{G}_n := \mathsf{G}(\boldsymbol{x}_n, \xi_n)$, such as GAdam (Chen et al., 2019) and AdaBelief (Zhuang et al., 2020) (see (Iiduka, 2022) for examples of H_n in step 6 of Algorithm 1). Our results show that the adaptive methods (i.e., SCG with $\gamma_n = 0$ and $\delta_n = 0$) using constant learning rates have $\mathcal{O}(1/N) + C_1\alpha + C_2\beta$ convergence, which is the same as in (Iiduka, 2022). Moreover, the adaptive methods using diminishing learning rates have $\mathcal{O}(1/\sqrt{N})$ convergence, the same as in (Iiduka, 2022). Therefore, our results are generalizations of the ones in (Iiduka, 2022).

The remainder of the paper is as follows. Section 2 provides the mathematical preliminaries. Section 3 presents convergence analyses of Algorithm 1. Section 4 provides numerical performance comparisons of the proposed method with the existing adaptive methods. Section 5 concludes the paper with a brief summary.

2. Mathematical Preliminaries

2.1 Notation and definitions

$\mathbb{N}$ is the set of non-negative integers. For $n \in \mathbb{N} \backslash \{0\}$, define $[n] := \{1, 2, \ldots, n\}$. The d-dimensional Euclidean space $\mathbb{R}^d$ has an inner product $\langle \cdot, \cdot \rangle$ inducing the norm $\| \cdot \|$. $\mathbb{S}^d$ denotes the set of $d \times d$ symmetric matrices: $\mathbb{S}^d = \{M \in \mathbb{R}^{d \times d} : M = M^\top\}$, where $^\top$ indicates the transpose operation. The set of $d \times d$ symmetric positive-definite matrices is denoted as $\mathbb{S}^d_{++} = \{M \in \mathbb{S}^d : M \succ O\}$, and the set of $d \times d$ diagonal matrices is denoted as $\mathbb{D}^d = \{M \in \mathbb{R}^{d \times d} : M = \mathsf{diag}(x_i), x_i \in \mathbb{R} \ (i \in [d])\}$.

2.2 Mathematical modeling in deep learning and assumptions

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 is measured by a differentiable nonconvex loss function $f(\boldsymbol{x}; z)$. We aim to minimize the expected loss defined for all $\boldsymbol{x} \in \mathbb{R}^d$ by

$$f(\boldsymbol{x}) = \mathbb{E}_{z \sim \mathcal{D}}[f(\boldsymbol{x}; z)] = \mathbb{E}[f_\xi(\boldsymbol{x})], \tag{3}$$

where $\mathcal{D}$ is a probability distribution over Z, ξ denotes a random variable with distribution function P, and $\mathbb{E}[\cdot]$ denotes the expectation taken with respect to ξ. A particularly interesting example of (3) is the empirical average loss defined for all $\boldsymbol{x} \in \mathbb{R}^d$ by

$$f(\boldsymbol{x}; S) = \frac{1}{T} \sum_{i \in [T]} f(\boldsymbol{x}; z_i) = \frac{1}{T} \sum_{i \in [T]} f_i(\boldsymbol{x}), \tag{4}$$

where $S = (z_1, z_2, \ldots, z_T)$ denotes the training set and $f_i(\cdot) := f(\cdot; z_i)$ denotes the loss function corresponding to the i-th training data z_i.

This paper considers optimization problems under the following assumptions.

Assumption 2.1

(A1) *$f_i \colon \mathbb{R}^d \to \mathbb{R}$ $(i \in [T])$ is continuously differentiable and $f \colon \mathbb{R}^d \to \mathbb{R}$ is defined for all $\boldsymbol{x} \in \mathbb{R}^d$ by*

$$f(\boldsymbol{x}) := \frac{1}{T} \sum_{i=1}^{T} f_i(\boldsymbol{x}),$$

where T denotes the number of samples.

(A2) *For each iteration n, the optimizers estimate the full gradient vector $\nabla f(\boldsymbol{x}_n)$ as the stochastic gradient vector $\mathsf{G}(\boldsymbol{x}_n, \xi_n)$ such that, for all $\boldsymbol{x} \in \mathbb{R}^d$, $\mathbb{E}[\mathsf{G}(\boldsymbol{x}, \xi_n)] = \nabla f(\boldsymbol{x})$.*

(A3) *There exists a positive number M such that, for all $\boldsymbol{x} \in \mathbb{R}^d$, $\mathbb{E}[\|\mathsf{G}(\boldsymbol{x}, \xi_n))\|^2] \leq M^2$.*

Assumption 2.1 (A1) is a standard one for nonconvex optimization in deep neural networks (see, e.g., (Chen et al., 2019, (2))). Assumption 2.1 (A2) is needed for the optimizers to work (see, e.g., (Chen et al., 2019, Section 2)), and Assumption 2.1 (A3) is used in the analysis of optimizers (see, e.g., (Chen et al., 2019, A2)).

2.3 Nonconvex optimization problem in deep neural networks

This paper deals with the following stationary point problem (Chen et al., 2019; Zhuang et al., 2020; Iiduka, 2022) for nonconvex optimization to minimize f defined in (A1).

Problem 2.1 *Under Assumption 2.1, we would like to find a stationary point $\boldsymbol{x}^\star$ of a nonconvex optimization problem that minimizes f over $\mathbb{R}^d$, i.e.,*

$$\boldsymbol{x}^\star \in X^\star := \left\{ \boldsymbol{x}^\star \in \mathbb{R}^d \colon \nabla f(\boldsymbol{x}^\star) = \boldsymbol{0} \right\} = \left\{ \boldsymbol{x}^\star \in \mathbb{R}^d \colon \langle \boldsymbol{x}^\star - \boldsymbol{x}, \nabla f(\boldsymbol{x}^\star) \rangle \le 0 \ \left(\boldsymbol{x} \in \mathbb{R}^d \right) \right\}.$$

The performance measure of the optimizers for Problem 2.1 is as follows:

$$\mathbb{E}\left[\langle \boldsymbol{x}_n - \boldsymbol{x}, \nabla f(\boldsymbol{x}_n) \rangle \right], \tag{5}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ and $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$ is the sequence generated by the optimizer.

Let us discuss the relationship between the performance measure (5) and the squared ℓ_2 norm of the gradient ∇f, which is the traditional performance measure. Assume that $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$ is bounded (see (A6)). First, we consider the case where there exists $n_0 \in \mathbb{N}$ such that, for all $n \in \mathbb{N}$ and all $\boldsymbol{x} \in \mathbb{R}^d$, $n \ge n_0$ implies that $\mathbb{E}[\langle \boldsymbol{x}_n - \boldsymbol{x}, \nabla f(\boldsymbol{x}_n) \rangle] < 0$. The boundedness condition of $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$ ensures that there exists a subsequence $(\boldsymbol{x}_{n_i})_{i\in\mathbb{N}}$ of $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$ such that $(\boldsymbol{x}_{n_i})_{i\in\mathbb{N}}$ converges to a point $\bar{\boldsymbol{x}} \in \mathbb{R}^d$. Hence, the continuity of ∇f (see (A1)) guarantees that, for all $\boldsymbol{x} \in \mathbb{R}^d$,

$$\mathbb{E}\left[\langle \bar{\boldsymbol{x}} - \boldsymbol{x}, \nabla f(\bar{\boldsymbol{x}}) \rangle \right] \le 0,$$

which, together with $\boldsymbol{x} := \bar{\boldsymbol{x}} - \nabla f(\bar{\boldsymbol{x}})$, implies that

$$\mathbb{E}\left[\|\nabla f(\bar{\boldsymbol{x}})\|^2 \right] = 0.$$

Accordingly, the convergent point $\bar{\boldsymbol{x}}$ of $(\boldsymbol{x}_{n_i})_{i\in\mathbb{N}}$ is a stationary point of f. Next, we consider the case where, for all $n_0 \in \mathbb{N}$, there exist $n \in \mathbb{N}$ and $\boldsymbol{x} \in \mathbb{R}^d$ such that $n \ge n_0$ and $\mathbb{E}[\langle \boldsymbol{x}_n - \boldsymbol{x}, \nabla f(\boldsymbol{x}_n) \rangle] \ge 0$. Then, there exists a subsequence $(\boldsymbol{x}_{n_j})_{j\in\mathbb{N}}$ of $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$ such that, for all $j \in \mathbb{N}$, $0 \le \mathbb{E}[\langle \boldsymbol{x}_{n_j} - \boldsymbol{x}, \nabla f(\boldsymbol{x}_{n_j}) \rangle]$. The boundedness condition of $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$ ensures that there exists a subsequence $(\boldsymbol{x}_{n_{j_k}})_{k\in\mathbb{N}}$ of $(\boldsymbol{x}_{n_j})_{j\in\mathbb{N}}$ such that $(\boldsymbol{x}_{n_{j_k}})_{k\in\mathbb{N}}$ converges to a point $\hat{\boldsymbol{x}} \in \mathbb{R}^d$. Hence, the continuity of ∇f (see (A1)) guarantees that

$$0 \le \lim_{k\to+\infty} \mathbb{E}\left[\langle \boldsymbol{x}_{n_{j_k}} - \boldsymbol{x}, \nabla f(\boldsymbol{x}_{n_{j_k}}) \rangle \right] \le \mathbb{E}\left[\langle \hat{\boldsymbol{x}} - \boldsymbol{x}, \nabla f(\hat{\boldsymbol{x}}) \rangle \right] \le \epsilon,$$

where $\epsilon > 0$ is the precision. The definition of the inner product implies that, if ϵ is small enough, then $\|\nabla f(\hat{\boldsymbol{x}})\|$ will be small.

Let $\boldsymbol{x} := \boldsymbol{x}^\star$ be a stationary point of f and consider the following performance measure:

$$\mathbb{E}\left[\langle \boldsymbol{x}_n - \boldsymbol{x}^\star, \nabla f(\boldsymbol{x}_n) \rangle \right], \tag{6}$$

which is a restricted measure of (5). When the optimizer can find $\boldsymbol{x}^\star$, the inner product of $\boldsymbol{x}_n - \boldsymbol{x}^\star$ and $\nabla f(\boldsymbol{x}_n)$ is positive. If the upper bound of (6) is small enough, then the definition of the inner product implies that the optimizer has fast convergence. Hence, the

performance measure (5) including (6) will be adequate for investigating the performances of deep-learning optimizers.

Let us consider the case where f is convex (Kingma and Ba, 2015; Reddi et al., 2018). In this case, Problem 2.1 involves doing the following:

$$\text{Find a point } \boldsymbol{x}^\star \text{ such that } f(\boldsymbol{x}^\star) \leq f(\boldsymbol{x}) \text{ for all } \boldsymbol{x} \in \mathbb{R}^d. \qquad (7)$$

The performance measure of Problem (7) is the regret, defined as

$$R(T) := \sum_{i=1}^{T} f_i(\boldsymbol{x}_i) - T f^\star,$$

where $f^\star$ denotes the optimal value of Problem (7) and $(\boldsymbol{x}_i)_{i=1}^T$ is the sequence generated by the optimizer. Let $\boldsymbol{x}^\star$ be a solution of Problem (7). When the upper bound of the performance measure (5) is $\epsilon > 0$, the convexity of f ensures that

$$\mathbb{E}\left[f(\boldsymbol{x}_n) - f^\star\right] \leq \mathbb{E}\left[\langle \boldsymbol{x}_n - \boldsymbol{x}^\star, \nabla f(\boldsymbol{x}_n)\rangle\right] \leq \epsilon.$$

Accordingly, the performance measure (5) leads to the one $\mathbb{E}[f(\boldsymbol{x}_n) - f^\star]$ used for convex optimization.

3. Convergence Analyses of Algorithm 1

3.1 Outline of our results

Let us outline our results (see also Table 1). First, we show that the SCG method (Algorithm 1) with constant learning rates ensures that there exist positive constants C_i ($i = 1, 2, 3$) such that

$$\frac{1}{n}\sum_{k=1}^{n} \mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq \mathcal{O}\left(\frac{1}{n}\right) + C_1\alpha + C_2\beta + C_3\delta$$

(see Theorem 1 and Table 1). This implies that, if f is convex, we have

$$\frac{R(T)}{T} \leq \mathcal{O}\left(\frac{1}{T}\right) + C_1\alpha + C_2\beta + C_3\delta$$

(see Proposition 2). Next, we show that the SCG method (Algorithm 1) with diminishing learning rates, such as $\alpha_n = \mathcal{O}(1/\sqrt{n})$, gives

$$\frac{1}{n}\sum_{k=1}^{n} \mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$$

(see Theorem 3 and Table 1). This implies that, if f is convex,

$$\frac{R(T)}{T} \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$$

(see Proposition 4).

3.2 Our detailed results

In order to analyze Algorithm 1, we will assume the following.

Assumption 3.1 *For the sequence* $(\mathsf{H}_n)_{n\in\mathbb{N}} \subset \mathbb{S}^d_{++} \cap \mathbb{D}^d$ *in Algorithm 1 defined by* $\mathsf{H}_n := \mathsf{diag}(h_{n,i})$, *the following conditions are satisfied:*

(A4) $h_{n+1,i} \geq h_{n,i}$ *for all* $n \in \mathbb{N}$ *and all* $i \in [d]$;

(A5) *For all* $i \in [d]$, *a positive number* B_i *exists such that* $\sup\{\mathbb{E}[h_{n,i}]\colon n \in \mathbb{N}\} \leq B_i$.

Moreover,

(A6) $D := \max_{i\in[d]} \sup\{(x_{n,i} - x_i)^2 \colon n \in \mathbb{N}\} < +\infty$ *for* $\boldsymbol{x} = (x_i) \in \mathbb{R}^d$.

Section 4 provides examples of H_n satisfying Assumption 3.1 (A4) and (A5) (see also (Iiduka, 2022)). Assumption 3.1 (A6) was used in the analysis of the existing adaptive methods (see (Kingma and Ba, 2015; Reddi et al., 2018; Iiduka, 2022)).

The following is a convergence analysis of Algorithm 1 with constant learning rates. The proof of Theorem 1 is given in Appendix A.

Theorem 1 *Suppose that Assumptions 2.1 and 3.1 hold and let* $\alpha_n := \alpha$, $\beta_n := \beta$, $\gamma_n := \gamma$, *and* $\delta_n := \delta$. *Then, Algorithm 1 is such that, for all* $\boldsymbol{x} \in \mathbb{R}^d$,

$$\liminf_{n\to+\infty} \mathbb{E}\left[\langle \boldsymbol{x}_n - \boldsymbol{x}, \nabla f(\boldsymbol{x}_n)\rangle\right] \leq \frac{\tilde{B}^2\tilde{M}^2}{2\tilde{b}\tilde{\gamma}\tilde{\zeta}^2}\alpha + \frac{\tilde{M}\sqrt{Dd}}{\tilde{b}\tilde{\gamma}\tilde{\zeta}}\beta + \frac{4\hat{M}\sqrt{Dd}}{\tilde{\gamma}\tilde{\zeta}}\delta,$$

where $\tilde{b} := 1-\beta$, $\tilde{\gamma} := 1+\gamma$, $\tilde{\zeta} := 1-\zeta$, $\tilde{M}^2 := \max\{\|\boldsymbol{m}_{-1}\|^2, M^2\}$, $\tilde{B} := \sup\{\max_{i\in[d]} h_{n,i}^{-1/2} \colon n \in \mathbb{N}\} < +\infty$, *and* $\hat{M} := \max\{M^2, \|\mathbf{G}_{-1}\|^2\}$. *Moreover, for all* $\boldsymbol{x} \in \mathbb{R}^d$ *and all* $n \in \mathbb{N}$,

$$\frac{1}{n}\sum_{k=1}^n \mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k)\rangle\right] \leq \frac{D\sum_{i=1}^d B_i}{2\tilde{b}\alpha n} + \frac{\tilde{B}^2\tilde{M}^2}{2\tilde{b}\tilde{\zeta}^2}\alpha + \frac{\tilde{M}\sqrt{Dd}}{\tilde{b}}\beta + \frac{4\hat{M}\sqrt{Dd}}{\tilde{\gamma}}\delta.$$

Theorem 1 implies the following proposition.

Proposition 2 *Under the assumptions in Theorem 1 and the convexity of* f_i $(i \in [T])$, *we have*

$$\liminf_{n\to+\infty} \mathbb{E}\left[f(\boldsymbol{x}_n) - f^\star\right] \leq \frac{\tilde{B}^2\tilde{M}^2}{2\tilde{b}\tilde{\gamma}\tilde{\zeta}^2}\alpha + \frac{\tilde{M}\sqrt{Dd}}{\tilde{b}\tilde{\gamma}\tilde{\zeta}}\beta + \frac{4\hat{M}\sqrt{Dd}}{\tilde{\gamma}\tilde{\zeta}}\delta,$$

where the positive constants are defined as in Theorem 1. Moreover, for all $n \in \mathbb{N}$,

$$\min_{k\in[n]} \mathbb{E}\left[f(\boldsymbol{x}_k) - f^\star\right] \leq \frac{D\sum_{i=1}^d B_i}{2\tilde{b}\alpha n} + \frac{\tilde{B}^2\tilde{M}^2}{2\tilde{b}\tilde{\zeta}^2}\alpha + \frac{\tilde{M}\sqrt{Dd}}{\tilde{b}}\beta + \frac{4\hat{M}\sqrt{Dd}}{\tilde{\gamma}}\delta,$$

$$\frac{R(T)}{T} \leq \frac{D\sum_{i=1}^d B_i}{2\tilde{b}\alpha T} + \frac{\tilde{B}^2\tilde{M}^2}{2\tilde{b}\tilde{\zeta}^2}\alpha + \frac{\tilde{M}\sqrt{Dd}}{\tilde{b}}\beta + \frac{4\hat{M}\sqrt{Dd}}{\tilde{\gamma}}\delta.$$

Additionally, if we define $\tilde{\boldsymbol{x}}_n$ *by* $\tilde{\boldsymbol{x}}_n := (1/n)\sum_{k=1}^n \boldsymbol{x}_k$, *then*

$$\mathbb{E}\left[f(\tilde{\boldsymbol{x}}_n) - f^\star\right] \leq \frac{D\sum_{i=1}^d B_i}{2\tilde{b}\alpha n} + \frac{\tilde{B}^2\tilde{M}^2}{2\tilde{b}\tilde{\zeta}^2}\alpha + \frac{\tilde{M}\sqrt{Dd}}{\tilde{b}}\beta + \frac{4\hat{M}\sqrt{Dd}}{\tilde{\gamma}}\delta.$$

The following is a convergence analysis of Algorithm 1 with diminishing learning rates. The proof of Theorem 3 is given in Appendix A.

Theorem 3 *Suppose that Assumptions 2.1 and 3.1 hold and let $(\alpha_n)_{n \in \mathbb{N}}$, $(\beta_n)_{n \in \mathbb{N}}$, $(\gamma_n)_{n \in \mathbb{N}}$, and $(\delta_n)_{n \in \mathbb{N}}$ satisfy*

(C1) $\sum_{n=0}^{+\infty} \alpha_n = +\infty$, $\sum_{n=0}^{+\infty} \alpha_n^2 < +\infty$, $\sum_{n=0}^{+\infty} \alpha_n \beta_n < +\infty$, $\sum_{n=0}^{+\infty} \alpha_n \gamma_n < +\infty$, *and* $\sum_{n=0}^{+\infty} \alpha_n \delta_n < +\infty$.

Then, Algorithm 1 satisfies, for all $\boldsymbol{x} \in \mathbb{R}^d$,

$$\liminf_{n \to +\infty} \mathbb{E}\left[\langle \boldsymbol{x}_n - \boldsymbol{x}, \nabla f(\boldsymbol{x}_n) \rangle\right] \leq 0.$$

Let $(\alpha_n)_{n \in \mathbb{N}}$, $(\beta_n)_{n \in \mathbb{N}}$, $(\gamma_n)_{n \in \mathbb{N}}$, and $(\delta_n)_{n \in \mathbb{N}}$ satisfy

(C2) $\lim_{n \to +\infty} 1/(n\alpha_n) = 0$, $\lim_{n \to +\infty}(1/n)\sum_{k=0}^{n} \alpha_k = 0$, $\lim_{n \to +\infty}(1/n)\sum_{k=0}^{n} \beta_k = 0$, $\lim_{n \to +\infty}(1/n)\sum_{k=0}^{n} \delta_k = 0$, *and* $\gamma_{n+1} \leq \gamma_n$ $(n \in \mathbb{N})$.

Then, for all $\boldsymbol{x} \in \mathbb{R}^d$,

$$\limsup_{n \to +\infty} \frac{1}{n} \sum_{k=1}^{n} \mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k) \rangle\right] \leq 0.$$

In particular, for the case of $\alpha_n := \mathcal{O}(1/n^\eta)$ $(\eta \in (0,1))$, $\beta_n := \beta^n$, and $\delta_n := \delta^n$, Algorithm 1 exhibits convergence such that, for all $\boldsymbol{x} \in \mathbb{R}^d$ and all $n \in \mathbb{N}$,

$$\frac{1}{n} \sum_{k=1}^{n} \mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \nabla f(\boldsymbol{x}_k) \rangle\right] \leq \frac{D \sum_{i=1}^{d} B_i}{2\tilde{b}n^{1-\eta}} + \frac{\tilde{B}^2 \tilde{M}^2}{2\tilde{b}\tilde{\zeta}^2(1-\eta)n^\eta} + \frac{\beta \tilde{M}\sqrt{Dd}}{\tilde{b}(1-\beta)n} + \frac{4\delta \hat{M}\sqrt{Dd}}{(1-\delta)n}.$$

Theorem 3 implies the following proposition.

Proposition 4 *Under the assumptions in Theorem 3 and the convexity of f_i $(i \in [T])$, Algorithm 1 with (C1) satisfies*

$$\liminf_{n \to +\infty} \mathbb{E}\left[f(\boldsymbol{x}_n) - f^\star\right] = 0.$$

Moreover, under (C2), any accumulation point of $(\tilde{\boldsymbol{x}}_n)_{n \in \mathbb{N}}$ defined by $\tilde{\boldsymbol{x}}_n := (1/n)\sum_{k=1}^{n} \boldsymbol{x}_k$ almost surely belongs to the solution set of Problem (7). In particular, in the case of $\alpha_n := \mathcal{O}(1/n^\eta)$ $(\eta \in (0,1))$, $\beta_n := \beta^n$, and $\delta_n := \delta^n$, Algorithm 1 exhibits convergence such that, for all $n \in \mathbb{N}$,

$$\min_{k \in [n]} \mathbb{E}\left[f(\boldsymbol{x}_k) - f^\star\right] \leq \frac{D \sum_{i=1}^{d} B_i}{2\tilde{b}n^{1-\eta}} + \frac{\tilde{B}^2 \tilde{M}^2}{2\tilde{b}\tilde{\zeta}^2(1-\eta)n^\eta} + \frac{\beta \tilde{M}\sqrt{Dd}}{\tilde{b}(1-\beta)n} + \frac{4\delta \hat{M}\sqrt{Dd}}{(1-\delta)n},$$

$$\frac{R(T)}{T} \leq \frac{D \sum_{i=1}^{d} B_i}{2\tilde{b}T^{1-\eta}} + \frac{\tilde{B}^2 \tilde{M}^2}{2\tilde{b}\tilde{\zeta}^2(1-\eta)T^\eta} + \frac{\beta \tilde{M}\sqrt{Dd}}{\tilde{b}(1-\beta)T} + \frac{4\delta \hat{M}\sqrt{Dd}}{(1-\delta)T},$$

$$\mathbb{E}\left[f(\tilde{\boldsymbol{x}}_n) - f^\star\right] \leq \frac{D \sum_{i=1}^{d} B_i}{2\tilde{b}n^{1-\eta}} + \frac{\tilde{B}^2 \tilde{M}^2}{2\tilde{b}\tilde{\zeta}^2(1-\eta)n^\eta} + \frac{\beta \tilde{M}\sqrt{Dd}}{\tilde{b}(1-\beta)n} + \frac{4\delta \hat{M}\sqrt{Dd}}{(1-\delta)n}.$$

9

4. Numerical Experiments

4.1 Algorithms

We define $\mathsf{H}_n^{\mathrm{Adam}}$ by

$$
\begin{aligned}
\boldsymbol{v}_n &:= \theta \boldsymbol{v}_{n-1} + (1-\theta)\mathbf{G}_n \odot \mathbf{G}_n, \\
\bar{\boldsymbol{v}}_n &:= (1-\theta^{n+1})^{-1}\boldsymbol{v}_n, \\
\hat{\boldsymbol{v}}_n &:= \max\{\hat{v}_{n-1,i}, \bar{v}_{n,i}\}, \\
\mathsf{H}_n^{\mathrm{Adam}} &:= \mathsf{diag}\left(\sqrt{\hat{v}_{n,i}}\right),
\end{aligned}
\tag{8}
$$

where $\theta \in [0,1)$, $\boldsymbol{v}_n = \hat{\boldsymbol{v}}_n = \mathbf{0}$, and $\boldsymbol{x} \odot \boldsymbol{x} := (x_i^2)_{i=1}^d$ for $\boldsymbol{x} = (x_i)_{i=1}^d \in \mathbb{R}^d$. Algorithm 1 with $\mathsf{H}_n = \mathsf{H}_n^{\mathrm{Adam}}$ and $\gamma_n = \delta_n = 0$ resembles Adam (Kingma and Ba, 2015)[1]. We also define $\mathsf{H}_n^{\mathrm{AMSGrad}}$ by

$$
\begin{aligned}
\boldsymbol{v}_n &:= \theta \boldsymbol{v}_{n-1} + (1-\theta)\mathbf{G}_n \odot \mathbf{G}_n, \\
\hat{\boldsymbol{v}}_n &:= \max\{\hat{v}_{n-1,i}, v_{n,i}\}, \\
\mathsf{H}_n^{\mathrm{AMSGrad}} &:= \mathsf{diag}\left(\sqrt{\hat{v}_{n,i}}\right).
\end{aligned}
\tag{9}
$$

Algorithm 1 with $\mathsf{H}_n = \mathsf{H}_n^{\mathrm{AMSGrad}}$ and $\gamma_n = \delta_n = 0$ is equivalent to AMSGrad (Reddi et al., 2018). We compared Algorithm 1 with SGD (Robbins and Monro, 1951), Momentum (Polyak, 1964; Nesterov, 1983), RMSprop (Tieleman and Hinton, 2012), Adagrad (Duchi et al., 2011), Adam (Kingma and Ba, 2015), AMSGrad (Reddi et al., 2018), and AdamW (Loshchilov and Hutter, 2019) (see Sections A.5 and A.6 for details).

The experiments used two Intel(R) Xeon(R) Gold 6148 2.4-GHz CPUs with 20 cores each and a 16-GB NVIDIA Tesla V100 900-Gbps GPU. The experimental code was written in Python 3.7.5 using the NumPy 1.21.6 package and PyTorch 1.7.1 package. Python implementations of the optimizers used in the numerical experiments are available at `https://github.com/iiduka-researches/202210-izumi`.

4.2 Image classification

For image classification, we used a residual neural network (ResNet), a relatively deep model derived from convolutional neural networks (CNNs). It was applied to the CIFAR-100 and CIFAR-10 benchmark datasets[2] for image classification. These datasets respectively comprise 100 and 10 classes and each contain 6,000 32×32 pixel images. Each dataset was separated into 50,000 training images and 10,000 test images (1,000 randomly selected images per class). A 18-layer ResNet was trained on CIFAR-100 and CIFAR-10 (He et al., 2015). Following common practice in image classification, cross-entropy was used as the loss function for model fitting.

The image classification results for Algorithm 1 with constant and diminishing learning rates are shown in Figures 1, 2, 6, and 7, where panels (a), (b), and (c) respectively show

---

1. While Adam (Kingma and Ba, 2015) used $\mathsf{H}_n := \mathsf{diag}(\sqrt{\bar{v}_{n,i}})$, we use $\mathsf{H}_n^{\mathrm{Adam}}$ to guarantee convergence (Iiduka, 2022).
2. `https://www.cs.toronto.edu/~kriz/cifar.html`

the training loss function value, training classification accuracy score, and test classification accuracy score as functions of the number of epochs. Figures 3, 4, 5, 8, 9, and 10 present box-plot comparisons of Algorithm 1 with constant (panel (a)) and diminishing (panel (b)) learning rates in terms of the training loss function value, training classification accuracy score, and test classification accuracy score. As shown, Algorithm 1 performed better with constant learning rates than with diminishing learning rates. For example, looking at Figures 1(a) and 3(a)(ResNet-18 on CIFAR-100), we see that SCGAdam performed best at minimizing the training loss function. Moreover, looking at Figures 6(a) and 8(a)(ResNet-18 on CIFAR-10), SCGAdam performed best at minimizing the training loss function. Figures 1(c) and 5(a)(ResNet-18 on CIFAR-100) also show that, when the number of epochs was 200, Momentum had a higher test classification accuracy score than those of the other algorithms. Meanwhile, AMSGrad and SCGAdam had 70 % test classification accuracy and was faster than the other algorithms (see Figure 1(c)). Figure 6(c) (ResNet-18 on CIFAR-10) also shows that, when the number of epochs was 200, Momentum had a higher classification accuracy than those of the other algorithms, while SCGAdam had 90 % test classification accuracy and was faster than the other algorithms.
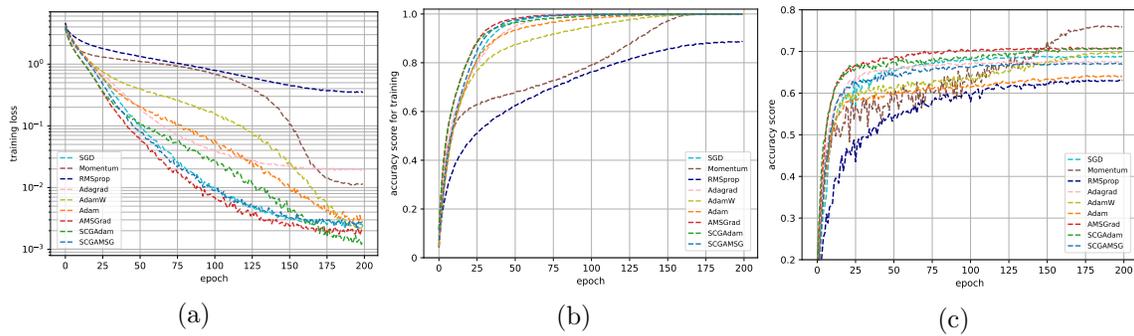
Figure 1: Results of Algorithm 1 with constant learning rates for training ResNet-18 on the CIFAR-100 dataset: (a) training loss function value, (b) training classification accuracy score, and (c) test classification accuracy score.
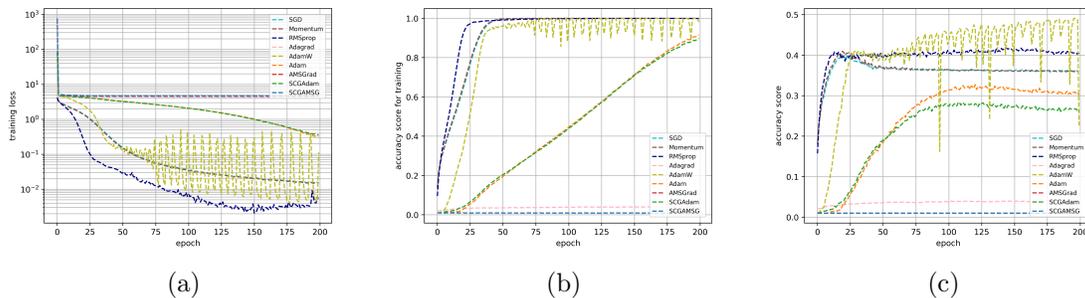
Figure 2: Results of Algorithm 1 with diminishing learning rates for training ResNet-18 on the CIFAR-100 dataset: (a) training loss function value, (b) training classification accuracy score, and (c) test classification accuracy score.

(a)

(b)
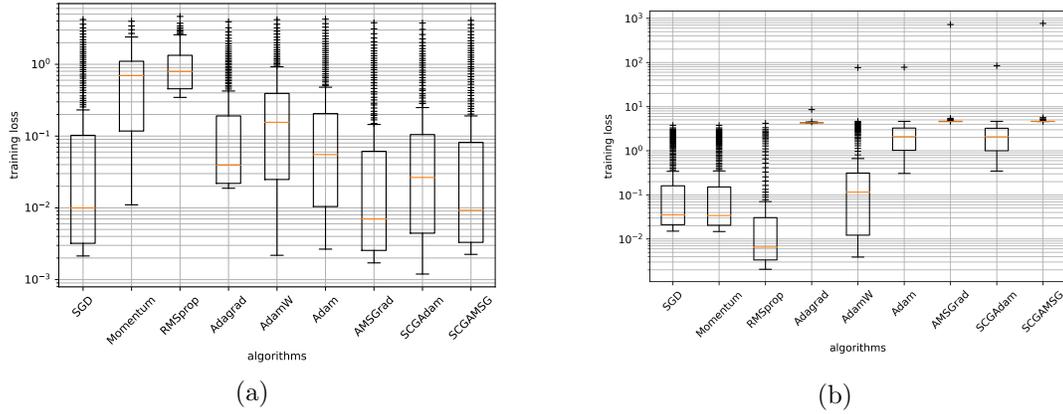
Figure 3: Box plots of training loss function values for Algorithm 1 for training ResNet-18 on the CIFAR-100 dataset: (a) constant learning rates and (b) diminishing learning rates.

(a)

(b)
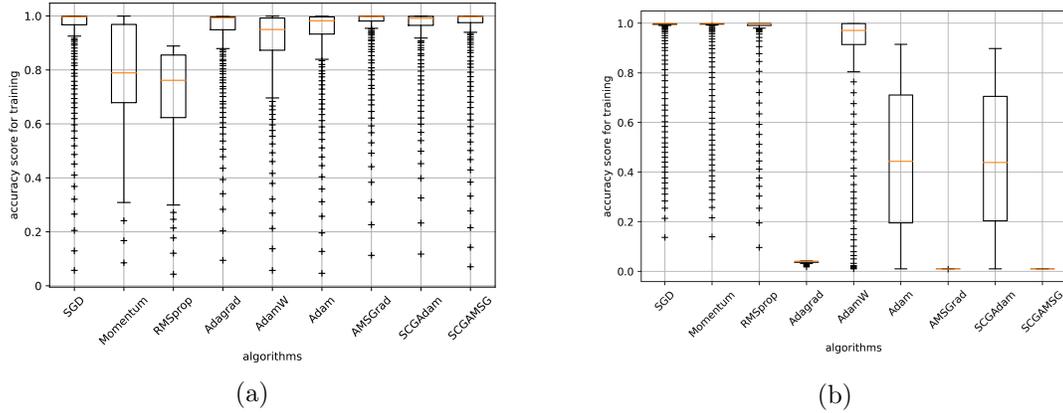
Figure 4: Box plots of training classification accuracy score for Algorithm 1 for training ResNet-18 on the CIFAR-100 dataset: (a) constant learning rates and (b) diminishing learning rates.

(a)

(b)
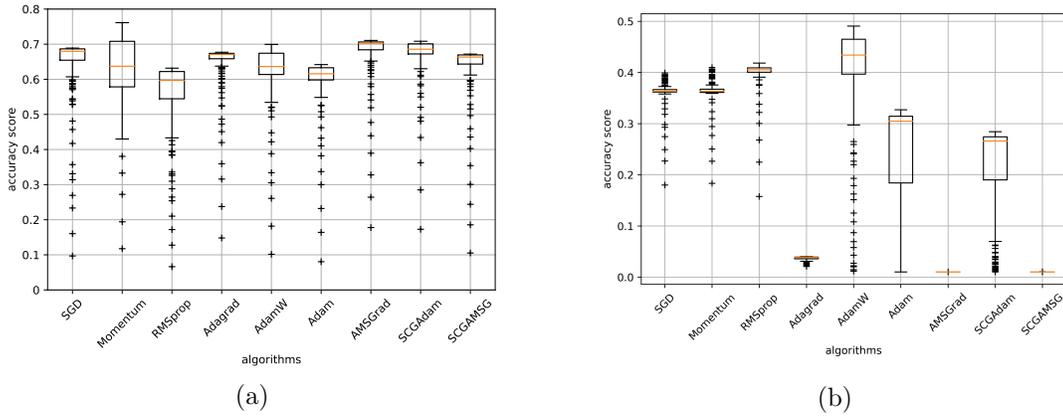
Figure 5: Box plots of test classification accuracy score for Algorithm 1 for training ResNet-18 on the CIFAR-100 dataset: (a) constant learning rates and (b) diminishing learning rates.
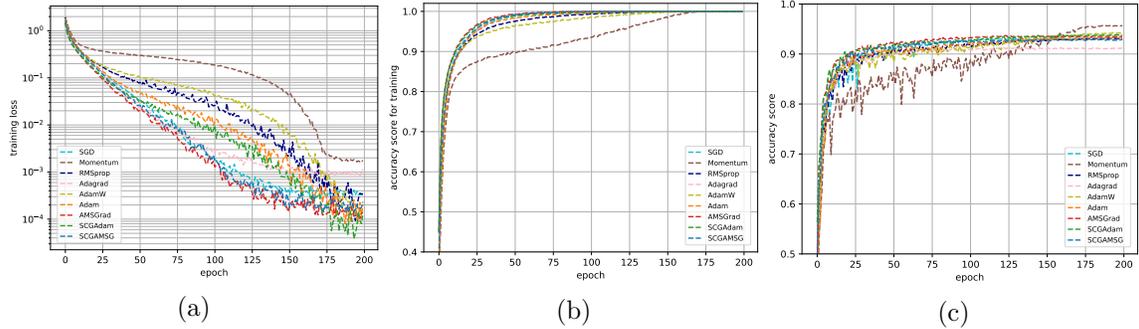
Figure 6: Results of Algorithm 1 with constant learning rates for training ResNet-18 on the CIFAR-10 dataset: (a) training loss function value, (b) training classification accuracy score, and (c) test classification accuracy score.
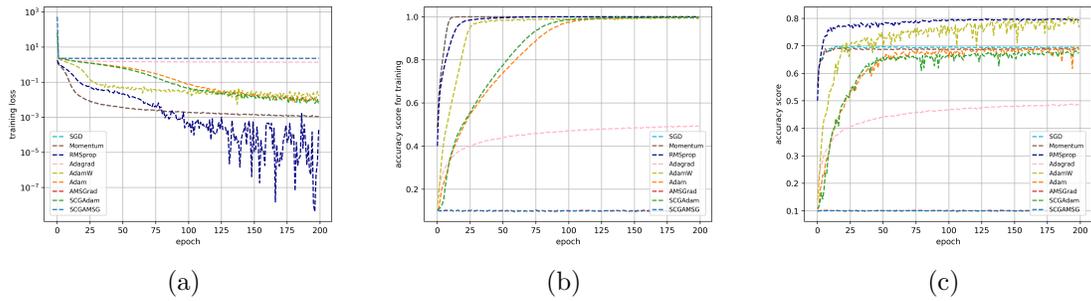
Figure 7: Results of Algorithm 1 with diminishing learning rates for training ResNet-18 on the CIFAR-10 dataset: (a) training loss function value, (b) training classification accuracy score, and (c) test classification accuracy score.
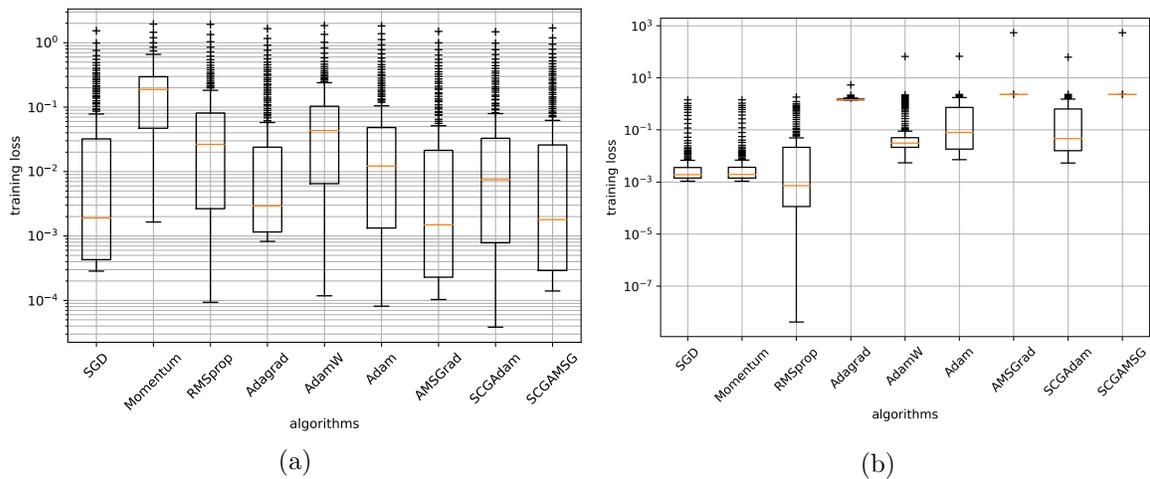
Figure 8: Box plots of training loss function values for Algorithm 1 for training ResNet-18 on the CIFAR-10 dataset: (a) constant learning rates and (b) diminishing learning rates.

(a)   (b)
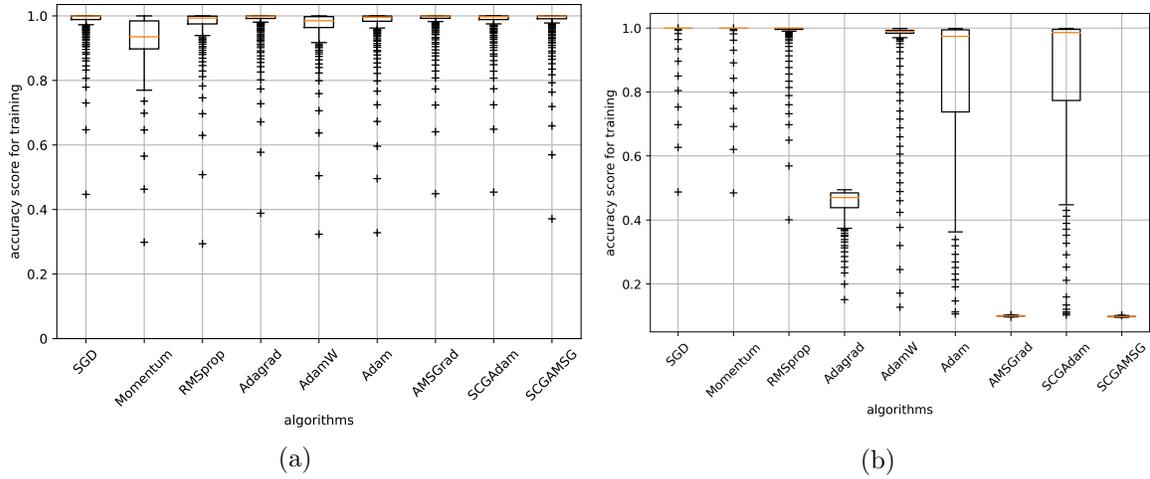
Figure 9: Box plots of training classification accuracy score for Algorithm 1 for training ResNet-18 on the CIFAR-10 dataset: (a) constant learning rates and (b) diminishing learning rates.

(a)   (b)
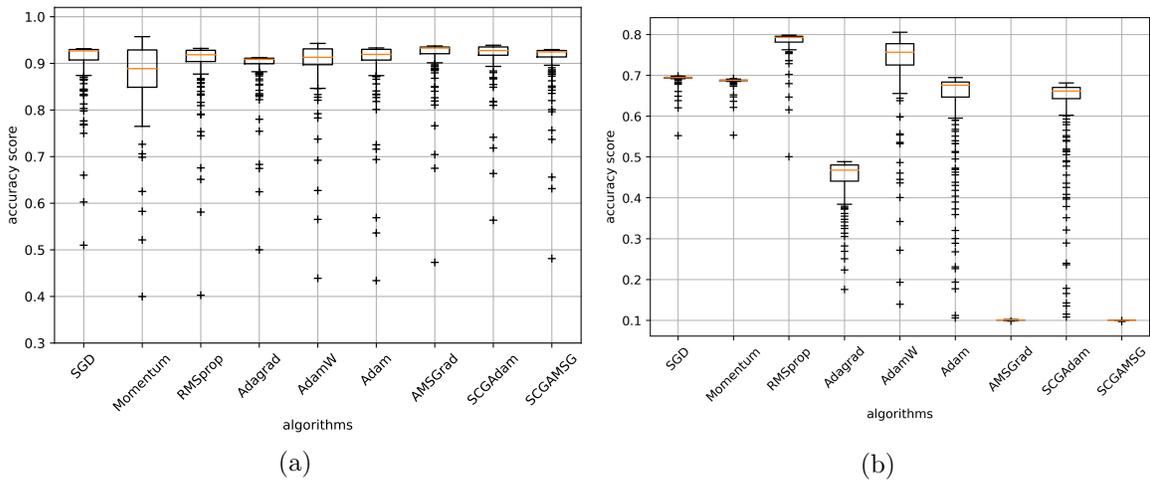
Figure 10: Box plots of test classification accuracy score for Algorithm 1 for training ResNet-18 on the CIFAR-10 dataset: (a) constant learning rates and (b) diminishing learning rates.

4.3 Text classification

For text classification, we used long short-term memory (LSTM), an artificial recurrent neural network (RNN) architecture used for deep learning that is based on natural language processing. The IMDb dataset (Maas et al., 2011) was used; this dataset comprises 50,000 movie reviews and associated binary sentiment polarity labels. The data were split into 25,000 training sets and 25,000 test sets. The data were classified using a multilayer LSTM neural network and AlphaDropout for overfitting suppression. This network included one affine layer and employed a sigmoid activation function for the output. As with the image classification, cross-entropy was used as the loss function for the model fitting.

The text classification results for Algorithm 1 with constant learning rates are shown in Figures 11 and 12, where panels (a), (b), and (c) respectively show the training loss function value, training classification accuracy score, and test classification accuracy score as functions of the number of epochs. Figures 13, 14, and 15 present box-plot comparisons of Algorithm 1 with constant (panel (a)) and diminishing (panel (b)) learning rates in terms of the training loss function value, training classification accuracy score, and test classification accuracy score. As was the case with the image datasets, Algorithm 1 performed better with constant learning rates than with diminishing learning rates. For example, looking at Figures 11(a), and 13(a), we see that SCGAdam, along with RMSprop, performed the best at minimizing the training loss function.

| (a) | (b) | (c) |
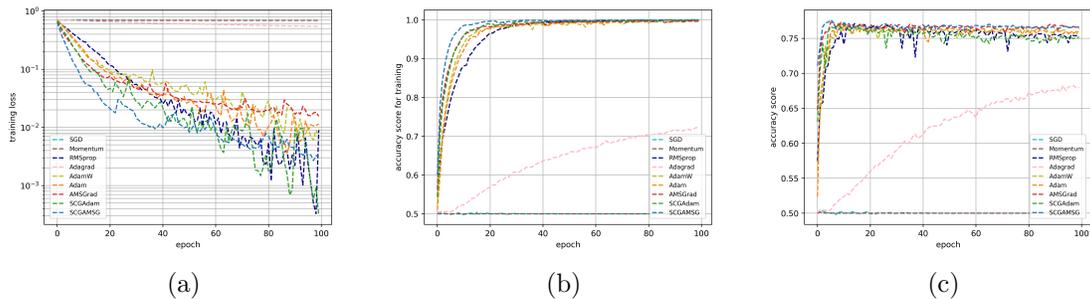
Figure 11: Results of Algorithm 1 with constant learning rates on the IMDb dataset: (a) training loss function value, (b) training classification accuracy score, and (c) test classification accuracy score.

| (a) | (b) | (c) |
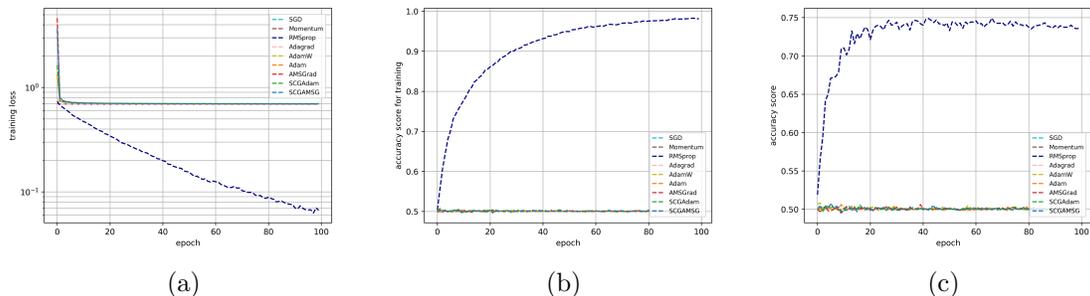
Figure 12: Results of Algorithm 1 with diminishing learning rates on the IMDb dataset: (a) training loss function value, (b) training classification accuracy score, and (c) test classification accuracy score.
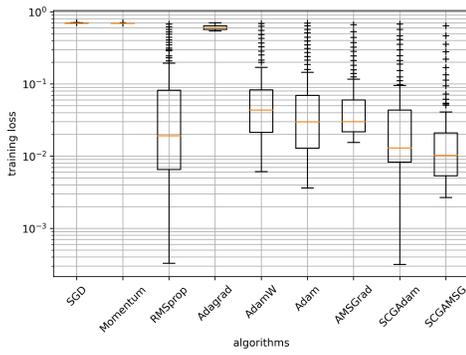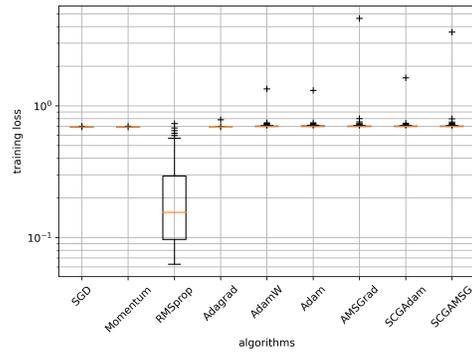
(a)   (b)

Figure 13: Box plots of training loss function values for Algorithm 1 on the IMDb dataset: (a) constant learning rates and (b) diminishing learning rates.
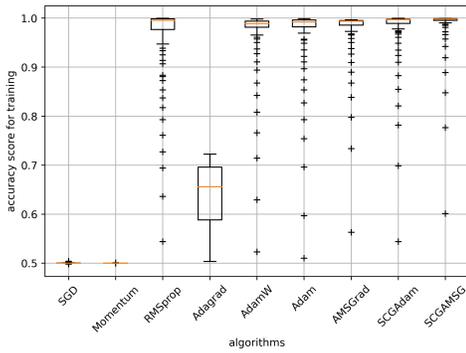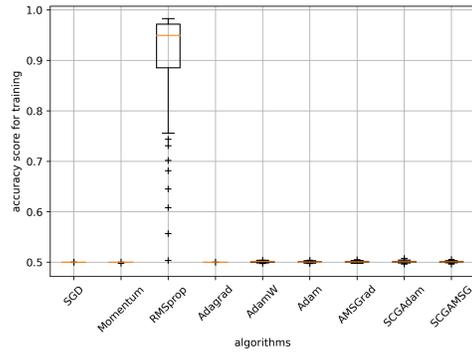
(a)   (b)

Figure 14: Box plots of training classification accuracy score for Algorithm 1 on the IMDb dataset: (a) constant learning rates and (b) diminishing learning rates.
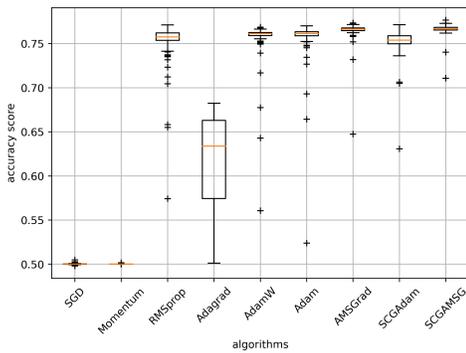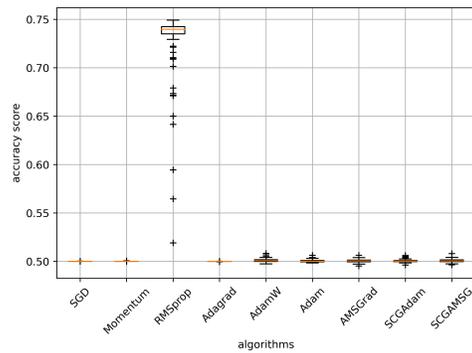
(a)   (b)

Figure 15: Box plots of test classification accuracy score for Algorithm 1 on the IMDb dataset: (a) constant learning rates and (b) diminishing learning rates.

16

4.4 Image generation with GANs

Generative adversarial networks (GANs) consist of two deep neural networks, a generator and a discriminator. The training of GANs is generally unstable because it is performed simultaneously by the generator and the discriminator in a minimax game. Since SGD and Momentum often cause mode collapse and their convergence is slow, most of the previous studies (Goodfellow, 2017; Gulrajani et al., 2017; Salimans et al., 2016; Zhuang et al., 2020) used adaptive methods such as Adam, RMSProp, and AdaBelief. Therefore, training of GANs is suitable for testing the stability of the optimizer. Here, we trained a deep convolutional GAN (DCGAN) (Radford et al., 2016) on the LSUN-Bedroom dataset (Yu et al., 2015), SNGAN (Miyato et al., 2018) on the CIFAR10 dataset, and Wasserstein GAN with gradient penalty (WGAN-GP) (Gulrajani et al., 2017) on the Tiny ImageNet dataset (Ya Le, 2015) with some adaptive optimizers. We used the widely used Fréchet inception distance (FID) (Heusel et al., 2017) to evaluate the quality of the generated images.

For training DCGAN, five 250,000-step trainings were performed with each optimizer under the optimal hyperparameter settings; the results are summarized in Figure 16. Figure 17 compares the box plots of mean FID scores for the last 50,000 steps, where the decrease in FID score has subsided. According to Figure 17, SCGAMSGrad achieves the lowest FID score on average, with AdaBelief and SCGAdam having second and third lowest scores. See Table 2 in Appendix A.7 for the hyperparameters used in the experiments.
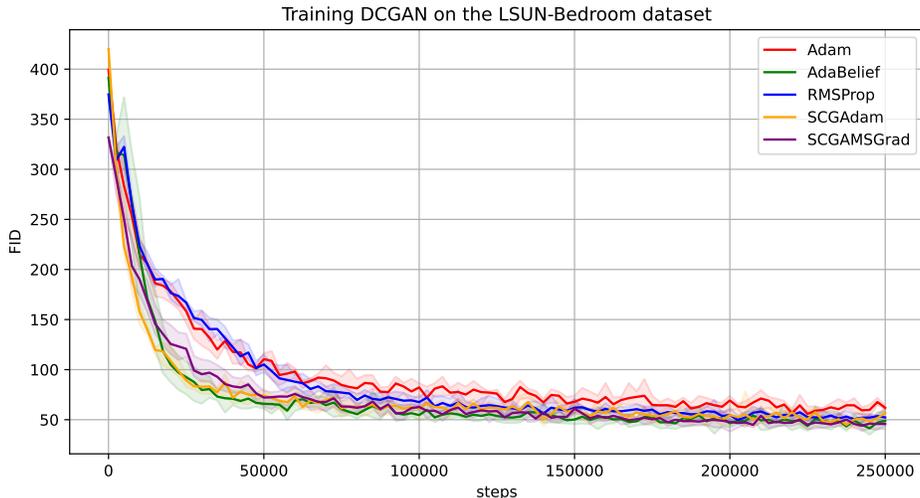
Figure 16: Mean FID (solid line) bounded by the maximum and minimum over 5 runs (shaded area) for training DCGAN on the LSUN-Bedroom dataset for five optimizers. For all runs, the batch size is 64 and the learning rate combinations are determined with a grid search (see Figure 22).

For training SNGAN, five 150,000-step trainings were performed with each optimizer under the optimal hyperparameter settings; the results are summarized in Figure 18. Moreover, Figure 19 compares the box plots of mean FID scores for the last 30,000 steps, where the decrease in FID score has subsided. According to this figure, SCGAMSGrad achieves

Training DCGAN on the LSUN-Bedroom dataset
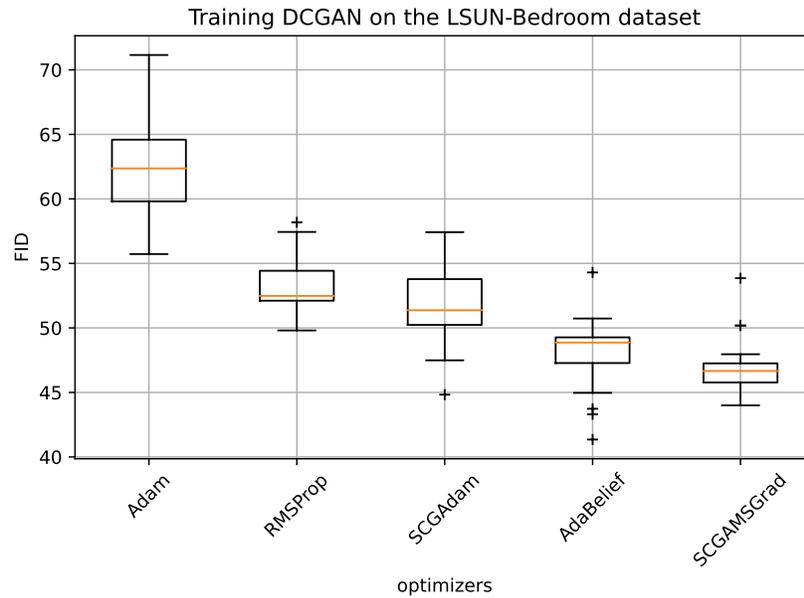
Figure 17: Box plots of mean FID scores for the last 50,000 steps in training DCGAN on the LSUN-Bedroom dataset for five optimizers.

the lowest FID score on average. See Table 3 in Appendix A.7 for the hyperparameters used in the experiments.

Training SNGAN on the CIFAR10 dataset
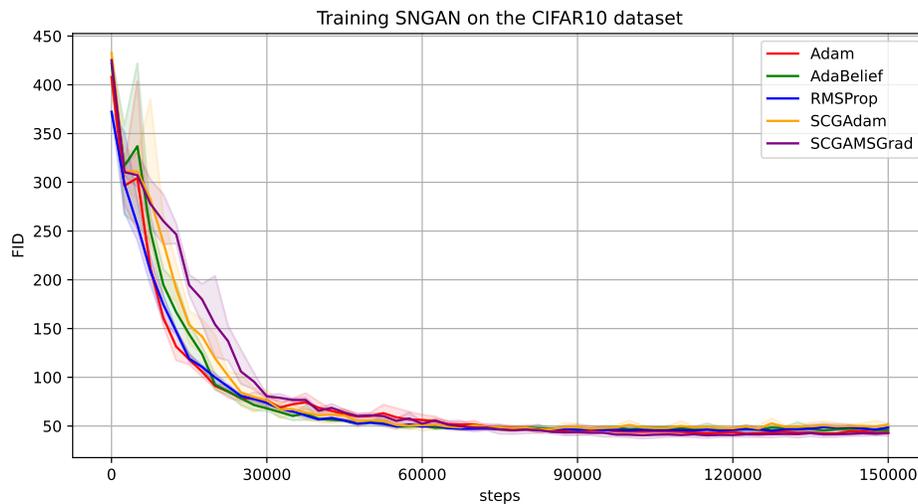
Figure 18: Mean FID (solid line) bounded by the maximum and minimum over 5 runs (shaded area) for training SNGAN on the CIFAR10 dataset for five optimizers. For all runs, the batch size is 64 and the learning rate combinations are determined with a grid search (see Figure 23).

For training WGAN-GP, five 400,000-step trainings were performed with each optimizer under the optimal hyperparameter settings; the results are summarized in Figure 20. More-
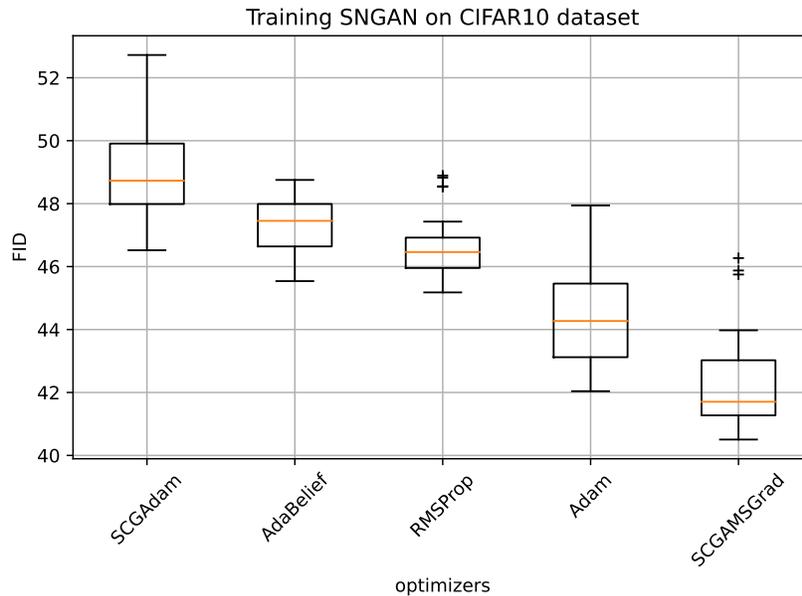
Figure 19: Box plots of mean FID scores for the last 30,000 steps in training SNGAN on the CIFAR10 dataset for five optimizers.

over, Figure 21 compares the box plots of mean FID scores for the last 80,000 steps, where the decrease in FID score has subsided. According to this figure, SCGAdam achieves the lowest FID score on average. See Table 4 in Appendix A.7 for the hyperparameters used in the experiments.
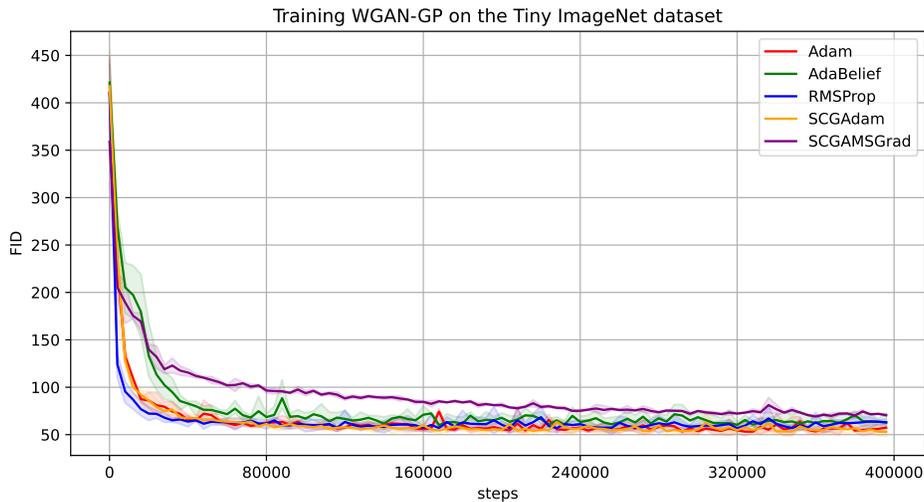
Figure 20: Mean FID (solid line) bounded by the maximum and minimum over 5 runs (shaded area) for training WGAN-GP on the Tiny ImageNet dataset for five optimizers. For all runs, the batch size is 64 and the learning rate combinations are determined with a grid search (see Figure 24).
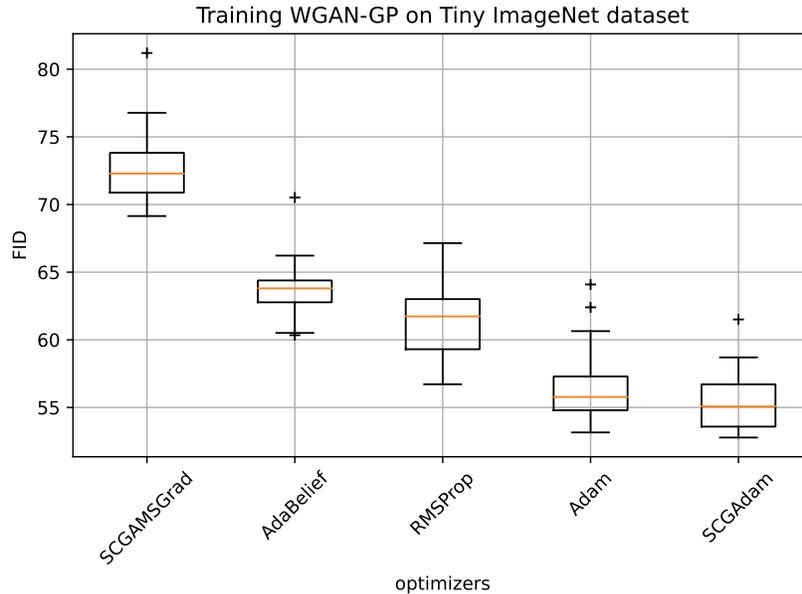
Figure 21: Box plots of mean FID scores for the last 80,000 steps in training WGAN-GP on the Tiny ImageNet dataset for five optimizers.

All of the results show that the proposed method successfully trains unstable GANs as well as other adaptive methods. They also show that the proposed method is able to achieve the lowest FID for certain datasets and models.

5. Conclusion

We proposed a scaled conjugate gradient method for nonconvex optimization in deep learning. With constant learning rates, this method has approximately $\mathcal{O}(1/N)$ convergence, and with diminishing learning rates, it has $\mathcal{O}(1/\sqrt{N})$ convergence, which is an improvement on the previous results reported in (Iiduka and Kobayashi, 2020). We also showed that the proposed method is suitable for practical applications, in particular image and text classification tasks. Using constant learning rates, it minimized the training loss functions faster than other methods. Furthermore, our experimental results show that the proposed method can achieve lowest FID score among adaptive optimizers in training several GANs.

Acknowledgments

Appendix A. Appendix A

Suppose that $H \in \mathbb{S}_{++}^d$. The H-inner product of $\mathbb{R}^d$ is defined for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ by $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_H := \langle \boldsymbol{x}, H\boldsymbol{y} \rangle$, and the H-norm is defined by $\|\boldsymbol{x}\|_H := \sqrt{\langle \boldsymbol{x}, H\boldsymbol{x} \rangle}$.

A.1 Lemmas

Lemma 5 *Under (A1) and (A2), for all $\boldsymbol{x} \in \mathbb{R}^d$ and all $n \in \mathbb{N}$,*

$$
\mathbb{E}\left[\|\boldsymbol{x}_{n+1} - \boldsymbol{x}\|_{\mathsf{H}_n}^2\right]
$$
$$
\leq \mathbb{E}\left[\|\boldsymbol{x}_n - \boldsymbol{x}\|_{\mathsf{H}_n}^2\right] + 2\alpha_n\left[\frac{\beta_n}{\tilde{\zeta}_n}\mathbb{E}\left[\langle \boldsymbol{x} - \boldsymbol{x}_n, \boldsymbol{m}_{n-1}\rangle\right] + \frac{\tilde{\beta}_n}{\tilde{\zeta}_n}\left\{(1 + \gamma_n)\mathbb{E}\left[\langle \boldsymbol{x} - \boldsymbol{x}_n, \nabla f(\boldsymbol{x}_n)\rangle\right]\right.\right.
$$
$$
\left.\left. + \delta_n\mathbb{E}\left[\langle \boldsymbol{x}_n - \boldsymbol{x}, \mathsf{G}_{n-1}\rangle\right]\right\}\right] + \alpha_n^2\mathbb{E}\left[\|\mathbf{d}_n\|_{\mathsf{H}_n}^2\right],
$$

where $\tilde{\zeta}_n := 1 - \zeta^{n+1}$ and $\tilde{\beta}_n := 1 - \beta_n$.

Proof Let $\boldsymbol{x} \in \mathbb{R}^d$ and $n \in \mathbb{N}$. We have that

$$
\|\boldsymbol{x}_{n+1} - \boldsymbol{x}\|_{\mathsf{H}_n}^2 = \|\boldsymbol{x}_n - \boldsymbol{x}\|_{\mathsf{H}_n}^2 + 2\alpha_n \langle \boldsymbol{x}_n - \boldsymbol{x}, \mathbf{d}_n \rangle_{\mathsf{H}_n} + \alpha_n^2 \|\mathbf{d}_n\|_{\mathsf{H}_n}^2. \tag{10}
$$

The definition of the H_n-inner product and the definition of Algorithm 1 ensure that

$$
\langle \boldsymbol{x}_n - \boldsymbol{x}, \mathbf{d}_n \rangle_{\mathsf{H}_n} = \frac{1}{\tilde{\zeta}_n} \langle \boldsymbol{x} - \boldsymbol{x}_n, \boldsymbol{m}_n \rangle = \frac{\beta_n}{\tilde{\zeta}_n} \langle \boldsymbol{x} - \boldsymbol{x}_n, \boldsymbol{m}_{n-1} \rangle + \frac{\tilde{\beta}_n}{\tilde{\zeta}_n} \langle \boldsymbol{x} - \boldsymbol{x}_n, \mathsf{G}_n \rangle.
$$

The definition of G_n implies that

$$
\langle \boldsymbol{x} - \boldsymbol{x}_n, \mathsf{G}_n \rangle = (1 + \gamma_n) \langle \boldsymbol{x} - \boldsymbol{x}_n, \mathsf{G}(\boldsymbol{x}_n, \xi_n) \rangle + \delta_n \langle \boldsymbol{x}_n - \boldsymbol{x}, \mathsf{G}_{n-1} \rangle.
$$

Define the history of process $\xi_0, \xi_1, \ldots$ to time step n by $\xi_{[n]} = (\xi_0, \xi_1, \ldots, \xi_n)$. The condition $\boldsymbol{x}_n = \boldsymbol{x}_n(\xi_{[n-1]})$ $(n \in \mathbb{N})$ and (A2) guarantee that

$$
\begin{aligned}
\mathbb{E}\left[\langle \boldsymbol{x} - \boldsymbol{x}_n, \mathsf{G}(\boldsymbol{x}_n, \xi_n) \rangle\right] &= \mathbb{E}\left[\mathbb{E}\left[\langle \boldsymbol{x} - \boldsymbol{x}_n, \mathsf{G}(\boldsymbol{x}_n, \xi_n) \rangle | \xi_{[n-1]}\right]\right] \\
&= \mathbb{E}\left[\left\langle \boldsymbol{x} - \boldsymbol{x}_n, \mathbb{E}\left[\mathsf{G}(\boldsymbol{x}_n, \xi_n) | \xi_{[n-1]}\right]\right\rangle\right] \\
&= \mathbb{E}\left[\langle \boldsymbol{x} - \boldsymbol{x}_n, \nabla f(\boldsymbol{x}_n) \rangle\right].
\end{aligned}
$$

Accordingly, taking the expectation of (10) gives us the following lemma. ∎

Lemma 6 *(Iiduka, 2022, Lemma 2) Under (A3), for all $n \in \mathbb{N}$,*

$$
\mathbb{E}\left[\|\boldsymbol{m}_n\|^2\right] \leq \tilde{M}^2 := \max\left\{\|\boldsymbol{m}_{-1}\|^2, M^2\right\}.
$$

Additionally, under (A4), for all $n \in \mathbb{N}$,

$$
\mathbb{E}\left[\|\mathbf{d}_n\|_{\mathsf{H}_n}^2\right] \leq \frac{\tilde{B}^2\tilde{M}^2}{\tilde{\zeta}^2},
$$

where $\tilde{\zeta} := 1 - \zeta$ and $\tilde{B} := \sup\{\max_{i=1,2,\ldots,d} h_{n,i}^{-1/2} : n \in \mathbb{N}\} < +\infty$.

Lemma 7 *Under (A3), for all $n \in \mathbb{N}$,*

$$\mathbb{E}\left[\|\mathbf{G}_n\|^2\right] \leq 16\hat{M}^2,$$

where $\hat{M} := \max\{M^2, \|\mathbf{G}_{-1}\|^2\}$.

Proof From the definition of $\mathbf{G}_n$, we find that, for all $n \in \mathbb{N}$,

$$\mathbb{E}\left[\|\mathbf{G}_n\|^2\right] \leq 2(1+\gamma_n)^2\mathbb{E}\left[\|\mathsf{G}(\boldsymbol{x}_n,\xi_n)\|^2\right] + 2\delta_n^2\mathbb{E}\left[\|\mathbf{G}_{n-1}\|^2\right]$$
$$\leq 8M^2 + 2\delta_n^2\mathbb{E}\left[\|\mathbf{G}_{n-1}\|^2\right],$$

where $\gamma_n \leq 1$ and (A3) have been used. In the case of $n = 0$, we have that $\mathbb{E}[\|\mathbf{G}_0\|^2] \leq 8\hat{M}^2 + 2(1/2)^2\hat{M}^2 \leq 16\hat{M}^2$, where $\delta_n \leq 1/2$. Assume that $\mathbb{E}[\|\mathbf{G}_n\|^2] \leq 16\hat{M}^2$ for some n. Then, we have

$$\mathbb{E}\left[\|\mathbf{G}_{n+1}\|^2\right] \leq 8M^2 + 2\delta_{n+1}^2\mathbb{E}\left[\|\mathbf{G}_n\|^2\right]$$
$$\leq 8\hat{M}^2 + 2 \cdot \frac{1}{4} \cdot 16\hat{M}^2 = 16\hat{M}^2.$$

Induction thus shows that, for all $n \in \mathbb{N}$, $\mathbb{E}[\|\mathbf{G}_n\|^2] \leq 16\hat{M}^2$. ∎

A.2 Theorem

Theorem 8 *Let $(\kappa_n)_{n\in\mathbb{N}}$ be defined by $\kappa_n := \alpha_n\tilde{\beta}_n(1+\gamma_n)/\tilde{\zeta}_n$ and $(\beta_n)_{n\in\mathbb{N}}$ satisfy $\kappa_{n+1} \leq \kappa_n$ $(n \in \mathbb{N})$ and $\limsup_{n\to+\infty}\beta_n < 1$. Define $V_n(\boldsymbol{x}) := \mathbb{E}\left[\langle\boldsymbol{x}_n - \boldsymbol{x}, \nabla f(\boldsymbol{x}_n)\rangle\right]$ for all $\boldsymbol{x} \in \mathbb{R}^d$ and all $n \in \mathbb{N}$. Under (A1)–(A6), for $\boldsymbol{x} \in \mathbb{R}^d$ and $n \geq 1$,*

$$\frac{1}{n}\sum_{k=1}^{n}V_k(\boldsymbol{x}) \leq \frac{D\sum_{i=1}^{d}B_i}{2\tilde{b}\alpha_n n} + \frac{\tilde{B}^2\tilde{M}^2}{2\tilde{b}\tilde{\zeta}^2 n}\sum_{k=1}^{n}\alpha_k + \frac{\tilde{M}\sqrt{Dd}}{\tilde{b}n}\sum_{k=1}^{n}\beta_k + \frac{4\hat{M}\sqrt{Dd}}{n}\sum_{k=1}^{n}\frac{\delta_k}{1+\gamma_k},$$

where $\tilde{b} := 1 - b$, $\tilde{\zeta} := 1 - \zeta$, $(\beta_n)_{n\in\mathbb{N}} \subset (0, b] \subset (0, 1)$, $\tilde{M}$, $\hat{M}$, and $\tilde{B}$ are defined in Lemmas 6 and 7, and D and B_i are defined in Assumption 3.1.

Proof Fix $\boldsymbol{x} \in X$ arbitrarily. Lemma 5 guarantees that, for all $n \geq 1$,

$$\sum_{k=1}^{n}V_k(\boldsymbol{x}) \leq \underbrace{\frac{1}{2}\sum_{k=1}^{n}\frac{1}{\kappa_k}\left\{\mathbb{E}\left[\|\boldsymbol{x}_k - \boldsymbol{x}\|_{\mathsf{H}_k}^2\right] - \mathbb{E}\left[\|\boldsymbol{x}_{k+1} - \boldsymbol{x}\|_{\mathsf{H}_k}^2\right]\right\}}_{K_n}$$

$$+ \underbrace{\sum_{k=1}^{n}\frac{\beta_k}{\tilde{\beta}_k(1+\gamma_k)}\mathbb{E}\left[\langle\boldsymbol{x} - \boldsymbol{x}_k, \boldsymbol{m}_{k-1}\rangle\right]}_{B_n} \tag{11}$$

$$+ \underbrace{\sum_{k=1}^{n}\frac{\delta_k}{(1+\gamma_k)}\mathbb{E}\left[\langle\boldsymbol{x}_k - \boldsymbol{x}, \mathbf{G}_{k-1}\rangle\right]}_{\Delta_n} + \underbrace{\frac{1}{2}\sum_{k=1}^{n}\frac{\alpha_k\tilde{\zeta}_k}{\tilde{\beta}_k}\mathbb{E}\left[\|\mathbf{d}_k\|_{\mathsf{H}_k}^2\right]}_{A_n}.$$

From the definition of K_n and $\mathbb{E}[\|\boldsymbol{x}_{n+1} - \boldsymbol{x}\|_{\mathsf{H}_n}^2]/\kappa_n \geq 0$,

$$K_n \leq \frac{\mathbb{E}\left[\|\boldsymbol{x}_1 - \boldsymbol{x}\|_{\mathsf{H}_1}^2\right]}{\kappa_1} + \underbrace{\sum_{k=2}^n \left\{ \frac{\mathbb{E}\left[\|\boldsymbol{x}_k - \boldsymbol{x}\|_{\mathsf{H}_k}^2\right]}{\kappa_k} - \frac{\mathbb{E}\left[\|\boldsymbol{x}_k - \boldsymbol{x}\|_{\mathsf{H}_{k-1}}^2\right]}{\kappa_{k-1}} \right\}}_{\tilde{K}_n}. \tag{12}$$

Since $\overline{\mathsf{H}}_k \in \mathbb{S}_{++}^d$ exists such that $\mathsf{H}_k = \overline{\mathsf{H}}_k^2$, we have $\|\boldsymbol{x}\|_{\mathsf{H}_k}^2 = \|\overline{\mathsf{H}}_k \boldsymbol{x}\|^2$ for all $\boldsymbol{x} \in \mathbb{R}^d$. Accordingly, we have

$$\tilde{K}_n = \mathbb{E}\left[\sum_{k=2}^n \left\{ \frac{\left\|\overline{\mathsf{H}}_k(\boldsymbol{x}_k - \boldsymbol{x})\right\|^2}{\kappa_k} - \frac{\left\|\overline{\mathsf{H}}_{k-1}(\boldsymbol{x}_k - \boldsymbol{x})\right\|^2}{\kappa_{k-1}} \right\}\right].$$

From $\mathsf{H}_k = \mathsf{diag}(h_{k,i})$, for all $k \in \mathbb{N}$ and all $\boldsymbol{x} := (x_i) \in \mathbb{R}^d$,

$$\overline{\mathsf{H}}_k = \mathsf{diag}\left(\sqrt{h_{k,i}}\right) \text{ and } \left\|\overline{\mathsf{H}}_k \boldsymbol{x}\right\|^2 = \sum_{i=1}^d h_{k,i} x_i^2. \tag{13}$$

Hence, for all $n \geq 2$,

$$\tilde{K}_n = \mathbb{E}\left[\sum_{k=2}^n \sum_{i=1}^d \left(\frac{h_{k,i}}{\kappa_k} - \frac{h_{k-1,i}}{\kappa_{k-1}} \right)(x_{k,i} - x_i)^2\right].$$

From $\kappa_k \leq \kappa_{k-1}$ ($k \geq 1$) and (A4), we have $h_{k,i}/\kappa_k - h_{k-1,i}/\kappa_{k-1} \geq 0$ ($k \geq 1, i \in [d]$). Accordingly, (A6) implies that, for all $n \geq 2$,

$$\tilde{K}_n \leq D\mathbb{E}\left[\sum_{k=2}^n \sum_{i=1}^d \left(\frac{h_{k,i}}{\kappa_k} - \frac{h_{k-1,i}}{\kappa_{k-1}} \right)\right] = D\mathbb{E}\left[\sum_{i=1}^d \left(\frac{h_{n,i}}{\kappa_n} - \frac{h_{1,i}}{\kappa_1} \right)\right].$$

Therefore, (12), $\mathbb{E}[\|\boldsymbol{x}_1 - \boldsymbol{x}\|_{\mathsf{H}_1}^2]/\kappa_1 \leq D\mathbb{E}[\sum_{i=1}^d h_{1,i}/\kappa_1]$, and (A5) imply, for all $n \in \mathbb{N}$,

$$K_n \leq D\mathbb{E}\left[\sum_{i=1}^d \frac{h_{1,i}}{\kappa_1}\right] + D\mathbb{E}\left[\sum_{i=1}^d \left(\frac{h_{n,i}}{\kappa_n} - \frac{h_{1,i}}{\kappa_1} \right)\right] \leq \frac{D}{\kappa_n}\mathbb{E}\left[\sum_{i=1}^d h_{n,i}\right] \leq \frac{D}{\kappa_n}\sum_{i=1}^d B_i.$$

Moreover, from $\gamma_n, \zeta \geq 0$ and $\tilde{b} := 1 - b$,

$$\kappa_n := \frac{\alpha_n \tilde{\beta}_n (1 + \gamma_n)}{\tilde{\zeta}_n} \geq \tilde{b}\alpha_n.$$

Hence,

$$K_n \leq \frac{D \sum_{i=1}^d B_i}{\tilde{b}\alpha_n}. \tag{14}$$

The Cauchy-Schwarz inequality and (A6) imply that

$$B_n \leq \sum_{k=1}^{n} \frac{\beta_k}{\tilde{\beta}_k(1+\gamma_k)} \mathbb{E}\left[\|\boldsymbol{x}-\boldsymbol{x}_k\| \|\boldsymbol{m}_{k-1}\|\right] \leq \frac{\sqrt{Dd}}{\tilde{b}} \sum_{k=1}^{n} \beta_k \mathbb{E}\left[\|\boldsymbol{m}_{k-1}\|\right] \leq \frac{\tilde{M}\sqrt{Dd}}{\tilde{b}} \sum_{k=1}^{n} \beta_k,$$

(15)

where the final inequality comes from Lemma 6 and Jensen's inequality, i.e., for all $n \in \mathbb{N}$, $\mathbb{E}[\|\boldsymbol{m}_n\|] \leq \tilde{M}$. A discussion similar to the one showing (15) implies that

$$\Delta_n \leq \sum_{k=1}^{n} \frac{\delta_k}{1+\gamma_k} \mathbb{E}\left[\|\boldsymbol{x}_k-\boldsymbol{x}\| \|\mathbf{G}_{k-1}\|\right] \leq 4\hat{M}\sqrt{Dd} \sum_{k=1}^{n} \frac{\delta_k}{1+\gamma_k},$$

(16)

where the final inequality comes from Lemma 7 and Jensen's inequality, i.e., for all $n \in \mathbb{N}$, $\mathbb{E}[\|\mathbf{G}_n\|] \leq 16\hat{M}$. Lemma 6 ensures that, for all $n \in \mathbb{N}$,

$$A_n := \sum_{k=1}^{n} \frac{\alpha_k \tilde{\zeta}_k}{\tilde{\beta}_k} \mathbb{E}\left[\|\mathbf{d}_k\|_{\mathsf{H}_k}^2\right] \leq \frac{\tilde{B}^2 \tilde{M}^2}{\tilde{b}\tilde{\zeta}^2} \sum_{k=1}^{n} \alpha_k.$$

(17)

Therefore, the assertion of Theorem 8 follows from (11) and (14)–(17). ∎

A.3 Proofs of Theorem 1 and Proposition 2

Proof [Proof of Theorem 1] Let $\boldsymbol{x} \in \mathbb{R}^d$, $\alpha_n := \alpha \in (0,1)$, $\beta_n := \beta = b \in (0,1)$, $\gamma_n := \gamma \in [0,+\infty)$, and $\delta_n := \delta \in [0,1/2]$. Define $X_n := \mathbb{E}[\|\boldsymbol{x}_n-\boldsymbol{x}\|_{\mathsf{H}_n}^2]$. Lemmas 5, 6, and 7 imply that, for all $n \in \mathbb{N}$,

$$X_{n+1} \leq X_n + \left(X_{n+1} - \mathbb{E}\left[\|\boldsymbol{x}_{n+1}-\boldsymbol{x}\|_{\mathsf{H}_n}^2\right]\right)$$
$$+ \frac{2\alpha\beta\tilde{M}\sqrt{Dd}}{\tilde{\zeta}} - \frac{2\alpha\tilde{b}\tilde{\gamma}}{1-\zeta^{n+1}}V_n(\boldsymbol{x}) + \frac{8\alpha\tilde{b}\delta\hat{M}\sqrt{Dd}}{\tilde{\zeta}} + \frac{\alpha^2\tilde{B}^2\tilde{M}^2}{\tilde{\zeta}^2}.$$

From $1-\zeta^{n+1} \leq 1$ and $(X_{n+1}-X_n)\zeta^{n+1} \leq X_{n+1}\zeta^{n+1}$, we have, for all $n \in \mathbb{N}$,

$$X_{n+1} \leq X_n + X_{n+1}\zeta^{n+1} + \left(X_{n+1} - \mathbb{E}\left[\|\boldsymbol{x}_{n+1}-\boldsymbol{x}\|_{\mathsf{H}_n}^2\right]\right)$$
$$+ \frac{2\alpha\beta\tilde{M}\sqrt{Dd}}{\tilde{\zeta}} - 2\alpha\tilde{b}\tilde{\gamma}V_n(\boldsymbol{x}) + \frac{8\alpha\tilde{b}\delta\hat{M}\sqrt{Dd}}{\tilde{\zeta}} + \frac{\alpha^2\tilde{B}^2\tilde{M}^2}{\tilde{\zeta}^2}.$$

(18)

We will show that, for all $\epsilon > 0$,

$$\liminf_{n\to+\infty} V_n(\boldsymbol{x}) \leq \frac{\alpha\tilde{B}^2\tilde{M}^2}{2\tilde{b}\tilde{\gamma}\tilde{\zeta}^2} + \frac{\beta\tilde{M}\sqrt{Dd}}{\tilde{b}\tilde{\gamma}\tilde{\zeta}} + \frac{4\delta\hat{M}\sqrt{Dd}}{\tilde{\gamma}\tilde{\zeta}} + \frac{Dd\epsilon}{2\tilde{b}\tilde{\gamma}} + \epsilon.$$

(19)

If (19) does not hold for all $\epsilon > 0$, then there exists $\epsilon_0 > 0$ such that

$$\liminf_{n\to+\infty} V_n(\boldsymbol{x}) > \frac{\alpha\tilde{B}^2\tilde{M}^2}{2\tilde{b}\tilde{\gamma}\tilde{\zeta}^2} + \frac{\beta\tilde{M}\sqrt{Dd}}{\tilde{b}\tilde{\gamma}\tilde{\zeta}} + \frac{4\delta\hat{M}\sqrt{Dd}}{\tilde{\gamma}\tilde{\zeta}} + \frac{Dd\epsilon_0}{2\tilde{b}\tilde{\gamma}} + \epsilon_0.$$

(20)

Assumptions (A4) and (A5) guarantee that there exists $n_0 \in \mathbb{N}$ such that $n \in \mathbb{N}$ with $n \geq n_0$ implies

$$\mathbb{E}\left[\sum_{i=1}^{d}(h_{n+1,i} - h_{n,i})\right] \leq \frac{d\alpha\epsilon_0}{2}, \tag{21}$$

which, together with (13), (21), and (A6), implies that, for all $n \geq n_0$,

$$X_{n+1} - \mathbb{E}\left[\|\boldsymbol{x}_{n+1} - \boldsymbol{x}\|_{\mathsf{H}_n}^2\right] \leq \frac{Dd\alpha\epsilon_0}{2}. \tag{22}$$

The condition $\zeta \in [0, 1)$ implies that there exists $n_1 \in \mathbb{N}$ such that, for all $n \geq n_1$,

$$X_{n+1}\zeta^{n+1} \leq \frac{Dd\alpha\epsilon_0}{2}. \tag{23}$$

Accordingly, (18), (22), and (23) ensure that, for all $n \geq \max\{n_0, n_1\}$,

$$\begin{aligned} X_{n+1} \leq X_n &+ Dd\alpha\epsilon_0 - 2\alpha\tilde{b}V_n(\boldsymbol{x}) \\ &+ \frac{2\alpha\beta\tilde{M}\sqrt{Dd}}{\tilde{\zeta}} + \frac{8\alpha\tilde{b}\delta\hat{M}\sqrt{Dd}}{\tilde{\zeta}} + \frac{\alpha^2\tilde{B}^2\tilde{M}^2}{\tilde{\zeta}^2}. \end{aligned} \tag{24}$$

Meanwhile, there exists $n_2 \in \mathbb{N}$ such that, for all $n \geq n_2$, $\liminf_{n\to+\infty} V_n(\boldsymbol{x}) - \epsilon_0/2 \leq V_n(\boldsymbol{x})$. Hence, (20) guarantees that, for all $n \geq n_2$,

$$V_n > \frac{\alpha\tilde{B}^2\tilde{M}^2}{2\tilde{b}\tilde{\gamma}\tilde{\zeta}^2} + \frac{\beta\tilde{M}\sqrt{Dd}}{\tilde{b}\tilde{\gamma}\tilde{\zeta}} + \frac{4\delta\hat{M}\sqrt{Dd}}{\tilde{\gamma}\tilde{\zeta}} + \frac{Dd\epsilon_0}{2\tilde{b}\tilde{\gamma}} + \frac{\epsilon_0}{2}. \tag{25}$$

Therefore, (24) and (25) ensure that, for all $n \geq n_3 := \max\{n_0, n_1, n_2\}$,

$$\begin{aligned} X_{n+1} &< X_n + Dd\alpha\epsilon_0 - 2\alpha\tilde{b}\tilde{\gamma}\left\{\frac{\alpha\tilde{B}^2\tilde{M}^2}{2\tilde{b}\tilde{\gamma}\tilde{\zeta}^2} + \frac{\beta\tilde{M}\sqrt{Dd}}{\tilde{b}\tilde{\gamma}\tilde{\zeta}} + \frac{4\delta\hat{M}\sqrt{Dd}}{\tilde{\gamma}\tilde{\zeta}} + \frac{Dd\epsilon_0}{2\tilde{b}\tilde{\gamma}} + \frac{\epsilon_0}{2}\right\} \\ &\quad + \frac{2\alpha\beta\tilde{M}\sqrt{Dd}}{\tilde{\zeta}} + \frac{8\alpha\tilde{b}\delta\hat{M}\sqrt{Dd}}{\tilde{\zeta}} + \frac{\alpha^2\tilde{B}^2\tilde{M}^2}{\tilde{\zeta}^2} \\ &= X_n - \alpha\tilde{b}\tilde{\gamma}\epsilon_0 < X_{n_3} - \alpha\tilde{b}\tilde{\gamma}\epsilon_0(n + 1 - n_3), \end{aligned}$$

which is a contradiction since the right-hand side of the final inequality approaches minus infinity as n approaches positive infinity. Therefore, (19) holds for all $\epsilon > 0$; that is, we have

$$\liminf_{n\to+\infty} V_n(\boldsymbol{x}) \leq \frac{\alpha\tilde{B}^2\tilde{M}^2}{2\tilde{b}\tilde{\gamma}\tilde{\zeta}^2} + \frac{\beta\tilde{M}\sqrt{Dd}}{\tilde{b}\tilde{\gamma}\tilde{\zeta}} + \frac{4\delta\hat{M}\sqrt{Dd}}{\tilde{\gamma}\tilde{\zeta}}.$$

Theorem 8 implies the assertions in Theorem 1. This completes the proof. ∎

25

Proof [Proof of Proposition 2] Since f_i is convex for $i \in [T]$, we have that, for all $\boldsymbol{x} \in \mathbb{R}^d$ and all $n \in \mathbb{N}$,

$$\mathbb{E}[f(\boldsymbol{x}_n) - f^\star] \leq V_n(\boldsymbol{x}),$$

$$\min_{k \in [n]} \mathbb{E}[f(\boldsymbol{x}_k) - f^\star], \mathbb{E}[f(\tilde{\boldsymbol{x}}_n) - f^\star] \leq \frac{1}{n} \sum_{k=1}^{n} V_k(\boldsymbol{x}),$$

$$R(T) = \sum_{t=1}^{T} (f_t(\boldsymbol{x}_t) - f_t(\boldsymbol{x}^\star)) \leq \sum_{t=1}^{T} \langle \boldsymbol{x}_t - \boldsymbol{x}^\star, \mathsf{G}(\boldsymbol{x}_t, \xi_t) \rangle.$$

Accordingly, Theorems 1 and 8 imply Proposition 2. ∎

A.4 Proofs of Theorem 3 and Proposition 4

Proof [Proof of Theorem 3] Lemmas 5 and 6 imply that, for all $\boldsymbol{x} \in \mathbb{R}^d$ and all $k \in \mathbb{N}$,

$$2\alpha_k V_k(\boldsymbol{x}) \leq X_k - X_{k+1} + X_{k+1}\zeta^{k+1}$$

$$+ \left(X_{k+1} - \mathbb{E}\left[\|\boldsymbol{x}_{k+1} - \boldsymbol{x}\|_{\mathsf{H}_k}^2 \right] \right) + \frac{\tilde{B}^2 \tilde{M}^2}{\tilde{\zeta}^2} \alpha_k^2 + 2\alpha_k \beta_k \mathbb{E}\left[\langle \boldsymbol{x} - \boldsymbol{x}_k, \boldsymbol{m}_{k-1} \rangle \right]$$

$$+ 2\alpha_k \tilde{\beta}_k \delta_k \mathbb{E}\left[\langle \boldsymbol{x}_k - \boldsymbol{x}, \mathsf{G}_{k-1} \rangle \right] - 2\{\alpha_k \gamma_k - \alpha_k \beta_k (1 + \gamma_k)\} V_k(\boldsymbol{x}),$$

which in turn implies that

$$2\alpha_k V_k(\boldsymbol{x}) \leq X_k - X_{k+1} + X^* \zeta^{k+1}$$

$$+ D\mathbb{E}\left[\sum_{i=1}^{d} (h_{k+1,i} - h_{k,i}) \right] + \frac{\tilde{B}^2 \tilde{M}^2}{\tilde{\zeta}^2} \alpha_k^2 + 2\tilde{M}\sqrt{Dd}\alpha_k\beta_k + 8\hat{M}\sqrt{Dd}\alpha_k\delta_k$$

$$+ 2F\{\alpha_k \gamma_k + \alpha_k \beta_k (1 + \gamma_k)\},$$

where $X^* := \sup\{X_n \colon n \in \mathbb{N}\}$ and $F := \sup\{|V_n(\boldsymbol{x})| \colon n \in \mathbb{N}\}$ are finite by (A1) and (A6). Summing the above inequality from $k = 0$ to $k = n$ implies that, for all $\boldsymbol{x} \in \mathbb{R}^d$,

$$2\sum_{k=0}^{n} \alpha_k V_k(\boldsymbol{x}) \leq X_0 + X^* \sum_{k=0}^{n} \zeta^{k+1} + D\mathbb{E}\left[\sum_{i=1}^{d} (h_{n+1,i} - h_{0,i}) \right]$$

$$+ \frac{\tilde{B}^2 \tilde{M}^2}{\tilde{\zeta}^2} \sum_{k=0}^{n} \alpha_k^2 + 2\tilde{M}\sqrt{Dd} \sum_{k=0}^{n} \alpha_k\beta_k$$

$$+ 8\hat{M}\sqrt{Dd} \sum_{k=0}^{n} \alpha_k\delta_k + 4F \sum_{k=0}^{n} \alpha_k\gamma_k$$

$$+ 2F \sum_{k=0}^{n} \alpha_k\beta_k.$$

From (A5), $\zeta \in [0,1)$, $\sum_{n=0}^{+\infty} \alpha_n^2 < +\infty$, $\sum_{n=0}^{+\infty} \alpha_n \beta_n < +\infty$, $\sum_{n=0}^{+\infty} \alpha_n \gamma_n < +\infty$, and $\sum_{n=0}^{+\infty} \alpha_n \delta_n < +\infty$, we have

$$\sum_{k=0}^{+\infty} \alpha_k V_k < +\infty, \tag{26}$$

which, together with $\sum_{n=0}^{+\infty} \alpha_n = +\infty$, implies that

$$\liminf_{n \to +\infty} V_n(\boldsymbol{x}) \leq 0.$$

Suppose that $(\alpha_n)_{n\in\mathbb{N}}$, $(\beta_n)_{n\in\mathbb{N}}$, $(\gamma_n)_{n\in\mathbb{N}}$, and $(\delta_n)_{n\in\mathbb{N}}$ satisfy $\lim_{n\to+\infty} 1/(n\alpha_n) = 0$, $\lim_{n\to+\infty}(1/n)\sum_{k=0}^{n}\alpha_k = 0$, $\lim_{n\to+\infty}(1/n)\sum_{k=0}^{n}\beta_k = 0$, $\lim_{n\to+\infty}(1/n)\sum_{k=0}^{n}\delta_k = 0$, and $\gamma_n \geq 0$ $(n \in \mathbb{N})$. Then, Theorem 8 implies that, for all $\boldsymbol{x} \in \mathbb{R}^d$,

$$\limsup_{n \to +\infty} \frac{1}{n} \sum_{k=1}^{n} V_k(\boldsymbol{x}) \leq 0.$$

In the case of $\alpha_n := 1/n^\eta$ $(\eta \in (0,1))$, $\beta_n := \beta^n$ $(\beta \in [0,1))$, $\delta_n := \delta^n$ $(\delta \in [0,1))$, and a monotone decreasing sequence $(\gamma_n)_{n\in\mathbb{N}}$, we have $\kappa_{n+1} \leq \kappa_n$ $(n \in \mathbb{N})$ and $\limsup_{n\to+\infty} \beta_n < 1$. We also have

$$\lim_{n \to +\infty} \frac{1}{n\alpha_n} = \lim_{n \to +\infty} \frac{1}{n^{1-\eta}} = 0,$$
$$\frac{1}{n} \sum_{k=1}^{n} \alpha_k \leq \frac{1}{n}\left(1 + \int_1^n \frac{\mathrm{d}t}{t^\eta}\right) \leq \frac{1}{(1-\eta)n^\eta}.$$

Furthermore, $\sum_{k=1}^{n} \beta_k \leq \sum_{k=1}^{+\infty} \beta_k = \beta/(1-\beta)$ and $\sum_{k=1}^{n} \delta_k \leq \sum_{k=1}^{+\infty} \delta_k = \delta/(1-\delta)$. Therefore, Theorem 8 gives us the convergence rates in Theorem 3. This completes the proof. ∎

Proof [Proof of Proposition 4] It is sufficient to show that any accumulation point of $(\tilde{\boldsymbol{x}}_n)_{n\in\mathbb{N}}$ belongs to $X^\star$ almost surely. Theorem 6 and the proof of Proposition 5 imply that $\lim_{n\to+\infty} \mathbb{E}[f(\tilde{\boldsymbol{x}}_n) - f^\star] = 0$. Let $\hat{\boldsymbol{x}} \in \mathbb{R}^d$ be an arbitrary accumulation point of $(\tilde{\boldsymbol{x}}_n)_{n\in\mathbb{N}} \subset \mathbb{R}^d$. Then, there exists $(\tilde{\boldsymbol{x}}_{n_i})_{i\in\mathbb{N}} \subset (\tilde{\boldsymbol{x}}_n)_{n\in\mathbb{N}}$ such that $(\tilde{\boldsymbol{x}}_{n_i})_{i\in\mathbb{N}}$ converges almost surely to $\hat{\boldsymbol{x}}$. The continuity of f implies that $\mathbb{E}[f(\hat{\boldsymbol{x}}) - f^\star] = 0$, and hence, $\hat{\boldsymbol{x}} \in X^\star$. The remaining assertions in Proposition 4 follow from the proof of Proposition 2. ∎

A.5 Algorithms for image classification

A.5.1 RESNET-18 ON CIFAR-100 DATASET

The training of ResNet-18 on the CIFAR-100 dataset is based on the results in `https://github.com/weiaicunzai/pytorch-cifar100`.

Constant learning rates: A learning rate with cosine annealing was used (Loshchilov and Hutter, 2017). The initial constant learning rates used in the experiments were determined with a grid search ($\alpha \in \{10^{-3}, 5 \times 10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}\}$).

- SGD (Robbins and Monro, 1951): $\alpha = 5 \times 10^{-2}$

- Momentum[3] (Polyak, 1964; Nesterov, 1983): $\alpha = 10^{-1}$, $\beta_n = 0.9$, and the weight decay is 5×10^{-4}

- RMSprop (Tieleman and Hinton, 2012): $\alpha = 10^{-2}$ and the parameter α is 0.9

- Adagrad (Duchi et al., 2011): $\alpha = 10^{-2}$

- AdamW (Loshchilov and Hutter, 2019): $\alpha = 10^{-3}$ and the weight decay is 10^{-2}

- Adam (Kingma and Ba, 2015): Algorithm 1 with (8), $\alpha = 10^{-2}$, $\zeta = \beta_n = 0.9$, $\theta = 0.999$, and $\gamma_n = \delta_n = 0$

- AMSGrad (Reddi et al., 2018): Algorithm 1 with (9), $\alpha = 10^{-3}$, $\zeta = 0$, $\beta_n = 0.9$, $\theta = 0.999$, and $\gamma_n = \delta_n = 0$

- SCGAdam-C: Algorithm 1 with (8), $\alpha = 10^{-3}$, $\zeta = \beta_n = 0.9$, $\theta = 0.999$, $\gamma_n = 10^{-1}$, and $\delta_n = 10^{-3}$

- SCGAMSG-C: Algorithm 1 with (9), $\alpha = 10^{-3}$, $\zeta = 0$, $\theta = 0.999$, $\beta_n = 0.9$, $\gamma_n = 10^{-1}$, and $\delta_n = 10^{-3}$

Diminishing learning rates: All optimizers used $\alpha_n = 1/\sqrt{n}$.

- SGD (Robbins and Monro, 1951)

- Momentum (Polyak, 1964; Nesterov, 1983): $\beta_n = 1/2^n$

- RMSprop (Tieleman and Hinton, 2012): the parameter α is $1/2^n$

- Adagrad (Duchi et al., 2011):

- AdamW (Loshchilov and Hutter, 2019): the weight decay is 10^{-2} and $\beta_n = 1/2^n$

- Adam (Kingma and Ba, 2015): Algorithm 1 with (8), $\zeta = 0.9$, $\theta = 0.999$, $\beta_n = 1/2^n$, and $\gamma_n = \delta_n = 0$

- AMSGrad (Reddi et al., 2018): Algorithm 1 with (9), $\zeta = 0$, $\theta = 0.999$, $\beta_n = 1/2^n$, and $\gamma_n = \delta_n = 0$

- SCGAdam-D: Algorithm 1 with (8), $\zeta = 0.9$, $\theta = 0.999$, and $\beta_n = \gamma_n = \delta_n = 1/2^n$

- SCGAMSG-D: Algorithm 1 with (8), $\zeta = 0$, $\theta = 0.999$, and $\beta_n = \gamma_n = \delta_n = 1/2^n$

---

3. `https://github.com/kuangliu/pytorch-cifar`

A.5.2 RESNET-18 ON CIFAR-10 DATASET

The training of ResNet-18 on the CIFAR-10 dataset is based on the results in `https://github.com/kuangliu/pytorch-cifar`.

Constant learning rates: A learning rate with cosine annealing was used (Loshchilov and Hutter, 2017). The initial constant learning rates used in the experiments were determined with a grid search ($\alpha \in \{10^{-3}, 5 \times 10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}\}$).

- SGD (Robbins and Monro, 1951): $\alpha = 5 \times 10^{-2}$

- Momentum[4] (Polyak, 1964; Nesterov, 1983): $\alpha = 10^{-1}$, $\beta_n = 0.9$, and the weight decay is 5×10^{-4}

- RMSprop (Tieleman and Hinton, 2012): $\alpha = 10^{-2}$ and the parameter α is 0.9

- Adagrad (Duchi et al., 2011): $\alpha = 10^{-2}$

- AdamW (Loshchilov and Hutter, 2019): $\alpha = 10^{-3}$ and the weight decay is 10^{-2}

- Adam (Kingma and Ba, 2015): Algorithm 1 with (8), $\alpha = 10^{-2}$, $\zeta = \beta_n = 0.9$, $\theta = 0.999$, and $\gamma_n = \delta_n = 0$

- AMSGrad (Reddi et al., 2018): Algorithm 1 with (9), $\alpha = 10^{-3}$, $\zeta = 0$, $\beta_n = 0.9$, $\theta = 0.999$, and $\gamma_n = \delta_n = 0$

- SCGAdam-C: Algorithm 1 with (8), $\alpha = 10^{-3}$, $\zeta = \beta_n = 0.9$, $\theta = 0.999$, $\gamma_n = 10^{-1}$, and $\delta_n = 10^{-2}$

- SCGAMSG-C: Algorithm 1 with (9), $\alpha = 10^{-3}$, $\zeta = 0$, $\theta = 0.999$, $\beta_n = 0.9$, $\gamma_n = 10^{-1}$, and $\delta_n = 10^{-2}$

Diminishing learning rates: All optimizers used $\alpha_n = 1/\sqrt{n}$.

- SGD (Robbins and Monro, 1951)

- Momentum (Polyak, 1964; Nesterov, 1983): $\beta_n = 1/2^n$

- RMSprop (Tieleman and Hinton, 2012): the parameter α is $1/2^n$

- Adagrad (Duchi et al., 2011):

- AdamW (Loshchilov and Hutter, 2019): the weight decay is 10^{-2} and $\beta_n = 1/2^n$

- Adam (Kingma and Ba, 2015): Algorithm 1 with (8), $\zeta = 0.9$, $\theta = 0.999$, $\beta_n = 1/2^n$, and $\gamma_n = \delta_n = 0$

- AMSGrad (Reddi et al., 2018): Algorithm 1 with (9), $\zeta = 0$, $\theta = 0.999$, $\beta_n = 1/2^n$, and $\gamma_n = \delta_n = 0$

- SCGAdam-D: Algorithm 1 with (8), $\zeta = 0.9$, $\theta = 0.999$, and $\beta_n = \gamma_n = \delta_n = 1/2^n$

- SCGAMSG-D: Algorithm 1 with (8), $\zeta = 0$, $\theta = 0.999$, and $\beta_n = \gamma_n = \delta_n = 1/2^n$

---

4. `https://github.com/kuangliu/pytorch-cifar`

A.6 Algorithms for text classification

A.6.1 LSTM ON IMDB DATASET

Constant learning rates: All optimizers used $\alpha = 10^{-3}$. The existing optimizers together with their default values are in `torch.optim`[5].

- SGD (Robbins and Monro, 1951)

- Momentum (Polyak, 1964; Nesterov, 1983): $\beta_n = 0.9$

- RMSprop (Tieleman and Hinton, 2012): the parameter α is 0.99

- Adagrad (Duchi et al., 2011)

- AdamW (Loshchilov and Hutter, 2019): the weight decay is 10^{-2}

- Adam (Kingma and Ba, 2015): Algorithm 1 with (8), $\zeta = \beta_n = 0.9$, $\theta = 0.999$, and $\gamma_n = \delta_n = 0$

- AMSGrad (Reddi et al., 2018): Algorithm 1 with (9), $\zeta = 0$, $\beta_n = 0.9$, $\theta = 0.999$, and $\gamma_n = \delta_n = 0$

- SCGAdam-C: Algorithm 1 with (8), $\zeta = \beta_n = 0.9$, $\theta = 0.999$, $\gamma_n = 1$, and $\delta_n = 10^{-2}$

- SCGAMSG-C: Algorithm 1 with (9), $\zeta = 0$, $\theta = 0.999$, $\beta_n = 0.9$, $\gamma_n = 1$, and $\delta_n = 10^{-3}$

Diminishing learning rates: All optimizers used $\alpha_n = 1/\sqrt{n}$. The existing optimizers together with their default values are in `torch.optim`.

- SGD (Robbins and Monro, 1951)

- Momentum (Polyak, 1964; Nesterov, 1983): $\beta_n = 1/2^n$

- RMSprop (Tieleman and Hinton, 2012): the parameter is $1/2^n$

- Adagrad (Duchi et al., 2011)

- AdamW (Loshchilov and Hutter, 2019): the weight decay is 10^{-2} and $\beta_n = 1/2^n$

- Adam (Kingma and Ba, 2015): Algorithm 1 with (8), $\zeta = 0.9$, $\theta = 0.999$, $\beta_n = 1/2^n$, and $\gamma_n = \delta_n = 0$

- AMSGrad (Reddi et al., 2018): Algorithm 1 with (9), $\zeta = 0$, $\theta = 0.999$, $\beta_n = 1/2^n$, and $\gamma_n = \delta_n = 0$

- SCGAdam-D: Algorithm 1 with (8), $\zeta = 0.9$, $\theta = 0.999$, and $\beta_n = \gamma_n = \delta_n = 1/2^n$

- SCGAMSG-D: Algorithm 1 with (8), $\zeta = 0$, $\theta = 0.999$, and $\beta_n = \gamma_n = \delta_n = 1/2^n$

Table 2: Hyperparameters of the optimizer used in the training DCGAN in Section 4.4.

optimizer	α^D	α^G	$\beta_1^G = \beta_1^D$	$\beta_2^G = \beta_2^D$	β_1 and β_2's reference
Adam	0.0003	0.0001	0.5	0.999	(Radford et al., 2016)
AdaBelief	0.00003	0.0003	0.5	0.999	(Zhuang et al., 2020)
RMSProp	0.00003	0.0001	0	0.99	
SCGAdam	0.0001	0.0003	0.5	0.999	
SCGAMSGrad	0.00005	0.0005	0.5	0.999	

Table 3: Hyperparameters of the optimizer used in the training SNGAN in Section 4.4.

optimizer	α^D	α^G	$\beta_1^G = \beta_1^D$	$\beta_2^G = \beta_2^D$
Adam	0.0001	0.0001	0.5	0.999
AdaBelief	0.0001	0.0001	0.5	0.999
RMSProp	0.00005	0.0001	0	0.99
SCGAdam	0.0001	0.0003	0.5	0.999
SCGAMSGrad	0.0001	0.0003	0.5	0.999

Table 4: Hyperparameters of the optimizer used in the training WGAN-GP in Section 4.4.

optimizer	α^D	α^G	$\beta_1^G = \beta_1^D$	$\beta_2^G = \beta_2^D$	β_1 and β_2's reference
Adam	0.0003	0.0001	0.5	0.999	(Gulrajani et al., 2017)
AdaBelief	0.00005	0.0005	0.5	0.999	(Zhuang et al., 2020)
RMSProp	0.0003	0.0005	0	0.99	
SCGAdam	0.0003	0.0001	0.5	0.999	
SCGAMSGrad	0.0003	0.0001	0.5	0.999	
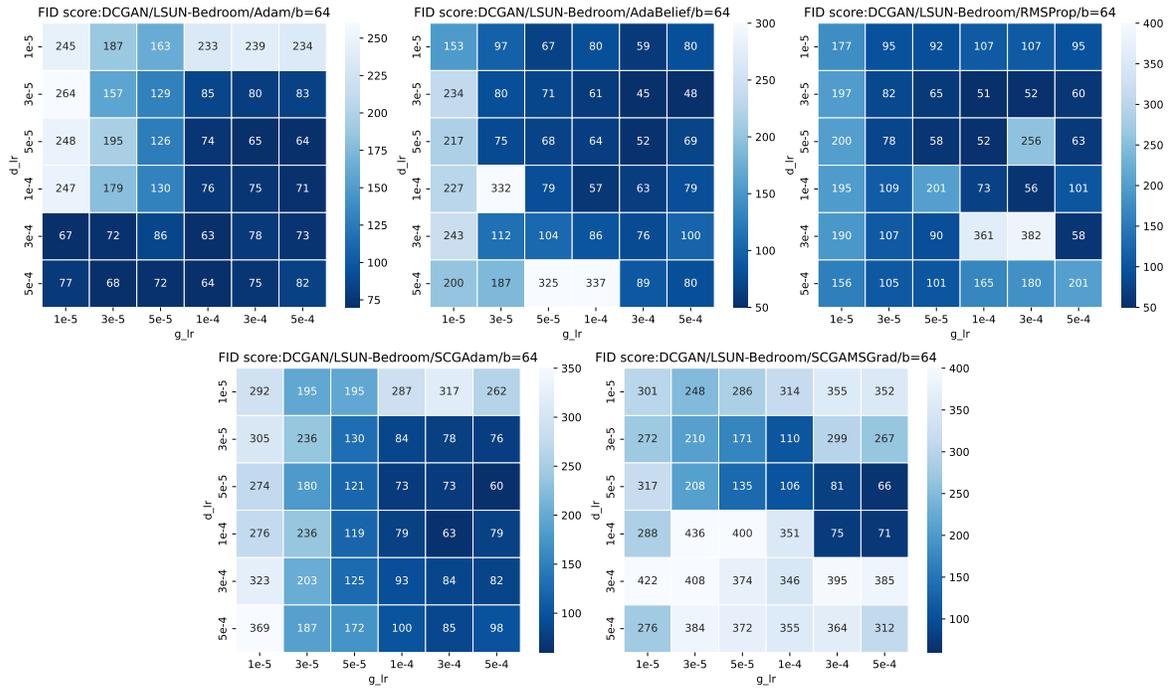
Figure 22: Analysis of the relationship between combination of learning rates and FID score in training DCGAN on the LSUN-Bedroom dataset: discriminator learning rate α^D on the vertical axis and generator learning rate α^G on the horizontal axis. The heatmap colors denote the FID scores; the darker the blue, the lower the FID, meaning that the training of the generator succeeded.
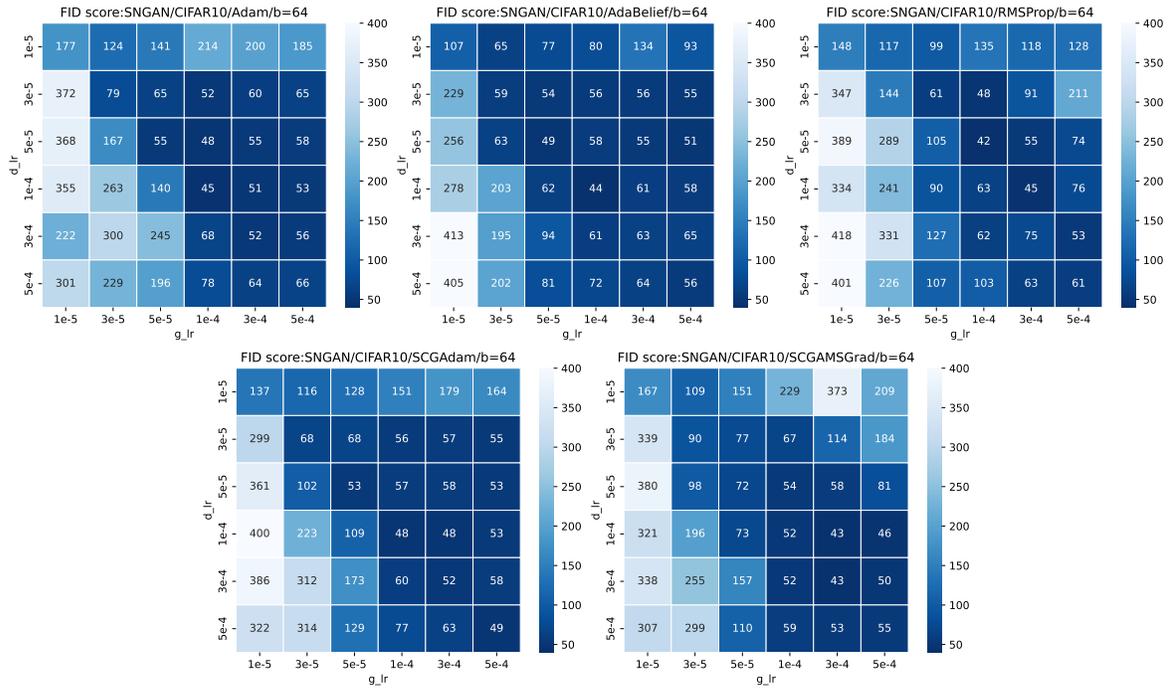
Figure 23: Analysis of the relationship between combination of learning rates and FID score in training SNGAN on the CIFAR10 dataset: discriminator learning rate α^D on the vertical axis and generator learning rate α^G on the horizontal axis. The heatmap colors denote the FID scores; the darker the blue, the lower the FID, meaning that the training of the generator succeeded.
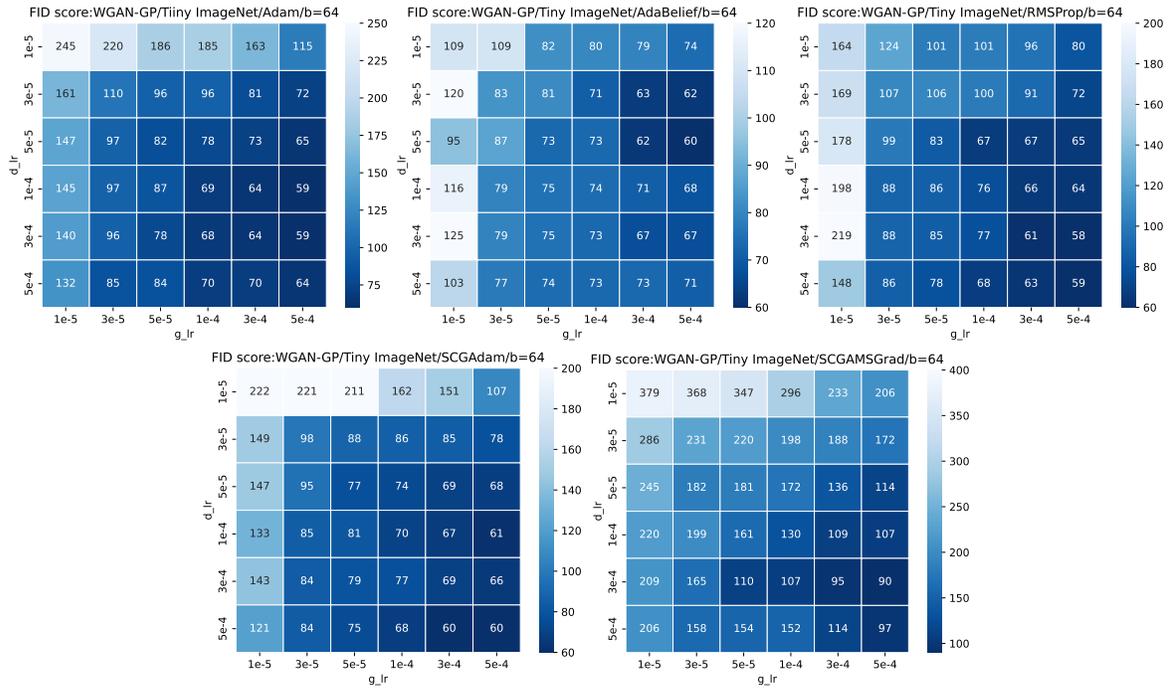
Figure 24: Analysis of the relationship between combination of learning rates and FID score in training WGAN-GP on the Tiny ImageNet dataset: discriminator learning rate α^D on the vertical axis and generator learning rate α^G on the horizontal axis. The heatmap colors denote the FID scores; the darker the blue, the lower the FID, meaning that the training of the generator succeeded.

A.7 Algorithms for image generation

The combinations of learning rates used in the experiments in Section 4.4 were determined using a grid search. Figures 22-24 shows the results of the grid search.

References

Neculai Andrei. Scaled conjugate gradient algorithms for unconstrained optimization. *Computational Optimization and Applications*, 38(3):401–416, 2007.

Hao Chen, Lili Zheng, Raed Al Kontar, and Garvesh Raskutti. Stochastic gradient descent in correlated settings: A study on Gaussian processes. In *Advances in Neural Information Processing Systems*, volume 33, pages 1–12, 2020.

Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of Adam-type algorithms for non-convex optimization. In *Proceedings of The International Conference on Learning Representations*, pages 1–30, 2019.

Wanyou Cheng. A two-term PRP-based descent method. *Numerical Functional Analysis and Optimization*, 28(11–12):1217–1230, 2007.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Benjamin Fehrman, Benjamin Gess, and Arnulf Jentzen. Convergence rates for the stochastic gradient descent method for non-convex objective functions. *Journal of Machine Learning Research*, 21(136):1–48, 2020.

Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. `http://arxiv.org/abs/1701.00160`, 2017.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, volume 30, pages 5769–5779, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, pages 770–778, 2015.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, volume 30, pages 6629–6640, 2017.

Hideaki Iiduka. Appropriate learning rates of adaptive learning rate optimization algorithms for training deep neural networks. *IEEE Transactions on Cybernetics*, 2022.

Hideaki Iiduka and Yu Kobayashi. Training deep neural networks using conjugate gradient-like methods. *Electronics*, 9(11), 2020.

---

5. `https://pytorch.org/docs/stable/optim.html`

Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *Proceedings of The International Conference on Learning Representations*, pages 1–15, 2015.

Nicolas Loizou, Sharan Vaswani, Issam Laradji, and Simon Lacoste-Julien. Stochastic polyak step-size for SGD: An adaptive learning rate for fast convergence. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130, pages 1–11, 2021.

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

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

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150. Association for Computational Linguistics, 2011.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.

Wataru Nakamura, Yasushi Narushima, and Hiroshi Yabe. Nonlinear conjugate gradient methods with sufficient descent properties for unconstrained optimization. *Journal of Industrial & Management Optimization*, 9(3):595–619, 2013.

Yasushi Narushima and Hiroshi Yabe. A survey of sufficient descent conjugate gradient methods for unconstrained optimization. *SUT Journal of Mathematics*, 50(2):167–203, 2014.

Yurii Evgen'evich Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Doklady Akademii Nauk SSSR*, 269:543–547, 1983.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, 2nd edition, 2006.

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

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of the International Conference on Learning Representations*, 2016.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of Adam and beyond. In *Proceedings of The International Conference on Learning Representations*, pages 1–23, 2018.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22:400–407, 1951.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Proceedings of the 29th Advances in Neural Information Processing Systems*, pages 2226–2234, 2016.

Kevin Scaman and Cédric Malherbe. Robustness analysis of non-convex stochastic gradient descent using biased expectations. In *Advances in Neural Information Processing Systems*, volume 33, pages 1–11, 2020.

Tijmen Tieleman and Geoffrey Hinton. RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Xuan Yang Ya Le. Tiny imagenet visual recognition challenge. *CS 231N*, 7, 2015.

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. `https://arxiv.org/abs/1506.03365`, 2015.

Li Zhang, Weijun Zhou, and Donghui Li. Global convergence of a modified Fletcher–Reeves conjugate gradient method with Armijo-type line search. *Numerische Mathematik*, 104 (4):561–572, 2006.

Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James S. Duncan. AdaBelief optimizer: Adapting stepsizes by the belief in observed gradients. In *Advances in Neural Information Processing Systems*, volume 33, pages 1–29, 2020.