2022

# Adaptive strategies in non-convex optimization

BOSTON UNIVERSITY

GRADUATE SCHOOL OF ARTS AND SCIENCES

Dissertation

# ADAPTIVE STRATEGIES IN NON-CONVEX

# OPTIMIZATION

by

## ZHENXUN ZHUANG

B.Eng., University of Science and Technology of China, 2016

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2022

Approved by

First Reader

Francesco Orabona, Ph.D.
Associate Professor of Electrical and Computer Engineering
Associate Professor of Computer Science
Associate Professor of Systems Engineering
Associate Professor of Computing and Data Sciences

Second Reader

Bryan A. Plummer, Ph.D.
Assistant Professor of Computer Science

Third Reader

Ioannis Ch. Paschalidis, Ph.D.
Distinguished Professor of Engineering
Professor of Electrical and Computer Engineering
Professor of Systems Engineering
Professor of Biomedical Engineering
Professor of Computing and Data Sciences

*God, give me grace to accept with serenity*
*the things that cannot be changed,*
*Courage to change the things*
*which should be changed,*
*and the wisdom to distinguish*
*the one from the other.*

*Living one day at a time,*
*Enjoying one moment at a time,*
*Accepting hardship as a pathway to peace,*
*Taking, as Jesus did,*
*This sinful world as it is,*
*Not as I would have it.*

<div align="right">Reinhold Niebuhr</div>

*She was still too young to know that life never gives anything for nothing,*
*and that a price is always exacted for what fate bestows.*

<div align="right">Stefan Zweig</div>

# Acknowledgments

The journey toward my Ph.D. has eventually come to an end. It is full of obstacles and setbacks, yet is also filled with joy and achievements. So many people have helped me along the way and I am forever indebted to them.

First and foremost, I would like to express my deepest gratitude to my adviser Francesco Orabona without whom this adventure would be impossible. I knew literally nothing about research in machine learning upon entering my Ph.D. and it is him who tirelessly and patiently taught me right from wrong and trained me to build a full skill-set on being a researcher. He will always be my role model for his passion for life and his rigor towards work.

I also thank all my collaborators: Ashok Cutkosky, Xiaoyu Li, Mingrui Liu, Songtao Lu, Yunlong Wang, Kezi Yu, and a lot more. I will always remember those inspiring discussions on new problems and those sleepless nights catching deadlines.

Special thanks go to my committee members Alina Ene, Yannis Paschalidis, and Bryan Plummer for their service and their attention to my work. I also thank all the people in the department and the university, some of whom I became friends with. Altogether you created a really enjoyable atmosphere to be working in.

Last but not least, I thank my parents Shaobing and Xiuyue for their unconditional support and my love Xiaoqing for fighting with me till the end.

# ADAPTIVE STRATEGIES IN NON-CONVEX OPTIMIZATION

## ZHENXUN ZHUANG

Boston University, Graduate School of Arts and Sciences, 2022

Major Professor: Francesco Orabona,
Associate Professor of Electrical and Computer Engineering
Associate Professor of Computer Science
Associate Professor of Systems Engineering
Associate Professor of Computing and Data Sciences

## ABSTRACT

Modern applications in machine learning have seen more and more usage of non-convex formulations in that they can often better capture the problem structure. One prominent example is the Deep Neural Networks which have achieved innumerable successes in various fields including computer vision and natural language processing. However, optimizing a non-convex problem presents much greater difficulties compared with convex ones. A vastly popular optimizer used for such scenarios is Stochastic Gradient Descent (SGD), but its performance depends crucially on the choice of its step sizes. Tuning of step sizes is notoriously laborious and the optimal choice can vary drastically across different problems. To save the labor of tuning, adaptive algorithms come to the rescue: An algorithm is said to be adaptive to a certain parameter (of the problem) if it does not need a priori knowledge of such parameter but performs competitively to those that know it.

This dissertation presents our work on adaptive algorithms in following scenarios:

1. In the stochastic optimization setting, we only receive stochastic gradients and the level of noise in evaluating them greatly affects the convergence rate. Tuning is typically required when without prior knowledge of the noise scale in order to achieve the optimal rate. Considering this, we designed and analyzed noise-adaptive algorithms that can automatically ensure (near)-optimal rates under different noise scales without knowing it.

2. In training deep neural networks, the scales of gradient magnitudes in each coordinate can scatter across a very wide range unless normalization techniques, like BatchNorm, are employed. In such situations, algorithms not addressing this problem of gradient scales can behave very poorly. To mitigate this, we formally established the advantage of scale-free algorithms that adapt to the gradient scales and presented its real benefits in empirical experiments.

3. Traditional analyses in non-convex optimization typically rely on the smoothness assumption. Yet, this condition does not capture the properties of some deep learning objective functions, including the ones involving Long Short-Term Memory (LSTM) networks and Transformers. Instead, they satisfy a much more relaxed condition, with potentially unbounded smoothness. Under this condition, we show that a generalized SignSGD (update using only the sign of each coordinate of the stochastic gradient vector when running SGD) algorithm can theoretically match the best-known convergence rates obtained by SGD with gradient clipping but does not need explicit clipping at all, and it can empirically match the performance of Adam and beat others. Moreover, it can also be made to automatically adapt to the unknown relaxed smoothness.

# Contents

# List of Tables

# List of Figures

# List of Abbreviations

| | | |
|---|---|---|
| ACM | . . . . . | Association for Computing Machinery |
| BN | . . . . . | Batch Normalization |
| CNN | . . . . . | Convolutional Neural Network |
| CV | . . . . . | Computer Vision |
| DNN | . . . . . | Deep Neural Network |
| $\mathbb{E}[\cdot]$ | . . . . . | Expectation |
| FTRL | . . . . . | Follow The Regularized Leader |
| GD | . . . . . | Gradient Descent |
| ICLR | . . . . . | International Conference on Learning Representations |
| ICML | . . . . . | International Conference on Machine Learning |
| IEEE | . . . . . | Institute of Electrical and Electronics Engineers |
| JMLR | . . . . . | Journal of Machine Learning Research |
| LSTM | . . . . . | Long Short-Term Memory |
| ML | . . . . . | Machine Learning |
| NeurIPS | . . . . . | Neural Information Processing Systems |
| NLP | . . . . . | Natural Language Processing |
| PL | . . . . . | Polyak-Łojasiewicz |
| PMLR | . . . . . | Proceedings of Machine Learning Research |
| $\mathbb{P}[\cdot]$ | . . . . . | Probability |
| $\mathbb{R}^d$ | . . . . . | Real coordinate space of dimension $d$ |
| r.h.s. | . . . . . | right hand side |
| SGD | . . . . . | Stochastic Gradient Descent |
| SOTA | . . . . . | State-Of-The-Art |
| w.r.t. | . . . . . | with respect to |

# Chapter 1

# Introduction

Recent decades have witnessed a surge of interest in the field of *machine learning* (Jordan and Mitchell, 2015). As the name suggests, this discipline strives to build machines that can improve automatically through experience obtained by learning from data. Apart from the efforts of researchers in developing novel theories and algorithms, its rapid advancement is also partly attributed to the accumulation of vast datasets and the explosive growth of the semiconductor industry which resulted in efficient and low-cost computations.

Most machine learning tasks can be modeled as the mathematical optimization problem

$$\min_{\boldsymbol{x} \in \mathcal{X}} F(\boldsymbol{x}), \tag{1.1}$$

where we assume the domain $\mathcal{X} \subseteq \mathbb{R}^d$ and call $F : \mathbb{R}^d \to \mathbb{R}$ the objective function. We also make the following assumption on $F$.

**Assumption 1.1.** *$F$ is differentiable and bounded from below by $F^*$.*

## 1.1 Convex and Non-convex Optimization

By assuming different conditions on $F$ and $\mathcal{X}$, we restrict our focus to different families or classes of optimization problems.

A classic and important class of optimization problems is convex optimization problems (Rockafellar, 1976; Boyd and Vandenberghe, 2004) in which the domain $\mathcal{X}$ is a convex set and the objective function $F$ is a convex function.

We say a set $\mathcal{X}$ is *convex* if the line segment between any two points in $\mathcal{X}$ lies entirely in $\mathcal{X}$, i.e., for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ and any $\theta$ with $0 \leq \theta \leq 1$, we have

$$\theta \boldsymbol{x}_1 + (1 - \theta)\boldsymbol{x}_2 \in \mathcal{X} \ .$$

We say a function $F : \mathcal{X} \to \mathbb{R}$ is *convex* if its domain $\mathcal{X}$ is a convex set and for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ and any $\theta$ with $0 \leq \theta \leq 1$, we have

$$F(\theta \boldsymbol{x}_1 + (1 - \theta)\boldsymbol{x}_2) \leq \theta F(\boldsymbol{x}_1) + (1 - \theta)F(\boldsymbol{x}_2) \ . \tag{1.2}$$

Moreover, when $F$ is differentiable, (1.2) is equivalent to that for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ it holds that

$$F(\boldsymbol{x}_1) + \langle \nabla F(\boldsymbol{x}_1), \boldsymbol{x}_2 - \boldsymbol{x}_1 \rangle \leq F(\boldsymbol{x}_2), \tag{1.3}$$

where $\nabla F(\boldsymbol{x})$ denotes the gradient of $F$ at $\boldsymbol{x}$ and $\langle \cdot, \cdot \rangle$ denotes the inner product of two vectors.

Additionally, a function $F : \mathcal{X} \to \mathbb{R}$ is said to be *$\mu$-strongly convex* (Nesterov, 2004, Theorem 2.1.9) if for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ we have

$$F(\theta \boldsymbol{x}_1 + (1 - \theta)\boldsymbol{x}_2) \leq \theta F(\boldsymbol{x}_1) + (1 - \theta)F(\boldsymbol{x}_2) - \frac{\mu}{2}\theta(1 - \theta)\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2^2 \ . \tag{1.4}$$

where $\|\cdot\|_2$ denotes the $\ell_2$ norm.

Many classical machine learning methods are inspired by or can be reduced to a convex optimization problem. Notable examples include least squeares (Gauss,
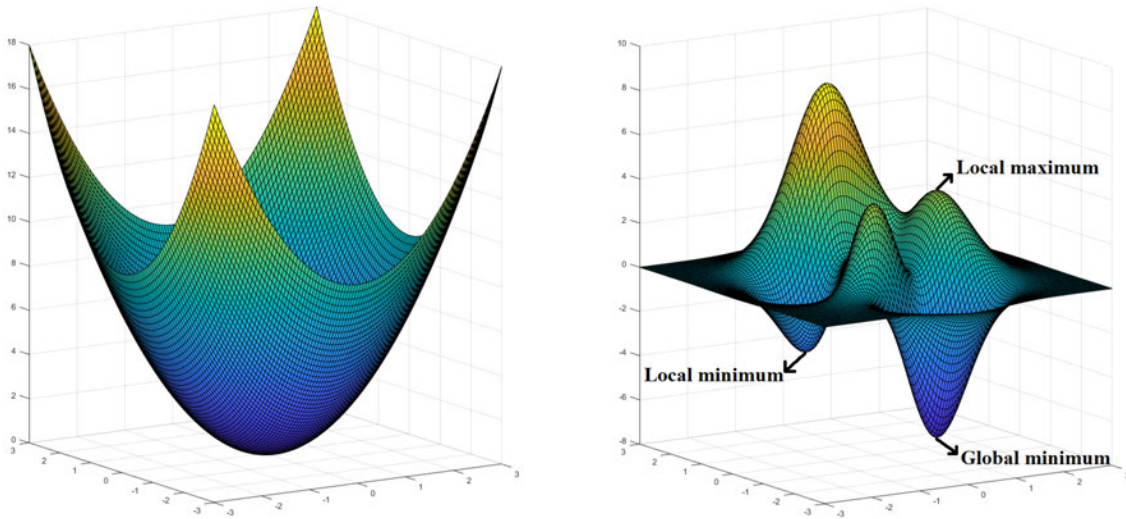
**(a)** A convex function.        **(b)** A non-convex function.

**Figure 1·1:** A convex function vs. a non-convex function.

1820), logistic regression (Verhulst, 1845; Cramer, 2002), and support vector machines (Cortes and Vapnik, 1995).

One nice and important property of convexity is that any stationary point with zero gradient is bound to be a global minimum point which is evident from the definition (1.3). Here we say $\boldsymbol{x}$ is a *stationary point* if $\nabla F(\boldsymbol{x}) = \boldsymbol{0}$ and say $\boldsymbol{x}$ is a *global minimum point* if $F(\boldsymbol{x}) \leq F(\boldsymbol{y})$ for any $\boldsymbol{y} \in \mathcal{X}$. We will also call $\boldsymbol{x}$ a *local minimum point* if for some $\epsilon > 0$ and some distance measure $dist(\cdot, \cdot)$ that $F(\boldsymbol{x}) \leq F(\boldsymbol{y})$ for any $\boldsymbol{y} \in \mathcal{X}$ with $dist(\boldsymbol{x}, \boldsymbol{y}) \leq \epsilon$, and similarly for a *local maximum point*. We can also show that (proof in Appendix A.1):

**Lemma 1.1.** *Let $F : \mathcal{X} \to \mathbb{R}$ be a convex function, then any local minimum point of $F$ in $\mathcal{X}$ is also a global minimum point.*

However, when the convexity condition no longer holds, the optimization problem becomes fundamentally harder and we enter into the wild world of *non-convex optimization*. The above nice property of convex functions is not true anymore, and the

global minimum points could be hidden among an infinite number of local minimum points. As Rockafellar (1993) puts it: "the great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity." As an example, we plot in Figure 1·1 a convex function and a non-convex function, from which it can be immediately seen that the non-convex function is much more complex than the convex one.

Indeed, Nemirovsky and Yudin (1983) showed that the information-based complexity of convex optimization problems is far lower than that of general nonconvex optimization problems. In fact, they showed in Section 1.6 that finding a global optimum for a general non-convex problem is NP-hard. The situation is made worse that even approximately solving a range of non-convex problems is NP-hard (Meka et al., 2008). Thus, in this dissertation, assuming $F$ to be differentiable (Assumption 1.1), following an established line of research, we settle for finding (first-order) $\epsilon$-stationary points, a.k.a. critical points, where the gradient goes to zero, namely some $\boldsymbol{x} \in \mathcal{X}$ with $\|\nabla F(\boldsymbol{x})\| \leq \epsilon$.

Despite the added difficulty in optimizing, modern applications in machine learning have seen more and more usage of non-convex formulations due to their better capability to capture the problem structure. One of the most prominent examples is the Deep Neural Networks, which have scored enormous successes in a vast range of tasks including but not limited to computer vision (Krizhevsky et al., 2012), language translation (Vaswani et al., 2019), speech recognition (Zhang et al., 2017), and recommendation systems (Zheng et al., 2017). Consequently, progresses in this field is much awaited.

It is worth stressing that non-convex functions are not characterized by a particular property, but rather by the lack of a specific property: convexity. In this sense, trying to carry out any meaningful analyses on the entire class of non-convex functions

is hopeless. Therefore, we typically focus on specific classes of non-convex problems by adding additional assumptions. Ideally, the assumptions we use shall balance the trade-off of *approximately* model many interesting machine learning problems while allowing us to restrict the class of non-convex functions to particular subsets where we can unearth interesting behaviors. One common assumption people use is the smoothness one presented below.

**Assumption 1.2.** *A differentiable function $F : \mathcal{X} \to \mathbb{R}$ is called L-smooth, if for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ we have $\|\nabla F(\boldsymbol{x}_1) - \nabla F(\boldsymbol{x}_2)\|_2 \le L\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2$ w.r.t. the $\ell_2$ norm. Note that this directly implies that (Nesterov, 2004, Lemma 1.2.3), for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ we have*

$$|F(\boldsymbol{x}_2) - F(\boldsymbol{x}_1) - \langle \nabla F(\boldsymbol{x}_1), \boldsymbol{x}_2 - \boldsymbol{x}_1 \rangle| \le \frac{L}{2}\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2^2 \ . \tag{1.5}$$

More in detail, the above smoothness assumption is considered "weak" and is ubiquitous in analyses of optimization algorithms in the non-convex setting. Admittedly, in many neural networks, it is only approximately true because ReLUs activation functions are non-smooth. However, if the number of training points is large enough, it is a good approximation of the loss landscape.

## 1.2 Black-box Oracles and Convergence Rates

To solve an optimization problem, we employ an *algorithm*, a finite sequence of rigorous instructions which, given information about the problem, will eventually output a solution to the problem. We are mainly interested in *iterative algorithms* which proceeds in discrete steps $t = 1, 2, \ldots$ and in each step operates on the results of previous steps resulting a sequence of updates $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$. Of course, the more information we are given, the more efficient we can solve the problem, with one extreme of revealing everything and the other extreme of giving nothing. Thus, it makes sense

to restrict our attention to certain classes of algorithms by specifying which information is accessible, as only then can we compare one with another and pick the best candidate.

The algorithms we focus on in this dissertation are those with access to a deterministic first-order black-box oracle (FO) (Nesterov, 2004):

$$Given \ \boldsymbol{x}, \ an \ FO \ returns \ [F(\boldsymbol{x}), \nabla F(\boldsymbol{x})],$$

which we call the *deterministic* setting, or a stochastic first-order black-box oracle (SFO):

$$Given \ \boldsymbol{x}, \ an \ SFO \ returns \ [f(\boldsymbol{x}, \xi), \nabla f(\boldsymbol{x}, \xi)]$$
$$with \ \mathbb{E}_\xi f(\boldsymbol{x}, \xi) = F(\boldsymbol{x}), \ \mathbb{E}_\xi \nabla f(\boldsymbol{x}, \xi) = \nabla F(\boldsymbol{x}),$$

where $\xi$ is a vector drawn by the oracle from an arbitrary set and we call it the *stochastic* setting.

In the stochastic setting, we will focus on the optimization problem

$$\min_{\boldsymbol{x} \in \mathbb{R}^d} F(x) := \mathbb{E}_{\xi \sim \mathcal{D}}[f(\boldsymbol{x}, \xi)], \tag{1.6}$$

where $\xi$ is a random variable representing a randomly selected data sample or random noise following an unknown distribution $\mathcal{D}$.

When comparing one algorithm to another, we would first check if they will *converge*, namely if they will approach the desired solution, and if so, how fast they converge, namely the *convergence rate*. There are two interchangeable forms that are widely used to describe the convergence rate:

- fix the precision $\epsilon > 0$ quantifying how close we want the output of the algorithm to be w.r.t. the desired solution, e.g., $F(\boldsymbol{x}_{output}) - F^* \leq \epsilon$ and compute the number of oracle calls $N$ to achieve such precision in terms of $\epsilon$ like $N = \mathcal{O}(\epsilon^{-2})$.

- fix the number of oracle calls an algorithm can make say $N$ and computes the precision of the output in terms of $N$ like $F(\boldsymbol{x}_{output}) - F^* = \mathcal{O}\left(\frac{1}{N}\right)$.

We will mainly use the second form in this dissertation.

## 1.3 Adaptive Optimization Algorithms

For the scenario of a deterministic first-order black-box oracle, a vastly popular optimizer is the **Gradient Descent** (GD) which can be traced back to Cauchy (1847). GD proceeds along the negative direction of the gradient based on the intuition that the gradient represents the direction of the fastest increase. Mathematically, given an initial point $\boldsymbol{x}_1 \in \mathcal{X}$, GD iteratively updates through

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_t \nabla F(\boldsymbol{x}_t),$$

where $\eta_t$ is called the *step size* at time $t$ and controls how far the algorithm moves. Examples of step size sequences including a constant schedule $\eta_t = \eta$, a polynomial schedule $\eta_t = \frac{1}{t}$, and an exponential schedule $\eta_t = \exp(-t)$.

Meanwhile, in the stochastic setting, when the true gradient is unavailable, we only receive a stochastic gradient $\nabla f(\boldsymbol{x}, \xi)$ where $\xi$ is a random variable denoting the stochasticity. Note that, when convenient, we will refer to $\nabla f(\boldsymbol{x}_t, \xi_t)$ as $\boldsymbol{g}_t$. Then, a counterpart of GD that is widely used is **Stochastic Gradient Descent** (Robbins

and Monro, 1951) which updates through

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_t \nabla f(\boldsymbol{x}_t, \xi_t), \tag{1.7}$$

Despite their wide usage, the performance of GD/SGD depends crucially on the choice of its step size. The tuning of the step size is notoriously laborious and the optimal choice of it can vary drastically across different problems.

To save the labor of tuning, adaptive algorithms come to the rescue. *An algorithm is said to be adaptive to a certain parameter (of the optimization problem) if it does not need a priori knowledge of such parameter but performs competitively to those that know it (up to some additional cost).*

Adaptation is a general concept and an algorithm can be adaptive to any characteristic of the optimization problem. The idea is formalized in (Nesterov, 2015) with the equivalent name of *universality*, but it goes back at least to the "self-confident" strategies in online convex optimization (Auer et al., 2002). Indeed, the famous AdaGrad algorithm (McMahan and Streeter, 2010; Duchi et al., 2010a) uses exactly this concept of adaptation to design an algorithm *adaptive to the gradients*. Nowadays, "adaptive step size" tend to denote coordinate-wise ones, with no guarantee of adaptation to any particular property. There is an abundance of adaptive optimization algorithm in the convex setting (e.g., McMahan and Streeter, 2010; Duchi et al., 2010a; Kingma and Ba, 2015; Reddi et al., 2018), while only a few in the more challenging non-convex setting (e.g., Chen et al., 2020a). The first analysis to show adaptivity to the noise of non-convex SGD with appropriate step sizes is in Li and Orabona (2019) and later in Ward et al. (2019) under stronger assumptions. Then, Li and Orabona (2020) studied the adaptivity to the noise of AdaGrad plus momentum, with a high probability analysis.

---

**Algorithm 1.1** AdaGrad with a global step size (Levy et al., 2018)

---

1: **Input** $\boldsymbol{x}_1 \in \mathcal{X}$, $D$ (diameter of $\mathcal{X}$)
2: **Set** $Q_0 = 0$
3: **for** $t = 1, 2, \ldots, T$ **do**
4:   **Update** $Q_t = Q_{t-1} + \|\nabla F(\boldsymbol{x}_t)\|^2$
5:   **Set** $\eta_t = \frac{D}{\sqrt{2Q_t}}$
6:   **Update** $\boldsymbol{x}_{t+1} = \prod_{\mathcal{X}}(\boldsymbol{x}_t - \eta_t \nabla F(\boldsymbol{x}_t))$
7: **end for**
8: **Output**: $\bar{\boldsymbol{x}} \triangleq \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{x}_t$

---

As an example describing what adaptivity is, we show a variant of AdaGrad in Algorithm 1.1 where $\prod_{\mathcal{X}}(\cdot)$ denotes the projection onto $\mathcal{X}$ namely $\prod_{\mathcal{X}}(\boldsymbol{x}) = \arg\min_{\boldsymbol{y} \in \mathcal{X}} \|\boldsymbol{y} - \boldsymbol{x}\|_2^2$. It has the following guarantee (Levy et al., 2018) (proof in Appendix A.1):

**Theorem 1.2.** *Assume $F$ to be differentiable, convex, and has $D$-bounded-domain namely $D := \max_{\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}} \|\boldsymbol{x} - \boldsymbol{y}\|_2$ which we call the diameter of $\mathcal{X}$, Algorithm 1.1 guarantees*

$$F(\bar{\boldsymbol{x}}) - F(\boldsymbol{x}^*) \leq \frac{\sqrt{2}D}{T} \sqrt{\sum_{t=1}^{T} \|\nabla F(\boldsymbol{x}_t)\|_2^2} \ . \tag{1.8}$$

It can *adapt* to the sum of the gradients norm squared! To see why this is good, consider the projected gradient descent algorithm with a constant step size which updates in the form of Line 6 of Algorithm 1.1 but with a fixed step size $\eta_t = \eta$. We can show the following guarantee for this algorithm.

**Theorem 1.3.** *Assume $F$ to be differentiable, convex, and has $D$-bounded-domain, the projected GD algorithm with a fixed step size $\eta$ guarantees for $\bar{\boldsymbol{x}} = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{x}_t$ that*

$$F(\bar{\boldsymbol{x}}) - F(\boldsymbol{x}^*) \leq \frac{D^2}{2\eta T} + \frac{\eta}{2T} \sum_{t=1}^{T} \|\nabla F(\boldsymbol{x}_t)\|_2^2 \ .$$

Obviously, to get a guarantee like (1.8), we would need to set $\eta = \frac{D}{\sqrt{\sum_{t=1}^{T} \|\nabla F(\boldsymbol{x}_t)\|_2^2}}$.

Yet, this step size requires knowledge of all updates which is clearly impossible amid running the algorithm. This immediately shows the advantage of adaptive algorithms.

## 1.4    Structure of the Dissertation

Chapter 2 is devoted to the adaptation to the level of noise in evaluating stochastic gradients under the stochastic optimization setting. There, we will present our work on designing/analyzing noise adaptive algorithms in both the general smooth non-convex setting and the setting with the additional PL condition.

Chapter 3 discusses the problem that the scales of gradient magnitudes can vary significantly across layers in training deep neural networks. We will identify scenarios where the renowned Adam optimizer (Kingma and Ba, 2015) is inferior to its variant AdamW (Loshchilov and Hutter, 2019), and then correlate this observation to the scale-freeness property AdamW enjoys while Adam does not. A connection between AdamW and proximal updates will then be revealed providing a potential explanation for where the scale-freeness comes from.

Chapter 4 focuses on the setting of relaxed smoothness where the gradients can change drastically. We will start by reporting empirical evidence showing that this condition captures the training of Transformers and propose to further refine it to a coordinate-wise level. A generalized SignSGD algorithm is then proposed which on one end recovers the SignSGD algorithm with matching theoretical convergence guarantees as SGD with gradient clipping, while on the other end closely mimics Adam with matching empirical performance. This algorithm can be made to adapt to the unknown parameter characterizing the relaxed smoothness condition.

Finally, Chapter 5 concludes this dissertation with major contributions.

## 1.5   Notations

We use bold lower-case letters to denote vectors and upper-case letters for matrices, e.g., $\boldsymbol{u} \in \mathbb{R}^d$, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. The $\mathrm{i}^{th}$ coordinate of a vector $\boldsymbol{u}$ is $u_i$. Unless otherwise noted, we study the Euclidean space $\mathbb{R}^d$ with the inner product $\langle \cdot, \cdot \rangle$, and all the norms are the Euclidean norms. The dual norm $\|\cdot\|_*$ is the norm defined by $\|\boldsymbol{v}\|_* = \sup_{\boldsymbol{u}} \{ \langle \boldsymbol{u}, \boldsymbol{v} \rangle :$ $\|\boldsymbol{u}\| \leq 1 \}$. $\mathbb{E}[\boldsymbol{u}]$ means the expectation with respect to the underlying probability distribution of a random variable $\boldsymbol{u}$, and $\mathbb{E}_t[\boldsymbol{u}]$ is the conditional expectation of $\boldsymbol{u}$ with respect to the past. The gradient of $F$ at $\boldsymbol{x}$ is denoted by $\nabla F(\boldsymbol{x})$. $\partial F$ denotes the set of subgradients. $[d]$ denotes the sequence $\{1, \ldots, d\}$.

# Chapter 2

# Adaptation to Noise

[The results in Section 2.2 appeared in Zhuang et al. (2019) and the results in Section 2.3 appeared in Li et al. (2021).]

Gradient Descent is an intuitive yet effective algorithm that enjoys vast popularity in the machine learning community and beyond. In practice, however, the true gradient is not always available, either because it is impossible to obtain at all, or because evaluating it would be too expensive. Under such scenarios, we do stochastic optimization in which we only access a stochastic gradient and run Stochastic Gradient Descent instead. Yet, the noise in evaluating the stochastic gradients slows down the convergence or even leads to divergence if we do not tune the step size carefully.

Formally, in this chapter, we focus on optimizing problem (1.1) using GD or (1.6) using SGD. Further, we will use following assumptions:

**Assumption 2.1.** *The stochastic gradient at each step t is unbiased given the past, that is,*

$$\mathbb{E}_t \left[ \nabla f(\boldsymbol{x}_t, \xi_t) \right] = \nabla F(\boldsymbol{x}_t) \ .$$

**Assumption 2.2.** *The stochastic gradient at each step t has finite variance with*

*respect to the $\ell_2$ norm given the past, that is,*

$$\mathbb{E}_t \left[ \|\nabla f(\boldsymbol{x}_t, \xi_t) - \nabla F(\boldsymbol{x}_t)\|^2 \right] = \sigma^2 \ .$$

The structure of this chapter is the following: we will first discuss how these two settings (deterministic vs. stochastic) are different and why adapting to noise is desirable in Section 2.1. Next, we will introduce our work (Zhuang et al., 2019) on designing an algorithm that uses no-regret online algorithms to compute optimal step sizes on the fly and guarantees convergence rates that are automatically adaptive to the level of noise in Section 2.2. Then, in Section 2.3, we will present our work (Li et al., 2021) showing that, under the added PL condition, SGD employing two vastly popular empirical step size schedules enjoys a faster convergence rate while still being adaptive to the level of noise.

## 2.1 Why Adaptation to Noise is Desirable

Intuitively, when using SGD, the noisy gradients pointing to a random direction would slow down the convergence speed. Indeed, there are already established lower bounds showing that stochastic optimization is fundamentally more difficult than the deterministic one. For example, in the general $L$-smooth non-convex scenario, when we can access the true gradient (the deterministic setting), the best possible worst-case rate of convergence to a stationary point for any first-order optimization algorithm is $O(\frac{1}{T})$ (Carmon et al., 2021), which can be obtained by Gradient Descent with a constant step size $\eta = \frac{1}{L}$. In contrast, when there is noise (assuming zero mean and bounded variance), namely the stochastic setting, no first-order algorithm can do better than the $O(\frac{\sigma}{\sqrt{T}})$ rate (Arjevani et al., 2022) and this rate can be achieved by

Stochastic Gradient Descent with a constant step size of $\eta = \frac{1}{\sigma\sqrt{T}}$ or a time-varying one $\eta_t = \frac{1}{\sigma\sqrt{t}}$.

Clearly, the deterministic and the stochastic settings are intrinsically different. Without knowing the noise level $\sigma$, especially if $\sigma = 0$ or not, there is no single step size schedule that will make GD/SGD converge at the optimal rate in both settings. We would have to spend a lot of time tuning the algorithm very carefully. Indeed, Ghadimi and Lan (2013) proved the following result for SGD with a constant step size in the smooth setting:

**Theorem 2.1.** *For a L-smooth function F (Assumption 1.2), under Assumption 2.1 and 2.2, SGD with a constant step size $\eta \leq \frac{1}{L}$ guarantees*

$$\mathbb{E}[\|\nabla F(\boldsymbol{x}_i)\|^2] \leq O\left(\frac{F(\boldsymbol{x}_1) - F^\star}{\eta T} + \eta\sigma^2\right),$$

*where $\boldsymbol{x}_i$ is uniformly randomly picked in $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$.*

From the above, it is immediate to see that we need a step size of the form $O(\min(\frac{\sqrt{F(\boldsymbol{x}_1) - F^\star}}{\sigma\sqrt{T}}, \frac{1}{L}))$ to have the best worst case convergence of $O(\frac{1}{T} + \frac{\sigma}{\sqrt{T}})$. In words, this means that we get a faster rate, $O(\frac{1}{T})$, when there is no noise, and a slower rate, $O(\frac{\sigma}{\sqrt{T}})$, in the presence of noise.

In practice, however, we usually do not know the variance of the noise $\sigma$, which makes the above optimal tuning of the step size difficult to achieve. Even worse, the variance can change over time. For example, it may decrease over time if $F(\boldsymbol{x}) = \mathbb{E}_j[F_j(\boldsymbol{x})]$ and each $F_j$ has zero gradient at the local optimum we are converging to. Moreover, even assuming the knowledge of the variance of the noise, the step sizes proposed in Ghadimi and Lan (2013) assume the knowledge of the unknown quantity $F(\boldsymbol{x}_1) - F^\star$.
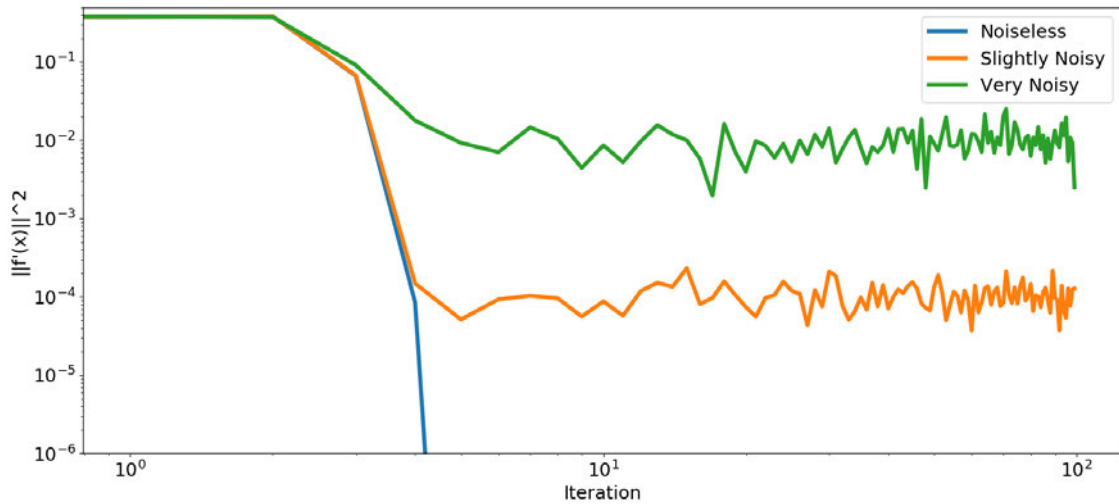
**Figure 2·1:** Running SGD with a fixed step size on a smooth non-convex function for different noise levels.

We use Figure 2·1 to depict the above result, in which we run SGD with a fixed constant step size $\eta = 0.5$ on a non-convex function $F(x) = \frac{x^2}{1+x^2}$ which is 2-smooth. When accessing the gradient, we add additive white Gaussian noise with different variances $\sigma$ $(0, 0.01, 0.1$, respectively). The figure clearly shows that one constant step size does not work for different noise settings and that tuning is necessary.

Another solution would be to obtain an explicit estimate of the variances of the noise, for example by applying some concentration inequality to the sample variance, and using it to set the step sizes. This approach is suboptimal because it does not directly optimize the convergence rates, relying instead on a loose worst-case analysis.

In light of this problem, in the next section, we will present an algorithm we designed that guarantees the optimal rates in both deterministic and stochastic settings automatically without knowing $\sigma$, namely adapting to noise.

## 2.2 Adaptation to Noise in the Smooth Non-convex Setting

### 2.2.1 Surrogate Losses

Before presenting the algorithm, we first introduce the notion of surrogate losses which motivates the key idea behind the design of our algorithm: we use the smoothness of the objective function to transform the problem of optimizing a non-convex objective function into the problem of optimizing a series of convex loss functions, which we solve by an online learning algorithm.

Specifically, consider using SGD with non-convex $L$-smooth losses starting from an initial point $\boldsymbol{x}_1$. At each time $t$, we define the *surrogate loss* $\ell_t : \mathbb{R}^d \to \mathbb{R}$ as

$$\ell_t(\eta) = -\eta \langle \nabla f(\boldsymbol{x}_t, \xi_t), \nabla f(\boldsymbol{x}_t, \xi_t') \rangle + \frac{L\eta^2}{2} \|\nabla f(\boldsymbol{x}_t, \xi_t)\|^2, \tag{2.1}$$

where $\nabla f(\boldsymbol{x}_t, \xi_t)$ and $\nabla f(\boldsymbol{x}_t, \xi_t')$ are the noisy stochastic gradients received from the black-box oracle at time $t$. Note that, when convenient, we will refer to $\nabla f(\boldsymbol{x}_t, \xi_t)$ and $\nabla f(\boldsymbol{x}_t, \xi_t')$ as $\boldsymbol{g}_t$ and $\boldsymbol{g}_t'$ respectively. We also assume that:

**Assumption 2.3.** *The two stochastic gradients at step t are independent given the past, i.e.,*

$$\mathbb{E}_t \left[ \langle \nabla f(\boldsymbol{x}_t, \xi_t), \nabla f(\boldsymbol{x}_t, \xi_t') \rangle \right] = \|\nabla F(\boldsymbol{x}_t)\|^2 .$$

It is clear that $\ell_t$ is convex. Moreover, the following key result shows that these surrogate losses upper bound the expected decrease of the function value $F$.

**Theorem 2.2.** *Assume Assumption 2.1 and 2.3 hold and $\eta_t$ is independent from $\xi_j$ and $\xi_j'$ for $j \geq t$. Then, for an $L$-smooth function, the SGD update in (1.7) gives us*

$$\mathbb{E} \left[ F(\boldsymbol{x}_{t+1}) - F(\boldsymbol{x}_t) \right] \leq \mathbb{E} \left[ \ell_t(\eta_t) \right] .$$

*Proof of Theorem 2.2.* The $L$-smoothness of $F$ gives us:

$$\mathbb{E}\left[F(\boldsymbol{x}_{t+1}) - F(\boldsymbol{x}_t)\right] \leq \mathbb{E}\left[\langle \nabla F(\boldsymbol{x}_t), \boldsymbol{x}_{t+1} - \boldsymbol{x}_t \rangle + \frac{L}{2}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|^2\right]$$

$$= \mathbb{E}\left[\langle \nabla F(\boldsymbol{x}_t), -\eta_t \nabla f(\boldsymbol{x}_t, \xi_t) \rangle + \frac{L}{2}\eta_t^2\|\nabla f(\boldsymbol{x}_t, \xi_t)\|^2\right]$$

$$= \mathbb{E}\left[\langle \nabla F(\boldsymbol{x}_t), \mathbb{E}_t\left[-\eta_t \nabla f(\boldsymbol{x}_t, \xi_t)\right] \rangle + \frac{L}{2}\eta_t^2\|\nabla f(\boldsymbol{x}_t, \xi_t)\|^2\right].$$

Now observe that $\nabla F(\boldsymbol{x}_t) = \mathbb{E}_t\left[\nabla f(\boldsymbol{x}_t, \xi_t')\right]$, so that

$$\mathbb{E}\left[\langle \mathbb{E}_t\left[\nabla f(\boldsymbol{x}_t, \xi_t')\right], \mathbb{E}_t\left[-\eta_t \nabla f(\boldsymbol{x}_t, \xi_t)\right] \rangle\right] = \mathbb{E}\left[\mathbb{E}_t\left[\langle \nabla f(\boldsymbol{x}_t, \xi_t'), -\eta_t \nabla f(\boldsymbol{x}_t, \xi_t) \rangle\right]\right]$$

$$= \mathbb{E}\left[\langle \nabla f(\boldsymbol{x}_t, \xi_t'), -\eta_t \nabla f(\boldsymbol{x}_t, \xi_t) \rangle\right].$$

Putting it all together, we have the stated inequality. □

The above theorem tells us that if we want to decrease the function $f$, we might instead try to minimize the convex surrogate losses $\ell_t$. In the following, we build upon this intuition to design an online learning procedure that adapts the step sizes of SGD to achieve the optimal convergence rate.

## 2.2.2   SGD with Online Learning

The surrogate losses allow us to design an online convex optimization procedure to learn the optimal step sizes. In each round, the step sizes are chosen by an online learning algorithm $\mathcal{A}$ fed with the surrogate losses $\ell_t$. The online learning algorithm will minimize the regret: the difference between the cumulative sum of the losses of the algorithm, $\ell(\eta_t)$, and the cumulative losses of any fixed point $\eta$. In formulas, for

---

**Algorithm 2.1** Stochastic Gradient Descent with Online Learning (SGDOL)

---

1: **Input:** $\boldsymbol{x}_1 \in \mathcal{X}$, $L$, an online learning algorithm $\mathcal{A}$
2: **for** $t = 1, 2, \ldots, T$ **do**
3:    **Compute** $\eta_t$ by running $\mathcal{A}$ on $\ell_i, i = 1, \ldots, t-1$, as defined in (2.1)
4:    **Receive** two independent unbiased estimates of $\nabla F(\boldsymbol{x}_t)$: $\boldsymbol{g}_t$, $\boldsymbol{g}_t'$
5:    **Update** $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_t \boldsymbol{g}_t$
6: **end for**
7: **Output**: uniformly randomly choose a $\boldsymbol{x}_k$ from $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$.

---

a 1-dimensional online convex optimization problem, the regret is defined as

$$\text{Regret}_T(\eta) = \sum_{t=1}^{T} (\ell_t(\eta_t) - \ell_t(\eta)) \ .$$

If the regret is small, we will have that the losses of the algorithm are not too big compared to the best losses, which implies that the step sizes chosen by the online algorithm are not too far from the optimal (unknown) step size.

We call this procedure Stochastic Gradient Descent with Online Learning (SG-DOL) and the pseudocode is in Algorithm 2.1.

To prove its convergence rate, we need the following assumption:

**Assumption 2.4.** *The stochastic gradients at each step t have bounded norms:*

$$\|\nabla f(\boldsymbol{x}_t, \xi_t)\| \leq G, \quad \|\nabla f(\boldsymbol{x}_t, \xi_t')\| \leq G \ .$$

Then, we can prove the following Theorem.

**Theorem 2.3.** *For an L-smooth function F, under Assumption 2.1 and 2.2, for any* $\eta > 0$, *SGDOL in Algorithm 2.1 satisfies*

$$\mathbb{E}\left[\left(\eta - \frac{L}{2}\eta^2\right) \sum_{t=1}^{T} \|\nabla F(\boldsymbol{x}_t)\|^2\right] \leq F(\boldsymbol{x}_1) - F^\star + \mathbb{E}\left[\text{Regret}_T(\eta)\right] + \frac{L\eta^2\sigma^2 T}{2} \ .$$

*Proof of Theorem 2.3.* Summing the inequality in Theorem 2.2 from 1 to $T$:

$$
\begin{aligned}
F^\star - F(\boldsymbol{x}_1) &\leq \mathbb{E}[F(\boldsymbol{x}_{T+1})] - F(\boldsymbol{x}_1) \\
&= \sum_{t=1}^{T} \mathbb{E}\left[F(\boldsymbol{x}_{t+1}) - F(\boldsymbol{x}_t)\right] \\
&\leq \sum_{t=1}^{T} \mathbb{E}\left[\ell_t(\eta_t)\right] \\
&= \sum_{t=1}^{T} \mathbb{E}\left[\ell_t(\eta_t) - \ell_t(\eta)\right] + \sum_{t=1}^{T} \mathbb{E}\left[\ell_t(\eta)\right] \\
&\leq \mathbb{E}\left[\mathrm{Regret}_T(\eta)\right] + \sum_{t=1}^{T} \mathbb{E}\left[\ell_t(\eta)\right] \ .
\end{aligned}
$$

Using the fact that

$$
\mathbb{E}_t[\ell_t(\eta)] = \left(-\eta + \frac{L}{2}\eta^2\right)\|\nabla F(\boldsymbol{x}_t)\|^2 + \frac{L}{2}\eta^2\sigma^2,
$$

we have the stated bound. $\qquad\square$

The only remaining ingredient for SGDOL is to decide on an online learning procedure. Given that the surrogate losses are strongly convex, we can use a Follow The Regularized Leader (FTRL) algorithm (Shalev-Shwartz, 2007; Abernethy et al., 2008, 2012; McMahan, 2017). Note that this is not the only possibility, for example, we could use an optimistic FTRL algorithm which achieves even smaller regret (Mohri and Yang, 2016). Yet, FTRL is enough to show the potential of our surrogate losses. As shown in Algorithm 2.2, in an online learning game in which we receive the convex losses $\ell_t$, FTRL constructs the predictions $\boldsymbol{v}_t$ by solving the optimization problem

$$
\boldsymbol{v}_{t+1} = \operatorname*{argmin}_{\boldsymbol{v}\in\mathbb{R}^d} \ r(\boldsymbol{v}) + \sum_{s=1}^{t} \ell_s(\boldsymbol{v}),
$$

---

**Algorithm 2.2** Follow the Regularized Leader (FTRL)

---

1: **Parameters**: $r(\boldsymbol{v}) \geq 0$
2: $\boldsymbol{v}_1 \leftarrow \mathrm{argmin}_{\boldsymbol{v} \in \mathbb{R}^d} \ r(\boldsymbol{v})$
3: **for** $t = 1, 2, \ldots$ **do**
4:     Observe convex loss function $\ell_t : \mathbb{R}^d \to \mathbb{R} \cup \{\infty\}$
5:     Incur loss $\ell_t(\boldsymbol{v}_t)$
6:     Update $\boldsymbol{v}_{t+1} \leftarrow \mathrm{argmin}_{\boldsymbol{v} \in \mathbb{R}^d} \ r(\boldsymbol{v}) + \sum_{s=1}^{t} \ell_s(\boldsymbol{v})$
7: **end for**

---

where $r : \mathbb{R}^d \to \mathbb{R}$ is a regularization function. We can upper bound the regret of FTRL with the following theorem.

**Theorem 2.4.** *(McMahan, 2017) For the FTRL Algorithm 2.2, suppose each $\ell_t$ is convex and differentiable and $r$ is chosen such that $h_t = r + \sum_{i=1}^{t} \ell_i$ is 1-strongly-convex w.r.t. some norm $\|\cdot\|_{(t)}$. Then, for any $\boldsymbol{x}^\star \in \mathbb{R}^d$ and for any $T > 0$,*

$$\mathrm{Regret}_T(\boldsymbol{x}^\star) \leq r(\boldsymbol{x}^\star) + \frac{1}{2} \sum_{t=1}^{T} \|\nabla \ell_t(\boldsymbol{x}_t)\|_{(t),\star}^2,$$

*where $\|\cdot\|_{(t),\star}$ is the dual norm of $\|\cdot\|_{(t)}$ namely $\|\boldsymbol{x}\|_{(t),\star} := \sup_{\boldsymbol{y}:\|\boldsymbol{y}\|_{(t)} \leq 1} \langle \boldsymbol{x}, \boldsymbol{y} \rangle$.*

We can now put it all together and get a convergence rate guarantee for SGDOL.

**Theorem 2.5.** *For an $L$-smooth function $F$, under Assumption 2.1, 2.2, 2.3, and 2.4 and using FTRL (Algorithm 2.2) by choosing $r(\eta) = \frac{L\alpha}{2}\left(\eta - \frac{1}{L}\right)^2 + \mathcal{I}\left(\eta \in \left[0, \frac{2}{L}\right]\right)$ with $\alpha > 0$ in Algorithm 2.1, for an uniformly randomly picked $\boldsymbol{x}_k$ from $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t$, we have:*

$$\mathbb{E}_k\left[\|\nabla F(\boldsymbol{x}_k)\|^2\right] \leq \frac{2L}{T}\left(F(\boldsymbol{x}_1) - F^\star + \frac{5G^2}{L}\ln\left(1 + \frac{G^2 T}{\alpha}\right)\right)$$
$$+ \frac{1}{\sqrt{T}}\sqrt{2L\sigma^2\left(f(\boldsymbol{x}_1) - f^\star + \frac{\alpha}{2L}\right)}$$
$$+ \frac{1}{\sqrt{T}}\sqrt{10G^2\sigma^2\ln\left(1 + \frac{G^2 T}{\alpha}\right)}.$$

Clearly, when $\sigma = 0$ we obtain the convergence rate of $\mathcal{O}\left(\frac{1}{T}\right)$, and when $\sigma > 0$ we can still converge in the rate of $\mathcal{O}\left(\frac{\sigma}{\sqrt{T}}\right)$. Note that we do not need to know $\sigma$ to achieve this result, thus our algorithm is adaptive to $\sigma$.

Before proving this theorem, we make some observations.

The FTRL update gives us a very simple strategy to calculate the step sizes $\eta_t$. In particular, the FTRL update has a closed-form:

$$\eta_t = \max\left\{0, \min\left\{\frac{\alpha + \sum_{j=1}^{t-1}\langle \boldsymbol{g}_j, \boldsymbol{g}_j' \rangle}{L\left(\alpha + \sum_{j=1}^{t-1}\|\boldsymbol{g}_j\|^2\right)}, \frac{2}{L}\right\}\right\} \, . \tag{2.2}$$

Note that this can be efficiently computed by keeping track of the quantities $\sum_{j=1}^{t-1}\langle \boldsymbol{g}_j, \boldsymbol{g}_j' \rangle$ and $\sum_{j=1}^{t-1}\|\boldsymbol{g}_j\|^2$.

While the computational complexity of calculating $\eta_t$ by FTRL is negligible, SG-DOL requires two unbiased gradients per step. This increases the computational complexity with respect to a plain SGD procedure by a factor of two.

The theorem also shows that the parameter $\alpha$ has a minor influence on the convergence rate: although it should optimally be set to any constant value on the order of $G^2$, it is safe to set it reasonably small without blowing up the log factor.

We can now prove the convergence rate in Theorem 2.5. For the proof, we need the following lemma.

**Lemma 2.6.** *Let $a_i \geq 0$ for $i = 0, \cdots, T$ and $h : [0, +\infty) \to [0, +\infty)$ be a nonincreasing function. Then*

$$\sum_{t=1}^{T} a_t h\left(a_0 + \sum_{i=1}^{t} a_i\right) \leq \int_{a_0}^{\sum_{t=0}^{T} a_t} h(x)dx \, .$$

*Proof of Lemma 2.6.* Denote by $s_t = \sum_{i=0}^{t} a_i$.

$$a_i h(s_i) = \int_{s_{i-1}}^{s_i} h(s_i) dx \leq \int_{s_{i-1}}^{s_i} h(x) dx .$$

Summing over $i = 1, \cdots, T$, we obtain the stated bound. $\qquad\square$

*Proof of Theorem 2.5.* As $\ell_t''(\eta) = L\|\boldsymbol{g}_t\|^2$, we have that $h_t = r + \sum_{i=1}^{t} \ell_i$ is 1-strongly-convex with respect to the norm $\sqrt{L\left(\alpha + \sum_{s=1}^{t} \|\boldsymbol{g}_s\|^2\right)}\|\cdot\|$.

Applying Theorem 2.4, we get that, for any $\eta \in \left[0, \frac{2}{L}\right]$,

$$\text{Regret}_T(\eta) \leq \frac{L\alpha}{2}\left(\eta - \frac{1}{L}\right)^2 + \frac{1}{2L}\sum_{t=1}^{T}\frac{(\ell_t'(\eta_t))^2}{\alpha + \sum_{s=1}^{t}\|\boldsymbol{g}_s\|^2} . \tag{2.3}$$

Now observe that

$$(\ell_t'(\eta_t))^2 = \left(-\langle \boldsymbol{g}_t, \boldsymbol{g}_t'\rangle + L\eta_t\|\boldsymbol{g}_t\|^2\right)^2$$

$$\leq 2\langle \boldsymbol{g}_t, \boldsymbol{g}_t'\rangle^2 + 2L^2\eta_t^2\|\boldsymbol{g}_t\|^4$$

$$\leq 2\|\boldsymbol{g}_t\|^2\|\boldsymbol{g}_t'\|^2 + 8\|\boldsymbol{g}_t\|^4 \leq 10G^2\|\boldsymbol{g}_t\|^2,$$

where in the third line we used the Cauchy-Schwarz inequality and $\eta_t \leq \frac{2}{L}$. Hence, the last term in (2.3) can be upper bounded as

$$\frac{1}{2}\sum_{t=1}^{T}\frac{(\ell_t'(\eta_t))^2}{L\left(\alpha + \sum_{s=1}^{t}\|\boldsymbol{g}_s\|^2\right)} \leq \frac{5G^2}{L}\sum_{t=1}^{T}\frac{\|\boldsymbol{g}_t\|^2}{\alpha + \sum_{s=1}^{t}\|\boldsymbol{g}_s\|^2}$$

$$\leq \frac{5G^2}{L}\ln\left(\frac{\alpha + \sum_{t=1}^{T}\|\boldsymbol{g}_t\|^2}{\alpha}\right)$$

$$\leq \frac{5G^2}{L}\ln\left(1 + \frac{G^2T}{\alpha}\right),$$

where in the first inequality we used Lemma 2.6.

Now put the last inequality above back into Theorem 2.3, to obtain

$$\mathbb{E}\left[\left(\eta - \frac{L}{2}\eta^2\right)\sum_{t=1}^{T}\|\nabla F(\boldsymbol{x}_t)\|^2\right] \leq F(\boldsymbol{x}_1) - F^\star + \frac{L\alpha}{2}\left(\eta - \frac{1}{L}\right)^2$$
$$+ \frac{5G^2}{L}\ln\left(1 + \frac{G^2T}{\alpha}\right) + \frac{L\eta^2\sigma^2T}{2} .$$

Denote $A \triangleq \sum_{t=1}^{T}\mathbb{E}\left[\|\nabla F(\boldsymbol{x}_t)\|^2\right]$, we can transform the above into a quadratic inequality of $\eta$:

$$0 \leq \frac{L}{2}\left(A + \alpha + \sigma^2T\right)\eta^2 - (A + \alpha)\eta + F(\boldsymbol{x}_1) - F^\star + \frac{5G^2}{L}\ln\left(1 + \frac{G^2T}{\alpha}\right) + \frac{\alpha}{2L} .$$

Choosing $\eta$ as the minimizer of the right hand side: $\eta^* = \frac{\alpha + A}{L(\alpha + A + \sigma^2T)}$ (which satisfies $\eta^* \leq \frac{2}{L}$) gives us

$$\frac{(\alpha + A)^2}{2L\left(\alpha + A + \sigma^2T\right)} \leq F(\boldsymbol{x}_1) - F^\star + \frac{\alpha}{2L} + \frac{5G^2}{L}\ln\left(1 + \frac{G^2T}{\alpha}\right) .$$

Solving this quadratic inequality of $A$ yields

$$A \leq 2L\left(F(\boldsymbol{x}_1) - F^\star + \frac{5G^2}{L}\ln\left(1 + \frac{G^2T}{\alpha}\right)\right)$$
$$+ \sqrt{2L\sigma^2T\left(F(\boldsymbol{x}_1) - F^\star + \frac{\alpha}{2L}\right)}$$
$$+ \sqrt{10G^2\sigma^2T\ln\left(1 + \frac{G^2T}{\alpha}\right)} .$$

By taking an $\boldsymbol{x}_k$ from $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t$ randomly, we get:

$$\mathbb{E}_k\left[\|\nabla F(\boldsymbol{x}_k)\|^2\right] = \mathbb{E}_k\left[\mathbb{E}\left[\|\nabla F(\boldsymbol{x}_k)\|^2\big|k\right]\right] = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\|\nabla F(\boldsymbol{x}_t)\|^2\right],$$

which completes the proof. □

Our step size schedule (2.2) is very unique; thus, to illustrate its behavior in practice, we experiment on fitting a classification model on the adult (a9a) dataset from the LibSVM website (Chang and Lin, 2001). The objective function is

$$F(\boldsymbol{x}) := \frac{1}{m} \sum_{i=1}^{m} \phi(\boldsymbol{a}_i^\top \boldsymbol{x} - y_i),$$

where $\phi(\theta) = \frac{\theta^2}{1+\theta^2}$, and $(\boldsymbol{a}_i, y_i)$ are the couples feature vector/label. The loss function $\phi$ is non-convex, 1-Lipschitz and 2-smooth w.r.t. the $\ell_2$ norm.

We consider the minimization problem with respect to all training samples. Also, as the dataset is imbalanced towards the group with an annual income of less than 50K, we subsample that group to balance the dataset, which results in 15682 samples with 123 features each. In addition, we append a constant element to each sample feature vector to introduce a constant bias. $\boldsymbol{x}_1$ is initialized to be all zeros. For each setting, we repeat the experiment with different random seeds but with the same initialization 5 times and plot the average of the relevant quantities.

In this experiment, the noise on the gradient is generated by the use of mini-batches. Specifically, we compare SGDOL with SGD on three different mini-batch sizes, namely different noise scales: using all samples (Noiseless), 50 i.i.d. samples (Moderate Noise), or 1 random sample (Heavy Noise) for evaluating the gradient at a point. The step size of SGD is selected as the one giving the best convergence rate when the full batch scheme, namely zero noise, is employed which turns out to be 0.1. We take the reciprocal of SGD's best step size as the parameter $L$ for SGDOL, and we set $\alpha = 10$ without any tuning based on our discussion on the influence of $\alpha$ above. These parameters are then employed in the other two noisy settings.

We report the results in Figure 2·2. Figures in the top row show $\mathbb{E}[\|\nabla F(\boldsymbol{x}_k)\|^2]$ vs. number of iterations, whereas those in the bottom row are per-round step sizes.
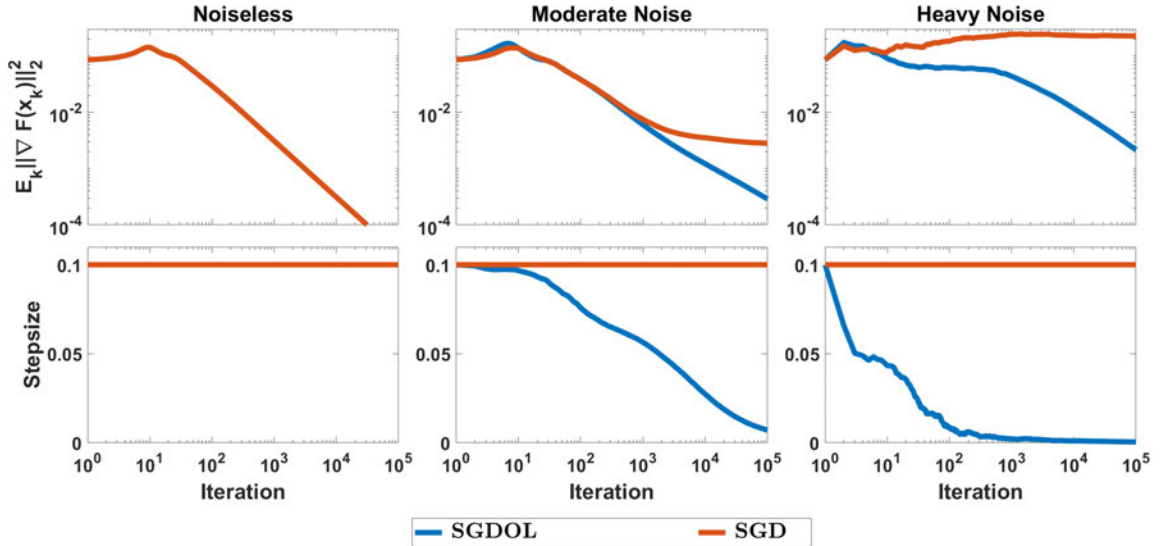
**Figure 2·2:** Comparison of SGDOL vs. SGD on optimizing a smooth non-convex function with various noise scales.

The x-axis in all figures and the y-axis in the top three are logarithmic.

As can be seen, the step size of SGDOL is the same as SGD at first, but gradually decreases automatically. Also, the larger the noise, the sooner the decreasing phase starts. The decrease of the step size makes the convergence of SGDOL possible. In particular, SGDOL recovers the performance of SGD in the noiseless case, while it allows convergence in the noisy cases through an automatic decrease of the step sizes. In contrast, when noise exists, after reaching the proximity of a stationary point, SGD oscillates thereafter without converging, and the value it oscillates around depends on the variance of the noise. This underlines the superiority of the surrogate losses, rather than choosing a step size based on a worst-case convergence rate bound.

### 2.2.3 Adapting Per-coordinate Step Sizes

In the previous subsection, we have shown how to use the surrogate loss functions to adapt a step size. Another common strategy in practice is to use a *per-coordinate*

step size. This kind of scheme is easily incorporated into our framework and we show that it can provide improved adaptivity to per-coordinate variances.

Specifically, we consider $\boldsymbol{\eta}_t$ now to be a vector in $\mathbb{R}^d$, $\boldsymbol{\eta}_t = (\eta_{t,1}, \ldots, \eta_{t,d})$, and use the update $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \boldsymbol{\eta}_t \boldsymbol{g}_t$ where $\boldsymbol{\eta}_t \boldsymbol{g}_t$ now indicates coordinate-wise product $(\eta_{t,1} g_{t,1}, \ldots, \eta_{t,d} g_{t,d})$. Then we define the surrogate losses to be

$$\ell_t(\boldsymbol{\eta}) = -\langle \boldsymbol{\eta} \nabla f(\boldsymbol{x}_t, \xi_t), \nabla f(\boldsymbol{x}_t, \xi_t') \rangle + \frac{L}{2} \|\boldsymbol{\eta} \nabla f(\boldsymbol{x}_t, \xi_t)\|^2 = \sum_{i=1}^d -\eta_i g_{t,i} g_{t,i}' + \frac{L}{2} \eta_i^2 g_{t,i}^2 \ .$$

To take advantage of this scenario, we need more detail about the variance, which we encapsulate in the following assumption:

**Assumption 2.5.** *The noisy gradients have finite variance in each coordinate:*

$$\mathbb{E}_t \left[ \left( g_{t,i} - \frac{\partial F}{\partial x_i}(\boldsymbol{x}_t) \right)^2 \right] = \sigma_i^2 \ .$$

Note that this assumption is not actually stronger than Assumption 2.2 because we can define $\sigma^2 = \sum_{i=1}^d \sigma_i^2$. This merely provides finer-grained variable names.

Also, we make the following assumption:

**Assumption 2.6.** *The noisy gradients have bounded coordinate values:*

$$|g_{t,i}| \leq G_i, \ |g_{t,i}'| \leq G_i \ .$$

Now the exact same argument as for Theorem 2.3 yields:

**Theorem 2.7.** *For an L-smooth function $F$, assume the two noisy gradients in each round $t$ to satisfy Assumption 2.1, 2.3, and 2.5. Then, for any $\boldsymbol{\eta} \in \mathbb{R}^d$ with $\eta_i > 0$*

*for all i, the per-coordinate variant of Algorithm 2.1 gives*

$$\mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{d}\left(\eta_i - \frac{L}{2}\eta_i^2\right)\left(\frac{\partial F}{\partial x_i}(\boldsymbol{x}_t)\right)^2\right] \leq F(\boldsymbol{x}_1) - F^\star + \mathbb{E}\left[\text{Regret}_T(\boldsymbol{\eta})\right] + \frac{LT}{2}\sum_{i=1}^{d}\eta_i\sigma_i^2 .$$

With this Theorem in hand, once again all that remains is to choose the online learning algorithm. To this end, observe that we can write $\ell_t(\eta) = \sum_{i=1}^{d}\ell_{t,i}(\eta_i)$ where

$$\ell_{t,i}(\eta_i) = -\eta_i g_{t,i} g'_{t,j} + \frac{L}{2}\eta_i^2 g_{t,i}^2 .$$

Thus, we can take our online learning algorithm to be a per-coordinate instantiation of Algorithm 2.2, and the total regret is simply the sum of the per-coordinate regrets. Each per-coordinate regret can be analyzed in exactly the same way as Algorithm 2.2, leading to

$$\text{Regret}_T(\eta) = \sum_{i=1}^{d}\text{Regret}_{T,i}(\eta_i),$$

$$\text{Regret}_{T,i}(\eta_i) \leq \frac{L\alpha}{2}\left(\eta_i - \frac{1}{L}\right)^2 + \frac{5G_i^2}{L}\ln\left(1 + \frac{G_i^2 T}{\alpha}\right) .$$

From these inequalities, we can make a per-coordinate bound on the gradient magnitudes. In words, the coordinates which have smaller variances $\sigma_{t,i}^2$ achieve smaller gradient values faster than coordinates with larger variances. Further, we preserve adaptivity to the full variance $\sigma^2$ in the rate of decrease of $\|\nabla F(x)\|$.

**Theorem 2.8.** *For an L-smooth function F, assume Assumption 2.1, 2.5, and 2.6, suppose we run a per-coordinate variant of Algorithm 2.1, with regularizer $r(\eta) = \frac{L\alpha}{2}\left(\eta_i - \frac{1}{L}\right)^2 + \mathcal{I}\left(\eta_i \in \left[0, \frac{2}{L}\right]\right)$ in each coordinate i with $\alpha > 0$. Then, for each $i \in$*

$\{1, \ldots, d\}$, *we have*

$$\mathbb{E}\left[\sum_{t=1}^{T}\left(\frac{\partial F}{\partial x_i}(\boldsymbol{x}_t)\right)^2\right] \le 2L\left(F(\boldsymbol{x}_1) - F^\star + \sum_{i=1}^{d}\frac{5G_i^2}{L}\ln\left(1 + \frac{G_i^2 T}{\alpha}\right)\right)$$

$$+ \sqrt{2L\sigma_i^2 T\left(F(\boldsymbol{x}_1) - F^\star + \frac{d\alpha}{2L}\right)}$$

$$+ \sqrt{10\sigma_i^2 T\sum_{i=1}^{d}G_i^2\ln\left(1 + \frac{G_i^2 T}{\alpha}\right)}$$

$$+ (d-1)\alpha .$$

*Further, with* $\sigma^2 = \sum_{t=1}^{T}\sigma_i^2$ *it also holds*

$$\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla F(\boldsymbol{x}_t)\|^2\right] \le 2L\left(F(\boldsymbol{x}_1) - F^\star + \frac{5}{L}\sum_{i=1}^{d}G_i^2\ln\left(1 + \frac{G_i^2 T}{\alpha}\right)\right)$$

$$+ \sqrt{2L\sigma^2 T\left(F(\boldsymbol{x}_1) - F^\star + \frac{d\alpha}{2L}\right)}$$

$$+ \sqrt{10\sigma^2 T\sum_{i=1}^{d}G_i^2\ln\left(1 + \frac{G_i^2 T}{\alpha}\right)} .$$

*Proof of Theorem 2.8.* The proof is nearly identical to that of Theorem 2.5. We have

$$\mathbb{E}\left[\sum_{i=1}^{d}\left(\eta_i - \frac{L}{2}\eta_i^2\right)\sum_{t=1}^{T}\left(\frac{\partial F}{\partial x_i}(\boldsymbol{x}_t)\right)^2\right] \le F(\boldsymbol{x}_1) - F^\star + \frac{L\alpha}{2}\sum_{i=1}^{d}\left(\eta_i - \frac{1}{L}\right)^2$$

$$+ \sum_{i=1}^{d}\frac{5G_i^2}{L}\ln\left(1 + \frac{G_i^2 T}{\alpha}\right) + \sum_{i=1}^{d}\frac{L\eta_i^2\sigma_i^2 T}{2} .$$

Define $A_i = \mathbb{E}\left[\sum_{t=1}^{T}\left(\frac{\partial F}{\partial x_i}(\boldsymbol{x}_t)\right)^2\right]$ and set $\eta_i = \frac{\alpha + A_i}{L(\alpha + A_i + \sigma_i^2 T)}$ to obtain

$$\sum_{i=1}^{d}\frac{(\alpha + A_i)^2}{2L(\alpha + A_i + \sigma_i^2 T)} \le F(\boldsymbol{x}_1) - F^\star + \frac{d\alpha}{2L} + \sum_{i=1}^{d}\frac{5G_i^2}{L}\ln\left(1 + \frac{G_i^2 T}{\alpha}\right) .$$

Now, the first statement of the Theorem follows by observing that each term on the LHS is non-negative so that the sum can be lower-bounded by any individual term. For the second statement, define

$$Q_i = \frac{(\alpha + A_i)^2}{2L\,(\alpha + A_i + \sigma_i^2 T)},$$

$$Q = F(\boldsymbol{x}_1) - F^\star + \frac{d\alpha}{2L} + \sum_{i=1}^{d} \frac{5G_i^2}{L} \ln\left(1 + \frac{G_i^2 T}{\alpha}\right),$$

so that $\sum_{i=1}^{d} Q_i \leq Q$. By the quadratic formula and definition of $Q_i$, we have

$$A_i \leq 2LQ_i + \sqrt{2LQ_i \sigma_i^2 T} - \alpha \;.$$

Thus,

$$\sum_{i=1}^{d} A_i \leq 2LQ - d\alpha + \sum_{i=1}^{d} \sqrt{2LQ_i \sigma_i^2 T}$$

$$\leq 2LQ - d\alpha + \sqrt{2L}\sqrt{\sum_{i=1}^{d} Q_i}\sqrt{\sum_{i=1}^{d} \sigma_i^2 T}$$

$$= 2LQ - d\alpha + \sqrt{2LQ\sigma^2 T} \;.$$

From which the second statement follows. $\qquad\square$

### 2.2.4  Summary

In summary, we have presented a novel way to reduce the adaptation of step sizes for the stochastic optimization of smooth non-convex functions to an online convex optimization problem. The reduction goes through the use of novel surrogate convex losses. This framework allows us to use no-regret online algorithms to learn step sizes

on the fly. The resulting algorithm has an optimal convergence guarantee for any level of noise, without the need to estimate the noise or tune the step sizes. The overall price to pay is a factor of 2 in the computation of the gradients. We also have presented a per-coordinate version of our algorithm that achieves faster convergence on the coordinates with less noise.

As a side note, the optimal convergence rate was also obtained by Ward et al. (2019) using AdaGrad global step sizes, without the need to tune parameters. Li and Orabona (2019) improves over the results of Ward et al. (2019) by removing the assumption of bounded gradients. However, both analyses focus on the adaptivity of non-per-coordinate updates and are somewhat complicated in order to deal with unbounded gradients or non-independence of the current step size from the current step gradient. In comparison, our technique is relatively simple, allowing us to easily show a nontrivial guarantee for per-coordinate updates.

The idea of tuning step sizes with online learning has been explored in the online convex optimization literature (Koolen et al., 2014; van Erven and Koolen, 2016). There, the possible step sizes are discretized and an expert algorithm is used to select the step size to use online. Instead, in our work the use of convex surrogate loss functions allows us to directly learn the optimal step size, without needing to discretize the range of step sizes.

## 2.3   Adaptation to Noise under the PL condition

In the previous section, we showed how to adapt to the noise in the general smooth non-convex scenario. In practice, we might learn additional properties of the problem which we can use to choose/set algorithms to make it converge faster under these conditions. For example, if $F$ is a convex function, then GD can obtain a convergence
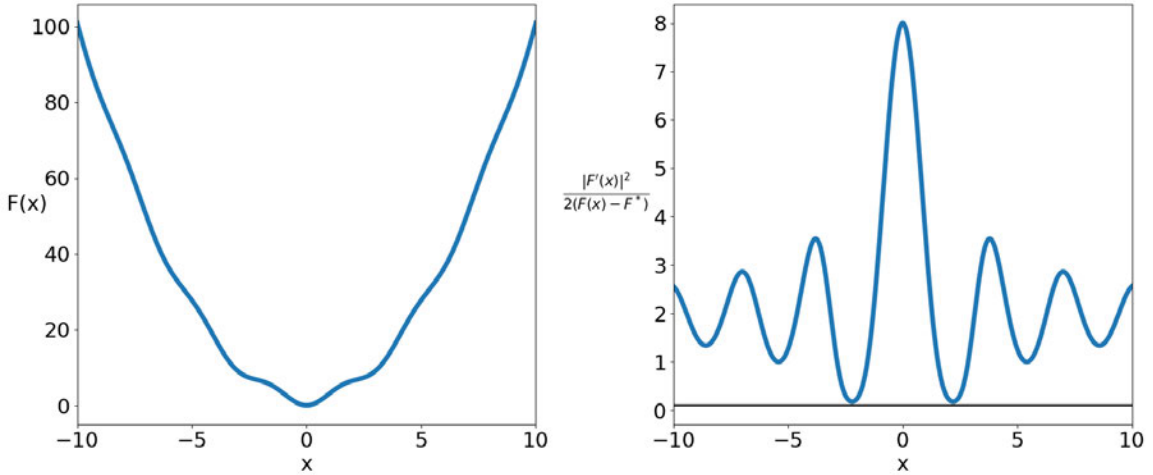
**Figure 2·3:** The left figure plots the function $F(x) = x^2 + 3\sin^2(x)$ and the right one shows that it satisfies the PL condition where the black line is $y = \frac{1}{10}$.

rate of $O(\frac{1}{\sqrt{T}})$ with a step size proportional to $\frac{1}{\sqrt{T}}$; while if $F$ is in addition $L$-smooth and $\mu$-strongly-convex, GD with a constant step size $\frac{1}{L}$ guarantees a linear convergence rate of $O(\exp(-\frac{\mu}{L}T))$. Similarly, for the non-convex setting, there are conditions under which we can also get a linear rate (Karimi et al., 2016). One popular option is the Polyak-Łojasiewicz (PL) condition (Polyak, 1963; Łojasiewicz, 1963):

**Assumption 2.7.** *A differentiable function $F : \mathbb{R}^d \to \mathbb{R}$ is said to satisfy the PL condition if for some $\mu > 0$ we have for any $\boldsymbol{x} \in \mathbb{R}^d$ that*

$$\frac{1}{2}\|\nabla F(\boldsymbol{x})\|^2 \geq \mu\left(F(\boldsymbol{x}) - F^\star\right) \ .$$

In words, the gradient grows as at least as a quadratic function of the sub-optimality. As an example, we show in Figure 2·3 a function $F(x) = x^2 + 3\sin^2(x)$ which satisfies the PL condition with $\mu = \frac{1}{10}$ (Karimi et al., 2016).

We want to stress that a function satisfying the PL condition is not necessarily convex as defined in (1.2), which can be clearly seen from Figure 2·3. Yet, the PL condition does imply the weaker condition of *invexity* (Hanson, 1981). Recall that a

function $F$ is invex if it is differentiable and there exists a vector valued function $\boldsymbol{v}$ such that for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ , it holds that:

$$F(\boldsymbol{y}) \geq F(\boldsymbol{x}) + \langle \nabla F(\boldsymbol{x}), \boldsymbol{v}(\boldsymbol{y}, \boldsymbol{x}) \rangle \ .$$

Obviously, invexity admits convexity as a special case of $\boldsymbol{v}(\boldsymbol{y}, \boldsymbol{x}) = \boldsymbol{y} - \boldsymbol{x}$. As a smooth function $F$ is *invex* if and only if every stationary point (namely a point where the gradient is zero) of $F$ is a global minimum (Craven and Glover, 1985), any smooth function that satisfies the PL condition must be invex (Karimi et al., 2016).

Though the PL condition is often considered a "strong" condition, it was formally proved to hold locally in a sufficiently large neighborhood of the random initialization when training deep neural networks in Allen-Zhu et al. (2019). Furthermore, Kleinberg et al. (2018) empirically observed that the loss surface of neural networks has good one-point convexity properties, and thus locally satisfies the PL condition, at least for the whole neighborhood along the SGD trajectory. For us, we only need it to hold along the optimization path and not over the entire space, as also pointed out in Karimi et al. (2016). So, while being strong, it actually models the cases we are interested in. Moreover, dictionary learning (Arora et al., 2015), phase retrieval (Chen and Candes, 2015), and matrix completion (Sun and Luo, 2016), all satisfy the one-point convexity locally (Zhu, 2018), and in turn they all satisfy the PL condition locally.

Apart from the linear rate GD obtains in the deterministic setting under the PL condition by using a constant step size, for the stochastic setting, the best rate we know of is $O(\frac{\sigma^2}{\mu^2 T})$ obtainable using SGD with the decaying step size $O(\frac{1}{\mu t})$ (Karimi et al., 2016; Li et al., 2021). Clearly, to achieve the best performance in each setting, we need two completely different step size decay schedules.

Below, we prove that using two popular empirical step size schedules, the expo-

nential and the cosine step sizes, SGD's convergence rate adapts to the noise.

### 2.3.1 Exponential and Cosine Step Sizes

Specifically, we use the following definition for the exponential step size

$$\eta_t = \eta_0 \cdot \alpha^t, \tag{2.4}$$

and for the cosine step size (Loshchilov and Hutter, 2017)

$$\eta_t = \frac{\eta_0}{2}\left(1 + \cos\frac{t\pi}{T}\right). \tag{2.5}$$

The exponential step size is simply an exponential decaying step size. It is less discussed in the optimization literature and it is also unclear who proposed it first, even if it has been known to practitioners for a long time and already included in many deep learning software libraries including TensorFlow (Abadi et al., 2015) and PyTorch (Paszke et al., 2019). Yet, no convergence guarantee has ever been proved for it. The closest strategy is the *stagewise step decay*, which corresponds to the discrete version of the exponential step size we analyze. The stagewise step decay uses a piece-wise constant step size strategy, where the step size is cut by a factor in each "stage". This strategy is known with many different names: "stagewise step size" (Yuan et al., 2019), "step decay schedule" (Ge et al., 2019), "geometrically decaying schedule" (Davis et al., 2021), and "geometric step decay" (Davis et al., 2019). In this paper, we will call it stagewise step decay. The stagewise step decay approach was first introduced in (Goffin, 1977) and used in many *convex* optimization problem (e.g., Hazan and Kale, 2014; Aybat et al., 2019; Kulunchakov and Mairal, 2019; Ge et al., 2019). Interestingly, Ge et al. (2019) also shows promising empirical results

on non-convex functions, but instead of using their proposed decay strategy, they use an exponentially decaying schedule, like the one we analyze here. The only use of the stagewise step decay for non-convex functions we know are for sharp functions (Davis et al., 2019) and weakly-quasi-convex functions (Yuan et al., 2019). However, they do not show any adaptation property and they still do not consider the exponential step size but rather its discrete version.

The cosine step size, which anneals the step size following a cosine function, has exhibited great power in practice but it does not have any theoretical justification. The cosine step size was originally presented in Loshchilov and Hutter (2017) with two tunable parameters. Later, He et al. (2019) proposed a simplified version of it with one parameter. However, there is no theory for this strategy though it is popularly used in the practical world (Liu et al., 2019; Zhang et al., 2019b; Cubuk et al., 2019; Zhao et al., 2020; You et al., 2020; Chen et al., 2020b; Grill et al., 2020).

### 2.3.2 Theoretical Analyses of Exponential and Cosine Step Sizes

We now prove the convergence guarantees for these two step sizes showing that they can adapt to noise. We would use the following assumption on noises:

**Assumption 2.8.** *For $t = 1, 2, \ldots, T$, we assume $\mathbb{E}_t[\|\nabla f(\boldsymbol{x}_t, \xi_t) - \nabla F(\boldsymbol{x}_t)\|^2] \leq a\|\nabla F(\boldsymbol{x}_t)\|^2 + b$, where $a, b \geq 0$.*

This assumption on the noise is strictly weaker than the common assumption of assuming a bounded variance (Assumption 2.2). Indeed, this assumption recovers the bounded variance case with $a = 0$ while also allowing for the variance to grow unboundedly far from the optimum when $a > 0$. This is indeed the case when the optimal solution has low training error and the stochastic gradients are generated by mini-batches. This relaxed assumption on the noise was first used by Bertsekas and

Tsitsiklis (1996) in the analysis of the asymptotic convergence of SGD.

**Theorem 2.9** (SGD with exponential step size). *Assume $F$ to be $L$-smooth and $\mu$-PL and Assumption 2.1 and 2.8 to hold. For a given $T \geq \max\{3, \beta\}$ and $\eta_0 = (L(1+a))^{-1}$, with step size (2.4), SGD guarantees*

$$\mathbb{E}F(\boldsymbol{x}_{T+1}) - F^{\star} \leq \frac{5LC(\beta)}{e^2\mu^2}\frac{\ln^2\frac{T}{\beta}}{T}b + C(\beta)\exp\left(-\frac{0.69\mu}{L+a}\left(\frac{T}{\ln\frac{T}{\beta}}\right)\right) \cdot (F(\boldsymbol{x}_1) - F^{\star}),$$

*where $C(\beta) \triangleq \exp\left((2\mu\beta)/(L(1+a)\ln T/\beta)\right)$.*

**Choice of $\beta$** Note that if $\beta = L(1+a)/\mu$, we get

$$\mathbb{E}F(\boldsymbol{x}_{T+1}) - F^{\star} \leq O\left(\exp\left(-\frac{\mu}{L+a}\left(\frac{T}{\ln\frac{\mu T}{L}}\right)\right) + \frac{b\ln^2\frac{\mu T}{L}}{\mu^2 T}\right).$$

In words, this means that we are basically free to choose $\beta$, but will pay an exponential factor in the mismatch between $\beta$ and $\frac{L}{\mu}$, which is basically the condition number for PL functions. This has to be expected because it also happens in the easier case of stochastic optimization of strongly convex functions (Bach and Moulines, 2011).

**Theorem 2.10** (SGD with cosine step size). *Assume $F$ to be $L$-smooth and $\mu$-PL and Assumption 2.1 and 2.8 to hold. For a given $T$ and $\eta_0 = (L(1+a))^{-1}$, with step size (2.5), SGD guarantees*

$$\mathbb{E}F(\boldsymbol{x}_{t+1}) - F^{\star} \leq \exp\left(-\frac{\mu(T-1)}{2L(1+a)}\right)(F(x_1) - F^{\star})$$
$$+ \frac{\pi^4 b}{32(1+a)T^4}\left(\left(\frac{8T^2}{\mu}\right)^{4/3} + \left(\frac{6T^2}{\mu}\right)^{\frac{5}{3}}\right).$$

**Adaptivity to Noise** From the above theorems, we can see that both the exponential step size and the cosine step size have a provable advantage over polynomial

ones: *adaptivity to the noise*. Indeed, when $b = 0$, namely there is only noise relative to the distance from the optimum, they both guarantee a linear rate. Meanwhile, if there is noise, using the *same step size without any tuning*, the exponential step size recovers the rate of $O\left(1/(\mu^2 T)\right)$ while the cosine step size achieves the rate of $O(1/(\mu^{\frac{5}{3}} T^{\frac{2}{3}}))$ (up to poly-logarithmic terms). In contrast, polynomial step sizes would require two different settings—decaying vs constant—in the noisy vs no-noise situation (Karimi et al., 2016). It is worth stressing that the rate in Theorem 2.9 is one of the first results in the literature on the stochastic optimization of smooth PL functions (Khaled and Richtárik, 2020).

It is worth reminding the reader that *any* polynomial decay of the step size does not give us this adaptation. So, let's gain some intuition on why this should happen with these two step sizes. In the early stage of the optimization process, we can expect that the disturbance due to the noise is relatively small compared to how far we are from the optimal solution. Accordingly, at this phase, a near-constant step size should be used. This is exactly what happens with (2.4) and (2.5). On the other hand, when the iterate is close to the optimal solution, we have to decrease the step size to fight the effects of the noise. In this stage, the exponential step size goes to 0 as $O\left(1/T\right)$, which is the optimal step size used in the noisy case. Meanwhile, the last $i$th cosine step size is $\eta_{T-i} = \frac{\eta_0}{2}(1 - \cos\frac{i\pi}{T}) = \eta_0 \sin^2\frac{i\pi}{2T}$, which amounts $O(1/T^2)$ when $i$ is much smaller than $T$.

**Optimality of the bounds** As far as we know, it is unknown if the rate we obtain for the optimization of non-convex smooth functions under the PL condition is optimal or not. However, up to poly-logarithmic terms, Theorem 2.9 matches at the same time the best-known rates for the noisy and deterministic cases (Karimi et al., 2016). We would remind the reader that this rate is not comparable with the one for strongly convex functions which is $O(1/(\mu T))$. Meanwhile, cosine step size

achieves a rate slightly worse in $T$ (but better in $\mu$) under the same assumptions.

Before proving the above theorems, we first introduce some technical lemmas.

**Lemma 2.11.** *Assume $F$ to be $L$-smooth, Assumption 2.1 and 2.8 to hold, and $\eta_t \leq \frac{1}{L(1+a)}$, then SGD guarantees*

$$\mathbb{E}F(\boldsymbol{x}_{t+1}) - \mathbb{E}F(\boldsymbol{x}_t) \leq -\frac{\eta_t}{2}\mathbb{E}\|\nabla F(\boldsymbol{x}_t)\|^2 + \frac{L\eta_t^2 b}{2} \ .$$

*Proof of Lemma 2.11.* By (1.5), we have

$$F(\boldsymbol{x}_{t+1}) \leq F(\boldsymbol{x}_t) - \langle \nabla F(\boldsymbol{x}_t), \eta_t \boldsymbol{g}_t \rangle + \frac{L}{2}\eta_t^2\|\boldsymbol{g}_t\|^2 \ .$$

Taking expectation on both sides, we get

$$\begin{aligned}
\mathbb{E}F(\boldsymbol{x}_{t+1}) - \mathbb{E}F(\boldsymbol{x}_t) &\leq -\left(\eta_t - \frac{L(a+1)}{2}\eta_t^2\right)\mathbb{E}\|\nabla F(\boldsymbol{x}_t)\|^2 + \frac{L}{2}\eta_t^2 b \\
&\leq -\frac{1}{2}\eta_t\mathbb{E}\|\nabla F(\boldsymbol{x}_t)\|^2 + \frac{L}{2}\eta_t^2 b,
\end{aligned}$$

where in the last inequality we used the fact that $\eta_t \leq \frac{1}{L(1+a)}$. $\qquad\square$

**Lemma 2.12.** *Assume $X_k, A_k, B_k \geq 0, k = 1, ...,$ and $X_{k+1} \leq A_k X_k + B_k$, we have*

$$X_{k+1} \leq \prod_{i=1}^{k} A_i X_1 + \sum_{i=1}^{k} \prod_{j=i+1}^{k} A_j B_i \ .$$

*Proof of Lemma 2.12.* When $k = 1$, $X_2 \leq A_1 X_1 + B_1$ satisfies. By induction, assume $X_k \leq \prod_{i=1}^{k-1} A_i X_1 + \sum_{i=1}^{k-1} \prod_{j=i+1}^{k-1} A_j B_i$, and we have

$$X_{k+1} \leq A_k \left( \prod_{i=1}^{k-1} A_i X_1 + \sum_{i=1}^{k-1} \prod_{j=i+1}^{k-1} A_j B_i \right) + B_k$$

$$= \prod_{i=1}^{k} A_i X_1 + \sum_{i=1}^{k-1} \prod_{j=i+1}^{k} A_j B_i + A_k B_k$$

$$= \prod_{i=1}^{k} A_i X_1 + \sum_{i=1}^{k} \prod_{j=i+1}^{k} A_j B_i . \qquad \square$$

**Lemma 2.13.** *For any $T \geq 1$, we have $\sum_{t=1}^{T} \cos \frac{t\pi}{T} = -1$.*

*Proof of Lemma 2.13.* If $T$ is odd, we have

$$\sum_{t=1}^{T} \cos \frac{t\pi}{T} = \cos \frac{T\pi}{T} + \sum_{t=1}^{(T-1)/2} \cos \frac{t\pi}{T} + \cos \frac{(T-t)\pi}{T} = \cos \pi = -1,$$

where in the second inequality we used the fact that $\cos(\pi - x) = -\cos(x)$ for any $x$.

If $T$ is even, we have

$$\sum_{t=1}^{T} \cos \frac{t\pi}{T} = \cos \frac{T\pi}{T} + \cos \frac{T\pi}{2T} + \sum_{t=1}^{T/2-1} \cos \frac{t\pi}{T} + \cos \frac{(T-t)\pi}{T} = \cos \pi = -1 .$$

$$\square$$

**Lemma 2.14.** *For $T \geq 3$, $\alpha \geq 0.69$ and $\frac{\alpha^{T+1}}{(1-\alpha)} \leq \frac{2\beta}{\ln \frac{T}{\beta}}$.*

*Proof of Lemma 2.14.* We have

$$\frac{\alpha^{T+1}}{(1-\alpha)} = \frac{\alpha\beta}{T(1-\alpha)} = \frac{\beta}{T\left(1 - \exp\left(-\frac{1}{T} \ln \frac{T}{\beta}\right)\right)} \leq \frac{2\beta}{\ln \frac{T}{\beta}},$$

where in the last inequality we used $\exp(-x) \leq 1 - \frac{x}{2}$ for $0 < x < \frac{1}{e}$ and the fact that $\frac{1}{T} \ln \left(\frac{T}{\beta}\right) \leq \frac{\ln T}{T} \leq \frac{1}{e}$. $\qquad \square$

**Lemma 2.15.** $1 - x \leq \ln \left(\frac{1}{x}\right), \forall x > 0$.

*Proof of Lemma 2.15.* It is enough to prove that $f(x) := x - 1 - \ln x \geq 0$. Observe that $f'(x)$ is increasing and $f'(1) = 0$, hence, we have $f(x) \geq f(1) = 0$. $\qquad \square$

**Lemma 2.16.** *Let $a, b \geq 0$. Then*

$$\sum_{t=0}^{T} \exp(-bt)t^a \leq 2\exp(-a) \left(\frac{a}{b}\right)^a + \frac{\Gamma(a+1)}{b^{a+1}} \ .$$

*Proof of Lemma 2.16.* Note that $f(t) = \exp(-bt)t^a$ is increasing for $t \in [0, a/b]$ and decreasing for $t \geq a/b$. Hence, we have

$$\sum_{t=0}^{T} \exp(-bt)t^a \leq \sum_{t=0}^{\lfloor a/b \rfloor - 1} \exp(-bt)t^a + \exp(-b\lfloor a/b \rfloor)\lfloor a/b \rfloor^a + \exp(-b\lceil a/b \rceil)\lceil a/b \rceil^a$$

$$+ \sum_{\lceil a/b \rceil + 1}^{T} \exp(-bt)t^a$$

$$\leq 2\exp(-a)(a/b)^a + \int_{0}^{\lfloor a/b \rfloor} \exp(-bt)t^a dt + \int_{\lceil a/b \rceil}^{T} \exp(-bt)t^a dt$$

$$\leq 2\exp(-a)(a/b)^a + \int_{0}^{T} \exp(-bt)t^a dt$$

$$\leq 2\exp(-a)(a/b)^a + \int_{0}^{\infty} \exp(-bt)t^a dt$$

$$= 2\exp(-a)(a/b)^a + \frac{1}{b^{a+1}}\Gamma(a+1) \ . \qquad \square$$

We can now prove both Theorem 2.9 and Theorem 2.10.

*Proof of Theorem 2.9 and Theorem 2.10.* Denote $\mathbb{E}[F(\boldsymbol{x}_t)] - F^\star$ by $\Delta_t$. From Lemma 2.11 and the PL condition, we get

$$\Delta_{t+1} \leq (1 - \mu\eta_t)\Delta_t + \frac{L}{2}\eta_t^2 b^2 \ .$$

By Lemma 2.12 and $1 - x \leq \exp(-x)$, we have

$$
\Delta_{T+1} \leq \prod_{t=1}^{T}(1 - \mu\eta_t)\Delta_1 + \frac{L}{2}\sum_{t=1}^{T}\prod_{i=t+1}^{T}(1 - \mu\eta_i)\eta_t^2 b
$$

$$
\leq \exp\left(-\mu\sum_{t=1}^{T}\eta_t\right)\Delta_1 + \frac{Lb}{2}\sum_{t=1}^{T}\exp\left(-\mu\sum_{i=t+1}^{T}\eta_i\right)\eta_t^2 .
$$

We then show that both the exponential step size and the cosine step size satisfy $\sum_{t=1}^{T}\eta_t = \Omega(T)$, which guarantees a linear rate in the noiseless case.

For the cosine step size (2.5), we observe that

$$
\sum_{t=1}^{T}\eta_t = \frac{\eta_0 T}{2} + \frac{\eta_0}{2}\sum_{t=1}^{T}\cos\frac{t\pi}{T} = \frac{\eta_0(T-1)}{2},
$$

where in the last equality we used Lemma 2.13.

Also, for the exponential step size (2.4), we can show that

$$
\sum_{t=1}^{T}\eta_t = \eta_0\frac{\alpha - \alpha^{T+1}}{1 - \alpha} \geq \frac{\eta_0\alpha}{1 - \alpha} - \frac{2\eta_0\beta}{\ln\frac{T}{\beta}} \geq T \cdot \frac{0.69\eta_0}{\ln\frac{T}{\beta}} - \frac{2\eta_0\beta}{\ln\frac{T}{\beta}},
$$

where we used Lemma 2.14 in the first inequality and Lemma 2.15 in the second.

Next, we upper bound $\sum_{t=1}^{T}\exp\left(-\mu\sum_{i=t+1}^{T}\eta_i\right)\eta_t^2$ for the two step sizes.

For the exponential step size, by Lemma 2.14, we obtain

$$
\sum_{t=1}^{T}\exp\left(-\mu\sum_{i=t+1}^{T}\eta_i\right)\eta_t^2 = \eta_0^2\sum_{t=1}^{T}\exp\left(-\mu\eta_0\frac{\alpha^{t+1} - \alpha^{T+1}}{1 - \alpha}\right)\alpha^{2t}
$$

$$
\leq \eta_0^2 C(\beta)\sum_{t=1}^{T}\exp\left(-\frac{\mu\eta_0\alpha^{t+1}}{1 - \alpha}\right)\alpha^{2t}
$$

$$
\leq \eta_0^2 C(\beta)\sum_{t=1}^{T}\left(\frac{e}{2}\frac{\mu\alpha^{t+1}}{L(1+a)(1-\alpha)}\right)^{-2}\alpha^{2t}
$$

$$\leq \frac{4L^2(1+a)^2}{e^2\mu^2} \sum_{t=1}^{T} \frac{1}{\alpha^2} \ln^2\left(\frac{1}{\alpha}\right)$$

$$\leq \frac{10L^2(1+a)^2 \ln^2 \frac{T}{\beta}}{e^2\mu^2 T},$$

where in the second inequality we used $\exp(-x) \leq \left(\frac{\gamma}{ex}\right)^\gamma, \forall x > 0, \gamma > 0$.

For the cosine step size, using the fact that $\sin x \geq \frac{2}{\pi}x$ for $0 \leq x \leq \frac{\pi}{2}$, we can lower bound $\sum_{i=t+1}^{T} \eta_i$ by

$$\sum_{i=t+1}^{T} \eta_i = \frac{\eta_0}{2} \sum_{i=t+1}^{T} \left(1 + \cos\frac{i\pi}{T}\right) = \frac{\eta_0}{2} \sum_{i=0}^{T-t-1} \sin^2\frac{i\pi}{2T} \geq \frac{\eta_0}{2T^2} \sum_{i=0}^{T-t-1} i^2 \geq \frac{\eta_0(T-t-1)^3}{6T^2} .$$

Then, we proceed to get

$$\sum_{t=1}^{T} \exp\left(-\mu \sum_{i=t+1}^{T} \eta_i\right) \eta_t^2 \leq \frac{\eta_0^2}{4} \sum_{t=1}^{T} \left(1 + \cos\frac{t\pi}{T}\right)^2 \exp\left(-\frac{\mu\eta_0(T-t-1)^3}{6T^2}\right)$$

$$= \frac{\eta_0^2}{4} \sum_{t=1}^{T-1} \left(1 - \cos\frac{t\pi}{T}\right)^2 \exp\left(-\frac{\eta_0\mu(t-1)^3}{6T^2}\right)$$

$$= \eta_0^2 \sum_{t=1}^{T-1} \sin^4\frac{t\pi}{2T} \exp\left(-\frac{\eta_0\mu(t-1)^3}{6T^2}\right)$$

$$\leq \frac{\eta_0^2\pi^4}{16T^4} \sum_{t=0}^{T-1} t^4 \exp\left(-\frac{\eta_0\mu t^3}{6T^2}\right)$$

$$\leq \frac{\eta_0\pi^4}{16T^4} \left(2\exp\left(-\frac{4}{3}\right) \left(\frac{8T^2}{\mu}\right)^{4/3} + \left(\frac{6T^2}{\mu}\right)^{\frac{5}{3}}\right),$$

where in the third line we used $\cos(\pi - x) = -\cos(x)$, in the forth line we used $1 - \cos(2x) = 2\sin^2(x)$, and in the last inequality we applied Lemma 2.16.

Putting things together, we get the stated bounds. □

### 2.3.3  Experiments Comparing Exponential and Cosine Step Sizes with Other Optimizers

The empirical performance of the exponential and the cosine step sizes is already well-known in the applied world and does not require additional validation. However, both step sizes are often missing as baselines in recent empirical evaluations. Hence, the main aim of this section is to provide a comparison of the exponential and the cosine step sizes to other popular state-of-the-art step size schedules. All experiments are done in PyTorch (Paszke et al., 2019) and the codes can be found at `https://github.com/zhenxun-zhuang/SGD-Exponential-Cosine-Stepsize`.

We conducted experiments using deep neural networks to do image classification tasks on various datasets with different network architectures.

**Datasets** We consider the image classification task on CIFAR-10/100 and FashionMNIST. For all datasets, we randomly select 10% training images for validation.

**Data Normalization and Augmentation** Images are normalized per channel using the means and standard deviations computed from all training images. For CIFAR-10/100, we adopt the data augmentation technique following Lee et al. (2015) (for training only): 4 pixels are padded on each side of an image and a $32 \times 32$ crop is randomly sampled from the padded image or its horizontal flip.

**Models** For FashionMNIST, we use a CNN model consisting of two alternating stages of $5 \times 5$ convolutional filters and $2 \times 2$ max-pooling followed by one fully connected layer of 1024 units. To reduce overfitting, 50% dropout noise is used during training. For CIFAR-10, we employ the 20-layer Residual Network model (He et al., 2016). For CIFAR-100, we utilize the DenseNet-BC model (Huang et al., 2017) with 100 layers and a growth rate of 12. The loss is cross-entropy.

**Training** During the validation stage, we tune each method using grid-search to

select the hyperparameters that work best according to their respective performance on the validation set. At the testing stage, the best performing hyperparameters from the validation stage are employed to train the model over all training images. The testing stage is repeated with random seeds 5 times to eliminate the influence of stochasticity.

We use Nesterov momentum (Nesterov, 1983) of 0.9 without dampening (if having this option), weight-decay of 0.0001 (FashionMNIST and CIFAR-10), and 0.0005 (CIFAR100), and use a batch size of 128. Regarding the employment of Nesterov momentum, we follow the setting of Ge et al. (2019). The use of momentum is essential to have a fair and realistic comparison in that the majority of practitioners would use it when using SGD.

**Optimization methods** We consider SGD with the following step size decay schedules:

$$
\begin{aligned}
\eta_t &= \eta_0 \cdot \alpha^t; \quad \eta_t = \eta_0(1 + \alpha\sqrt{t})^{-1}; \\
\eta_t &= \eta_0(1 + \alpha t)^{-1}; \quad \eta_t = \eta_0/2\left(1 + \cos\left(t\pi/T\right)\right),
\end{aligned}
\tag{2.6}
$$

where $t$ is the iteration number (instead of the number of epochs). We also compare with Adam (Kingma and Ba, 2015), SGD+Armijo (Vaswani et al., 2019), PyTorch's ReduceLROnPlateau scheduler[1] and stagewise step decay. We will call the place of decreasing the step size in stagewise step decay a **milestone**. (As a side note, since we use Nesterov momentum in all SGD variants, the stagewise step decay basically covers the performance of multistage accelerated algorithms (e.g., Aybat et al., 2019).)

**Hyperparameter tuning** We tune the hyperparameters on the validation set using the following two-stage grid searching strategy. First, search over a coarse grid and

---

[1] `https://pytorch.org/docs/stable/optim.html`

select the one yielding the best validation results. Next, continue searching in a fine grid centering at the best-performing hyperparameters found in the coarse stage, and in turn, take the best one as the final choice.

For the starting step size $\eta_0$, the coarse searching grid is {0.00001, 0.0001, 0.001, 0.01, 0.1, 1}, and the fine grid is like {0.006, 0.008, 0.01, 0.02, 0.04} if the best one in the coarse stage is 0.01.

For the $\alpha$ value, we set its searching grid so that the ratio $\eta_T/\eta_0$, where $\eta_T$ is the step size in the last iteration, is first searched over the coarse grid of {0.00001, 0.0001, 0.001, 0.01, 0.1, 1}, and then over a fine grid centered at the best one of the coarse stage. Note that we try all pairs of $(\eta_0, \alpha)$ from their respective searching grids.

For the stagewise step decay, to make the tuning process more thorough, we modify as follows the one employed in Section 6.1 (specifically on tuning SGD V1) of Yuan et al. (2019), where they first set two milestones and then tune the starting step size. Put it explicitly and take the experiment on CIFAR-10 as an example, we first run vanilla SGD with a constant step size to search for a good range of starting step size on the grid {0.00001, 0.0001, 0.001, 0.01, 0.1, 1}, and find 0.01 and 0.1 work well. Based on this, we set the fine searching grid of starting step sizes as {0.007, 0.01, 0.04, 0.07, 0.1, 0.4}. For each of them, we run three settings with an increasing number of milestones: vanilla SGD (with no milestone), SGD with 1 milestone, and SGD with 2 milestones. The searching grid for milestones is {16k, 24k, 32k, 40k, 48k, 56k} (number of iterations). For the 1 milestone setting, the milestone can be any of them. For the 2 milestones, they can be any combination of two different elements from the searching grid, like (16k, 32k) or (32k, 48k). The grid search strategy for FashionMNIST and CIFAR-100 is similar but with the searching grid for milestones over {3k, 6k, 9k, 12k, 15k, 18k}.

The PyTorch ReduceLROnPlateau scheduler takes multiple arguments, among
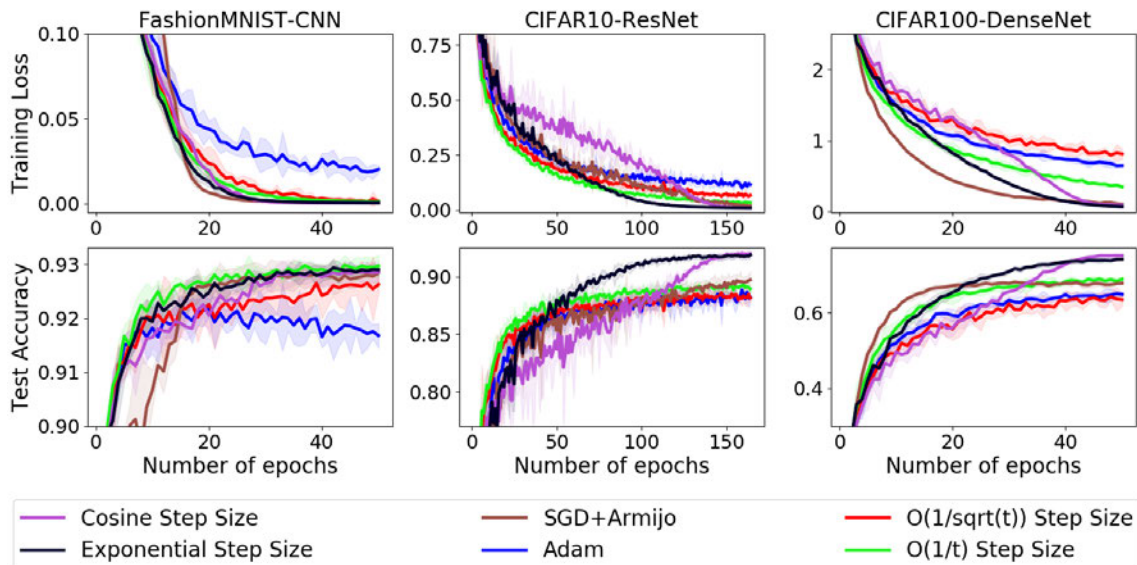
**Figure 2·4:** Training loss (top plots) and test accuracy (bottom plots) curves on employing different step size schedules to do image classification using a simple CNN for FashionMNIST (left), a 20-layer ResNet for CIFAR-10 (middle), and a 100-layer DenseNet on CIFAR-100 (right). *(The shading of each curve represents the 95% confidence interval computed across five independent runs from random initial starting points.)*

which we tune the starting step size, the factor argument which decides by which the step size will be reduced, the patience argument which controls the number of epochs with no improvement after which the step size will be reduced, and the threshold argument which measures the new optimum to only focus on significant changes. We choose the searching grid for the starting step size using the same strategy for stagewise step decay above: run SGD with a constant step size to search for a good starting step size, then search over a grid centering on the found value, which results in the grid {0.004, 0.007, 0.01, 0.04, 0.07} (FashionMNIST) and {0.01, 0.04, 0.07, 0.1, 0.4} (CIFAR10/100). We also explore the searching grid of the factor argument over {0.1, 0.5}, the patience argument over {5, 10} (CIFAR10) or {3, 6} (FashionMNIST/CIFAR100), and the threshold argument over {0.0001, 0.001, 0.01, 0.1}.

For each setting, we choose the combination of hyperparameters that gives the
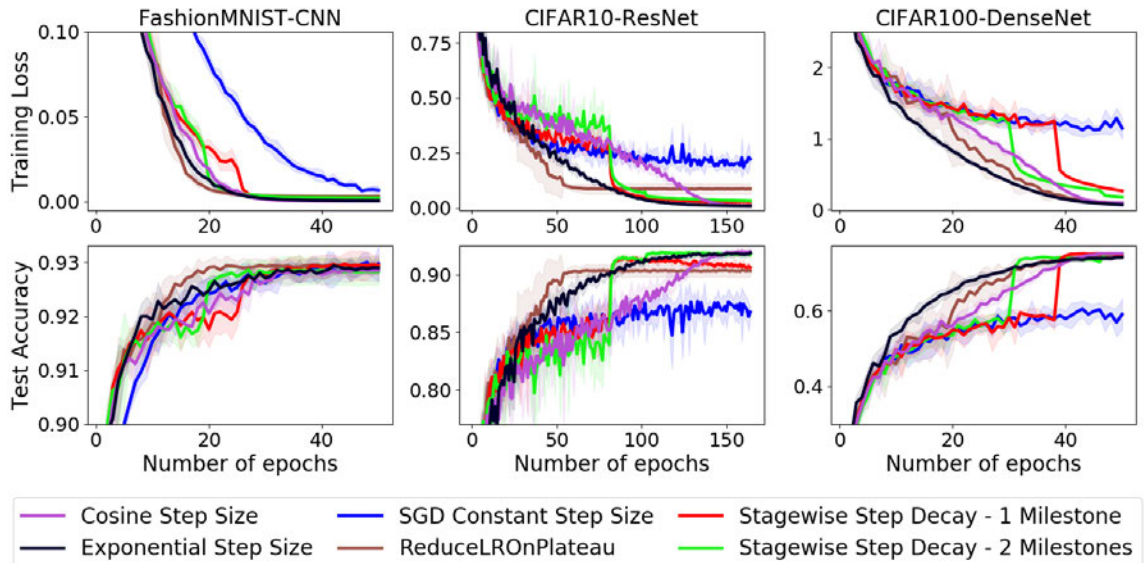
**Figure 2·5:** Training loss (top plots) and test accuracy (bottom plots) curves comparing the exponential and cosine step sizes with stagewise step decay for image classification using a simple CNN for FashionMNIST (left), a 20-layer ResNet for CIFAR-10 (middle), and a 100-layer DenseNet on CIFAR-100 (right). *(The shading of each curve represents the 95% confidence interval computed across five independent runs from random initial starting points.)*

best final validation loss to be used in testing. Also, whenever the best-performing hyperparameters lie in the boundary of the searching grid, we always extend the grid to make the final best-performing hyperparameters fall into the interior of the grid.

**Results and discussions** The exact loss and accuracy values are reported in Table 2.1. To avoid overcrowding the figures, we compare the algorithms in groups of baselines. The comparison of performance between step size schemes listed in (2.6), Adam, and SGD+Armijo are shown in Figure 2·4. As can be seen, the *only* two methods that perform well on *all* 3 datasets are cosine and exponential step sizes. In particular, the cosine step size performs the best across datasets both in training loss and test accuracy, with the exponential step size following closely.

On the other hand, as we noted above, stagewise step decay is a very popular decay schedule in deep learning. Thus, our second group of baselines in Figure 2·5 is

**Table 2.1:** Average final training loss and test accuracy achieved by each method when optimizing respective models on each dataset. The $\pm$ shows 95% confidence intervals of the mean loss/accuracy value over 5 runs starting from different random seeds.

**(a)** FashionMNIST – 3 layer CNN

| Methods | Training loss | Test accuracy |
|---|---|---|
| SGD Constant Step Size | $0.0068 \pm 0.0023$ | $0.9297 \pm 0.0033$ |
| $O(1/t)$ Step Size | $0.0013 \pm 0.0004$ | $\mathbf{0.9297 \pm 0.0021}$ |
| $O(1/\sqrt{t})$ Step Size | $0.0016 \pm 0.0005$ | $0.9262 \pm 0.0014$ |
| Adam | $0.0203 \pm 0.0021$ | $0.9168 \pm 0.0023$ |
| SGD+Armijo | $\mathbf{0.0003 \pm 0.0000}$ | $0.9284 \pm 0.0016$ |
| ReduceLROnPlateau | $0.0031 \pm 0.0009$ | $0.9294 \pm 0.0015$ |
| Stagewise - 1 Milestone | $0.0007 \pm 0.0002$ | $0.9294 \pm 0.0018$ |
| Stagewise - 2 Milestones | $0.0023 \pm 0.0005$ | $0.9283 \pm 0.0024$ |
| Exponential Step Size | $0.0006 \pm 0.0001$ | $0.9290 \pm 0.0009$ |
| Cosine Step Size | $0.0004 \pm 0.0000$ | $0.9285 \pm 0.0019$ |

**(b)** CIFAR10 – 20 layer Resnet

| Methods | Training loss | Test accuracy |
|---|---|---|
| SGD Constant Step Size | $0.2226 \pm 0.0169$ | $0.8674 \pm 0.0048$ |
| $O(1/t)$ Step Size | $0.0331 \pm 0.0028$ | $0.8894 \pm 0.0040$ |
| $O(1/\sqrt{t})$ Step Size | $0.0672 \pm 0.0086$ | $0.8814 \pm 0.0034$ |
| Adam | $0.1161 \pm 0.0111$ | $0.8823 \pm 0.0041$ |
| SGD+Armijo | $0.0185 \pm 0.0043$ | $0.8973 \pm 0.0071$ |
| ReduceLROnPlateau | $0.0867 \pm 0.0230$ | $0.9033 \pm 0.0049$ |
| Stagewise - 1 Milestone | $0.0269 \pm 0.0017$ | $0.9062 \pm 0.0020$ |
| Stagewise - 2 Milestones | $0.0322 \pm 0.0008$ | $0.9174 \pm 0.0020$ |
| Exponential Step Size | $\mathbf{0.0098 \pm 0.0010}$ | $0.9188 \pm 0.0033$ |
| Cosine Step Size | $0.0106 \pm 0.0008$ | $\mathbf{0.9199 \pm 0.0029}$ |

**(c)** CIFAR100 – 100 layer DenseNet-BC

| Methods | Training loss | Test accuracy |
|---|---|---|
| SGD Constant Step Size | $1.1467 \pm 0.1437$ | $0.5896 \pm 0.0404$ |
| $O(1/t)$ Step Size | $0.3489 \pm 0.0263$ | $0.6874 \pm 0.0076$ |
| $O(1/\sqrt{t})$ Step Size | $0.8147 \pm 0.0717$ | $0.6336 \pm 0.0169$ |
| Adam | $0.6513 \pm 0.0154$ | $0.6478 \pm 0.0054$ |
| SGD+Armijo | $0.1063 \pm 0.0153$ | $0.6768 \pm 0.0044$ |
| ReduceLROnPlateau | $0.0927 \pm 0.0085$ | $0.7435 \pm 0.0076$ |
| Stagewise - 1 Milestone | $0.2673 \pm 0.0084$ | $0.7459 \pm 0.0030$ |
| Stagewise - 2 Milestones | $0.1783 \pm 0.0030$ | $0.7487 \pm 0.0025$ |
| Exponential Step Size | $\mathbf{0.0714 \pm 0.0041}$ | $0.7398 \pm 0.0037$ |
| Cosine Step Size | $0.0949 \pm 0.0053$ | $\mathbf{0.7497 \pm 0.0044}$ |

composed of the stagewise step decay, ReduceLROnPlateau, and SGD with constant step size. The results show that exponential and cosine step sizes can still match or exceed the best of them with a fraction of their needed time to find the best hyperparameters. Indeed, we need 4 hyperparameters for two milestones, 3 for one

milestone, and at least 4 for ReduceLROnPlateau. In contrast, the cosine step size requires only 1 hyperparameter and the exponential one needs 2.

Note that we do not pretend that our benchmark of the stagewise step decay is exhaustive. Indeed, there are many unexplored (potentially infinite!) possible hyperparameter settings. For example, it is reasonable to expect that adding even more milestones at the appropriate times could lead to better performance. However, this would result in a linear growth of the number of hyperparameters leading to an exponential increase in the number of possible location combinations.

This in turn causes the rapid growth of tuning time in selecting a good set of milestones in practice. Worse still, even the intuition that one should decrease the step size once the test loss curve stops decreasing is not always correct. Indeed, we observed in experiments (see Figure 2·6) that doing this will, after the initial drop of the curve in response to the step size decrease, make the test loss curve gradually go up again.
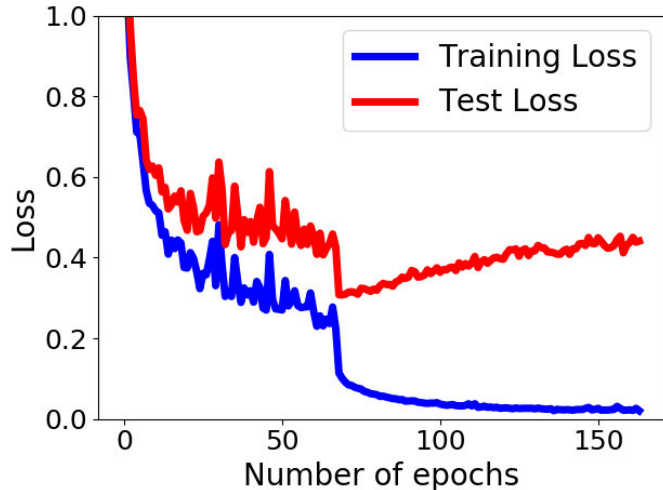


**Figure 2·6:** Plot showing that decreasing the step size too soon would lead to overfitting (ResNet20 on CIFAR10).

### 2.3.4 Summary

We have analyzed theoretically and empirically the exponential and the cosine step sizes, two successful step size decay schedules for the stochastic optimization of non-

convex functions. We have shown that, in the case of functions satisfying the PL condition, they are both adaptive to the level of noise. Furthermore, we have validated our theoretical findings on real-world tasks, showing that these two step sizes consistently match or outperform other strategies, while at the same time requiring only 1 (cosine) / 2 (exponential) hyperparameters to tune.

## 2.4   Conclusion

In this chapter, we introduced the notion of adaptation to noise as automatically guaranteeing (near) optimal convergence rates for different levels of noise without knowing it nor needing to tune any parameter. We then presented our works on designing/analyzing algorithms that can adapt to the noise in both the general smooth non-convex setting and the setting with the additional PL condition.

# Chapter 3

# Adaptation to Gradient Scales

[The results in this chapter appeared in Zhuang et al. (2022).]

This chapter studies the phenomenon that the scales of gradient magnitudes in each layer can scatter across a very wide range in training deep neural networks. We will first discuss when this variation becomes too severe and why it will be a problem in Section 3.1. In Section 3.2, we will report our observation that the popular Adam optimizer performs worse than its variant AdamW. Then, in Section 3.3, we will show evidence suggesting understanding AdamW's advantage through its scale-freeness property which ensures that its updates are invariant to component-wise rescaling of the gradients thus adapting to gradient scales. Section 3.4 presents the surprising connection between AdamW and proximal updates, providing a potential explanation of where its scale-freeness property comes from. Finally, we will show another merit of scale-free algorithms in Section 3.5: they can "adapt" to the condition number in certain scenarios.

## 3.1   When Varying Gradient Scales Become a Problem

Neural networks are known to suffer from vanishing/exploding gradients (Bengio et al., 1994; Glorot and Bengio, 2010; Pascanu et al., 2013). This leads to the scales of gradient magnitudes being very different across layers, especially between

**(a)** First Convolution Layer

**(b)** Last Convolution Layer
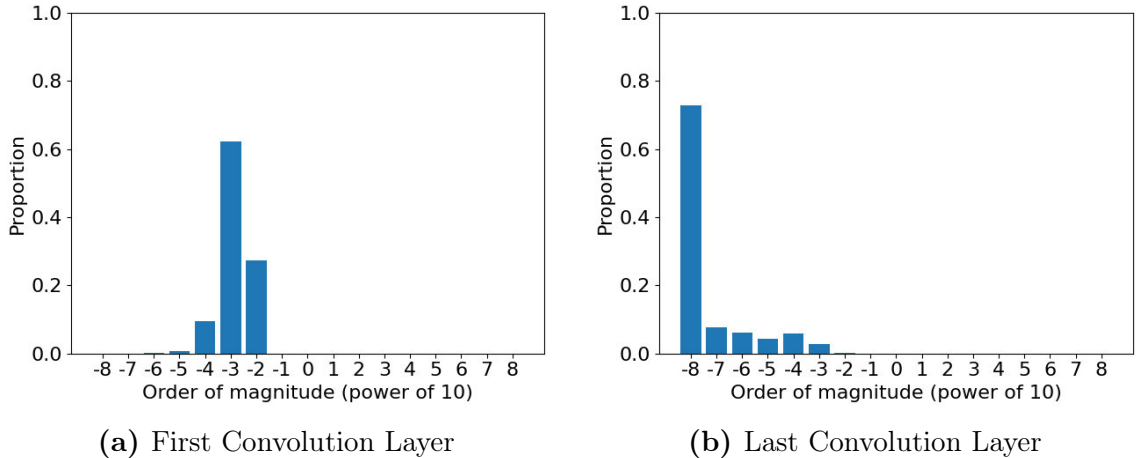
**Figure 3·1:** The histograms of the magnitudes of gradients of the first convolution layer vs. the last convolution layer in a time step during training a 110-layer Resnet with batch normalization disabled.

the first and the last layers. This problem is particularly severe when the model is not equipped with normalization mechanisms like Batch Normalization (BN) (Ioffe and Szegedy, 2015). As an example, Figure 3·1 shows the huge difference in gradient magnitude scales between the first and the last convolution layers in training a 110-layer Resnet (He et al., 2016) with BN disabled.

BN works by normalizing the input to each layer across the mini-batch to make each coordinate have zero-mean and unit-variance. While BN can greatly help reduce the variation in gradient scales between different layers, it comes with a price. For example, it introduces added memory overhead (Bulò et al., 2018) and training time (Gitman and Ginsburg, 2017) as well as a discrepancy between training and inferencing (Singh and Shrivastava, 2019). BN has also been found to be not suitable for many cases including distributed computing with a small minibatch per GPU (Wu and He, 2018; Goyal et al., 2017), sequential modeling tasks (Ba et al., 2016), and contrastive learning algorithms (Chen et al., 2020b). Actually, there is already active research in the setting of removing BN (De and Smith, 2020; Zhang et al., 2019a).

Also, there are already SOTA architectures that do not use BN including the BERT model (Devlin et al., 2019) and the Vision Transformer (Dosovitskiy et al., 2021).

In the scenario when the gradient scales vary significantly from layer to layer, the widely used SGD algorithm is not able to handle such a situation nicely. The reason is that SGD adopts a single step size value for all layers, and this value could be too large for some layers with large gradients leading to divergence while at the same time being too small for other layers with small gradients resulting in slow progress.

A natural idea is to use individual "step sizes" for each layer or even each coordinate and to make these layer-wise/coordinate-wise step sizes to take into account the gradient scales of corresponding layers or coordinates. A popular optimizer operating in such fashion is Adam (Kingma and Ba, 2015), a method that operates coordinate-wisely and utilizes first- and second-order moments of gradients to compute the step size. It has been empirically shown to achieve remarkable results across a variety of problems even by simply adopting the default hyperparameter setting. However, we observed that it does not entirely solve the problem; in the next section, we show that it performs worse than AdamW (Loshchilov and Hutter, 2019), a variant of Adam.

## 3.2   When Adam Performs Worse than AdamW

Since its debut, Adam has gained tremendous popularity due to less hyperparameter tuning and great performance. In practice, to improve the generalization ability, Adam is typically combined with a $\ell_2$ regularization which adds the squared $\ell_2$ norm of the model weights on top of the loss function (which we will call $Adam\text{-}\ell_2$ hereafter). This technique is usually referred to as weight decay because when using SGD, the $\ell_2$ regularization works by first shrinking the model weights by a constant factor in addition to moving along the negative gradient direction in each step. By biasing

---

**Algorithm 3.1** Adam with $\ell_2$ regularization (Adam-$\ell_2$) and Adam with decoupled weight decay (AdamW) *(All operations on vectors are element-wise.)*

---

1: **Given** $\alpha$, $\beta_1$, $\beta_2$, $\epsilon$, $\lambda \in \mathbb{R}$, $\{\eta_t\}_{t \geq 0}$.
2: **Initialize:** $\boldsymbol{x}_0 \in \mathbb{R}^d$, first moment vector $\boldsymbol{m}_0 = 0$, second moment vector $\boldsymbol{v}_0 = 0$
3: **for** $t = 1, 2, \ldots, T$ **do**
4:     Compute a stochastic evaluation of the true gradient $\nabla f(\boldsymbol{x}_{t-1})$ and denote it as $\nabla f_t(\boldsymbol{x}_{t-1})$
5:     $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{x}_{t-1}) + \lambda \boldsymbol{x}_{t-1}$
6:     $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t, \quad \boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2)\boldsymbol{g}_t^2$
7:     $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t/(1 - \beta_1^t), \quad \hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t/(1 - \beta_2^t)$
8:     $\boldsymbol{x}_t \leftarrow \boldsymbol{x}_{t-1} - \eta_t \lambda \boldsymbol{x}_{t-1} - \eta_t \alpha \hat{\boldsymbol{m}}_t/(\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon)$
9: **end for**

---

the optimization towards solutions with small norms, weight decay has long been a standard technique to improve the generalization ability in machine learning (Krogh and Hertz, 1992; Bos and Chug, 1996) and is still widely employed in training modern deep neural networks (Devlin et al., 2019; Tan and Le, 2019).

However, as pointed out in Loshchilov and Hutter (2019), for Adam, there is no fixed regularization that achieves the same effect $\ell_2$ regulation has on SGD. To address this, they provide a method called AdamW that decouples the gradient of the $\ell_2$ regularization from the update of Adam and directly decays the weights. These two algorithms are presented in Algorithm 3.1. Although AdamW is very popular (Kuen et al., 2019; Lifchitz et al., 2019; Carion et al., 2020) and it frequently outperforms Adam-$\ell_2$, it is currently unclear why it works so well. Worse still, recently, Bjorck et al. (2021) applied AdamW in Natural Language Processing and Reinforcement Learning problems and found no significant improvement over sufficiently tuned Adam-$\ell_2$.

Consequently, we conducted deep learning experiments to identify a scenario when AdamW exhibits concrete advantages over Adam-$\ell_2$ for isolating AdamW's unique merits. The search leads to the setting of training very deep neural networks with Batch Normalization disabled on image classification tasks as we reported below.

**Data Normalization and Augmentation:** We consider the image classification task on CIFAR-10/100 datasets. Images are normalized per channel using the means and standard deviations computed from all training images. We adopt the data augmentation technique following Lee et al. (2015) (for training only): 4 pixels are padded on each side of an image and a $32 \times 32$ crop is randomly sampled from the padded image or its horizontal flip.

**Models:** For both the CIFAR-10 and CIFAR-100 datasets, we employ the Residual Network model (He et al., 2016) of 20/44/56/110/218 layers; and for CIFAR-100, we additionally utilize the DenseNet-BC model (Huang et al., 2017) with 100 layers and a growth-rate of 12. The loss is the cross-entropy loss.

**Hyperparameter tuning:** For both Adam-$\ell_2$ and AdamW, we set $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ as suggested in the Adam paper (Kingma and Ba, 2015). To set the initial step size $\alpha$ and the weight decay parameter $\lambda$, we grid search over $\{0.00005, 0.0001, 0.0005, 0.001, 0.005\}$ for $\alpha$ and $\{0, 0.00001, 0.00005, 0.0001, 0.0005, 0.001\}$ for $\lambda$. Whenever the best performing hyperparameters lie in the boundary of the searching grid, we always extend the grid to ensure that the final best-performing hyperparameters fall into the interior of the grid.

**Training:** For each experiment configuration (e.g., 110-layer Resnet without BN on CIFAR-10), we randomly select an initialization of the model to use as a fixed starting point for all optimizers and hyperparameter settings. We use a mini-batch of 128, and train 300 epochs unless otherwise specified.

**With BN, Adam-$\ell_2$ is on par with AdamW** Recently, Bjorck et al. (2021) found that AdamW has no improvement in absolute performance over sufficiently tuned Adam-$\ell_2$ in some reinforcement learning experiments. We also discover the same phenomenon in several image classification tasks, see Figure 3·2. Indeed, the best weight decay parameter is 0 for all cases and AdamW coincides with Adam-$\ell_2$

**(a)** 20 Layer Resnet on CIFAR10

**(b)** 44 Layer Resnet on CIFAR10

**(c)** 56 Layer Resnet on CIFAR10

**(d)** 110 Layer Resnet on CIFAR10

**(e)** 218 Layer Resnet on CIFAR10

**(f)** 100 layer DenseNet-BC on CIFAR100
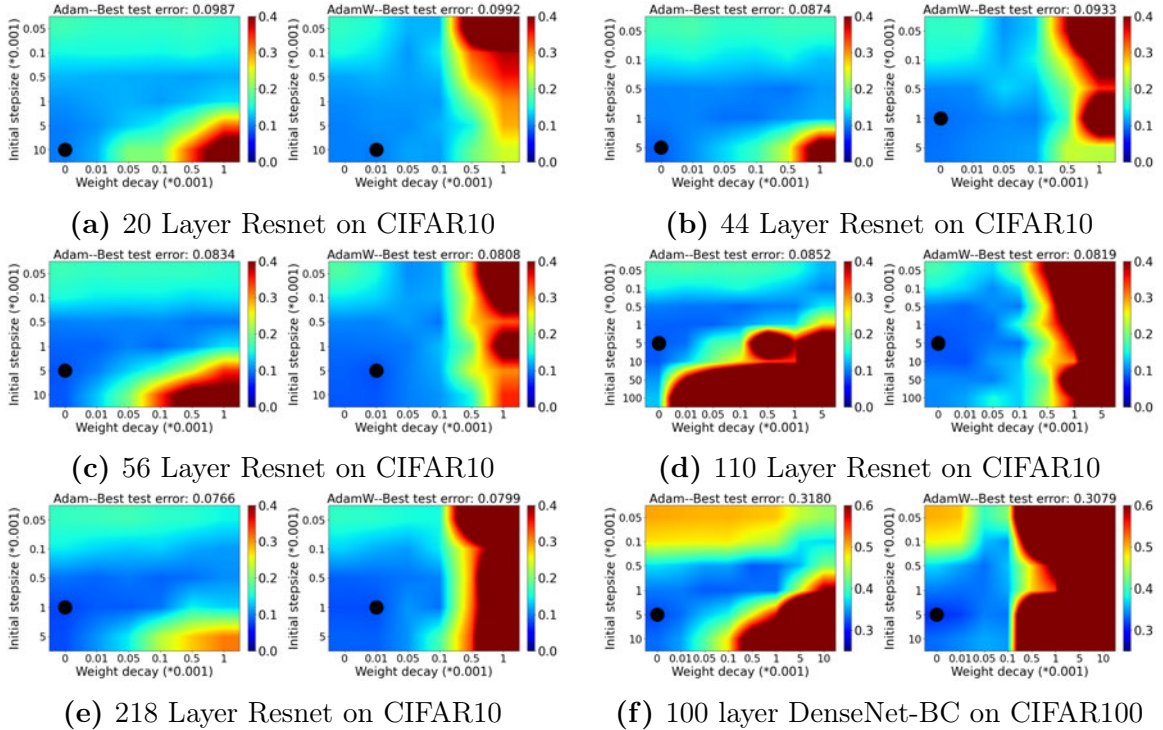
**Figure 3·2:** The final Top-1 test error on using AdamW vs. Adam-$\ell_2$ on training a Resnet/DenseNet with Batch Normalization on CIFAR10/100 (*the black circle denotes the best setting*). Note how close are the best performing hyperparameter combinations and the smallest testing error each optimizer obtains between Adam-$\ell_2$ and AdamW for each setting, suggesting they perform similarly when BN is enabled.

in these cases. Nevertheless, AdamW does decouple the optimal choice of the weight decay parameter from the initial step size much better than Adam-$\ell_2$ in all cases.

**Removing BN** Notice that the models used in Figure 3·2 all employ BN. Without BN, deep neural networks are known to suffer from gradient explosion and vanishing (Schoenholz et al., 2017). This means each coordinate of the gradient will have very different scales, especially between the first and the last layers. As we will detail in the next section, for Adam-$\ell_2$, the update to the network weights will be affected and each coordinate will proceed at a different pace, whereas AdamW is robust to such issues as the scaling of any single coordinate will not affect the update. Thus, we consider the case where BN is removed as that is where AdamW and Adam-$\ell_2$ will

**(a)** 20 Layer Resnet



**(b)** 44 Layer Resnet



**(c)** 56 Layer Resnet



**(d)** 110 Layer Resnet



**(e)** 218 Layer Resnet
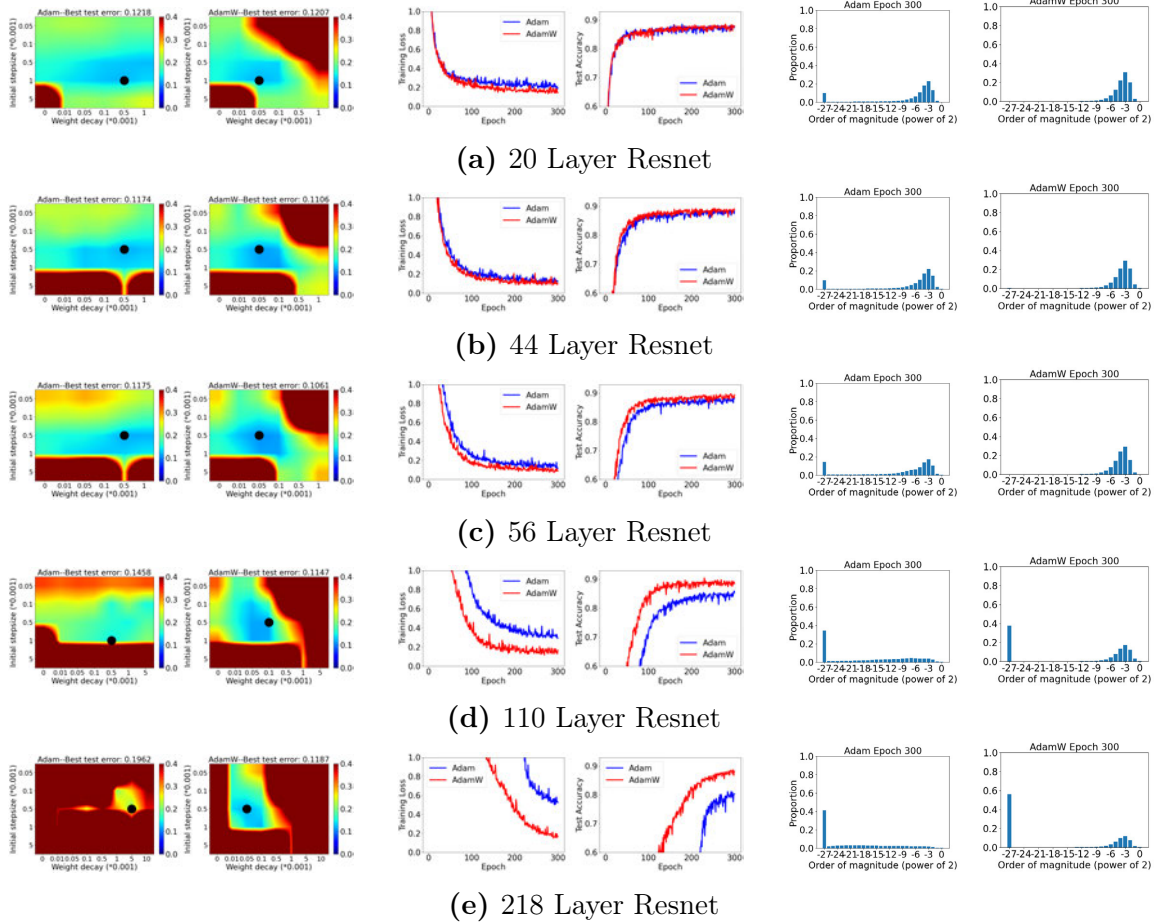
**Figure 3·3:** On using AdamW vs. Adam-$\ell_2$ to train a Resnet with Batch Normalizatin disabled on CIFAR10. (Left two) The final Top-1 test error (*the black circle denotes the best setting*). (Middle two) The training loss and test accuracy curves when using the initial step size and the weight decay parameter that gives the smallest test error. (Right two) The histogram of update magnitudes of all coordinates near the end of the training when using the initial step size and the weight decay parameter that gives the smallest test error. Note that as the depth of the neural network increases, Adam-$\ell_2$'s updates scatter more evenly over the entire spectrum while AdamW's updates are still concentrated in a small range, and AdamW's advantage in both training and testing over Adam-$\ell_2$ becomes more significant.

**(a)** 20 Layer Resnet



**(b)** 44 Layer Resnet



**(c)** 56 Layer Resnet



**(d)** 110 Layer Resnet



**(e)** 218 Layer Resnet



**(f)** 100 layer DenseNet-BC

**Figure 3·4:** On using AdamW vs. Adam-$\ell_2$ to train Resnet/DenseNet with Batch Normalization disabled on CIFAR100. (Left two) The final Top-1 test error (*the black circle denotes the best setting*). (Middle two) The training loss and test accuracy curves when using the initial step size and the weight decay parameter that gives the smallest test error. (Right two) The histogram of update magnitudes of all coordinates near the end of the training when using the initial step size and the weight decay parameter that gives the smallest test error. Note that as the depth of the neural network increases, Adam-$\ell_2$'s updates scatter more evenly over the entire spectrum while AdamW's updates are still concentrated in a small range, and AdamW's advantage in both training and testing over Adam-$\ell_2$ becomes more significant.

show very different patterns due to scale-freeness.

**Without BN, AdamW Outperforms Adam-$\ell_2$** In fact, without BN, AdamW outperforms Adam-$\ell_2$ even when both are finely tuned, especially on relatively deep neural networks (see Figure 3·3 and 3·4). AdamW not only obtains a much better test accuracy but also trains much faster. For example, Figure 3·3d shows that, when training a 110 layer ResNet (He et al., 2016) with Batch Normalization disabled to do image classification on the CIFAR10 dataset, even when both are finely tuned, AdamW gains a 3% improvement over Adam-$\ell_2$ in test errors as well as converging much faster during training.

In the next section, we propose to understand through the scale-freeness property why this different way of employing regularization leads to AdamW's advantage.

## 3.3 Understanding AdamW through its Scale-freeness

An optimization algorithm is said to be *scale-free* if its iterates do not change when one multiplies any coordinate of all the gradients by a positive constant (Orabona and Pál, 2015). The scale-free property was first proposed in the online learning field (Cesa-Bianchi et al., 2007; Orabona and Pál, 2015). There, they do not need to know a priori the Lipschitz constant of the functions, while still being able to obtain optimal convergence rates. We stress that the scale-freeness is an important but largely overlooked property of an optimization algorithm. It has already been utilized to explain the success of AdaGrad (Orabona and Pál, 2015). Recently, Agarwal et al. (2020) also provides theoretical and empirical support for setting the $\epsilon$ in the denominator of AdaGrad to be 0, thus making the update exactly scale-free.

It turns out that the update of AdamW is scale-free when $\epsilon = 0$. This is evident as the scaling factor for any coordinate of the gradient is kept in both $\hat{\boldsymbol{m}}_t$ and $\sqrt{\hat{\boldsymbol{v}}_t}$ and

will be canceled out when dividing them. In contrast, for Adam-$\ell_2$, the addition of the gradient of the $\ell_2$ regularization to the gradient (Line 5 of Algorithm 3.1) destroys this property.

We want to emphasize the comparison between Adam-$\ell_2$ and AdamW: once Adam-$\ell_2$ adopts a non-zero $\lambda$, it loses the scale-freeness property; in contrast, AdamW enjoys this property for arbitrary $\lambda$. The same applies to any AdaGrad-type and Adam-type algorithm that incorporates the squared $\ell_2$ regularizer by simply adding the gradient of the $\ell_2$ regularizer directly to the gradient of the loss function, as in Adam-$\ell_2$ which is implemented in Tensorflow and Pytorch. Such algorithms are scale-free only when they do not employ regularization.

Nevertheless, one may notice that in practice, the $\epsilon$ factor in the AdamW update is typically small but not 0, in our case $1e$-8, thus preventing it from being completely scale-free. Below, we verify that the effect of such an $\epsilon$ on the scale-freeness is negligible.

As a simple empirical verification of the scale-freeness, we consider the scenario where we multiply the loss function by a positive number. Note that any other method to test scale-freeness would be equally good. For a feed-forward neural network without BN, this means the gradient would also be scaled up by that factor. In this case, the updates of a scale-free optimization algorithm would remain exactly the same, whereas they would change for an optimization algorithm that is not scale-free.

Figure 3·5 shows the results of the loss function being multiplied by 10 and 100 respectively on optimizing a 110-layer Resnet with BN *disabled*. For results of the original loss see Figure 3·3d. We can see that AdamW has almost the same performance across the range of initial step sizes and weight decay parameters, and most importantly, the best values of these two hyperparameters remain the same. This verifies that, even when employing a (small) non-zero $\epsilon$, AdamW is still approximately
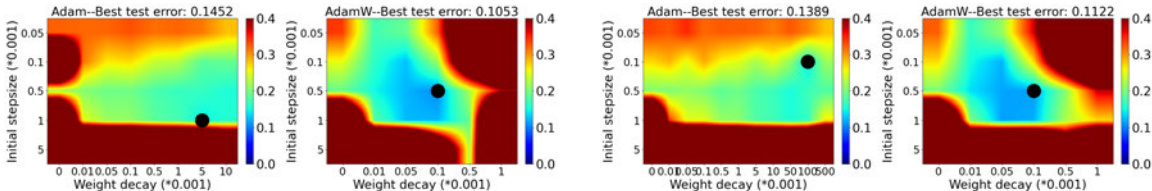
**(a)** Loss multiplied by 10        **(b)** Loss multiplied by 100

**Figure 3·5:** The final top-1 test error of AdamW vs. Adam-$\ell_2$ on optimizing a 110-layer Resnet with BN *removed* on CIFAR-10 with the loss function multiplied by 10 (left two figures) and 100 (right two figures). Note how the best performing hyperparameter combinations of AdamW remain the same for different loss multiplication factors as well as the shape of the heatmap being very similar. In contrast, Adam-$\ell_2$'s performance as well as the best performing hyperparameter combinations vary dramatically for different loss multiplication factors.

scale-free. In contrast, Adam-$\ell_2$ is not scale-free and we can see that its behavior varies drastically with the best initial step sizes and weight decay parameters in each setting totally different.

With that said, our main claim is *the lack of scale-freeness seems to harm Adam-$\ell_2$'s performance in certain scenarios in deep learning, while AdamW preserves the scale-freeness even with non-zero regularization.*

This is exactly verified empirically as illustrated in the 5th & 6th columns of figures in Figure 3·3 and 3·4. There, we report the histograms of the absolute value of updates of Adam-$\ell_2$ vs. AdamW of all coordinates near the end of training (for their comparison over the whole training process please refer to the Appendix A.2).

Indeed, the optimization processes of these two optimizers show the effects of with or without the scale-freeness. As can be seen, the magnitudes of AdamW's updates are much more concentrated than that of Adam-$\ell_2$ throughout the training. This means that a scale-free algorithm like AdamW ensures that each layer is updated at a similar pace; in contrast, for a non-scale-free optimization algorithm like Adam-$\ell_2$, different layers will proceed at very different speeds.

We also observe that the advantage of AdamW becomes more evident as the network becomes deeper. Recall that as the depth grows, without BN, the gradient explosion and vanishing problem becomes more severe. This means that for the non-scale-free Adam-$\ell_2$, the updates of each coordinate will be dispersed on a wider range of scales even when the same weight decay parameter is employed. In contrast, the scales of the updates of AdamW will be much more concentrated in a smaller range.

This correlation between the advantage of AdamW over Adam-$\ell_2$ and the different spread of update scales which is induced by the scale-freeness property of AdamW provides empirical evidence on when AdamW excels over Adam-$\ell_2$.

As a side note, the reader might wonder why SGD is known to provide state-of-the-art performance on many deep learning architectures (e.g., He et al., 2016; Huang et al., 2017) *without* being scale-free. At first blush, this seems to contradict our claims that scale-freeness correlates with good performance. In reality, the good performance of SGD in very deep models is linked to the use of BN that normalizes the gradients. Indeed, we verified empirically that SGD fails spectacularly when BN is not used. For example, on training the 110 layer Resnet without BN using SGD with momentum and weight decay of 0.0001, even a step size of $1e - 10$ will lead to divergence.

## 3.4 AdamW and Proximal Updates

The scale-freeness property of AdamW may seem a natural consequence of the way it constructs its update. Yet, in this section, we reveal the surprising connection between AdamW and proximal updates (Parikh and Boyd, 2014), suggesting another potential explanation of where AdamW's scale-freeness comes from.

A proximal algorithm is an algorithm for solving a convex optimization problem

that uses the proximal operators of the objective function. The *proximal operator* $\text{prox}_h : \mathbb{R}^d \to \mathbb{R}^d$ of a convex function $h$ is defined for any $\boldsymbol{y} \in \mathbb{R}^d$ as $\text{prox}_h(\boldsymbol{y}) = \arg\min_{\boldsymbol{x} \in \mathbb{R}^d}(h(\boldsymbol{x}) + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2)$. The use of proximal updates in the batch optimization literature dates back at least to 1965 (Moreau, 1965; Martinet, 1970; Rockafellar, 1976; Parikh and Boyd, 2014) and they are used more recently even in the stochastic setting (Toulis and Airoldi, 2017; Asi and Duchi, 2019).

Now consider that we want to minimize the objective function

$$F(\boldsymbol{x}) = \tfrac{\lambda}{2}\|\boldsymbol{x}\|_2^2 + f(\boldsymbol{x}), \tag{3.1}$$

where $\lambda > 0$ and $f(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$ is a function bounded from below. We could use a stochastic optimization algorithm that updates in the following fashion

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} - \eta_t \boldsymbol{p}_t, \tag{3.2}$$

where $\eta_t$ is a learning rate schedule, e.g., the constant one or the cosine annealing (Loshchilov and Hutter, 2017) and $\boldsymbol{p}_t$ denotes any update direction. This update covers many cases, where $\alpha$ denotes the initial step size:

1. $\boldsymbol{p}_t = \alpha \boldsymbol{g}_t$ gives us the vanilla SGD;

2. $\boldsymbol{p}_t = \alpha \boldsymbol{g}_t / (\sqrt{\sum_{i=1}^t \boldsymbol{g}_i^2 + \epsilon})$ gives the AdaGrad algorithm (Duchi et al., 2010a);

3. $\boldsymbol{p}_t = \alpha \hat{\boldsymbol{m}}_t / (\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon)$ recovers Adam (Kingma and Ba, 2015), where $\hat{\boldsymbol{m}}_t$ denotes the bias corrected first moment of past gradients and $\hat{\boldsymbol{v}}_t$ denotes the bias corrected second moment of past gradients as updated in Line 6-7 in Algorithm 3.1.

Note that in the above we use $\boldsymbol{g}_t$ to denote the stochastic gradient of the entire objective function: $\boldsymbol{g}_t = \nabla f_t(\boldsymbol{x}_{t-1}) + \lambda \boldsymbol{x}_{t-1}$ ($\lambda = 0$ if the regularizer is not present), where $\nabla f_t(\boldsymbol{x}_{t-1})$ is a stochastic evaluation of the true gradient $\nabla f(\boldsymbol{x}_{t-1})$.

This update rule (3.2) is given by the following online mirror descent update (Nemirovsky and Yudin, 1983; Warmuth and Jagota, 1997; Beck and Teboulle, 2003):

$$\boldsymbol{x}_t = \underset{\boldsymbol{x} \in \mathbb{R}^d}{\operatorname{argmin}} \ \frac{\lambda}{2}\|\boldsymbol{x}_{t-1}\|_2^2 + f(\boldsymbol{x}_{t-1}) + \boldsymbol{p}_t^\top(\boldsymbol{x} - \boldsymbol{x}_{t-1}) + \frac{1}{2\eta_t}\|\boldsymbol{x} - \boldsymbol{x}_{t-1}\|_2^2 \ . \tag{3.3}$$

This approximates minimizing a first-order Taylor approximation of $F$ centered in $\boldsymbol{x}_{t-1}$ plus a term that measures the distance between the $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t-1}$ according to the $\ell_2$ norm. The approximation becomes exact when $\boldsymbol{p}_t = \nabla f(\boldsymbol{x}_{t-1}) + \lambda \boldsymbol{x}_{t-1}$.

Yet, this is not the only way to construct first-order updates for the objective (3.1). An alternative route is to linearize only $f$ and to keep the squared $\ell_2$ norm in its functional form:

$$\begin{aligned}
\boldsymbol{x}_t &= \underset{\boldsymbol{x} \in \mathbb{R}^d}{\operatorname{argmin}} \ \frac{\lambda}{2}\|\boldsymbol{x}\|_2^2 + f(\boldsymbol{x}_{t-1}) + \boldsymbol{p}_t^\top(\boldsymbol{x} - \boldsymbol{x}_{t-1}) + \frac{1}{2\eta_t}\|\boldsymbol{x} - \boldsymbol{x}_{t-1}\|_2^2 \\
&= \operatorname{prox}_{\frac{\lambda\eta_t}{2}\|\cdot\|_2^2}(\boldsymbol{x}_{t-1} - \eta_t \boldsymbol{p}_t),
\end{aligned} \tag{3.4}$$

which uses the proximal operator of the convex function $\frac{\lambda\eta_t}{2}\|\cdot\|_2^2$.

It is intuitive why this would be a better update: *We directly minimize the squared $\ell_2$ norm instead of approximating it.* We also would like to note that, similar to (3.3), the proximal updates of (3.4) can be shown to minimize the objective $F$ under appropriate conditions. However, we do not include the convergence analysis of (3.4) as this is already well-studied in the literature. For example, when $\boldsymbol{p}_t = \nabla f(\boldsymbol{x}_{t-1})$ in (3.4) and $f$ is convex and smooth, the update becomes a version of the (non-accelerated) iterative shrinkage-thresholding algorithm. This algorithm guarantees $F(\boldsymbol{x}_t) - F^* \leq O(1/t)$, which is in the same order as obtained by gradient descent on minimizing $f$ alone (Beck and Teboulle, 2009).

From the first-order optimality condition, the update is

$$\boldsymbol{x}_t = (1 + \lambda\eta_t)^{-1}(\boldsymbol{x}_{t-1} - \eta_t\boldsymbol{p}_t) \; . \tag{3.5}$$

When $\lambda = 0$, the update in (3.2) and this one coincide. Yet, when $\lambda \neq 0$, they are no longer the same.

We now show how the update in (3.5) generalizes the one in AdamW. The update of AdamW is

$$\boldsymbol{x}_t = (1 - \lambda\eta_t)\boldsymbol{x}_{t-1} - \eta_t\alpha\hat{\boldsymbol{m}}_t/(\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon) \; . \tag{3.6}$$

On the other hand, using $\boldsymbol{p}_t = \alpha\hat{\boldsymbol{m}}_t/(\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon)$ in (3.5) gives:

$$\boldsymbol{x}_t = (1 + \lambda\eta_t)^{-1}(\boldsymbol{x}_{t-1} - \eta_t\alpha\hat{\boldsymbol{m}}_t/(\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon)), \tag{3.7}$$

which we will call *AdamProx* hereafter. Its first-order Taylor approximation around $\eta_t = 0$ is

$$\boldsymbol{x}_t \approx (1 - \lambda\eta_t)\boldsymbol{x}_{t-1} - \eta_t\alpha\hat{\boldsymbol{m}}_t/(\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon),$$

exactly the AdamW update (3.6). Hence, AdamW is a first-order approximation of a proximal update.

The careful reader might notice that the approximation from AdamW to the AdamProx update in (3.7) becomes less accurate when $\eta_t$ becomes too large, and so be concerned about whether this approximation is practical at all. Fortunately, in practice, $\eta_t$ is never large enough for this to be an issue. The remainder term of this approximation is $O(\lambda\eta_t^2)$ which we should always expect to be small as both $\lambda$ and $\eta_t$ are small. So, we can expect AdamW and AdamProx to perform similarly for step size schedules $\eta_t$ commonly employed in practice.

**(a)** ResNet on CIFAR-10          **(b)** DenseNet-BC on CIFAR-100
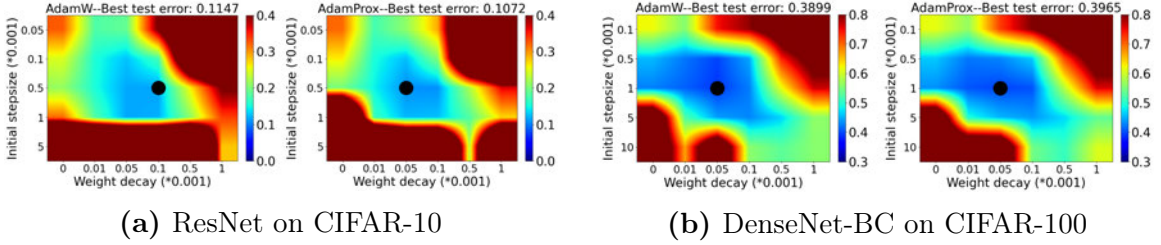
**Figure 3·6:** The final Top-1 test error of using AdamW vs. AdamProx on training (*the black circle denotes the best setting*). (Top row) a 110-layer ResNet with BN *removed* on CIFAR-10 (trained for 300 epochs). (Bottom row) a 100-layer DenseNet-BC with BN *removed* on CIFAR-100 (trained for 100 epochs). Note how similar are the shapes of the heatmaps, the best performing hyperparameter combinations, and the test errors between AdamW and AdamProx.

Indeed, we verified empirically that the approximation is good as reported in Figure 3·6, where we consider the case when $\eta_t = 1$ for all $t$, a relatively large constant step size schedule. In such cases, AdamW and AdamProx still behave very similarly. This suggests that for most step size schedules, e.g., cosine, exponential, polynomial, and step decay, which all monotonously decrease from $\eta_0 = 1$, AdamProx will remain a very good approximation to AdamW. Thus, it is reasonable to use the more classically-linked AdamProx to try to understand AdamW.

Let's now derive the consequences of this connection with proximal updates.

First of all, at least in the convex case, the convergence rate of the proximal updates will depend on $\|\nabla f(\boldsymbol{x}_t)\|_2^2$ rather than on $\|\nabla f(\boldsymbol{x}_t) + \lambda \boldsymbol{x}_t\|_2^2$ (Duchi et al., 2010b). This could be a significant improvement: the regularized loss function is never Lipschitz, so the regularized gradients $\nabla f(\boldsymbol{x}_t) + \lambda \boldsymbol{x}_t$ could be much larger than $\nabla f(\boldsymbol{x}_t)$ when $f$ itself is Lipschitz.

Also, as we wrote above, AdamW can be seen as the first-order Taylor approximation on $\eta_t = 0$ of the AdamProx update in (3.7); in turn, the scale-freeness of AdamProx directly comes from the proximal updates. Of course, there may be other ways to design scale-free updates solving (3.1); yet, for AdamW, its scale-free property

derives directly from the proximal update.

More importantly, proximal updates are fundamentally better at keeping the weights small. Let us consider a couple of simple examples to see how this could be. First, suppose the weights are *already zero*. Then, when taking an update according to (3.2), we increase the weights to $-\eta_t \boldsymbol{p}_t$. In contrast, update (3.5) clearly leads to a smaller value. This is because it computes an update using the regularizer rather than its gradient. As an even more disturbing, yet actually more realistic example, consider the case that $\boldsymbol{x}_{t-1}$ is non-zero, but $\boldsymbol{g}_t = \boldsymbol{0}$. In this case, taking an update using (3.2) may actually *increase* the weights by causing $\boldsymbol{x}_t$ to *overshoot* the origin. In contrast, the proximal update will never demonstrate such pathological behavior. Notice that this pathological behavior of (3.2) can be mitigated by properly tuning the step size. However, one of the main attractions of adaptive optimizers is that we should not need to tune the step size as much. Thus, *the proximal update can be viewed as augmenting the adaptive methods with an even greater degree of learning-rate robustness.*

## 3.5 Scale-free Algorithms can Adapt to the Condition Number

Interestingly, scale-freeness comes with another benefit: it can effectively reduce the effects of the condition number in certain scenarios, as detailed below.

For a twice continuously differentiable function $F$, its Hessian matrix is symmetric and its *condition number* $\kappa$ is defined as the ratio of its largest absolute value eigenvalue to its smallest one. It is well-known that the best convergence rate when minimizing such $F$ using a first-order optimization algorithm (e.g., gradient descent)

must depend on the condition number (Theorem 2.1.13, Nesterov, 2004), formally,

$$\|\boldsymbol{x}_t - \boldsymbol{x}^*\|_2^2 \geq \left( \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^{2t} \|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2 .$$

In particular, a problem with a small $\kappa$ can be solved more efficiently than one with a big $\kappa$.

One way to reduce the effect of the condition number is to use a *precondi-tioner* (Nocedal and Wright, 2006). While originally designed for solving systems of linear equations, preconditioning can be extended to the optimization of non-linear functions and it should depend on the Hessian of the function (Boyd and Vandenberghe, 2004; Li, 2018). However, it is unclear how to set the preconditioner given that the Hessian might not be constant (Section 9.4.4 Boyd and Vandenberghe, 2004) and in stochastic optimization the Hessian cannot be easily estimated (Li, 2018).

In the following theorem, we show that scale-freeness gives similar advantages to the use of an optimal diagonal preconditioner, *for free.* Specifically, a scale-free algorithm can automatically transform solving the original problem into solving a problem with a potentially much smaller condition number and thus could provide substantial improvements over non-scale-free ones.

**Theorem 3.1.** *Let $F$ be a twice continuously differentiable function and $\boldsymbol{x}^*$ such that $\nabla F(\boldsymbol{x}^*) = \boldsymbol{0}$. Next, let $\tilde{F}_\Lambda$ be the family of functions such that $\nabla \tilde{F}_\Lambda(\boldsymbol{x}^*) = \boldsymbol{0}$, and $\nabla^2 \tilde{F}_\Lambda(\boldsymbol{x}) = \Lambda \nabla^2 F(\boldsymbol{x})$, where $\Lambda = diag(\lambda_1, \ldots, \lambda_d) \succeq 0$. Then, running any scale-free optimization algorithm on $F$ and $\tilde{F}_\Lambda$ will result exactly in the same iterates, assuming the same noise on the gradients. Moreover, any dependency on the condition number of the scale-free algorithm will be reduced to the smallest condition number among all the functions $\tilde{F}_\Lambda$.*

*Proof of Theorem 3.1.* From the Fundamental Theorem of Calculus we have:

$$\nabla F(\boldsymbol{x}) = \nabla F(\boldsymbol{x}^*) + \int_0^1 \nabla^2 F(\boldsymbol{x}^* + t(\boldsymbol{x} - \boldsymbol{x}^*))(\boldsymbol{x} - \boldsymbol{x}^*) dt$$

$$= \int_0^1 \nabla^2 F(\boldsymbol{x}^* + t(\boldsymbol{x} - \boldsymbol{x}^*))(\boldsymbol{x} - \boldsymbol{x}^*) dt \ .$$

Thus, for any function $\tilde{F}_\Lambda(\boldsymbol{x})$ whose Hessian is $\Lambda \nabla^2 F(\boldsymbol{x})$ and $\nabla \tilde{F}_\Lambda(\boldsymbol{x}^*) = 0$, we have $\nabla \tilde{F}_\Lambda(\boldsymbol{x}) = \Lambda \nabla F(\boldsymbol{x})$.

Now, from the definition of a scale-free algorithm, the iterates of such an algorithm do not change when one multiplies each coordinate of all the gradients by a positive constant. Thus, a scale-free algorithm optimizing $F$ behaves the same as if it is optimizing $\tilde{F}_\Lambda$. □

To give an example of when this is advantageous, consider when $\nabla^2 F(\boldsymbol{x})$ is a diagonal matrix:

$$\nabla^2 F(\boldsymbol{x}) = diag(g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \dots, g_d(\boldsymbol{x})) \ .$$

Assume $0 < \mu \le \mu_i \le g_i(\boldsymbol{x}) \le L_i \le L$ for $i \in \{1, \dots, d\}$. Denote $j = \arg\max_i L_i/\mu_i$. Choose $\lambda_i$ s.t. $\mu_j \le \lambda_i \mu_i \le \lambda_i g_i(\boldsymbol{x}) \le \lambda_i L_i \le L_j$ then $\Lambda \nabla^2 F(\boldsymbol{x})$ has a condition number $\kappa' = L_j/\mu_j$. This gives scale-free algorithms a big advantage when $\max_i L_i/\mu_i \ll L/\mu$.

Another example is one of the quadratic functions.

**Corollary 3.2.** *For quadratic problems $F(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top H \boldsymbol{x} + \boldsymbol{b}^\top \boldsymbol{x} + c$, with $H$ diagonal and positive definite, any scale-free algorithm will not differentiate between minimizing $f$ and $\tilde{F}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{x} + (H^{-1}\boldsymbol{b})^\top \boldsymbol{x} + c$. As the condition number of $\tilde{F}$ is 1, the operation, and most importantly, the convergence, of a scale-free algorithm will not be affected by the condition number of $F$ at all.*

**(a)** Condition number 1.

**(b)** Condition number 100000.
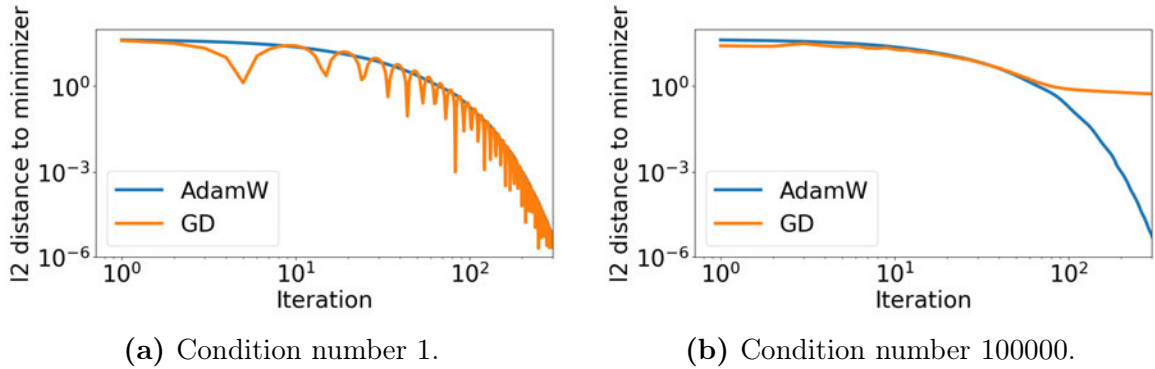
**Figure 3·7:** Non-scale-free GD v.s. scale-free AdamW on optimizing quadratic functions with different condition numbers.

Figure 3·7 illustrates Corollary 3.2: we compare GD (non-scale-free) with AdamW (scale-free) on optimizing two quadratic functions with the same minimizer, but one's Hessian matrix is a rescaled version of the other's, resulting in different condition numbers. The figure clearly shows that, even after tuning the step sizes, the updates of AdamW (starting from the same point) and thus its convergence to the minimizer, are completely unaffected by the condition number, while GD's updates change drastically and its performance deteriorates significantly when the condition number is large. It is not hard to imagine that such poor training performance would likely also lead to poor testing performance.

This can also explain AdaGrad's improvements over SGD in certain scenarios. Note that the folklore justification for such improvements is that the step size of Ada-Grad approximates the inverse of the Hessian matrix, but this is incorrect: AdaGrad does not compute Hessians and there is no reason to believe it approximates them in general. As an additional example, we analyze below a variant of AdaGrad with restarts (Algorithm 3.3) and show its convergence rate guarantee on strongly convex functions (ones satisfying (1.4)) which exhibits a dependency on the condition number $\kappa$ and thus can benefit from its scale-freeness.

---

**Algorithm 3.2** AdaGrad (Duchi et al., 2010a; McMahan and Streeter, 2010) *(All operations on vectors are element-wise.)*

---

**Input**: Number of iterations $T$, a set $\mathcal{K}$, $\boldsymbol{x}_1 \in \mathcal{K}$, step size $\eta$

**for** $t = 1 \ldots T$ **do**

    Receive: $\nabla F(\boldsymbol{x}_t)$

    Set: $\boldsymbol{\eta}_t = \frac{\eta}{\sqrt{\sum_{i=1}^{t}(\nabla F(\boldsymbol{x}_i))^2}}$

    Update: $\boldsymbol{x}_{t+1} = \Pi_{\mathcal{K}}\left(\boldsymbol{x}_t - \boldsymbol{\eta}_t \nabla F(\boldsymbol{x}_t)\right)$ where $\Pi_{\mathcal{K}}$ is the projection onto $\mathcal{K}$

**end for**

Output: $\bar{\boldsymbol{x}} = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t$

---

**Algorithm 3.3** AdaGrad with Restart

---

**Input**: Number of rounds $N$, $\boldsymbol{x}_0 \in \mathbb{R}^d$, upper bound on $\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_\infty$ as $D_\infty$, strong convexity $\mu$, smoothness $L$

Set: $\bar{\boldsymbol{x}}_0 = \boldsymbol{x}_0$

**for** $i = 1 \ldots N$ **do**

    Run Algorithm 3.2 to get $\bar{\boldsymbol{x}}_i$ with $T = 32d\frac{L}{\mu}$, $\boldsymbol{x}_1 = \bar{\boldsymbol{x}}_{i-1}$, $\mathcal{K} = \{\boldsymbol{x} : \|\boldsymbol{x} - \bar{\boldsymbol{x}}_{i-1}\|_\infty^2 \leq \frac{D_\infty^2}{4^{i-1}}\}$, $\eta = \frac{D_\infty/\sqrt{2}}{2^{i-1}}$

**end for**

Output: $\bar{\boldsymbol{x}}_N$

---

**Theorem 3.3.** *Let $\mathcal{K}$ be a hypercube with $\|\boldsymbol{x} - \boldsymbol{y}\|_\infty \leq D_\infty$ for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{K}$. For a convex function $F$, set $\eta = \frac{D_\infty}{\sqrt{2}}$, then Algorithm 3.2 guarantees for any $\boldsymbol{x} \in \mathcal{K}$:*

$$\sum_{t=1}^{T} F(\boldsymbol{x}_t) - F(\boldsymbol{x}) \leq \sqrt{2dD_\infty^2 \sum_{t=1}^{T}\|\nabla F(\boldsymbol{x}_t)\|^2} \ . \tag{3.8}$$

*Proof of Theorem 3.3.*

$$\sum_{t=1}^{T} F(\boldsymbol{x}_t) - F(\boldsymbol{x})$$

$$\leq \sum_{t=1}^{T} \langle \nabla F(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{x} \rangle$$

$$= \sum_{t=1}^{T} \sum_{j=1}^{d} \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \times (x_{t,j} - x_j)$$

$$= \sum_{t=1}^{T} \sum_{j=1}^{d} \frac{(x_{t,j} - x_j)^2 - \left(x_{t,j} - \eta_{t,j}\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) - x_j\right)^2}{2\eta_{t,j}} + \sum_{t=1}^{T} \sum_{j=1}^{d} \frac{\eta_{t,j}}{2}\left(\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right)^2$$

$$\leq \sum_{t=1}^{T} \sum_{j=1}^{d} \frac{(x_{t,j} - x_j)^2 - (x_{t+1,j} - x_j)^2}{2\eta_{t,j}} + \sum_{t=1}^{T} \sum_{j=1}^{d} \frac{\eta_{t,j}}{2}\left(\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right)^2$$

$$\leq \sum_{j=1}^{d} \sum_{t=1}^{T} \frac{(x_{t,j} - x_j)^2}{2}\left(\frac{1}{\eta_{t,j}} - \frac{1}{\eta_{t-1,j}}\right) + \sum_{j=1}^{d} \sum_{t=1}^{T} \frac{\eta_{t,j}}{2}\left(\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right)^2$$

$$\leq \frac{D_\infty^2}{2\eta} \sum_{j=1}^{d} \sum_{t=1}^{T} \left(\sqrt{\sum_{i=1}^{t}\left(\frac{\partial F}{\partial x_j}(\boldsymbol{x}_i)\right)^2} - \sqrt{\sum_{i=1}^{t-1}\left(\frac{\partial F}{\partial x_j}(\boldsymbol{x}_i)\right)^2}\right)$$

$$+ \sum_{j=1}^{d} \sum_{t=1}^{T} \frac{\eta}{2\sqrt{\sum_{i=1}^{t}\left(\frac{\partial F}{\partial x_j}(\boldsymbol{x}_i)\right)^2}}\left(\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right)^2$$

$$\leq \sum_{j=1}^{d} \left(\frac{D_\infty^2}{2\eta}\sqrt{\sum_{t=1}^{T}\left(\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right)^2} + \eta\sqrt{\sum_{t=1}^{T}\left(\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right)^2}\right)$$

$$= \sum_{j=1}^{d} \sqrt{2D_\infty^2 \sum_{t=1}^{T}\left(\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right)^2}$$

$$\leq \sqrt{2dD_\infty^2 \sum_{t=1}^{T} \sum_{j=1}^{d}\left(\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right)^2}$$

$$= \sqrt{2dD_\infty^2 \sum_{t=1}^{T} \|\nabla F(\boldsymbol{x}_t))\|^2},$$

where the first inequality is by convexity, the second one by the projection lemma as the projection onto a hypercube equals performing the projection independently for each coordinate, the fifth one by Lemma 5 in (McMahan and Streeter, 2010), and the last one by the concavity of $\sqrt{\cdot}$. $\qquad\square$

**Theorem 3.4.** *For a $\mu$ strongly convex and $L$ smooth function $F$, denote its unique minimizer as $\boldsymbol{x}^* \in \mathbb{R}^d$. Given $\boldsymbol{x}_0 \in \mathbb{R}^d$, assume that $\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_\infty \leq D_\infty$, then*

*Algorithm 3.3 guarantees*

$$\|\bar{\boldsymbol{x}}_N - \boldsymbol{x}^*\|_\infty^2 \leq \frac{D_\infty^2}{4^N} \ .$$

*Thus, to get a $\boldsymbol{x}$ s.t. $\|\boldsymbol{x} - \boldsymbol{x}^*\|_\infty^2 \leq \epsilon$, we need at most $32d\frac{L}{\mu}\log_4\left(D_\infty^2/\epsilon\right)$ gradient calls.*

*Proof of Theorem 3.4.* Consider round $i$ and assume $\mathcal{K}$ passed to Algorithm 3.2 is bounded w.r.t. $\ell_\infty$ norm by $D_{\infty_i}$. When $F$ is $\mu$ strongly convex and $L$ smooth, let $\boldsymbol{x} = \boldsymbol{x}^*$, then (3.8) becomes

$$\sum_{t=1}^T F(\boldsymbol{x}_t) - F(\boldsymbol{x}^*) \leq \sqrt{2dD_{\infty_i}^2 \sum_{t=1}^T \|\nabla F(\boldsymbol{x}_t)\|^2} \leq \sqrt{4LdD_{\infty_i}^2 \sum_{t=1}^T (F(\boldsymbol{x}_t) - F(\boldsymbol{x}^*))},$$

where the second inequality is by the $L$ smoothness of $F$. This gives

$$\sum_{t=1}^T F(\boldsymbol{x}_t) - F(\boldsymbol{x}^*) \leq 4LdD_{\infty_i}^2 \ .$$

Let $\bar{\boldsymbol{x}}_i = \frac{1}{T}\sum_{t=1}^T \boldsymbol{x}_t$ we have by the $\mu$-strong-convexity that

$$\|\bar{\boldsymbol{x}}_i - \boldsymbol{x}^*\|_\infty^2 \leq \|\bar{\boldsymbol{x}}_i - \boldsymbol{x}^*\|^2 \leq \frac{2}{\mu}(F(\bar{\boldsymbol{x}}) - F(\boldsymbol{x}^*)) \leq \frac{2}{\mu}\frac{1}{T}\sum_{t=1}^T (F(\boldsymbol{x}_t) - F(\boldsymbol{x}^*)) \leq \frac{8LdD_{\infty_i}^2}{\mu T} \ .$$

Put $T = 32d\frac{L}{\mu}$ in the above inequality, we have that $\|\bar{\boldsymbol{x}}_i - \boldsymbol{x}^*\|_\infty^2 \leq \frac{D_{\infty_i}^2}{4}$. Thus, after each round, the $\ell_\infty$ distance between the update $\bar{\boldsymbol{x}}_i$ and $\boldsymbol{x}^*$ is shrinked by half, which in turn ensures that $\boldsymbol{x}^*$ is still inside the $\mathcal{K}$ passed to Algorithm 3.2 in the next round with $D_{\infty_{i+1}} = \frac{D_{\infty_i}}{2}$. This concludes the proof. $\square$

## 3.6   Conclusion

In this chapter, we discussed the problem of gradient scales varying too much across layers during training deep neural networks without effective normalization tech-

niques. After reporting Adam-$\ell_2$'s inferior performance compared with AdamW on a scenario we identified, we correlated this with the scale-freeness property AdamW enjoys while Adam-$\ell_2$ does not with theoretical and empirical evidence. We then revealed the connection between AdamW and proximal updates for understanding where the scale-freeness of AdamW comes from. Finally, we introduced a theoretical merit of scale-free algorithms, namely that they can reduce the effects of the condition number of the objective function on convergence rates in certain cases.

# Chapter 4

# Adaptation to Unbounded Smoothness

[The results in this chapter appeared in Crawshaw et al. (2022).]

Traditional analyses in non-convex optimization typically rely on the smoothness assumption, namely requiring the gradients to be Lipschitz. However, recent evidence shows that this smoothness condition does not capture the properties of some deep learning objective functions, including the ones involving Recurrent Neural Networks and LSTMs. Instead, they satisfy a much more relaxed condition, with potentially unbounded smoothness. Under this relaxed assumption, it has been theoretically and empirically shown that the gradient-clipped SGD has an advantage over the vanilla one. In this paper, we show that clipping is not indispensable for Adam-type algorithms in tackling such scenarios: we theoretically prove that a generalized SignSGD algorithm can obtain similar convergence rates as SGD with clipping but does not need explicit clipping at all. This family of algorithms on one end recovers SignSGD and on the other end closely resembles the popular Adam algorithm. Our analysis underlines the critical role that momentum plays in analyzing SignSGD-type and Adam-type algorithms: it not only reduces the effects of noise, thus removing the need for large mini-batch in previous analyses of SignSGD-type algorithms, but it also substantially reduces the effects of unbounded smoothness and gradient norms. To the best of our knowledge, this work is the first one showing the benefit of Adam-type algorithms compared with non-adaptive gradient algorithms such as gradient

descent in the unbounded smoothness setting. We also compare these algorithms with popular optimizers on a set of deep learning tasks, observing that we can match the performance of Adam while beating others.

The layout of this chapter is as follows: we first discuss why we study a relaxed smoothness condition and report empirical evidence showing Transformers observe such condition in Section 4.1. Then, in Section 4.2 we show that refining the relaxed smoothness condition to the coordinate-wise granularity is necessary. Next, we introduce a generalized SignSGD algorithm to handle this scenario in Section 4.3, with a detailed discussion on the convergence rates it guarantees and the role of momentum. The proofs are contained in Section 4.4. We then report the experimental results in Section 4.5, comparing our algorithm with some popular competitors in deep learning tasks. Finally, we draw some conclusions and discuss the limitations of our work in Section 4.6.

## 4.1 A Relaxed Smoothness Assumption and Transformers

In previous chapters, we consider objective functions that are smooth, namely with Lipschitz gradients. The smoothness assumption is ubiquitous in analyses of optimization algorithms in the non-convex setting and has enabled many important theoretical results. For example, as we already discussed in Section 2.1, Ghadimi and Lan (2013) showed that, for a smooth function, Stochastic Gradient Descent can converge, in expectation, to a stationary point with $\|\nabla F(\boldsymbol{x})\| \leq \epsilon$ in the order of $\epsilon^{-4}$, which was later proven to be optimal (Arjevani et al., 2022).

Nevertheless, this assumption might be too restrictive. For example, simple polynomials such as $F(x) = x^4$ are not smooth unless the domain is bounded. Moreover, Zhang et al. (2020b) found that the smoothness assumption is not a good char-

acterization of the landscapes for objective functions in training deep neural networks including LSTMs (Hochreiter and Schmidhuber, 1997). In those scenarios, the gradient does not vary uniformly over the loss landscape and the local gradient Lipschitz constant can cross multiple orders of magnitudes for different points. Instead, the landscapes are observed to be better captured by a relaxed version of smoothness:

**Assumption 4.1.** *A second order differentiable function $F$ is $(L_0, L_1)$-smooth if for all $\boldsymbol{x} \in \mathbb{R}^d$ we have*

$$\|\nabla^2 F(\boldsymbol{x})\| \leq L_0 + L_1 \|\nabla F(\boldsymbol{x})\| . \tag{4.1}$$

Note that when $L_1 = 0$, it reduces to the original smoothness assumption. Also, all univariate polynomials (which can possibly be non-convex) like $F(x) = x^4$ are $(L_0, L_1)$-smooth.

Under this assumption, Zhang et al. (2020b) proved that the well-known gradient clipping technique can ensure SGD's convergence. Later, their results were improved by Zhang et al. (2020a) to show that SGD with clipping can be made unaffected by the $L_1$ in (4.1) and is able to recover the optimal convergence rate of SGD under the original smoothness setting.

Recently, Liu et al. (2022) also analyzed SGD with a local gradient clipping algorithm in the distributed learning setting under this condition and showed its linear parallel speedup property both theoretically and empirically, which means that the iteration complexity of the algorithm is reduced by a factor of $N$, the number of machines. This corroborated the motivation for studying the above assumption.

As observed by Zhang et al. (2020b), LSTMs empirically observe the $(L_0, L_1)$ smoothness condition (4.1). LSTMs have long been a powerhouse behind many machine learning tasks, especially the ones of Natural Language Processing (NLP) (Sundermeyer et al., 2012; Tai et al., 2015; Zhou et al., 2016). Yet, recently, Transform-

(a) Wikitext-2        (b) WMT'16 de-en
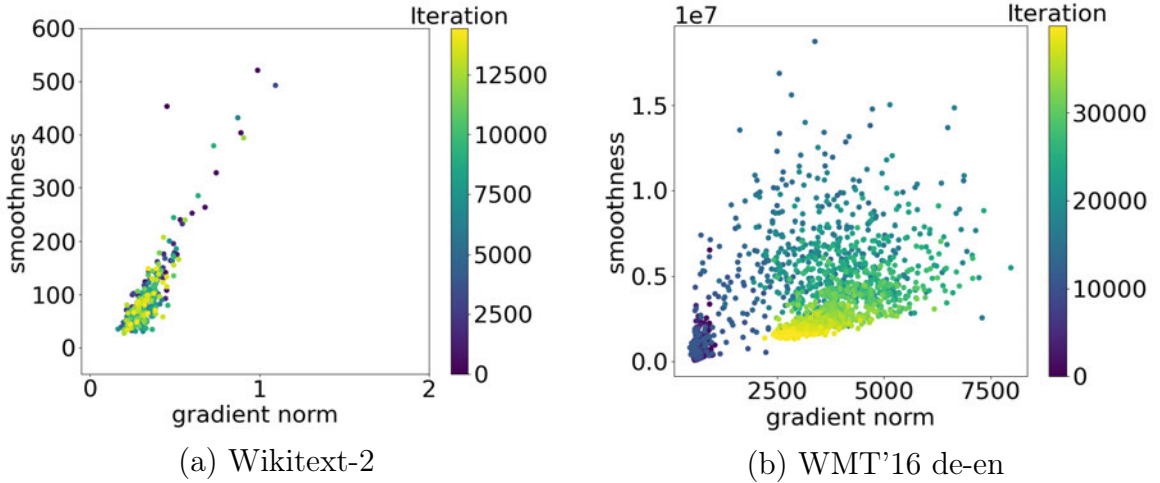
**Figure 4·1:** Local gradient Lipschitz constant vs. gradient norm on training (a) a 2-layer transformer encoder model on Wikitext-2 (b) a 6-layer Transformer on WMT'16 Multimodal Machine Translation de-en dataset. The colorbar indicates the number of iterations during training.

ers (Vaswani et al., 2017) are becoming more and more popular and have been consistently achieving SOTA results in the NLP field and beyond (Devlin et al., 2019; Dosovitskiy et al., 2021). Motivated by this, we empirically verified that losses employing Transformers also seem to satisfy the $(L_0, L_1)$-smooth assumption (4.1), see Figure 4·1.

For plotting Figure 4·1, we followed the method in Section H.3 of Zhang et al. (2020b). Specifically, given $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t+1}$, denote $\boldsymbol{d} := \boldsymbol{x}_{t+1} - \boldsymbol{x}_t$. We estimate the smoothness at $\boldsymbol{x}_t$ by

$$\hat{L}_t = \max_{\gamma \in \{\delta_1, \delta_2, \ldots, \delta_N\}} \frac{\|\nabla F(\boldsymbol{x}_t + \gamma \boldsymbol{d}) - \nabla F(\boldsymbol{x}_t)\|_2}{\|\gamma \boldsymbol{d}\|_2},$$

where $\{\delta_1, \delta_2, \ldots, \delta_N\}$ denotes the sample locations and we use $\{\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}\}$.

Figure 4·1(a) is on training a 2-layer Transformer Encoder to do language modeling on the Wikitext-2 dataset. The implementation, settings, and parameter choices

**(a)** Encoder First Layer

**(b)** Encoder Last Layer

**(c)** Decoder Second Layer

**(d)** Decoder Last Layer
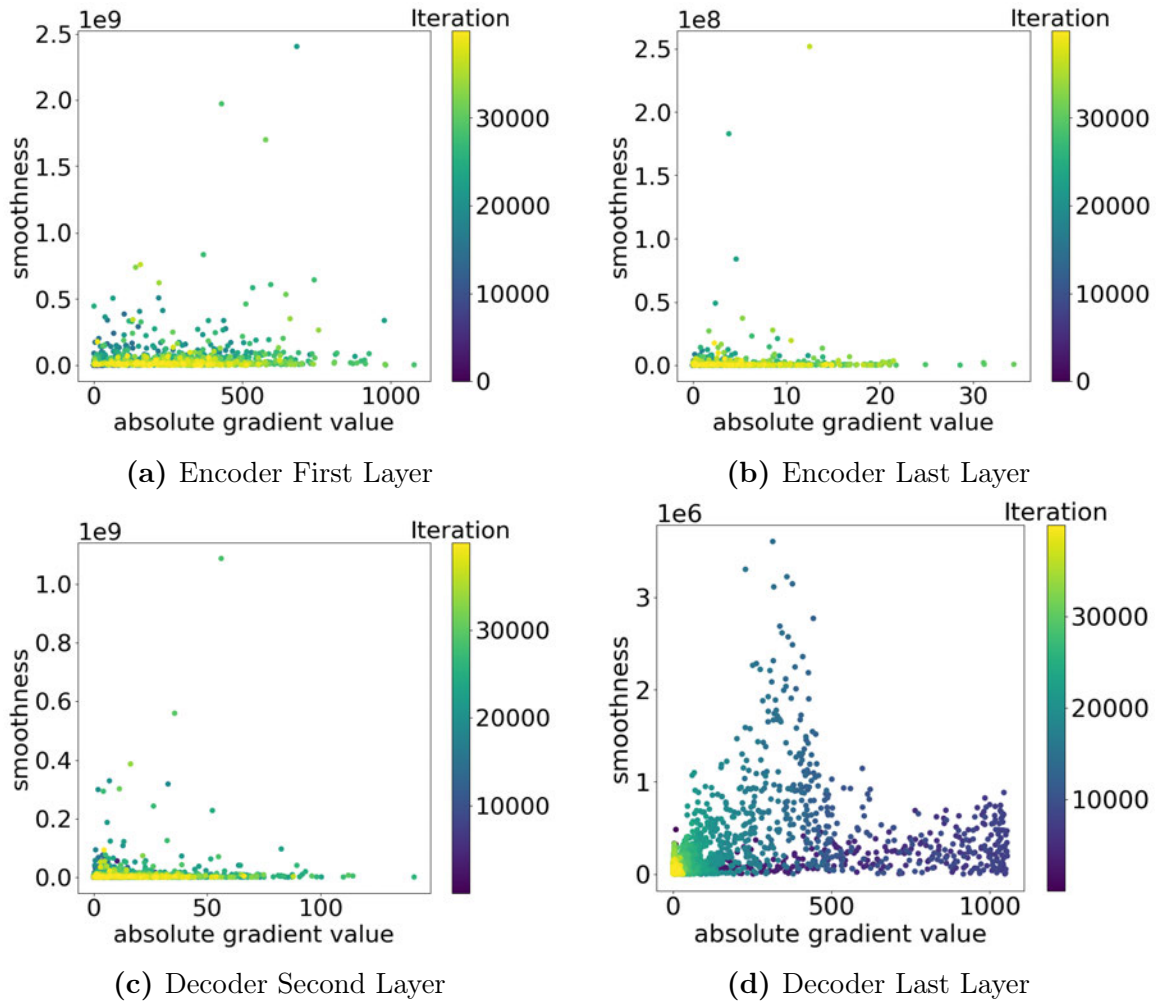
**Figure 4·2:** Local gradient Lipschitz constant vs. absolute gradient value on training a 6-layer Transformer on WMT'16 Multimodal Translation de-en dataset. Each figure represents a randomly picked coordinate in corresponding layers. The colorbar indicates the number of iterations during training.

follow this.[1] We only plot the first 5 training epochs. Figure 4·1(b) and 4·2 are on training a 6-layer Transformer to do machine translation on the WMT'16 Multimodal Machine Translation Task German-English dataset. The implementation of the transformer is forked from here[2] and we also follow their default settings. The mini-batch size is 256 and we trained for 400 epochs using Adam and report the whole training trajectory.

Yet, this is not the end of the story, as we will show in the next section: Transformers actually satisfy a more fine-grained relaxed smooth condition.

## 4.2    A Coordinate-wise Relaxed Smoothness Condition

The verification that Transformers observe the $(L_0, L_1)$-smooth condition in the previous section is exciting. However, after further investigation, we noticed that different coordinates, especially when they are in different layers of the model, exhibit very distinct degrees of variation of gradients as shown in Figure 4·2, in which we compared $\frac{\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_{t+1}) - \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right|}{|x_{t+1,j} - x_{t,j}|}$ vs. $\min\left(\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right|, \left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_{t+1})\right|\right)$.

Consequently, we propose to refine the $(L_0, L_1)$ condition in (4.1) to a coordinate-wise version to better capture the loss surface when training deep neural networks like Transformers.

**Assumption 4.2.** *We say that a differentiable function $F(\boldsymbol{x})$ is $(\boldsymbol{L_0}, \boldsymbol{L_1})$-smooth coordinate-wisely, if for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ for which $\|\boldsymbol{x} - \boldsymbol{y}\|_2 \leq \frac{1}{\|\boldsymbol{L_1}\|_\infty}$, we have for any $j \in [d]$ that*

$$\left|\frac{\partial F}{\partial x_j}(\boldsymbol{y}) - \frac{\partial F}{\partial x_j}(\boldsymbol{x})\right| \leq \left(\frac{L_{0,j}}{\sqrt{d}} + L_{1,j}\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x})\right|\right)\|\boldsymbol{y} - \boldsymbol{x}\|_2 .$$

---

[1]`https://pytorch.org/tutorials/beginner/transformer_tutorial.html`
[2]`https://github.com/jadore801120/attention-is-all-you-need-pytorch`

We will denote $\boldsymbol{L_0} := [L_{0,1}, L_{0,2}, \ldots, L_{0,d}]^T$ and $\boldsymbol{L_1} := [L_{1,1}, L_{1,2}, \ldots, L_{1,d}]^T$.

The following Lemma shows that our coordinate-wise $(\boldsymbol{L_0}, \boldsymbol{L_1})$ smooth assumption 4.2 is equivalent to the original $(L_0, L_1)$ smooth assumption (4.1) at least in 1-d case.

**Lemma 4.1.** *Let* $F : \mathbb{R} \to \mathbb{R}$ *be a twice continuously differentiable function. Then if (1) there exists some* $K_0, K_1 \geq 0$ *such that it holds for any* $x, y \in \mathbb{R}$ *with* $|y - x| \leq \frac{1}{K_1}$ *that* $|F'(y) - F'(x)| \leq (K_0 + K_1|F'(x)|)|y - x|$*, then (2) there exists some* $L_0, L_1 \geq 0$ *such that it holds for any* $x \in \mathbb{R}^d$ *that* $|F''(x)| \leq L_0 + L_1|F'(x)|$*, and vice versa.*

*Proof of Lemma 4.1.* **(1)** $\Rightarrow$ **(2)** By definition, for any $x \in \mathbb{R}$, we know that

$$
\begin{aligned}
F''(x) = \lim_{h \to 0} \frac{F'(x+h) - F'(x)}{h} &\leq \lim_{h \to 0} \frac{|F'(x+h) - F'(x)|}{|h|} \\
&\leq \lim_{h \to 0} \frac{(K_0 + K_1|F'(\boldsymbol{x})|)|h|}{|h|} = K_0 + K_1|F'(x)| \ .
\end{aligned}
$$

**(2)** $\Rightarrow$ **(1)** This is a special case for 1-d and $c = 1$ of Corollary A.4 in Zhang et al. (2020a). $\qquad\square$

Another motivation for this assumption comes from (Remark 2.3, Zhang et al., 2020a) where they noted that (4.1) can be relaxed to an assumption on gradient differences: there exists $K_0, K_1 > 0$ s.t. $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ with $\|\boldsymbol{x} - \boldsymbol{y}\|_2 \leq \frac{1}{K_1}$, we have

$$
\|\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})\|_2 \leq (K_0 + K_1\|\nabla F(\boldsymbol{x})\|_2)\|\boldsymbol{x} - \boldsymbol{y}\|_2. \tag{4.2}
$$

Indeed, our Assumption 4.2 implies (4.2) when $L_{0,j} = L_0$ and $L_{1,j} = L_1$ for all $j \in [d]$, up to constants (See Lemma 4.2 below). Note that the $\frac{1}{\sqrt{d}}$ factor in ours is exactly for easy comparison with (4.2).

**Lemma 4.2.** *When $L_{0,j} = L_0$ and $L_{1,j} = L_1$ for all $j \in [d]$, Assumption 4.2 implies (4.2) (up to constants).*

*Proof.* Suppose all $L_{0,j}, L_{1,j}$ are the same across $j$, then we have

$$
\begin{aligned}
\|\nabla F(\boldsymbol{y}) - \nabla F(\boldsymbol{x})\|_2 &= \sqrt{\sum_{j=1}^d \left| \frac{\partial F}{\partial x_j}(\boldsymbol{y}) - \frac{\partial F}{\partial x_j}(\boldsymbol{x}) \right|^2} \\
&\leq \sqrt{\sum_{j=1}^d \left( \frac{L_{0,j}}{\sqrt{d}} + L_{1,j} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}) \right| \right)^2 \times \|\boldsymbol{y} - \boldsymbol{x}\|_2^2} \\
&\leq \sqrt{\sum_{j=1}^d \left( \frac{2L_{0,j}^2}{d} + 2L_{1,j}^2 \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}) \right|^2 \right) \times \|\boldsymbol{y} - \boldsymbol{x}\|_2^2} \\
&\leq \sqrt{2}L_0 \|\boldsymbol{y} - \boldsymbol{x}\|_2 + \sqrt{2}L_1 \|\boldsymbol{y} - \boldsymbol{x}\|_2 \sqrt{\sum_{j=1}^d \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}) \right|^2} \\
&= \left( \sqrt{2}L_0 + \sqrt{2}L_1 \|\nabla F(\boldsymbol{x})\|_2 \right) \times \|\boldsymbol{y} - \boldsymbol{x}\|_2 . \qquad \square
\end{aligned}
$$

The original $(L_0, L_1)$ smoothness assumption (4.1) in Zhang et al. (2020b) was proposed as a generalization of the more common smoothness assumption, which says that the gradient should be Lipschitz. Indeed, when $L_1$ is zero, we recover the smoothness assumption. In contrast, when $L_{1,j}$ are non-zero, the smoothness of the function is potentially *unbounded*. However, Zhang et al. (2020b) works with norms and applies to the global scale, while ours is more fine-grained and it applies to each coordinate separately. We also note in passing that the smoothness assumption has been generalized in orthogonal directions in other work (Richtárik and Takác, 2014; Bernstein et al., 2018; Khaled and Richtárik, 2020).

One merit of Assumption 4.2 is that it gives us the following descent lemma.

**Lemma 4.3.** *Let $F$ be $(\boldsymbol{L_0}, \boldsymbol{L_1})$-smooth coordinate-wisely. For any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ for*

which $\|\boldsymbol{x} - \boldsymbol{y}\|_2 \le \frac{1}{\|\boldsymbol{L_1}\|_\infty}$, we have

$$F(\boldsymbol{y}) \le F(\boldsymbol{x}) + \langle \nabla F(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \sum_{j=1}^{d} \frac{\left( \frac{L_{0,j}}{\sqrt{d}} + L_{1,j} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}) \right| \right) \|\boldsymbol{y} - \boldsymbol{x}\|_2}{2} |y_j - x_j| \ .$$

Proof of Lemma 4.3.

$F(\boldsymbol{y})$

$$= F(\boldsymbol{x}) + \int_0^1 \langle \nabla F(\boldsymbol{x} + u(\boldsymbol{y} - \boldsymbol{x})), \boldsymbol{y} - \boldsymbol{x} \rangle du$$

$$= F(\boldsymbol{x}) + \langle \nabla F(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \int_0^1 \langle \nabla F(\boldsymbol{x} + u(\boldsymbol{y} - \boldsymbol{x})) - \nabla F(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle du$$

$$\le F(\boldsymbol{x}) + \langle \nabla F(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \left| \int_0^1 \langle \nabla F(\boldsymbol{x} + u(\boldsymbol{y} - \boldsymbol{x})) - \nabla F(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle du \right|$$

$$\le F(\boldsymbol{x}) + \langle \nabla F(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \int_0^1 |\langle \nabla F(\boldsymbol{x} + u(\boldsymbol{y} - \boldsymbol{x})) - \nabla F(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle| \, du$$

$$\le F(\boldsymbol{x}) + \langle \nabla F(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \int_0^1 \sum_{j=1}^{d} \left| \left[ \frac{\partial F}{\partial x_j}(\boldsymbol{x} + u(\boldsymbol{y} - \boldsymbol{x})) - \frac{\partial F}{\partial x_j}(\boldsymbol{x}) \right] (y_j - x_j) \right| du$$

$$\le F(\boldsymbol{x}) + \langle \nabla F(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \int_0^1 \sum_{j=1}^{d} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x} + u(\boldsymbol{y} - \boldsymbol{x})) - \frac{\partial F}{\partial x_j}(\boldsymbol{x}) \right| |y_j - x_j| du$$

$$= F(\boldsymbol{x}) + \langle \nabla F(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \sum_{j=1}^{d} \int_0^1 \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x} + u(\boldsymbol{y} - \boldsymbol{x})) - \frac{\partial F}{\partial x_j}(\boldsymbol{x}) \right| |y_j - x_j| du$$

$$\le F(\boldsymbol{x}) + \langle \nabla F(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \sum_{j=1}^{d} \int_0^1 u \left( \frac{L_{0,j}}{\sqrt{d}} + L_{1,j} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}) \right| \right) \|\boldsymbol{y} - \boldsymbol{x}\|_2 |y_j - x_j| du$$

$$= F(\boldsymbol{x}) + \langle \nabla F(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \sum_{j=1}^{d} \frac{\left( \frac{L_{0,j}}{\sqrt{d}} + L_{1,j} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}) \right| \right) \|\boldsymbol{y} - \boldsymbol{x}\|_2}{2} |y_j - x_j| \ ,$$

where the second inequality uses the fact that $\left|\int_a^b F(x)dx\right| \leq \int_a^b |F(x)|dx$ and the final inequality is due to Assumption 4.2. $\qquad\square$

Under this refined Assumption 4.2, we introduce in the next section an algorithm that can theoretically match the convergence rates of SGD with gradient clipping and empirically match the performance of Adam (Kingma and Ba, 2015) which is widely used in training LSTMs and Transformers.

## 4.3  A Generalized SignSGD Algorithm

When training LSTMs and Transformers on NLP tasks, a common practice is to use the Adam optimizer (Kingma and Ba, 2015). Moreover, given that these models observe the $(L_0, L_1)$ assumption, it would be natural to use some clipping procedure.

The algorithm and analysis of gradient clipping can be traced back to (Alber et al., 1998; Shor, 2012; Ermoliev, 1988) under the assumption that the function is convex and rapidly growing. Hazan et al. (2015) considered gradient clipping in quasi-convex optimization. Mai and Johansson (2021) showed the stability and convergence of stochastic gradient clipping algorithms for convex problems without the smoothness condition. Gradient clipping is a standard technique in training deep neural networks (Pascanu et al., 2013) such as RNNs and LSTMs. The theoretical analysis of gradient clipping for nonconvex models is pioneered by Zhang et al. (2020b), in which the authors analyzed the convergence of gradient clipping under the relaxed smoothness assumption rather than the standard smoothness assumption. Zhang et al. (2020a) further improved the convergence rate bound under the same assumption as in Zhang et al. (2020b). Gradient clipping is also used when there is a heavy tail noise in the stochastic gradient to establish high probability convergence rates (Cutkosky and Mehta, 2021; Gorbunov et al., 2020; Zhang et al., 2020c). Also, Cutkosky and
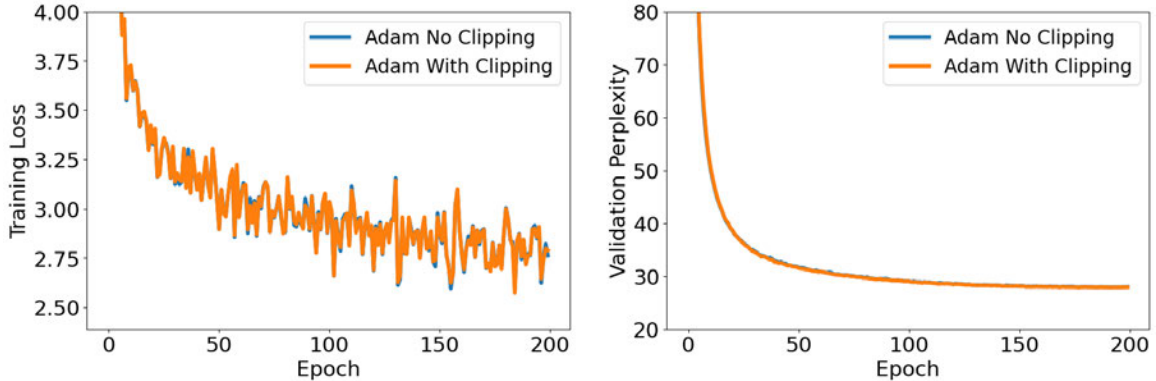
**Figure 4·3:** Training GPT-2 on Wikitext-103 using Adam with or without gradient clipping.

Mehta (2020) proved that normalized momentum improves normalized SGD under a second-order smoothness condition.

However, we found out that the use of clipping on Adam, while carried out in common practice (e.g., Wolf, 2019), *has no effect on the training and testing performance on optimizing a large transformer model as shown in Figure 4·3.*

Specifically, we conduct the experiment on the Wikitext-103 (103 million tokens, 180MB) (Merity et al., 2017) language modeling task, with a 16-layer GPT-2 transformer model (Radford et al., 2019). This GPT-2 model has an input length of 256 tokens, 410-dimension word embedding, 16 Attention layers with 10 Attention heads and 2100 hidden dimensions. The model size is 201.58 MB. The vocabulary size is 28996. We use the hyper-parameter settings prescribed in (Wolf, 2019): batch size 256, warm up step size from 0 to $2.5 \times 10^{-4}$ in the first 64000 samples (i.e., 250 iterations) and then cosine-anneal step size to zero, on top of an Adam optimizer. It takes about 40 hours to train 200 epochs on 8 V100 GPUs. We use clipping threshold max_norm 0.25 for the entire model as prescribed in the literature (Wolf, 2019). We also count that with this clipping scheme, clipping occurs in every single batch. As we can see from Figure 4·3, neither training loss (2.79 vs 2.76) nor perplexity score (27.92

---

**Algorithm 4.1** Generalized SignSGD *(All operations on vectors are element-wise.)*

---

1: Inputs: $\boldsymbol{x}_1, \beta_1, \beta_2, \eta$
2: $\boldsymbol{m}_0 = 0, \boldsymbol{v}_0 = 0$
3: **for** $t = 1, \ldots, T$ **do**
4:    Compute an unbiased estimate $\nabla f(\boldsymbol{x}_t, \xi_t)$ of $\nabla F(\boldsymbol{x}_t)$, denoted as $\boldsymbol{g}_t$
5:    $\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t$
6:    $\boldsymbol{v}_t = \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2)\boldsymbol{m}_t^2$
7:    $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta \frac{\boldsymbol{m}_t}{\sqrt{\boldsymbol{v}_t}}$
8: **end for**

---

vs 27.97) differs much in the clipping and the non-clipping cases, which suggests that Adam naturally achieves gradients clipping effect.

In retrospect, this might not be surprising: It is known that Adam has an implicit clipping behavior due to the normalization by the estimated second moment of the gradients. Indeed, Adam can be interpreted as a variant of SignSGD (Balles and Hennig, 2018).

Inspired by this, we present in Algorithm 4.1 a generalized SignSGD algorithm. This algorithm encompasses a variety of optimization algorithms.

At first sight, it seems very similar to Adam. Indeed, if we employ $\boldsymbol{g}_t^2$ in computing $\boldsymbol{v}_t$ instead of $\boldsymbol{m}_t^2$, then it is exactly Adam, except for the bias correction terms. We would like to clarify that the idea of this change has been explored before. Reddi et al. (2021) adopted this change to prove the convergence of Adam in a federated learning setting; yet, they only consider the smooth setting and require a large $\epsilon$ to obtain convergence in contrast to the original Adam. Later, Wang et al. (2021) explored this idea in more detail, but their analyses are still restricted to the smooth setting. The intuition is that $\boldsymbol{m}_t$ represents a better update direction than $\boldsymbol{g}_t$ and can thus better capture the second-moment information. Yet, in this work, the motivation for adopting this idea comes from the known effect of momentum on reducing the influence of noises (Cutkosky and Mehta, 2020). Indeed, in our analysis the difference

between $\boldsymbol{m}_t$ and $\nabla F(\boldsymbol{x}_t)$ is much more controllable than between $\boldsymbol{g}_t$ and $\nabla F(\boldsymbol{x}_t)$. Thus, we consider employing $\boldsymbol{m}_t$ in computing $\boldsymbol{v}_t$ a better choice.

On the other end, the careful reader might observe that Algorithm 4.1 recovers the SignSGD with Momentum algorithm, also called SIGNUM in Bernstein et al. (2018), when setting $\beta_2 = 0$. Sign-based algorithms are naturally suited to distributed learning (Li et al., 2014) and the idea dated back to at least Rprop (Riedmiller and Braun, 1993). The convergence to a stationary point (with $\ell_1$ norm) under a coordinate-wise smoothness condition has been established for SignSGD with/without the momentum in Bernstein et al. (2018) though they necessitate large mini-batches to control the variance of the noise. Yet, we are more interested in their property of the update size being bounded without the need for explicit gradient clipping.

Note that both SignSGD and Adam are good candidates for optimization algorithms whose updates must be bounded on functions that satisfy the $(\boldsymbol{L_0}, \boldsymbol{L_1})$ condition. Indeed, SignSGD can be seen as an extreme form of gradient clipping. On the other hand, as said above, Adam does not seem to require gradient clipping at all when used to train the large Transformer model as shown in Figure 4·3.

Consequently, we expect our algorithm, a generalization of SignSGD and a close resemblance to Adam, can enjoy the merits of both and be robust to the unbounded smoothness in the $(\boldsymbol{L_0}, \boldsymbol{L_1})$ scenario. This claim is formalized in Theorem 4.4 for which we require the following assumption.

**Assumption 4.3.** *For each $j \in [d]$, there exists $\sigma_j > 0$ such that for all $\boldsymbol{x} \in \mathbb{R}^d$ and $\xi \sim \mathcal{D}$, the noise satisfies $\left| \frac{\partial f}{\partial x_j}(\boldsymbol{x}, \xi) - \frac{\partial F}{\partial x_j}(\boldsymbol{x}) \right| \leq \sigma_j$ with probability $1$. We will denote $\boldsymbol{\sigma} := [\sigma_1, \sigma_2, \ldots, \sigma_d]^T$.*

**Theorem 4.4.** *Under Assumptions 1.1, 4.2, and 4.3, assume $M_j := \sup \left\{ \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}) \right| : F(\boldsymbol{x}) \leq F(\boldsymbol{x}_1) \right\}$ is finite for each $j \in [d]$, let $\Delta$ be any upper*

*bound on* $F(\boldsymbol{x}_1) - F^*$, $\alpha = \min\left(\frac{\sqrt{\|\boldsymbol{L_0}\|_1}\sqrt{\Delta}}{\|\boldsymbol{\sigma}\|_1\sqrt{T}}, 1\right)$, $\beta_1 = 1 - \alpha$, $\frac{\sqrt{\beta_2}}{\beta_1} < 1$, $\rho = 1 - \frac{\sqrt{\beta_2}}{\beta_1}$,

$\eta = \frac{\sqrt{\Delta\alpha}}{\sqrt{\|\boldsymbol{L_0}\|_1}\sqrt{T}}$, *for* $T \geq \max\left(\frac{100d\Delta\|\boldsymbol{L_1}\|_\infty^2}{(1-\beta_2)\rho^2\|\boldsymbol{L_0}\|_1}, \frac{10000d^2\Delta\|\boldsymbol{\sigma}\|_1^2\|\boldsymbol{L_1}\|_\infty^4}{(1-\beta_2)^2\rho^4\|\boldsymbol{L_0}\|_1^3}\right)$, *Algorithm 4.1 guaran-*

*tees, with probability at least* $1 - \delta$, *that*

$$
\min_{t\in[T]} \|\nabla F(\boldsymbol{x}_t)\|_1 = \mathcal{O}\left(\frac{\sqrt{\log(dT/\delta)}\|\boldsymbol{L_0}\|_1^{1/4}\Delta^{1/4}\|\boldsymbol{\sigma}\|_1^{1/2}}{\rho\sqrt{1-\beta_2}T^{1/4}} + \frac{\log(dT/\delta)\sqrt{\|\boldsymbol{L_0}\|_1\Delta}}{\rho\sqrt{T}}\right)
$$

$$
+ \mathcal{O}\left(\frac{\|\boldsymbol{M}\|_1 + \|\boldsymbol{\sigma}\|_1}{\rho}\exp\left(-\frac{\sqrt{1-\beta_2}\|\boldsymbol{L_0}\|_1^{3/4}}{\sqrt{d}\|\boldsymbol{L_1}\|_\infty\|\boldsymbol{\sigma}\|_1^{1/2}\Delta^{1/4}}T^{1/4}\right)\right)
$$

$$
+ \mathcal{O}\left(\frac{\|\nabla F(\boldsymbol{x}_1)\|_1}{T}\right).
$$

*Furthermore, for the case when* $\beta_2 = 0$, *we have the following refined guarantee:*

$$
\min_{t\in[T]} \|\nabla F(\boldsymbol{x}_t)\|_1 = \mathcal{O}\left(\frac{\sqrt{\log(dT/\delta)}\|\boldsymbol{L_0}\|_1^{1/4}\Delta^{1/4}\|\boldsymbol{\sigma}\|_1^{1/2}}{T^{1/4}} + \frac{\log(dT/\delta)\sqrt{\|\boldsymbol{L_0}\|_1\Delta}}{\sqrt{T}}\right)
$$

$$
+ \mathcal{O}\left(\frac{\|\nabla F(\boldsymbol{x}_1)\|_1}{\sqrt{T}}\left(\frac{1}{\sqrt{T}} + \frac{\|\boldsymbol{\sigma}\|_1}{\sqrt{\|\boldsymbol{L_0}\|_1\Delta}}\right) + \frac{\|\boldsymbol{\sigma}\|_1}{T}\right).
$$

Here, $M_j$ denotes the maximum absolute value of the partial derivative of $F$ for coordinate $j$ among the sub-level set of $F(\boldsymbol{x}_1)$, namely any point $\boldsymbol{x}$ with $F(\boldsymbol{x}) \leq F(\boldsymbol{x}_1)$. In other words, we assume gradients to be bounded in the sub-level set of $F(\boldsymbol{x}_1)$; yet, we do not make any restriction on gradients outside of this set. We believe this is not a strong assumption, for example, when the sub-level set of $F(\boldsymbol{x}_1)$ is bounded, then by the assumed continuity of gradients it trivially holds. Also, we just require an upper bound and it can even be exponentially large as we have an exponentially decaying coefficient to counteract it: notice how the term $\|\boldsymbol{M}\|_1$ is multiplied by a term that decays exponentially with $T$. Better still, when $\beta_2 = 0$, we no longer even need this assumption and the algorithm is entirely free of the influence of $\|\boldsymbol{M}\|_1$. To see why this is good, we show a refined lower bound of Gradient

Descent under the relaxed smoothness scenario below which is originally in Zhang et al. (2020b).

**Theorem 4.5.** *Fix $\epsilon > 0, L_0 > 0, L_1 > 0, M \geq \max(\frac{L_0}{L_1}, \epsilon)$, and $x_0 \in \mathbb{R}$. Pick any constant step size $\eta$ for GD, with the knowledge of the above constants. Then, there exists a 1-d $(L_0, L_1)$-smooth function $F$, bounded from below by $F^*$ (finite), and such that $\sup\{|F'(x)| : F(x) \leq F(x_0)\} \leq M$ on which the number of iterations $T$ of GD with step size $\eta$ to guarantee $|F'(x_T)| < \epsilon$ is at least*

$$\frac{ML_1(F(x_0) - F^* - \frac{15\epsilon^2}{16L_0})}{2\epsilon^2 \left(\ln \frac{ML_1}{L_0} + 1\right)} .$$

*Proof of Theorem 4.5.* By Lemma 4.1, we know that, in 1-d case, our coordinate-wise $(\boldsymbol{L_0}, \boldsymbol{L_1})$ assumption 4.2 is equivalent as the original one (4.1). Thus, without loss of generality, we use the original condition (4.1) in the proof. We will construct two different $(L_0, L_1)$-smooth functions based on the value of $\eta$.

**Case** $\eta > \frac{2}{ML_1} \left(\ln \frac{ML_1}{L_0} + 1\right)$. In this case, we can construct a function on which GD does not converge, hence the lower bound is trivially true. Consider the function

$$F(x) = \begin{cases} L_0 \frac{e^{-L_1 x - 1}}{L_1^2} & x < -\frac{1}{L_1} \\ \\ L_0 \frac{x^2}{2} + \frac{L_0}{2L_1^2} & x \in [-\frac{1}{L_1}, \frac{1}{L_1}] \\ \\ L_0 \frac{e^{L_1 x - 1}}{L_1^2} & x > \frac{1}{L_1} \end{cases}$$

Note that $F$ is $(L_0, L_1)$-smooth. Without loss of generality, we can assume $x_0 = \frac{1}{L_1} \left(\ln \frac{ML_1}{L_0} + 1\right)$, in fact if this is not the case we can translate the function $F$ ac-

cordingly. This setting of $x_0$ guarantees that the bound on the gradient is correct. Moreover, with this choice, we claim the function will diverge. To see this, we use mathematical induction to show that $|x_{t+1}| > |x_t|$ and $\text{sign}(x_{t+1}) \neq \text{sign}(x_t)$ for any $t \geq 0$. First, for the case when $t = 0$, we have

$$x_1 = x_0 - \eta F'(x_0) = x_0 - \frac{\eta L_0}{L_1} e^{L_1 x_0 - 1} = x_0 - \eta M < x_0 - 2x_0 = -x_0 \ .$$

Then suppose the condition holds up until $t$ and we prove for $t+1$. From the formula of $F$, we have that $\text{sign}(F'(x)) = \text{sign}(x)$ and that $F$ is monotonically increasing with $|x|$. Thus, from the update of gradient descent which moves along the negative direction of the gradient, if we can show that $|x_{t+1}| > |x_t|$, then $\text{sign}(x_{t+1}) \neq \text{sign}(x_t)$. This leads to

$$|x_{t+1}| = |x_t - \eta F'(x_t)| > |x_t| \Leftarrow \eta |F'(x_t)| > 2|x_t| \Leftarrow \eta L_0 > \frac{2|x_t|L_1}{\exp(L_1|x_t| - 1)} \ .$$

Now, note that $\psi(x) = \frac{2|x|L_1}{\exp(L_1|x|-1)}$ is decreasing for $x > \frac{1}{L_1}$ and increasing for $x < -\frac{1}{L_1}$. Hence, we have that

$$\eta L_0 > \frac{2|x_0|L_1}{\exp(L_1|x_0| - 1)} > \frac{2|x_t|L_1}{\exp(L_1|x_t| - 1)},$$

where the first inequality is true by the choice of $x_0 > \frac{1}{L_1}$ and the condition on $\eta$ and the second one is true by the induction hypothesis.

**Case** $\eta \leq \frac{2}{ML_1}\left(\ln \frac{ML_1}{L_0} + 1\right)$.

Now, consider

$$
F(x) = \begin{cases}
-\epsilon x, & x < -\frac{3\epsilon}{2L_0} \\[2mm]
\frac{L_0}{2}x^2 - \frac{L_0^3 x^4}{27\epsilon^2} + \frac{9\epsilon^2}{16L_0}, & x \in [-\frac{3\epsilon}{2L_0}, \frac{3\epsilon}{2L_0}] \\[2mm]
\epsilon x, & x > \frac{3\epsilon}{2L_0}
\end{cases}
$$

We have that $F$ is $(L_0, 0)$-smooth, hence also $(L_0, L_1)$-smooth. Note that the presence of the fourth power makes this function twice differentiable. Moreover, the maximum gradient in this case is $\epsilon \leq M$.

As before, without loss of generality, let the initial point $x_0 = \frac{3\epsilon}{2L_0} + \Delta$, where $\Delta > 0$. We have that $F(x_0) - F^* = \epsilon \left( \Delta + \frac{3\epsilon}{2L_0} \right) - \frac{9\epsilon^2}{16L_0}$, hence $\Delta = \frac{1}{\epsilon}\left( F(x_0) - F^* \right) - \frac{15\epsilon}{16L_0}$. Now, while we stay on the last branch of the function, we have

$$
x_{t+1} = x_t - \eta\epsilon \geq x_t - \epsilon\frac{2}{ML_1}\left( \ln\frac{ML_1}{L_0} + 1 \right) .
$$

Hence, we have that, for

$$
t \leq \frac{ML_1\Delta}{2\epsilon\left( \ln\frac{ML_1}{L_0} + 1 \right)} = \frac{ML_1\left( F(x_0) - F^* - \frac{15\epsilon^2}{16L_0} \right)}{2\epsilon^2\left( \ln\frac{ML_1}{L_0} + 1 \right)},
$$

we guarantee $|F'(x_t)| = \epsilon$. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

On a side note, Theorem 4.5 is a fixed version of the lower bound in Zhang et al. (2020b). First of all, they have a logarithm of a quantity with units, $M$, which is an undefined mathematical operation. A closer look at the proof reveals that, differently from the statement of their theorem, they construct a function with $L_0 = L_1$, which explains why these terms are missing in the logarithm. Moreover, it is also unclear if the second constructed function satisfies the assumptions of the theorem. We correct

all these issues by properly scaling the constructed functions so that they always satisfy the $(L_0, L_1)$ condition and all the units are coherent. This result in the correct term inside the logarithm and the right conditions on $L_0$, $L_1$, $M$, and $\epsilon$.

Theorem 4.5 shows that in the relaxed smoothness setting, GD with any constant step size will suffer from a linear term depending on $L_1 M$. Compared with GD, our algorithm only has an exponentially decaying dependence on $L_1 M$. We consider this to be a substantial merit of our algorithm. Furthermore, when $\beta_2 = 0$ in which case we recover the SignSGD with Momentum algorithm, we can even show that it completely removes the effects of the unbounded gradient norms. Also notice that in such case we actually no longer need the assumption of $M_j := \sup \left\{ \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}) \right| : F(\boldsymbol{x}) \leq F(\boldsymbol{x}_1) \right\}$ being finite for each $j \in [d]$ anymore, and the $\|\boldsymbol{L_1}\|_\infty$ term does not appear in the final bound anymore.

We also would like to point out that this bound closely resembles the one achieved by SGD with gradient clipping algorithm (Zhang et al., 2020a) except that we consider the coordinate-wise setting: take the setting of $\beta_2 = 0$ for example, we need at most $\mathcal{O}\left( \Delta \max \left\{ \frac{\|\boldsymbol{\sigma}\|_1^2 \|\boldsymbol{L_0}\|_1}{\epsilon^4}, \frac{d^2 \|\boldsymbol{\sigma}\|_1^2 \|\boldsymbol{L_1}\|_\infty^4}{\|\boldsymbol{L_0}\|_1^3}, \frac{d \|\boldsymbol{L_1}\|_\infty^2}{\|\boldsymbol{L_0}\|_1} \right\} \right)$ to get a point $\boldsymbol{x}$ with $\|\nabla F(\boldsymbol{x})\|_1 \leq \epsilon$ with high probability.

Careful readers might be concerned on the relations between $\alpha$, $\beta_1$, $\beta_2$, $\rho$, and $T$ when $\alpha \neq 1$. We would like to note that, when $\beta_2$ is fixed, $\alpha$ is inversely proportional to $\sqrt{T}$. In turn, the definition of $\rho$ means that as $T$ grows, $\rho$ grows and approaches $1 - \sqrt{\beta_2}$. Thus, the two conditions for $T$ decreases when $T$ grows. This means that there must exists a threshold of $T$ above which the two conditions on $T$ always hold. In summary, Theorem 4.4 conveys the same message as Zhang et al. (2020a) that as long as the expected $\epsilon$ is sufficiently small, the complexity no longer has a dependency on $\boldsymbol{L_1}$. Also note that we do not need the knowledge of $\boldsymbol{L_1}$ to set the algorithm, and in this case, as long as $\epsilon$ is small enough, the above bound reduces

to $\mathcal{O}\left(\Delta\|\boldsymbol{\sigma}\|_1^2\|\boldsymbol{L_0}\|_1\epsilon^{-4}\right)$ with no dependency on $\boldsymbol{L_1}$ at all. We can thus consider the algorithm to be *adaptive* to $\boldsymbol{L_1}$.

Finally, as a side note, the almost surely bounded assumption 4.3 of the noise can be relaxed to sub-gaussian noise, using standard extensions of the Freedman inequality (e.g., Harvey et al., 2019).

## 4.4  Convergence Analyses of our Generalized SignSGD Algorithm

The proof of the theorem is highly technical and it uses recent advancements in the analysis of momentum methods (Cutkosky and Mehta, 2020), key techniques to deal with the $(L_0, L_1)$ assumption (Zhang et al., 2020a), as well as a novel and essential inductive argument to control the norm of past gradients.

We first write down some notations that we will use heavily for easier reference:

$$\bar{\tau} = \frac{\sqrt{1 - \beta_2}}{\eta\sqrt{d}\|\boldsymbol{L_1}\|_\infty}, \quad \alpha = 1 - \beta_1, \quad \rho = 1 - \beta_2^{1/2}\beta_1^{-1},$$

$$\boldsymbol{\epsilon}_t = \boldsymbol{m}_t - \nabla F(\boldsymbol{x}_t), \quad \tilde{\boldsymbol{\epsilon}}_t = \boldsymbol{g}_t - \nabla F(\boldsymbol{x}_t),$$

$$E_j = 6\sigma_j \max(1, \log(1/\delta)) + \frac{6}{\sqrt{1 - \beta_1^2}}\sqrt{\sigma_j^2 \max(1, \log(1/\delta))},$$

$$B_j = \frac{\eta L_{0,j}}{\sqrt{1 - \beta_2}(1 - \beta_1)} + \beta_1^{\bar{\tau}}(M_j + \sigma_j) + (1 - \beta_1)E_j,$$

$$C_j = 1 + \frac{\eta\sqrt{d}L_{1,j}}{(1 - \beta_1)\sqrt{1 - \beta_2}}, \quad D = 1 - \frac{2\eta\sqrt{d}\|\boldsymbol{L_1}\|_\infty}{\sqrt{1 - \beta_2}(1 - \beta_1)},$$

$$A = \frac{\rho}{10\sqrt{1 - \beta_2}} .$$

Also, we would need the following formula many times:

$$\beta_1^{\bar{\tau}} = (1 - \alpha)^{\frac{1}{\alpha}\frac{\alpha\sqrt{1-\beta_2}}{\eta\sqrt{d}\|\boldsymbol{L_1}\|_\infty}} \le e^{-\frac{\alpha\sqrt{1-\beta_2}}{\eta\sqrt{d}\|\boldsymbol{L_1}\|_\infty}}, \tag{4.3}$$

where in the first inequality we used the fact that $(1-x)^{1/x} \leq \frac{1}{e}$ for $0 < x < 1$.

**Lemma 4.6.** *With the notations in Algorithm 4.1, for each $j \in [d]$ we have*

$$m_{t,j} = (1-\beta_1) \sum_{\tau=1}^{t} \beta_1^{t-\tau} g_{\tau,j}, \quad v_{t,j} = (1-\beta_2) \sum_{\tau=1}^{t} \beta_2^{t-\tau} m_{\tau,j}^2, \quad \frac{|m_{t,j}|}{\sqrt{v_{t,j}}} \leq \frac{1}{\sqrt{1-\beta_2}} .$$

*Proof of Lemma 4.6.* For all $t \geq 1$, we have

$$m_{t,j} = \beta_1 m_{t-1,j} + (1-\beta_1) g_{t,j}$$

$$= \beta_1 [\beta_1 m_{t-2,j} + (1-\beta_1) g_{t-1,j}] + (1-\beta_1) g_{t,j}$$

$$= \ldots = (1-\beta_1) \sum_{\tau=1}^{t} \beta_1^{t-\tau} g_{\tau,j} .$$

Similarly for $v_{t,j}$. Next,

$$\frac{|m_{t,j}|}{\sqrt{v_{t,j}}} = \frac{|m_{t,j}|}{\sqrt{(1-\beta_2) \sum_{\tau=1}^{t} \beta_2^{t-\tau} m_{\tau,j}^2}} \leq \frac{1}{\sqrt{1-\beta_2}} . \qquad \square$$

The following lemma shows when we can apply Assumption 4.2 and Lemma 4.3.

**Lemma 4.7.** *With notations in Algorithm 4.1, for $\tau \leq \bar{\tau} = \frac{\sqrt{1-\beta_2}}{\eta\sqrt{d}\|\boldsymbol{L_1}\|_\infty}$, we have* $\|\boldsymbol{x}_{t-\tau} - \boldsymbol{x}_t\|_2 \leq \frac{1}{\|\boldsymbol{L_1}\|_\infty}$.

*Proof of Lemma 4.7.* Using Lemma 4.6 we have

$$|x_{t-\tau,j} - x_{t,j}| \leq \sum_{i=1}^{\tau} |x_{t-i,j} - x_{t-i+1,j}| \leq \frac{\eta\tau}{\sqrt{1-\beta_2}} \leq \frac{1}{\sqrt{d}\|\boldsymbol{L_1}\|_\infty}$$

$$\Rightarrow \|\boldsymbol{x} - \boldsymbol{y}\|_2 \leq \frac{1}{\|\boldsymbol{L_1}\|_\infty} . \qquad \square$$

The following two lemmas are the major tools we use to analyze the effects of noises in which Lemma 4.8 is from Cutkosky and Mehta (2021, Lemma 12).

**Lemma 4.8.** *Suppose $X_1, \ldots, X_T$ is a martingale difference sequence in a Hilbert space and $\|X_t\| \leq R$ almost surely for some constant $R$. Further, assume $\mathbb{E}_t[\|X_t\|^2] \leq \sigma_t^2$ with probability 1 for some constants $\sigma_t$, where $\mathbb{E}_t[\cdot] \triangleq \mathbb{E}[\cdot|\xi_1, \xi_2, \ldots, \xi_{t-1}]$ denotes the expectation conditioned on all past randomnesses. Then, with probability at least $1 - 3\delta$, for all $k \leq T$ we have*

$$\left\| \sum_{t=1}^{k} X_t \right\| \leq 3R \max(1, \log(1/\delta)) + 3\sqrt{\sum_{t=1}^{k} \sigma_t^2 \max(1, \log(1/\delta))} \ .$$

**Lemma 4.9.** *Assume Assumption 4.3. With the notation of Algorithm 4.1, let $j \in [d]$ and $\beta_1 < 1$. Then, with probability at least $1 - 3\delta$, for any $t_0 \in [t]$, we have*

$$\left| \sum_{\tau=1}^{t_0} \beta_1^{t-\tau} \left( g_{\tau,j} - \frac{\partial F}{\partial x_j}(\boldsymbol{x}_\tau) \right) \right| \leq 3\sigma_j \max(1, \log(1/\delta))$$
$$+ \frac{3}{\sqrt{1-\beta_1^2}} \sqrt{\sigma_j^2 \max(1, \log(1/\delta))} \ .$$

*Proof of Lemma 4.9.* Recall Assumption 4.3 and notice that $\beta_1^{t-\tau} \leq 1$ for all $\tau \in [1, t]$, we know that $\left| \beta_1^{t-\tau} \left( g_{\tau,j} - \frac{\partial F}{\partial x_j}(\boldsymbol{x}_\tau) \right) \right| \leq \sigma_j$ almost surely. It also means $\mathbb{E}_\tau \left[ \left( \beta_1^{t-\tau} \left( g_{\tau,j} - \frac{\partial F}{\partial x_j}(\boldsymbol{x}_\tau) \right) \right)^2 \right] \leq \beta_1^{2(t-\tau)} \sigma_j^2$. Now, in Algorithm 4.1 we noted that $\boldsymbol{g}_\tau$ is an unbiased estimate of $\nabla F(\boldsymbol{x}_\tau)$ which means $\mathbb{E}_\tau \left[ \beta_1^{t-\tau} \left( g_{\tau,j} - \frac{\partial F}{\partial x_j}(\boldsymbol{x}_\tau) \right) \right] = 0$. Thus, $\left\{ \beta_1^{t-\tau} \left( g_{\tau,j} - \frac{\partial F}{\partial x_j}(\boldsymbol{x}_\tau) \right) \right\}_{1,\ldots,t}$ is a martingale difference sequence. Then, using Lemma 4.8, with probability at least $1 - 3\delta$, we have for all $t_0 \in [t]$ that

$$\left| \sum_{\tau=1}^{t_0} \beta_1^{t-\tau} \left( g_{\tau,j} - \frac{\partial F}{\partial x_j}(\boldsymbol{x}_\tau) \right) \right|$$
$$\leq 3\sigma_j \max(1, \log(1/\delta)) + 3\sqrt{\sum_{\tau=1}^{t_0} \beta_1^{2(t-\tau)} \sigma_j^2 \max(1, \log(1/\delta))}$$
$$\leq 3\sigma_j \max(1, \log(1/\delta)) + \frac{3}{\sqrt{1-\beta_1^2}} \sqrt{\sigma_j^2 \max(1, \log(1/\delta))} \ . \qquad \square$$

The following lemma upper bounds the differences between recent true gradients and the current one.

**Lemma 4.10.** *With the notation of Algorithm 4.1 and under the assumptions in Theorem 4.4, for any $j \in [d]$ and any $t_0$ with $t - t_0 \leq \bar{\tau} = \frac{\sqrt{1-\beta_2}}{\eta\sqrt{d}\|\boldsymbol{L_1}\|_\infty}$, we have*

$$\sum_{\tau=t_0}^{t} \beta_1^{t-\tau} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) - \frac{\partial F}{\partial x_j}(\boldsymbol{x}_\tau) \right| \leq \left( L_{0,j} + L_{1,j}\sqrt{d} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \right) \frac{\eta}{(1-\beta_1)^2\sqrt{1-\beta_2}} .$$

*Proof of Lemma 4.10.*

$$\sum_{\tau=t_0}^{t} \beta_1^{t-\tau} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) - \frac{\partial F}{\partial x_j}(\boldsymbol{x}_\tau) \right|$$

$$\leq \sum_{\tau=t_0}^{t} \beta_1^{t-\tau} \left( \frac{L_{0,j}}{\sqrt{d}} + L_{1,j} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \right) \|x_t - x_\tau\|_2$$

$$\leq \sum_{\tau=t_0}^{t} \beta_1^{t-\tau} \left( L_{0,j} + L_{1,j}\sqrt{d} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \right) \frac{\eta(t-\tau)}{\sqrt{1-\beta_2}}$$

$$= \left( L_{0,j} + L_{1,j}\sqrt{d} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \right) \frac{\eta}{\sqrt{1-\beta_2}} \sum_{\tau=t_0}^{t} (t-\tau)\beta_1^{t-\tau}$$

$$\leq \left( L_{0,j} + L_{1,j}\sqrt{d} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \right) \frac{\eta}{(1-\beta_1)^2\sqrt{1-\beta_2}} ,$$

where the first inequality is due to Assumption 4.2 and Lemma 4.7, the second inequality uses Lemma 4.6, and the final inequality uses the fact that $\sum_{k=1}^{N} ka^k \leq \frac{1}{(1-a)^2}$ for any $0 < a < 1$. $\qquad\square$

The following lemma upper bounds a past momentum with the current one.

**Lemma 4.11.** *With the notation of Algorithm 4.1 and under the assumptions of Theorem 4.4, for any $\tau \leq \bar{\tau} = \frac{\sqrt{1-\beta_2}}{\eta\sqrt{d}\|\boldsymbol{L_1}\|_\infty}$, with probability at least $1 - 3\delta$, it holds that*

$$|m_{t-\tau,j}| \leq \beta_1^{-\tau} \left( |m_{t,j}| + \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| + (1-\beta_1)E_j \right)$$

$$+ \beta_1^{-\tau} \left( \left( L_{0,j} + L_{1,j}\sqrt{d} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \right) \frac{\eta}{(1-\beta_1)\sqrt{1-\beta_2}} \right) .$$

*Proof of Lemma 4.11.* Denoting by $\tilde{\boldsymbol{\epsilon}}_t = \boldsymbol{g}_t - \nabla F(\boldsymbol{x}_t)$ and using Lemma 4.6, we have

$$|m_{t-\tau,j} - \beta_1^{-\tau} m_{t,j}|$$

$$= \left| (1-\beta_1) \sum_{i=1}^{t} \beta_1^{t-\tau-i} g_{i,j} - (1-\beta_1) \sum_{i=1}^{t-\tau} \beta_1^{t-\tau-i} g_{i,j} \right|$$

$$= (1-\beta_1) \left| \sum_{i=t-\tau+1}^{t} \beta_1^{t-\tau-i} g_{i,j} \right|$$

$$\leq (1-\beta_1) \left| \sum_{i=t-\tau+1}^{t} \beta_1^{t-\tau-i} \frac{\partial F}{\partial x_j}(\boldsymbol{x}_i) \right| + (1-\beta_1) \left| \sum_{i=t-\tau+1}^{t} \beta_1^{t-\tau-i} \tilde{\epsilon}_{i,j} \right| . \qquad (4.4)$$

We now upper bound the first term of (4.4) using Lemma 4.10 by using the fact that $\tau \leq \frac{\sqrt{1-\beta_2}}{\eta\|\boldsymbol{L_1}\|_1}$:

$$\left| \sum_{i=t-\tau+1}^{t} \beta_1^{t-\tau-i} \frac{\partial F}{\partial x_j}(\boldsymbol{x}_i) \right|$$

$$\leq \left| \sum_{i=t-\tau+1}^{t} \beta_1^{t-\tau-i} \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| + \sum_{i=t-\tau+1}^{t} \beta_1^{t-\tau-i} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) - \frac{\partial F}{\partial x_j}(\boldsymbol{x}_i) \right|$$

$$\leq \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \frac{\beta_1^{-\tau}}{1-\beta_1} + \left( L_{0,j} + L_{1,j}\sqrt{d} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \right) \frac{\eta\beta_1^{-\tau}}{(1-\beta_1)^2\sqrt{1-\beta_2}} .$$

Finally, the second term of (4.4) can be bounded using Lemma 4.9 by noticing that

$$\left| \sum_{i=t-\tau+1}^{t} \beta_1^{t-\tau-i} \tilde{\epsilon}_{i,j} \right| \leq \beta_1^{-\tau} \left( \left| \sum_{i=1}^{t} \beta_1^{t-i} \tilde{\epsilon}_{i,j} \right| + \left| \sum_{i=1}^{t-\tau} \beta_1^{t-i} \tilde{\epsilon}_{i,j} \right| \right) . \qquad \square$$

The following Lemma is adapted from (Zou et al., 2021). Yet, they only considered Adam under the $L$-smooth setting and when there is no noise. The existence of noise and the relaxed smoothness assumption makes the proofs substantially more

challenging. With this lemma, we know that either the true gradient is small or that the update of our Algorithm 4.1 can be lower bounded.

**Lemma 4.12.** *With the notation of Algorithm 4.1 and under the assumptions of Theorem 4.4, if $\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_\tau)\right| \leq M_j$ holds for all $\tau \leq t$ and $j \in [d]$, and $D > 0$, then, with probability at least $1 - 3t\delta$ we have that,*

$$either \left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right| < \frac{5B_j}{D} \ or \ \frac{|m_{t,j}|}{\sqrt{v_{t,j}}} \geq \frac{\rho D}{5\sqrt{1 - \beta_2}} \ .$$

*Proof of Lemma 4.12.* Given that $\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_\tau)\right| \leq M_j$ for any $\tau \leq t$ and $j \in [d]$, using Lemma 4.6 and Assumption 4.3, it is immediate to show that $|m_{t,j}| \leq M_j + \sigma_j$. Then, denote $\hat{\tau} = \lfloor \bar{\tau} \rfloor = \left\lfloor \frac{\sqrt{1 - \beta_2}}{\eta \sqrt{d} \|\boldsymbol{L_1}\|_\infty} \right\rfloor$ namely the largest integer that is no greater than $\bar{\tau}$, from Lemma 4.6, we have

$$
\begin{aligned}
\frac{|m_{t,j}|}{\sqrt{v_{t,j}}} &= \frac{|m_{t,j}|}{\sqrt{(1 - \beta_2) \sum_{\tau=0}^{t-1} \beta_2^\tau m_{t-\tau,j}^2}} \\
&= \frac{|m_{t,j}|}{\sqrt{1 - \beta_2} \sqrt{\sum_{\tau=\hat{\tau}+1}^{t-1} \beta_2^\tau m_{t-\tau,j}^2 + \sum_{\tau=0}^{\hat{\tau}} \beta_2^\tau m_{t-\tau,j}^2}} \\
&\geq \frac{|m_{t,j}|}{\sqrt{1 - \beta_2} \sqrt{(M_j + \sigma_j)^2 \frac{\beta_2^{\hat{\tau}+1}}{1 - \beta_2} + \sum_{\tau=0}^{\hat{\tau}} \beta_2^\tau m_{t-\tau,j}^2}} \\
&\geq \frac{|m_{t,j}|}{(M_j + \sigma_j)\beta_2^{\bar{\tau}/2} + \sqrt{1 - \beta_2} \sum_{\tau=0}^{\hat{\tau}} \beta_2^{\tau/2} |m_{t-\tau,j}|} \\
&> \frac{|m_{t,j}|}{(M_j + \sigma_j)\beta_1^{\bar{\tau}} + \sqrt{1 - \beta_2} \sum_{\tau=0}^{\hat{\tau}} \beta_2^{\tau/2} |m_{t-\tau,j}|},
\end{aligned}
$$

where the final inequality uses the assumption that $\sqrt{\beta_2} < \beta_1$. Using Lemma 4.11 and the definition of $\rho = 1 - \beta_2^{1/2} \beta_1^{-1} \in (0, 1]$, with probability at least $1 - 3t\delta$, as we

need to invoke Lemma 4.9 for at most $t$ times, we have

$$\frac{\sqrt{v_{t,j}}}{\sqrt{1-\beta_2}}$$

$$\leq (|m_{t,j}| + (1-\beta_1)E_j) \sum_{\tau=0}^{\hat{\tau}} \beta_2^{\tau/2}\beta_1^{-\tau} + \frac{\beta_1^{\bar{\tau}}(M_j + \sigma_j)}{\sqrt{1-\beta_2}}$$

$$+ \left( \left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right| + \left(L_{0,j} + L_{1,j}\sqrt{d}\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right|\right) \right) \frac{\eta}{(1-\beta_1)\sqrt{1-\beta_2}} \right) \sum_{\tau=0}^{\hat{\tau}} \beta_2^{\tau/2}\beta_1^{-\tau}$$

$$\leq \left( |m_{t,j}| + C_j \left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right| + B_j \right) \frac{1}{\rho},$$

where in the last inequality we used the fact that $\frac{1}{\rho} \geq \frac{1}{\sqrt{1-\beta_2}}$.

Thus, we consider following two cases depending on the relative size of $|m_{t,j}|$ vs. $C_j \left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right| + B_j$.

**Case 1**: $|m_{t,j}| > C_j \left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right| + B_j$, then

$$\frac{|m_{t,j}|}{\sqrt{v_{t,j}}} > \frac{\rho}{2\sqrt{1-\beta_2}} \ . \tag{4.5}$$

**Case 2**: $|m_{t,j}| \leq C_j \left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right| + B_j$, then we have

$$\sqrt{v_{t,j}} \leq \frac{2\sqrt{1-\beta_2}}{\rho} \left( C_j \left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right| + B_j \right) \ .$$

Also, for $|m_{t,j}|$ we have from Lemma 4.6 that

$$|m_{t,j}| = (1-\beta_1) \left|\sum_{\tau=1}^{t} \beta_1^{t-\tau} g_{\tau,j}\right| \geq (1-\beta_1) \left|\sum_{\tau=t-\hat{\tau}}^{t} \beta_1^{t-\tau} g_{\tau,j}\right| - (1-\beta_1) \left|\sum_{\tau=1}^{t-\hat{\tau}-1} \beta_1^{t-\tau} g_{\tau,j}\right|$$

$$\geq \underbrace{(1-\beta_1) \left|\sum_{\tau=t-\hat{\tau}}^{t} \beta_1^{t-\tau} \frac{\partial F}{\partial x_j}(\boldsymbol{x}_\tau)\right|}_{R_1} - \underbrace{(1-\beta_1) \left|\sum_{\tau=t-\hat{\tau}}^{t} \beta_1^{t-\tau} \left(\frac{\partial F}{\partial x_j}(\boldsymbol{x}_\tau) - g_{\tau,j}\right)\right|}_{R_2}$$

$$- (1 - \beta_1) \underbrace{\left| \sum_{\tau=1}^{t-\hat{\tau}-1} \beta_1^{t-\tau} g_{\tau,j} \right|}_{R_3} \; .$$

The first term can be bounded below by using Lemma 4.10 and that $\hat{\tau} + 1 \geq \bar{\tau}$:

$$R_1 \geq (1 - \beta_1) \left| \sum_{\tau=t-\hat{\tau}}^{t} \beta_1^{t-\tau} \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| - (1 - \beta_1) \left| \sum_{\tau=t-\hat{\tau}}^{t} \beta_1^{t-\tau} \left( \frac{\partial F}{\partial x_j}(\boldsymbol{x}_\tau) - \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right) \right|$$

$$\geq \left( 1 - \beta_1^{\bar{\tau}} - \frac{\sqrt{d} L_{1,j} \eta}{(1 - \beta_1)\sqrt{1 - \beta_2}} \right) \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| - \frac{L_{0,j} \eta}{(1 - \beta_1)\sqrt{1 - \beta_2}} \; .$$

The second term can be bounded using Lemma 4.9. Thus,

$$|m_{t,j}| \geq \left( 1 - \beta_1^{\bar{\tau}} - \frac{\sqrt{d} L_{1,j} \eta}{(1 - \beta_1)\sqrt{1 - \beta_2}} \right) \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| - \frac{L_{0,j} \eta}{(1 - \beta_1)\sqrt{1 - \beta_2}}$$

$$- \beta_1^{\bar{\tau}}(M_j + \sigma_j) - (1 - \beta_1)E_j$$

$$\geq D \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| - B_j \; ,$$

where we used (4.3) and that $e^{-x} \leq \frac{1}{x}$ for $x > 0$.

Therefore, with probability at least $1 - 3t\delta$ we have

$$\frac{|m_{t,j}|}{\sqrt{v_{t,j}}} \geq \frac{\rho \left( D \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| - B_j \right)}{2\sqrt{1 - \beta_2} \left( C_j \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| + B_j \right)} \; . \tag{4.6}$$

Given that $D > 0$, depending on the relative size of $\left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right|$ vs. $B_j$, we have following two cases.

**Case 2.1**: $\left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| < \frac{5B_j}{D}$.

**Case 2.2**: $\left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \geq \frac{5B_j}{D}$, using the fact that the r.h.s of (4.6) is decreasing in

$B_j$, we have

$$\frac{|m_{t,j}|}{\sqrt{v_{t,j}}} \geq \frac{\frac{4D}{5}\rho \left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right|}{2\sqrt{1-\beta_2}\left(C_j + \frac{D}{5}\right)\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right|} \geq \frac{2\rho D}{5\sqrt{1-\beta_2}(C_j + D)} \geq \frac{\rho D}{5\sqrt{1-\beta_2}},$$

where in the last inequality we used the fact that $C_j + D \leq 2$. Note that $D < 1$ so the above lower bound is smaller than (4.5). □

The following two lemmas are for the special case of $\beta_2 = 0$.

**Lemma 4.13.** *With choices of parameters in Theorem 4.4, when $\beta_2 = 0$, we have* $\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|_2 = \eta\sqrt{d} \leq \frac{1}{\|\boldsymbol{L_1}\|_\infty}$.

*Proof of Lemma 4.13.* Using the fact that $\alpha \leq 1$ and the condition on $T$, we have

$$\eta = \frac{\sqrt{\Delta\alpha}}{\sqrt{\|\boldsymbol{L_0}\|_1}\sqrt{T}} \leq \frac{\sqrt{\Delta}}{\sqrt{\|\boldsymbol{L_0}\|_1}\sqrt{T}} \leq \frac{\sqrt{\Delta}}{\sqrt{\|\boldsymbol{L_0}\|_1}}\frac{\sqrt{\|\boldsymbol{L_0}\|_1}}{10\sqrt{d}\sqrt{\Delta}\|\boldsymbol{L_1}\|_\infty} \leq \frac{1}{\sqrt{d}\|\boldsymbol{L_1}\|_\infty} . \quad □$$

The following lemma is adapted from the proof of Theorem 2 in (Cutkosky and Mehta, 2021).

**Lemma 4.14.** *Under Assumptions 1.1, 4.2, and 4.3, using the settings of the hyperparameters in Theorem 4.4, denoting $\alpha = 1 - \beta_1$ and $\boldsymbol{\epsilon}_t = \boldsymbol{m}_t - \nabla F(\boldsymbol{x}_t)$, for all $t \geq 1$ and $j \in [d]$ we have, with probability at least $1 - 3\delta$,*

$$|\epsilon_{t+1,j}| \leq (1-\alpha)^t \left(\alpha\sigma_j + (1-\alpha)\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_1)\right|\right) + \frac{\eta L_{0,j}}{\alpha} + \alpha E_j$$
$$+ (1-\alpha)\eta\sqrt{d}L_{1,j}\sum_{\tau=0}^{t-1}(1-\alpha)^\tau\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_{t-\tau})\right| .$$

*Proof of Lemma 4.14.* Denote $\tilde{\boldsymbol{\epsilon}}_t = \boldsymbol{g}_t - \nabla F(\boldsymbol{x}_t)$ and $S_j(\boldsymbol{a}, \boldsymbol{b}) = \frac{\partial F}{\partial x_j}(\boldsymbol{a}) - \frac{\partial F}{\partial x_j}(\boldsymbol{b})$.

Then, from Assumption 4.2 and Lemma 4.13, for all $t \geq 1$ and all $j \in [d]$ we have

$$\boldsymbol{\epsilon}_1 = \alpha\tilde{\boldsymbol{\epsilon}}_1 - (1-\alpha)\nabla F(\boldsymbol{x}_1), \tag{4.7}$$

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|_2 \leq \frac{1}{\|\boldsymbol{L_1}\|_\infty}$$

$$\Rightarrow |S_j(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)| \leq \left( \frac{L_{0,j}}{\sqrt{d}} + L_{1,j} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \right) \|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|_2. \tag{4.8}$$

We can derive the following recursive formulation for any $t \geq 1$:

$$\begin{aligned} m_{t+1,j} &= (1-\alpha)m_{t,j} + \alpha g_{t+1,j} \\ &= (1-\alpha)\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) + (1-\alpha)\epsilon_{t,j} + \alpha\frac{\partial F}{\partial x_j}(\boldsymbol{x}_{t+1}) + \alpha\tilde{\epsilon}_{t+1,j} \\ &= \frac{\partial F}{\partial x_j}(\boldsymbol{x}_{t+1}) + (1-\alpha)S_j(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}) + (1-\alpha)\epsilon_{t,j} + \alpha\tilde{\epsilon}_{t+1,j}, \end{aligned}$$

which implies

$$\epsilon_{t+1,j} = (1-\alpha)\epsilon_{t,j} + (1-\alpha)S_j(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}) + \alpha\tilde{\epsilon}_{t+1,j}. \tag{4.9}$$

Unravel (4.9) from 1 to $t$ gives us

$$\epsilon_{t+1,j} = (1-\alpha)^t\epsilon_{1,j} + (1-\alpha)\sum_{\tau=0}^{t-1}(1-\alpha)^\tau S_j(\boldsymbol{x}_{t-\tau}, \boldsymbol{x}_{t+1-\tau}) + \alpha\sum_{\tau=0}^{t-1}(1-\alpha)^\tau\tilde{\epsilon}_{t+1-\tau,j}.$$

Take the absolute value of both sides, to obtain

$$\begin{aligned} |\epsilon_{t+1,j}| \leq &(1-\alpha)^t|\epsilon_{1,j}| + (1-\alpha)\sum_{\tau=0}^{t-1}(1-\alpha)^\tau|S_j(\boldsymbol{x}_{t-\tau}, \boldsymbol{x}_{t+1-\tau})| \\ &+ \alpha\left|\sum_{\tau=0}^{t-1}(1-\alpha)^\tau\tilde{\epsilon}_{t+1-\tau,j}\right| \\ \leq &(1-\alpha)^t|\epsilon_{1,j}| + \alpha E_j \end{aligned}$$

$$+ (1-\alpha)\sum_{\tau=0}^{t-1}(1-\alpha)^\tau\left(\frac{L_{0,j}}{\sqrt{d}} + L_{1,j}\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_{t-\tau})\right|\right)\|\boldsymbol{x}_{t+1-\tau} - \boldsymbol{x}_{t-\tau}\|_2$$

$$\leq (1-\alpha)^t|\epsilon_{1,j}| + (1-\alpha)\eta L_{0,j}\sum_{\tau=0}^{t-1}(1-\alpha)^\tau$$

$$+ (1-\alpha)\eta\sqrt{d}L_{1,j}\sum_{\tau=0}^{t-1}(1-\alpha)^\tau\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_{t-\tau})\right| + \alpha E_j$$

$$\leq (1-\alpha)^t\left(\alpha\sigma_j + (1-\alpha)\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_1)\right|\right) + \frac{\eta L_{0,j}}{\alpha}$$

$$+ (1-\alpha)\eta\sqrt{d}L_{1,j}\sum_{\tau=0}^{t-1}(1-\alpha)^\tau\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_{t-\tau})\right| + \alpha E_j \ ,$$

where the second inequality uses (4.8) and Lemma 4.9, the fourth and fifth inequalities use (4.8), and the final one is due to (4.7). □

*Proof of Theorem 4.4 for $\beta_2 = 0$.* From Lemma 4.13 we know that $\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|_2 \leq \frac{1}{\|\boldsymbol{L_1}\|_\infty}$ for all $t \in [T]$. Thus, we can apply Lemma 4.3 to have

$$F(\boldsymbol{x}_{t+1}) - F(\boldsymbol{x}_t)$$

$$\leq \langle\nabla F(\boldsymbol{x}_t), \boldsymbol{x}_{t+1} - \boldsymbol{x}_t\rangle + \sum_{j=1}^{d}\frac{\left(\frac{L_{0,j}}{\sqrt{d}} + L_{1,j}\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right|\right)\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|_2}{2}|x_{t+1,j} - x_{t,j}|$$

$$= \langle\nabla F(\boldsymbol{x}_t), -\eta\,\mathrm{sign}(\boldsymbol{m}_t)\rangle + \sum_{j=1}^{d}\frac{L_{0,j} + L_{1,j}\sqrt{d}\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right|}{2}\eta^2$$

$$= -\eta\|\nabla F(\boldsymbol{x}_t)\|_1 + \eta\langle\nabla F(\boldsymbol{x}_t), \mathrm{sign}(\nabla F(\boldsymbol{x}_t)) - \mathrm{sign}(\boldsymbol{m}_t)\rangle$$

$$+ \sum_{j=1}^{d}\frac{L_{0,j} + L_{1,j}\sqrt{d}\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right|}{2}\eta^2$$

$$= -\eta\|\nabla F(\boldsymbol{x}_t)\|_1 + 2\eta\sum_{j=1}^{d}\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right|\mathbb{I}\left[\mathrm{sign}\left(\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right) \neq \mathrm{sign}(m_{t,j})\right]$$

$$+ \sum_{j=1}^{d}\frac{L_{0,j} + L_{1,j}\sqrt{d}\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right|}{2}\eta^2 \ ,$$

where $\mathbb{I}(\cdot)$ is the indicator function and the first inequality uses Lemma 4.3,

Now, note that

$$\mathbb{I}\left[\text{sign}\left(\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right) \neq \text{sign}(m_{t,j})\right] \leq \mathbb{I}\left[\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) - m_{t,j}\right| \geq \left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right|\right]$$
$$\leq \frac{\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) - m_{t,j}\right|}{\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right|} .$$

Thus, denoting by $\boldsymbol{\epsilon}_t = \boldsymbol{m}_t - \nabla F(\boldsymbol{x}_t)$ gives

$$F(\boldsymbol{x}_{t+1}) - F(\boldsymbol{x}_t)$$
$$\leq -\eta\|\nabla F(\boldsymbol{x}_t)\|_1 + 2\eta\|\boldsymbol{\epsilon}_t\|_1 + \sum_{j=1}^{d} \frac{L_{0,j} + L_{1,j}\sqrt{d}\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right|}{2}\eta^2$$
$$= -\eta\|\nabla F(\boldsymbol{x}_t)\|_1 + 2\eta\|\boldsymbol{\epsilon}_t\|_1 + \frac{\|\boldsymbol{L_0}\|_1\eta^2}{2} + \frac{\eta^2\sqrt{d}}{2}\sum_{j=1}^{d} L_{1,j}\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right| .$$

Sum both sides over $t = 1, \ldots, T$, to have

$$F^* - F(\boldsymbol{x}_1) \leq -\eta\sum_{t=1}^{T}\|\nabla F(\boldsymbol{x}_t)\|_1 + 2\eta\sum_{t=1}^{T}\|\boldsymbol{\epsilon}_t\|_1 + \frac{\|\boldsymbol{L_0}\|_1\eta^2 T}{2}$$
$$+ \frac{\eta^2\sqrt{d}}{2}\sum_{t=1}^{T}\sum_{j=1}^{d}L_{1,j}\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right| . \tag{4.10}$$

Use Lemma 4.14 to bound each coordinate of $\sum_{t=1}^{T}\|\boldsymbol{\epsilon}_t\|_1$:

$$\sum_{t=0}^{T-1}|\epsilon_{t+1,j}| \leq \sum_{t=0}^{T-1}\left[(1-\alpha)^t\left(\alpha\sigma_j + (1-\alpha)\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_1)\right|\right) + \frac{\eta L_{0,j}}{\alpha} + \alpha E_j\right]$$
$$+ (1-\alpha)\eta\sqrt{d}L_{1,j}\sum_{t=0}^{T-1}\sum_{\tau=0}^{t-1}(1-\alpha)^\tau\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_{t-\tau})\right|$$
$$= \sigma_j + \frac{1}{\alpha}\left|\frac{\partial F}{\partial x_j}(\boldsymbol{x}_1)\right| + \frac{\eta L_{0,j}T}{\alpha} + \alpha E_j T$$

$$+ (1-\alpha)\eta\sqrt{d}L_{1,j} \sum_{t=1}^{T-1} \sum_{\tau'=1}^{t} (1-\alpha)^{t-\tau'} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_{\tau'}) \right|$$

$$= \sigma_j + \frac{1}{\alpha}\left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_1) \right| + \frac{\eta L_{0,j}T}{\alpha} + \alpha E_j T$$

$$+ (1-\alpha)\eta\sqrt{d}L_{1,j} \sum_{\tau'=1}^{T-1} \left( \sum_{t=\tau'}^{T-1} (1-\alpha)^t \right) (1-\alpha)^{-\tau'} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_{\tau'}) \right|$$

$$\leq \sigma_j + \frac{1}{\alpha}\left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_1) \right| + \frac{\eta L_{0,j}T}{\alpha} + \alpha E_j T$$

$$+ \frac{(1-\alpha)\eta\sqrt{d}L_{1,j}}{\alpha} \sum_{t=1}^{T-1} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \ .$$

The above one holds with probability at least $1 - 3T\delta$ as we invoked Lemma 4.14 for $T$ times which in turns means invoking Lemma 4.9 for $T$ times. Yet, the above inequality would need to sum from $j = 1$ to $d$, meaning in total we would invoke Lemma 4.9 for $dT$ times. Thus, following results hold with probability at least $1 - 3dT\delta$.

Now, put the above inequality back into (4.10) to have

$$F^* - F(\boldsymbol{x}_1)$$

$$\leq -\eta \sum_{t=1}^{T} \|\nabla F(\boldsymbol{x}_t)\|_1 + \frac{\|\boldsymbol{L_0}\|_1\eta^2 T}{2} + \frac{\eta^2\sqrt{d}}{2} \sum_{t=1}^{T}\sum_{j=1}^{d} L_{1,j}\left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| + 2\eta\sum_{j=1}^{d}\sigma_j$$

$$+ 2\eta \sum_{j=1}^{d} \left( \frac{1}{\alpha}\left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_1) \right| + \frac{\eta L_{0,j}T}{\alpha} + \alpha E_j T + \frac{(1-\alpha)\eta\sqrt{d}L_{1,j}}{\alpha} \sum_{t=1}^{T}\left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \right)$$

$$= -\eta \sum_{t=1}^{T} \|\nabla F(\boldsymbol{x}_t)\|_1 + \frac{\|\boldsymbol{L_0}\|_1\eta^2 T}{2} + \frac{\eta^2\sqrt{d}}{2} \sum_{t=1}^{T}\sum_{j=1}^{d} L_{1,j}\left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| + 2\eta\|\boldsymbol{\sigma}\|_1$$

$$+ \frac{2\eta}{\alpha}\|\nabla F(\boldsymbol{x}_1)\|_1 + \frac{2\eta^2\|\boldsymbol{L_0}\|_1 T}{\alpha} + 2\eta\alpha T \sum_{j=1}^{d} E_j + \frac{2\eta^2\sqrt{d}}{\alpha} \sum_{t=1}^{T}\sum_{j=1}^{d} L_{1,j}\left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right|$$

$$= -\eta \sum_{t=1}^{T} \|\nabla F(\boldsymbol{x}_t)\|_1 + 2\eta\|\boldsymbol{\sigma}\|_1 + \frac{2\eta}{\alpha}\|\nabla F(\boldsymbol{x}_1)\|_1 + 2\eta\alpha T \sum_{j=1}^{d} E_j$$

$$+ \left(\frac{1}{2} + \frac{2}{\alpha}\right) \|\boldsymbol{L_0}\|_1 \eta^2 T + \left(\frac{1}{2} + \frac{2}{\alpha}\right) \eta^2 \sqrt{d} \sum_{t=1}^{T} \sum_{j=1}^{d} L_{1,j} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| .$$

Now, using the definitions of $\eta$ and $\alpha$, and the conditions on $T$, we have

$$\left(\frac{1}{2} + \frac{2}{\alpha}\right) \eta^2 \sqrt{d} L_{1,j}$$

$$\leq \frac{\eta}{2} \left(1 + \frac{4}{\alpha}\right) \frac{\sqrt{d}\sqrt{\Delta\alpha}}{\sqrt{T}} \frac{\|\boldsymbol{L_1}\|_\infty}{\sqrt{\|\boldsymbol{L_0}\|_1}} \leq \frac{\eta}{2} \frac{5\sqrt{d}\sqrt{\Delta}}{\sqrt{\alpha T}} \frac{\|\boldsymbol{L_1}\|_\infty}{\sqrt{\|\boldsymbol{L_0}\|_1}}$$

$$= \frac{\eta}{2} \frac{5\sqrt{d}\sqrt{\Delta}}{\sqrt{T}} \frac{\|\boldsymbol{L_1}\|_\infty}{\sqrt{\|\boldsymbol{L_0}\|_1}} \cdot \max\left(\frac{\sqrt{\|\boldsymbol{\sigma}\|_1} T^{1/4}}{\|\boldsymbol{L_0}\|_1^{1/4}\Delta^{1/4}}, 1\right)$$

$$= \frac{\eta}{2} \max\left(\frac{5\sqrt{d}\sqrt{\|\boldsymbol{\sigma}\|_1}\|\boldsymbol{L_1}\|_\infty \Delta^{1/4}}{\|\boldsymbol{L_0}\|_1^{3/4} T^{1/4}}, \frac{5\sqrt{d}\sqrt{\Delta}}{\sqrt{T}} \frac{\|\boldsymbol{L_1}\|_\infty}{\sqrt{\|\boldsymbol{L_0}\|_1}}\right) \leq \frac{\eta}{2} .$$

Thus, we have

$$F^* - F(\boldsymbol{x}_1) \leq -\frac{\eta}{2} \sum_{t=1}^{T} \|\nabla F(\boldsymbol{x}_t)\|_1 + 2\eta\|\boldsymbol{\sigma}\|_1 + \frac{2\eta}{\alpha}\|\nabla F(\boldsymbol{x}_1)\|_1$$

$$+ \left(\frac{1}{2} + \frac{2}{\alpha}\right) \|\boldsymbol{L_0}\|_1 \eta^2 T + 2\eta\alpha T \sum_{j=1}^{d} E_j .$$

Divide both sides by $T$ and rearrange terms to give

$$\frac{1}{T} \sum_{t=1}^{T} \|\nabla F(\boldsymbol{x}_t)\|_1$$

$$\leq \frac{2}{\eta T}[F(\boldsymbol{x}_1) - F^*] + \frac{4}{T}\|\boldsymbol{\sigma}\|_1 + \frac{4}{\alpha T}\|\nabla F(\boldsymbol{x}_1)\|_1 + \frac{5}{\alpha}\|\boldsymbol{L_0}\|_1\eta$$

$$+ 24\|\boldsymbol{\sigma}\|_1 (\alpha \max(1, \log(1/\delta)) + \sqrt{\alpha}\sqrt{\max(1, \log(1/\delta))}) .$$

Now, we need to consider the following two cases:

1. $\|\boldsymbol{\sigma}\|_1 < \frac{\sqrt{\|\boldsymbol{L_0}\|_1}\sqrt{\Delta}}{\sqrt{T}}$: then $\alpha = 1$ and $\eta = \frac{\sqrt{\Delta}}{\sqrt{\|\boldsymbol{L_0}\|_1}\sqrt{T}}$

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla F(\boldsymbol{x}_t)\|_1$$

$$\leq \frac{2\sqrt{\|\boldsymbol{L_0}\|_1}}{\sqrt{\Delta}\sqrt{T}}[F(\boldsymbol{x}_1) - F^*] + \frac{5\|\boldsymbol{L_0}\|_1\sqrt{\Delta}}{\sqrt{\|\boldsymbol{L_0}\|_1}\sqrt{T}}$$

$$+ \frac{4\sqrt{\|\boldsymbol{L_0}\|_1}\sqrt{\Delta}}{T^{3/2}} + \frac{4}{T}\|\nabla F(\boldsymbol{x}_1)\|_1$$

$$+ \frac{24\sqrt{\|\boldsymbol{L_0}\|_1}\sqrt{\Delta}(\max(1,\log(1/\delta)) + \sqrt{\max(1,\log(1/\delta))})}{\sqrt{T}}$$

$$\leq \frac{59\max(1,\log(1/\delta))\sqrt{\|\boldsymbol{L_0}\|_1\Delta}}{\sqrt{T}} + \frac{4}{T}\|\nabla F(\boldsymbol{x}_1)\|_1 \ . \qquad (4.11)$$

2. $\|\boldsymbol{\sigma}\|_1 \geq \frac{\sqrt{\|\boldsymbol{L_0}\|_1\Delta}}{\sqrt{T}}$: then $\alpha = \frac{\sqrt{\|\boldsymbol{L_0}\|_1\Delta}}{\|\boldsymbol{\sigma}\|_1\sqrt{T}} \leq 1$ and $\eta = \frac{\Delta^{3/4}}{\|\boldsymbol{L_0}\|_1^{1/4}\sqrt{\|\boldsymbol{\sigma}\|_1}T^{3/4}}$

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla F(\boldsymbol{x}_t)\|_1 \leq \frac{2\|\boldsymbol{L_0}\|_1^{1/4}\sqrt{\|\boldsymbol{\sigma}\|_1}}{\Delta^{3/4}T^{1/4}}[F(\boldsymbol{x}_1) - F^*] + \frac{4\|\boldsymbol{\sigma}\|_1}{T}$$

$$+ \frac{4\|\boldsymbol{\sigma}\|_1}{\sqrt{\|\boldsymbol{L_0}\|_1\Delta T}}\|\nabla F(\boldsymbol{x}_1)\|_1$$

$$+ \frac{5\|\boldsymbol{L_0}\|_1^{1/4}\sqrt{\|\boldsymbol{\sigma}\|_1}\Delta^{1/4}}{T^{1/4}}$$

$$+ \frac{24\max(1,\log(1/\delta))\sqrt{\|\boldsymbol{L_0}\|_1\Delta}}{\sqrt{T}}$$

$$+ \frac{24\sqrt{\max(1,\log(1/\delta))}\|\boldsymbol{L_0}\|_1^{1/4}\sqrt{\|\boldsymbol{\sigma}\|_1}\Delta^{1/4}}{T^{1/4}}$$

$$\leq \frac{31\sqrt{\max(1,\log(1/\delta))}\|\boldsymbol{L_0}\|_1^{1/4}\Delta^{1/4}\sqrt{\|\boldsymbol{\sigma}\|_1}}{T^{1/4}}$$

$$+ \frac{24\max(1,\log(1/\delta))\sqrt{\|\boldsymbol{L_0}\|_1\Delta}}{\sqrt{T}}$$

$$+ \frac{4\|\boldsymbol{\sigma}\|_1\|\nabla F(\boldsymbol{x}_1)\|_1}{\sqrt{\|\boldsymbol{L_0}\|_1\Delta T}} + \frac{4\|\boldsymbol{\sigma}\|_1}{T} \ .$$

Put $\delta' = \frac{\delta}{3dT}$ concludes the proof. $\qquad\qquad\qquad \square$

*Proof of Theorem 4.4 for general $\beta_2$.* Note that when $\|\boldsymbol{\sigma}\|_1 \leq \frac{\sqrt{\|\boldsymbol{L_0}\|_1 \Delta}}{\sqrt{T}}$, $\alpha = 1$, $\beta_2 < \beta_1^2 = 0$. Then our Generalized SignSGD algorithm 4.1 reduces to the SignSGD algorithm, and thus has the same guarantee of (4.11). Therefore, we only consider the other case when $\|\boldsymbol{\sigma}\|_1 \geq \frac{\sqrt{\|\boldsymbol{L_0}\|_1 \Delta}}{\sqrt{T}}$.

Note that the only randomness comes from evaluating stochastic gradients. In the following proof, we will need to invoke Lemma 4.9 for $T$ times for each coordinate $j \in [d]$. Therefore, the following results hold with probability at least $1 - 3dT\delta$. For simplicity, we use the term "with high probability" later in the proof to denote this.

We derive the following quantity which will be used multiple times later:

$$\frac{\eta}{\alpha} = \frac{1}{\sqrt{\|\boldsymbol{L_0}\|_1}} \cdot \frac{\sqrt{\Delta}}{\sqrt{T}} \cdot \frac{1}{\sqrt{\alpha}} = \frac{1}{\sqrt{\|\boldsymbol{L_0}\|_1}} \cdot \frac{\sqrt{\Delta}}{\sqrt{T}} \cdot \frac{\|\boldsymbol{\sigma}\|_1^{1/2} T^{1/4}}{\|\boldsymbol{L_0}\|_1^{1/4} \Delta^{1/4}} = \frac{\|\boldsymbol{\sigma}\|_1^{1/2} \Delta^{1/4}}{\|\boldsymbol{L_0}\|_1^{3/4} T^{1/4}} \; . \quad (4.12)$$

First, from Lemma 4.12 we have $D = 1 - \frac{2\sqrt{d}\|\boldsymbol{L_1}\|_\infty \eta}{(1-\beta_1)\sqrt{1-\beta_2}}$. Then, from the choice of the hyperparameters, we have

$$\frac{\sqrt{d}\|\boldsymbol{L_1}\|_\infty \eta}{(1-\beta_1)\sqrt{1-\beta_2}} = \frac{\|\boldsymbol{L_1}\|_\infty}{\sqrt{1-\beta_2}} \cdot \frac{\sqrt{d}\|\boldsymbol{\sigma}\|_1^{1/2} \Delta^{1/4}}{\|\boldsymbol{L_0}\|_1^{3/4} T^{1/4}} \leq \frac{\rho}{10} \leq \frac{1}{10} \; .$$

Thus, we have $D \geq 1 - \frac{1}{5} \geq \frac{1}{2}$ and, as $\sqrt{\beta_2}/\beta_1 < 1$,

$$\frac{\rho D}{5\sqrt{1-\beta_2}} \geq \frac{\rho}{10\sqrt{1-\beta_2}} = A \; . \quad (4.13)$$

Also, for those coordinates with small gradients $\left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| < \frac{5B_j}{D} \leq 10B_j$, we have

$$\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \cdot (x_{t+1,j} - x_{t,j})$$

$$= -\eta \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \cdot \frac{m_{t,j}}{\sqrt{v_{t,j}}}$$

$$= -A\eta \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| + \eta \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \cdot \left( A - \text{sign}\left( \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right) \frac{m_{t,j}}{\sqrt{v_{t,j}}} \right)$$

$$\leq -A\eta \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| + 10B_j\eta \left( \frac{1}{\sqrt{1-\beta_2}} + A \right) . \tag{4.14}$$

We are now ready to prove the theorem. We will need to use Lemma 4.12, hence we first need to show that all past true gradients are bounded by $M_j$, namely that, for any $t$, $\left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_\tau) \right| \leq M_j$ holds for all $\tau \leq t$ and all $j \in [d]$. From the definition of $M_j$ stated in the theorem, in order to guarantee this, we only need to prove that $F(\boldsymbol{x}_\tau) \leq F(\boldsymbol{x}_1)$ for all $\tau \leq t$. We will prove this by induction.

For $t = 1$ the condition is trivially true.

We then assume that the condition holds for $t$ and prove based on this that it still holds for $t + 1$.

For those coordinates with $\left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \geq \frac{5B_j}{D}$, denote $\hat{\tau} = \lfloor \bar{\tau} \rfloor = \left\lfloor \frac{\sqrt{1-\beta_2}}{\eta\sqrt{d}\|\boldsymbol{L_1}\|_\infty} \right\rfloor$, with high probability, we have

$$\left| m_{t,j} - \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| = \left| (1-\beta_1) \sum_{\tau=1}^{t} \beta_1^{t-\tau} g_{\tau,j} - \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right|$$

$$\leq \left| (1-\beta_1) \sum_{\tau=1}^{t-\hat{\tau}-1} \beta_1^{t-\tau} g_{\tau,j} \right| + \left| (1-\beta_1) \sum_{\tau=t-\hat{\tau}}^{t} \beta_1^{t-\tau} g_{\tau,j} - \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right|$$

$$\leq \beta_1^{\bar{\tau}}(M_j + \sigma_j) + \beta_1^{\bar{\tau}} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right|$$

$$+ (1-\beta_1) \left| \sum_{\tau=t-\hat{\tau}}^{t} \beta_1^{t-\tau} \left( \frac{\partial F}{\partial x_j}(\boldsymbol{x}_\tau) - \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right) \right|$$

$$+ (1-\beta_1) \left| \sum_{\tau=t-\hat{\tau}}^{t} \beta_1^{t-\tau} \left( g_{\tau,j} - \frac{\partial F}{\partial x_j}(\boldsymbol{x}_\tau) \right) \right|$$

$$\leq (1-D) \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| + B_j$$

$$\leq \left( 1 - \frac{4D}{5} \right) \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \leq \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right|, \tag{4.15}$$

where the first equality comes from Lemma 4.6; for the second inequality, the third term can be bounded using Lemma 4.10, and the final term can be bounded using Lemma 4.9; for the third inequality, we used (4.3) and that $e^{-x} \leq \frac{1}{x}$ for $x > 0$. This inequality implies that $\text{sign}(m_{t,j}) = \text{sign}\left(\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t)\right)$ with high probability.

Denote $\mathcal{U}_t = \left\{ j \in [d] : \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \geq \frac{5B_j}{D} \right\}$. From the choices of hyperparameters we can show that $1 \leq \bar{\tau}$ which means $\|\boldsymbol{x}_{t+1} - x_t\|_2 \leq \frac{1}{\|\boldsymbol{L_1}\|_\infty}$ (Lemma 4.7). Thus, using Lemma 4.3, with high probability we have

$$
F(\boldsymbol{x}_{t+1}) - F(\boldsymbol{x}_t)
$$

$$
\leq \langle \nabla F(\boldsymbol{x}_t), \boldsymbol{x}_{t+1} - \boldsymbol{x}_t \rangle + \sum_{j=1}^{d} \frac{\left( \frac{L_{0,j}}{\sqrt{d}} + L_{1,j} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \right) \|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|_2}{2} |x_{t+1,j} - x_{t,j}|
$$

$$
= \sum_{j=1}^{d} \left( -\frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \cdot \eta \text{sign}(m_{t,j}) \frac{|m_{t,j}|}{\sqrt{v_{t,j}}} \right)
$$

$$
+ \sum_{j=1}^{d} \left( \frac{\left( \frac{L_{0,j}}{\sqrt{d}} + L_{1,j} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \right) \|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|_2}{2} |x_{t+1,j} - x_{t,j}| \right)
$$

$$
\leq \sum_{j \in \mathcal{U}_t} -\eta \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| \cdot \frac{|m_{t,j}|}{\sqrt{v_{t,j}}} + \sum_{j \notin \mathcal{U}_t} \left( -A\eta \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right| + 10\eta B_j \left( \frac{1}{\sqrt{1 - \beta_2}} + A \right) \right)
$$

$$
+ \sum_{j=1}^{d} \frac{L_{0,j} + L_{1,j}\sqrt{d} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right|}{2(1 - \beta_2)} \eta^2
$$

$$
\leq -A\eta \|\nabla F(x_t)\|_1 + \sum_{j=1}^{d} \frac{L_{0,j} + L_{1,j}\sqrt{d} \left| \frac{\partial F}{\partial x_j}(\boldsymbol{x}_t) \right|}{2(1 - \beta_2)} \eta^2
$$

$$
+ 10\eta \left( \frac{1}{\sqrt{1 - \beta_2}} + A \right) \sum_{j=1}^{d} B_j, \tag{4.16}
$$

where the second inequality uses (4.14), (4.15), and Lemma 4.6, and the third inequality uses Lemma 4.12 and (4.13).

Now, noticing the conditions on $\eta$, $\alpha$, $\beta_2 < \beta_1^2 < \beta_1$, and $T$, use (4.12) to have

$$\frac{\eta^2\sqrt{d}L_{1,j}}{2(1-\beta_2)} \leq \frac{\eta\sqrt{d}\|\boldsymbol{L_1}\|_\infty}{2}\cdot\frac{\eta}{\alpha} = \frac{\eta}{2}\frac{\sqrt{d}\|\boldsymbol{L_1}\|_\infty\|\boldsymbol{\sigma}\|_1^{1/2}\Delta^{1/4}}{\|\boldsymbol{L_0}\|_1^{3/4}T^{1/4}} \leq \frac{\eta}{2}\frac{\rho\sqrt{1-\beta_2}}{10} \leq \frac{\eta}{2}A \ .$$

Thus, (4.16) becomes

$$F(\boldsymbol{x}_{t+1}) - F(\boldsymbol{x}_t) \leq -\frac{A\eta}{2}\|\nabla F(x_t)\|_1 + \frac{\eta^2\|\boldsymbol{L_0}\|_1}{2(1-\beta_2)}$$

$$+ 10\eta\left(\frac{1}{\sqrt{1-\beta_2}} + A\right)\sum_{j=1}^{d} B_j \ . \tag{4.17}$$

Therefore, either $F(\boldsymbol{x}_{t+1}) - F(\boldsymbol{x}_t) \leq 0$ or

$$\|\nabla F(\boldsymbol{x}_t)\|_1 \leq \frac{\eta\|\boldsymbol{L_0}\|_1}{A(1-\beta_2)} + 20\left(\frac{1}{A\sqrt{1-\beta_2}} + 1\right)\sum_{j=1}^{d} B_j \ . \tag{4.18}$$

This concludes the mathematical induction up until (4.18) is met for the first time which we denote as $T_0$. In the following, we will explain that if (4.18) holds then the algorithm has found an approximate stationary point.

Now, suppose $T \leq T_0$, then (4.17) holds all the time and we sum both sides of it from 1 to $T$ to have, with high probability,

$$F^* - F(\boldsymbol{x}_1) \leq -\frac{A\eta}{2}\sum_{t=1}^{T}\|\nabla F(\boldsymbol{x}_t)\|_1 + \frac{\|\boldsymbol{L_0}\|_1\eta^2 T}{2(1-\beta_2)} + 10\eta T\left(\frac{1}{\sqrt{1-\beta_2}} + A\right)\sum_{j=1}^{d} B_j \ .$$

Rearrange terms to obtain

$$\min_{t\in[T]}\|\nabla F(\boldsymbol{x}_t)\|_1 \leq \frac{1}{T}\sum_{t=1}^{T}\|\nabla F(\boldsymbol{x}_t)\|_1 \leq \frac{2}{A\eta T}[F(\boldsymbol{x}_1) - F^*] + \frac{\eta\|\boldsymbol{L_0}\|_1}{A(1-\beta_2)}$$

$$+ 20\left(\frac{1}{A\sqrt{1-\beta_2}} + 1\right)\sum_{j=1}^{d} B_j \ . \tag{4.19}$$

Note that RHS of (4.18) is less than RHS of (4.19). Thus, for the other case of $T > T_0$, (4.19) still holds.

Recalling that $\rho = 1 - \frac{\sqrt{\beta_2}}{\beta_1}$, $A = \frac{\rho}{10\sqrt{1-\beta_2}}$, $\beta_2 < \beta_1^2 < \beta_1$, and $B_j \triangleq \frac{\eta L_{0,j}}{(1-\beta_1)\sqrt{1-\beta_2}} + \beta_1^{\bar{\tau}}(M_j + \sigma_j) + 6(1-\beta_1)\sigma_j \max(1, \log(1/\delta)) + \frac{6(1-\beta_1)}{\sqrt{1-\beta_1^2}}\sqrt{\sigma_j^2 \max(1, \log(1/\delta))}$, we have

$$
\begin{aligned}
\min_{t\in[T]} \|\nabla F(\boldsymbol{x}_t)\|_1 \leq & \frac{20}{\rho\eta T}[F(\boldsymbol{x}_1) - F^*] + \frac{10\eta\|\boldsymbol{L_0}\|_1}{\rho\sqrt{1-\beta_1}} \\
& + 20\left(\frac{10}{\rho}+1\right)\left(\frac{\|\boldsymbol{L_0}\|_1\eta}{(1-\beta_1)\sqrt{1-\beta_2}} + \beta_1^{\bar{\tau}}(\|\boldsymbol{M}\|_1 + \|\boldsymbol{\sigma}\|_1)\right) \\
& + 120\|\boldsymbol{\sigma}\|_1(1-\beta_1)\left(\frac{10}{\rho}+1\right)\max(1, \log(1/\delta)) \\
& + 120\|\boldsymbol{\sigma}\|_1(1-\beta_1)\left(\frac{10}{\rho}+1\right)\frac{1}{\sqrt{1-\beta_1^2}}\sqrt{\max(1, \log(1/\delta))} \ .
\end{aligned}
$$

When $\|\boldsymbol{\sigma}\|_1 \geq \frac{\sqrt{\|\boldsymbol{L_0}\|_1\Delta}}{\sqrt{T}}$, then $\alpha = \frac{\sqrt{\|\boldsymbol{L_0}\|_1\Delta}}{\|\boldsymbol{\sigma}\|_1\sqrt{T}}$ and $\eta = \frac{\Delta^{3/4}}{\|\boldsymbol{L_0}\|_1^{1/4}\sqrt{\|\boldsymbol{\sigma}\|_1}T^{3/4}}$. Hence,

$$
\begin{aligned}
& \min_{t\in[T]} \|\nabla F(\boldsymbol{x}_t)\|_1 \\
\leq & \frac{20\Delta^{1/4}\|\boldsymbol{L_0}\|_1^{1/4}\|\boldsymbol{\sigma}\|_1^{1/2}T^{3/4}}{\rho T} + \frac{10\sqrt{\|\boldsymbol{L_0}\|_1\Delta}}{\rho\sqrt{T}} \\
& + 20\left(\frac{10}{\rho}+1\right)\left(\frac{\Delta^{1/4}\|\boldsymbol{\sigma}\|_1^{1/2}\|\boldsymbol{L_0}\|_1}{\sqrt{1-\beta_2}\|\boldsymbol{L_0}\|_1^{3/4}T^{1/4}} + \beta_1^{\bar{\tau}}(\|\boldsymbol{M}\|_1 + \|\boldsymbol{\sigma}\|_1)\right) \\
& + 120\left(\frac{10}{\rho}+1\right)\frac{\sqrt{\|\boldsymbol{L_0}\|_1\Delta}}{\sqrt{T}}\max(1, \log(1/\delta)) \\
& + 120\left(\frac{10}{\rho}+1\right)\frac{\Delta^{1/4}\|\boldsymbol{L_0}\|_1^{1/4}\|\boldsymbol{\sigma}\|_1^{1/2}}{T^{1/4}}\sqrt{\max(1, \log(1/\delta))} \\
\leq & \left(\frac{20}{\rho T^{1/4}} + \left(\frac{10}{\rho}+1\right)\frac{20 + 120\sqrt{\max(1, \log(1/\delta))}}{\sqrt{1-\beta_2}T^{1/4}}\right)\Delta^{1/4}\|\boldsymbol{L_0}\|_1^{1/4}\|\boldsymbol{\sigma}\|_1^{1/2} \\
& + \left(\frac{10}{\rho} + 120\left(\frac{10}{\rho}+1\right)\max(1, \log(1/\delta))\right)\frac{\sqrt{\|\boldsymbol{L_0}\|_1\Delta}}{\sqrt{T}} \\
& + 20\left(\frac{10}{\rho}+1\right)\beta_1^{\bar{\tau}}(\|\boldsymbol{M}\|_1 + \|\boldsymbol{\sigma}\|_1)
\end{aligned}
$$

$$\leq \frac{1560\Delta^{1/4}\|\boldsymbol{L_0}\|_1^{1/4}\|\boldsymbol{\sigma}\|_1^{1/2}\sqrt{\max(1,\log(1/\delta))}}{\rho\sqrt{1-\beta_2}T^{1/4}}$$

$$+ \frac{1330\max(1,\log(1/\delta))\sqrt{\|\boldsymbol{L_0}\|_1\Delta}}{\rho\sqrt{T}}$$

$$+ \frac{220}{\rho}(\|\boldsymbol{M}\|_1 + \|\boldsymbol{\sigma}\|_1)\exp\left(-\frac{\sqrt{1-\beta_2}\|\boldsymbol{L_0}\|_1^{3/4}}{\sqrt{d}\|\boldsymbol{L_1}\|_\infty\|\boldsymbol{\sigma}\|_1^{1/2}\Delta^{1/4}}T^{1/4}\right) .$$

Finally, taking $\delta' = \frac{\delta}{3dT}$, we obtain the stated result. $\qquad\square$

## 4.5 Experiments Comparing our Generalized SignSGD with Others

To validate the efficacy of our Algorithm 4.1, we compare it with Adam (Kingma and Ba, 2015), SGD (Robbins and Monro, 1951), SGDClipGrad, and SGDClipMomentum. The latter two are from Algorithm 1 in (Zhang et al., 2020a) where SGDClipGrad corresponds to the case where $\nu = 0$ and SGDClipMomentum corresponds to the case when $\nu = 1$.

**Training** Unless otherwise specified, we use grid-search to fine-tune the initial step size for all optimizers, as well as the clipping threshold for SGDClipGrad and SGDClipMomentum, and $\beta_2$ for Adam and our algorithm, to select the one giving the best validation performance on a separated validation set. We then employ the best performing hyperparameters to train the model over all training data and report the testing performance. The testing is repeated with random seeds 5 times to eliminate the influence of stochasticity.

**Hyperparameter Tuning** During the validation stage, we used grid-search to fine-tune respective hyperparameters and choose the ones that yield the best validation results. We tuned the hyperparameters using the following two-stage grid searching strategy: First, search over a coarse grid, and select the one yielding the

113

**Table 4.1:** Hyperparameter grid search ranges and choices yielding the highest validation accuracy for each optimizer for training a 20-layer Resnet to do image classification on CIFAR-10. ("lr" denotes the initial step size, "clip" denotes the clipping parameter $\gamma$ in Algorithm 1 of Zhang et al. (2020a), and "$\beta_2$" is defined in Kingma and Ba (2015) for Adam and in Algorithm 4.1 for ours.)

| Optimizer | Grid Search Range | Best Choice |
|---|---|---|
| SGD Momentum | lr {1e-5, 0.0001, 0.001, 0.01, 0.05, 0.07, 0.1, 0.2, 0.3, 1, 10} | lr=0.07 |
| SGDClipGrad | lr {0.001, 0.01, 0.05, 0.1, 0.5, 1, 10} | lr=0.5 |
| | clip {0.1, 1, 10} | clip=1 |
| SGDClipMomentum | lr {0.001, 0.01, 0.1, 1, 5, 10, 20, 50} | lr=10 |
| | clip {0.01, 0.1, 1, 10} | clip=0.1 |
| Adam | lr {1e-5, 0.0001, 0.0007, 0.0009, 0.001, 0.002, 0.003, 0.01, 0.1} | lr=0.0009 |
| | $\beta_2$ {0.4, 0.8, 0.999} | $\beta_2$=0.999 |
| Our Algorithm 4.1 | lr {5e-5, 8e-5, 0.0001, 0.0002, 0.0005, 0.001, 0.01} | lr=0.0002 |
| | $\beta_2$ {0.4, 0.8, 0.999} | $\beta_2 = 0.999$ |
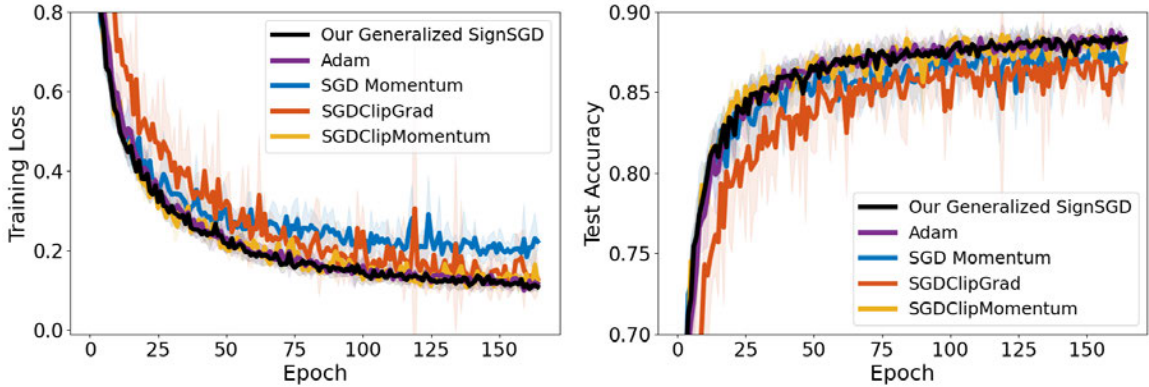


**Figure 4·4:** Training a 20-layer Resnet on CIFAR10. The shading of each curve represents the 95% confidence interval computed across 5 independent runs from different random seeds.

best validation result. Next, continue searching in a fine grid centering at the best-performing hyperparameters found in the coarse stage, and in turn, take the best one as the final choice. Also, whenever the best-performing hyperparameters lie in the boundary of the searching grid, we always extend the grid to make the final best-performing hyperparameters fall into the interior of the grid, if possible.

**Resnet for Image Classification on CIFAR-10** We employ the 20-layer Residual Network model (He et al., 2016) to do image classification on the CIFAR-10 dataset. Images are normalized per channel using the means and standard deviations computed from all training images. We adopt the data augmentation technique

**Table 4.2:** Hyperparameter grid search ranges and choices yielding the lowest validation perplexity for each optimizer on training an AWD-LSTM to do language modeling on Penn Treebank. ("wd" denotes the weight decay value, "lr" denotes the initial step size, "clip" denotes the clipping parameter $\gamma$ in Algorithm 1 of Zhang et al. (2020a), and "$\beta_2$" is defined in Kingma and Ba (2015) for Adam and in Algorithm 4.1 for ours.)

| Optimizer | Grid Search Range | Best Choice |
|---|---|---|
| SGD Momentum | wd {1e-7, 1.2e-6, 5e-6, 1e-5, 1e-4, 1e-3} | wd=1e-5 |
| | lr {0.001, 0.01, 0.1, 0.5, 0.8, 1, 2, 4, 5} | lr=1 |
| SGDClipGrad | wd {1e-7, 1.2e-6, 5e-6, 1e-5} | wd=1.2e-6 |
| | lr {0.1, 0.5, 1, 5, 10, 20, 30, 40, 50, 60, 70} | lr=50 |
| | clip {1, 2.5, 7.5, 10, 15, 20} | clip=10 |
| SGDClipMomentum | wd {1e-7, 1.2e-6, 5e-6, 1e-5} | wd=1.2e-6 |
| | lr {5, 10, 20, 30, 50, 100} | lr=20 |
| | clip {1, 2.5, 7.5} | clip=2.5 |
| Adam | wd {1e-7, 1.2e-6, 5e-6, 1e-5} | wd=5e-6 |
| | lr {0.0001, 0.001, 0.002, 0.003, 0.01, 0.1} | lr=0.002 |
| | $\beta_2$ {0.4, 0.8, 0.999} | $\beta_2$=0.999 |
| Our Algorithm 4.1 | wd {1e-7, 1.2e-6, 5e-6, 1e-5} | wd=1.2e-6 |
| | lr {0.0001, 0.001, 0.002, 0.003, 0.01, 0.1} | lr=0.001 |
| | $\beta_2$ {0.4, 0.8, 0.999} | $\beta_2$=0.999 |

following (Lee et al., 2015) (for training only): 4 pixels are padded on each side of an image and a $32 \times 32$ crop is randomly sampled from the padded image or its horizontal flip. The mini-batch size is 128 and we train all algorithms for 164 epochs. We do not employ any step size decay schedule in order to focus on the comparison of the optimizers themselves. We fixed the weight decay value to be 0.0001 and the momentum parameter ($\beta_1$) to be 0.9. We randomly selected 10% images from the training dataset for validation. Yet, during testing, we trained on the whole training dataset. The detailed search ranges and the hyperparameter choices yielding the highest validation accuracy for each optimizer are listed in Table 4.1. Figure 4·4 and Table 4.3 report the training and testing performance for each algorithm, showing that our algorithm closely matches Adam and is among the best of all.

**LSTM for Language Modeling on Penn Treebank** We adopt a 3-layer AWD-
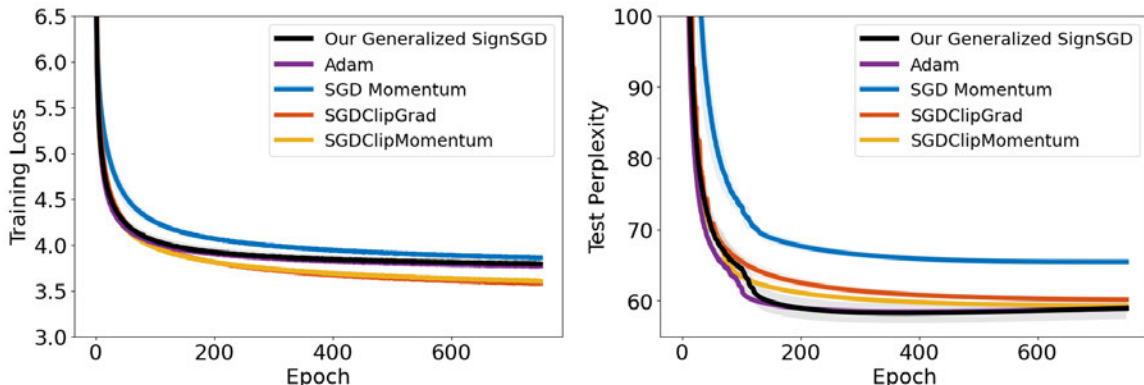
**Figure 4·5:** Training an AWD-LSTM model to do language modeling (word level) on Penn Treebank. The shading of each curve represents the 95% confidence interval computed across five independent runs from different random seeds.

**Table 4.3:** Average final training loss and test accuracy achieved by each method when optimizing respective models on each dataset. The ± shows 95% confidence intervals of the mean loss/accuracy/perplexity value over 5 runs starting from different random seeds.

| Methods | CIFAR10 | | Penn Treebank | |
|---|---|---|---|---|
| | Training loss | Test accuracy | Training loss | Test perplexity |
| SGD Momentum | $0.2226 \pm 0.0169$ | $0.8674 \pm 0.0048$ | $3.8587 \pm 0.0058$ | $65.4622 \pm 0.3842$ |
| SGDClipGrad | $0.1288 \pm 0.0403$ | $0.8677 \pm 0.0106$ | $\mathbf{3.5774 \pm 0.0081}$ | $60.1604 \pm 0.2797$ |
| SGDClipMomentum | $0.1220 \pm 0.0162$ | $0.8809 \pm 0.0022$ | $3.6038 \pm 0.0102$ | $59.3052 \pm 0.2798$ |
| Adam | $0.1161 \pm 0.0111$ | $0.8823 \pm 0.0041$ | $3.7692 \pm 0.0062$ | $\mathbf{58.9005 \pm 0.3058}$ |
| Our Algorithm 4.1 | $\mathbf{0.1086 \pm 0.0129}$ | $\mathbf{0.8835 \pm 0.0032}$ | $3.7928 \pm 0.0425$ | $58.9661 \pm 1.5218$ |

LSTM (Merity et al., 2018) to do language modeling on the Penn Treebank (PTB) dataset (Marcus et al., 1993)(word level). The mini-batch size is 40 and we trained each algorithm for 750 epochs. We used the original train-validation-test split that comes with the dataset. Apart from the hyperparameters we stated above, we fixed The momentum parameter ($\beta_1$) to be 0.9 except for SGDClipGrad which does not use momentum, and further fine-tuned the weight decay value for all algorithms as we noticed its significant influence on the performance. We choose the set of hyperparameters that give the smallest final validation perplexity. The detailed search ranges and the hyperparameter choices yielding the lowest validation perplexity for each optimizer are listed in Table 4.2. We report the results in Figure 4·5 and Table 4.3. It can be seen that we can match the performance of Adam while beating the others.

## 4.6  Conclusion

Smoothness has been a widely adopted condition for proving convergence rates of algorithms in the non-convex optimization scenario. Yet, it has been found that this assumption does not capture losses when employing some deep learning models including RNNs and LSTMs. In light of this, a relaxed smoothness assumption was proposed that aligns well with the practice. We observed that the loss surface of training using Transformers also exhibits this relaxed smoothness. Under this assumption, SGD with clipped gradient has been proven to work well. However, we found that clipping is not necessary for achieving convergence in such a setting. Indeed, we showed that a generalized SignSGD algorithm does not require explicit clipping but can almost guarantee the same bound as SGD with clipping. In the analyses, we identified the key effect of using momentum in analyzing Adam-type algorithms, that it reduces both the noise and the unbounded gradient norms. Finally, we conducted a variety of deep learning tasks showing that our algorithm can match Adam's performance while exceeding others.

**Limitations** The current work is in no way a perfect one and there are many directions worth exploring beyond it. First of all, though our algorithm could be seen as a close resemblance to the original Adam algorithm, they are still not equal. Considering the huge popularity of Adam and its established effectivity in practice, it is worth studying whether Adam in its original form can converge in the relaxed smooth setting. Second, while our Theorem 4.4 are upper bounds and cannot be directly compared between the two cases of $\beta_2$, it does suggests that $\beta_2 = 0$ minimizes the worst-case convergence rate. However, it still does not fully explain the phenomenon that a choice of $\beta_2$ close to 1 yields better performance in using our Algorithm 4.1 as well as Adam in practice. Third, despite there are lower bounds showing

that, for example, GD with a constant step size can be arbitrarily worse than GD with clipping, it would be more meaningful to study whether the relaxed smooth condition is inherently more difficult, possibly by establishing a lower bound for all first-order optimization algorithms. Fourth, we did show that Transformers observe the relaxed smoothness condition, but we consider it more beneficial to research in-depth what properties or structures make a model satisfy such conditions. Finally, when conducting our experiments, we observed that the weight decay value plays a prominent role in each optimizer's performance, and that the best weight decay value varies for different optimizers. Thus, one potential direction would be to explore different ways of incorporating the regularization in a way to preserve the scale-freeness (Orabona and Pál, 2015, 2018) of Algorithm 4.1, just as AdamW does (Zhuang et al., 2022).

# Chapter 5

# Conclusions

Non-convex optimization problems have been attracting much attention in recent years, especially with the successes of deep neural networks. A dominant optimization algorithm for such a scenario is gradient descent/stochastic gradient descent which necessitates specifying a parameter called the step size. This parameter plays a critical role in the performance of GD/SGD and often requires very careful tuning for each problem individually. The tuning is notoriously tedious and time and resources consuming. To ease this burden, adaptive algorithms are proposed which can automatically guarantee near-optimal convergence even without knowledge of certain properties of the objective problem.

In this dissertation, we discussed our work on studying adaptive strategies in non-convex optimization in three scenarios. We first designed and analyzed algorithms that can adapt to the level of noise in evaluating stochastic gradients in the general smooth non-convex setting and the setting with an additional PL condition. We then addressed the scenario when gradient scales can vary drastically across layers/coordinates, identified scenes when Adam performs worse than AdamW, and unearthed the correlation between AdamW's advantage and its scale-freeness property which makes it adaptive to the gradient scales. Finally, we attacked the relaxed smoothness setting by reporting empirical evidence showing that Transformers observe such a condition and introducing a generalized SignSGD algorithm that incor-

porates the empirical excellence of Adam and the theoretical merits of SGD with gradient clipping including the adaptivity to the smoothness parameter.

Nevertheless, there are still abundant directions for studying adaptivity in non-convex optimization that is yet to be explored. We believe the progress in this topic of adaptivity would be really beneficial to the field of non-convex optimization as adaptivity means less tuning needed which leads to faster exploration and iteration on making novel findings and on applying existing methods to new tasks.

# Appendix A

# Supporting Materials

## A.1 Omitted Proofs of Lemmas and Theorems

*Proof of Lemma 1.1.* We will prove by contradiction. Suppose there exists a global minimum point $\boldsymbol{x}^*$ and a local minimum point $\tilde{\boldsymbol{x}}$ with $F(\tilde{\boldsymbol{x}}) > F(\boldsymbol{x}^*)$.

From the definition of a local minimum point, there exists a neighborhood $\mathcal{N} \subset \mathcal{X}$ around $\tilde{\boldsymbol{x}}$ such that $F(\boldsymbol{x}) \geq F(\tilde{\boldsymbol{x}})$ for any $\boldsymbol{x} \in \mathcal{N}$.

Consider a point $\boldsymbol{y} = \theta\tilde{\boldsymbol{x}} + (1-\theta)\boldsymbol{x}^*$ for some $\theta$ with $0 \leq \theta \leq 1$. From the definition of a convex function, $\mathcal{X}$ is a convex set thus $\boldsymbol{y} \in \mathcal{X}$. Also, we have

$$F(\boldsymbol{y}) \leq \theta F(\tilde{\boldsymbol{x}}) + (1-\theta)F(\boldsymbol{x}^*) < \theta F(\tilde{\boldsymbol{x}}) + (1-\theta)F(\tilde{\boldsymbol{x}}) = F(\tilde{\boldsymbol{x}}) \ .$$

Pick $\theta$ to be sufficiently close to 1 such that $\boldsymbol{y} \in \mathcal{N}$, we get a contradiction. $\square$

*Proof of Theorem 1.3.*

$$
\begin{aligned}
F(\bar{\boldsymbol{x}}) - F(\boldsymbol{x}^*) &= F\left(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t\right) - F(\boldsymbol{x}^*) \\
&\leq \frac{1}{T}\sum_{t=1}^{T}F\left(\boldsymbol{x}_t\right) - F(\boldsymbol{x}^*) \\
&\leq \frac{1}{T}\sum_{t=1}^{T}\langle\nabla F(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{x}^*\rangle
\end{aligned}
$$

$$\begin{aligned}
&= \sum_{t=1}^{T} \frac{\|\boldsymbol{x}_t - \boldsymbol{x}^*\|_2^2 - \|\boldsymbol{x}_t - \eta\nabla F(\boldsymbol{x}_t) - \boldsymbol{x}^*\|_2^2 + \eta^2\|\nabla F(\boldsymbol{x}_t)\|_2^2}{2\eta T} \\
&\leq \sum_{t=1}^{T} \frac{\|\boldsymbol{x}_t - \boldsymbol{x}^*\|_2^2 - \|\boldsymbol{x}_{t+1} - \boldsymbol{x}^*\|_2^2 + \eta^2\|\nabla F(\boldsymbol{x}_t)\|_2^2}{2\eta T} \\
&= \frac{\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|_2^2 - \|\boldsymbol{x}_{T+1} - \boldsymbol{x}^*\|_2^2}{2\eta T} + \frac{\eta}{2T}\sum_{t=1}^{T}\|\nabla F(\boldsymbol{x}_t)\|_2^2 \\
&\leq \frac{D^2}{2\eta T} + \frac{\eta}{2T}\sum_{t=1}^{T}\|\nabla F(\boldsymbol{x}_t)\|_2^2,
\end{aligned}$$

where the first inequality uses the Jensen's inequality for convex functions (Boyd and Vandenberghe, 2004, Section 3.1.8), the second inequality uses the convexity of $F$, the third one uses the projection lemma (Shalev-Shwartz and Ben-David, 2014, Lemma 14.9), and the last one uses the bounded domain assumption. □

## A.2 The Histograms of Update Scales of each Coordinate during the Entire Training Phase of Adam vs. AdamW

In this section, we report the histograms of the absolute value of updates of Adam-$\ell_2$ vs. AdamW of all coordinates divided by the initial step size $\alpha$ during the whole training process. From the figures shown below, we can clearly see that AdamW's updates remain in a much more concentrated scale range than Adam-$\ell_2$ during the entire training. Moreover, as the depth of the network grows, Adam-$\ell_2$'s updates become more and more dispersed, while AdamW's updates are still concentrated. *(Note that the leftmost bin contains all values equal to or less than $2^{-27} \approx 10^{-8.1}$ and the rightmost bin contains all values equal to or larger than 1.)*
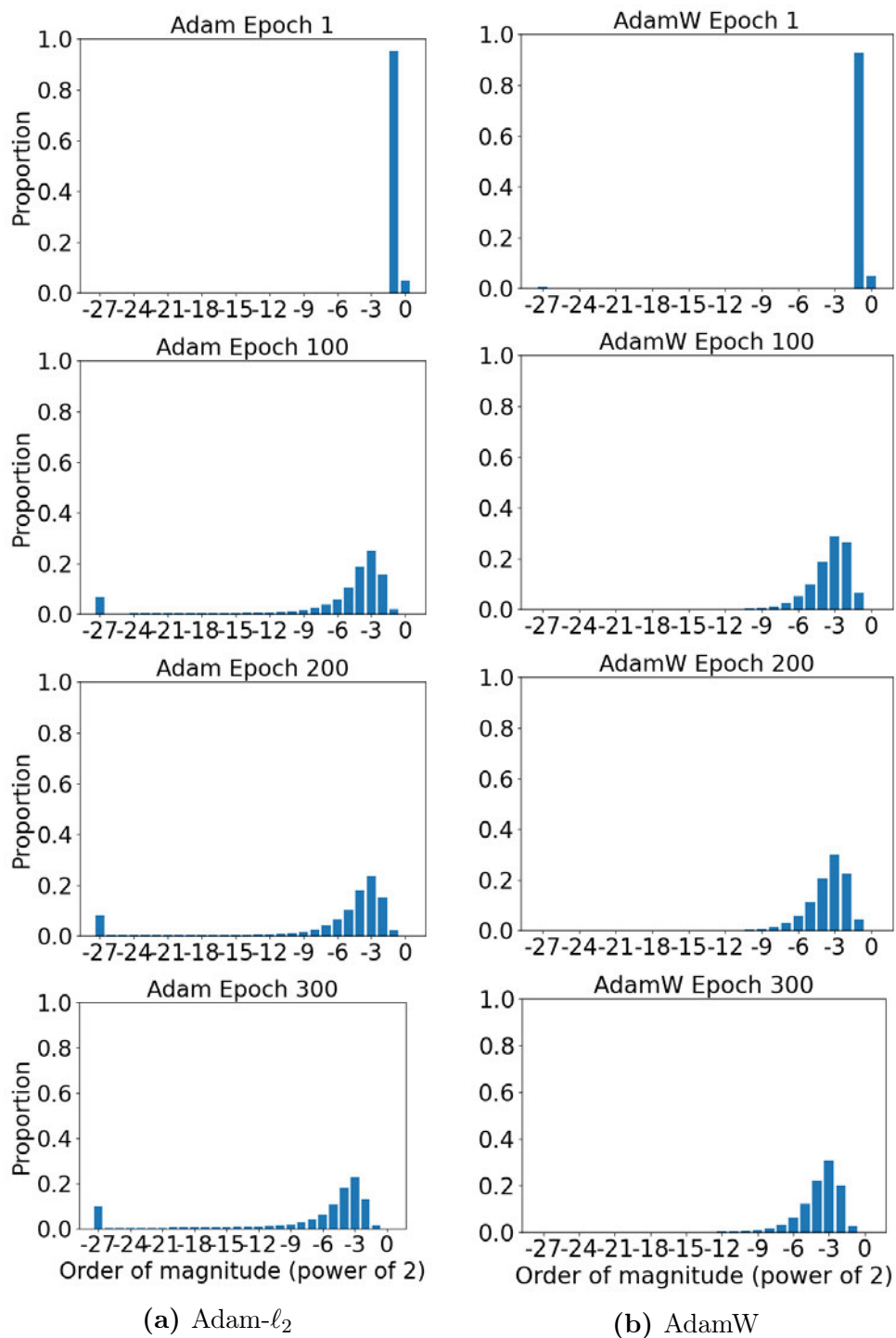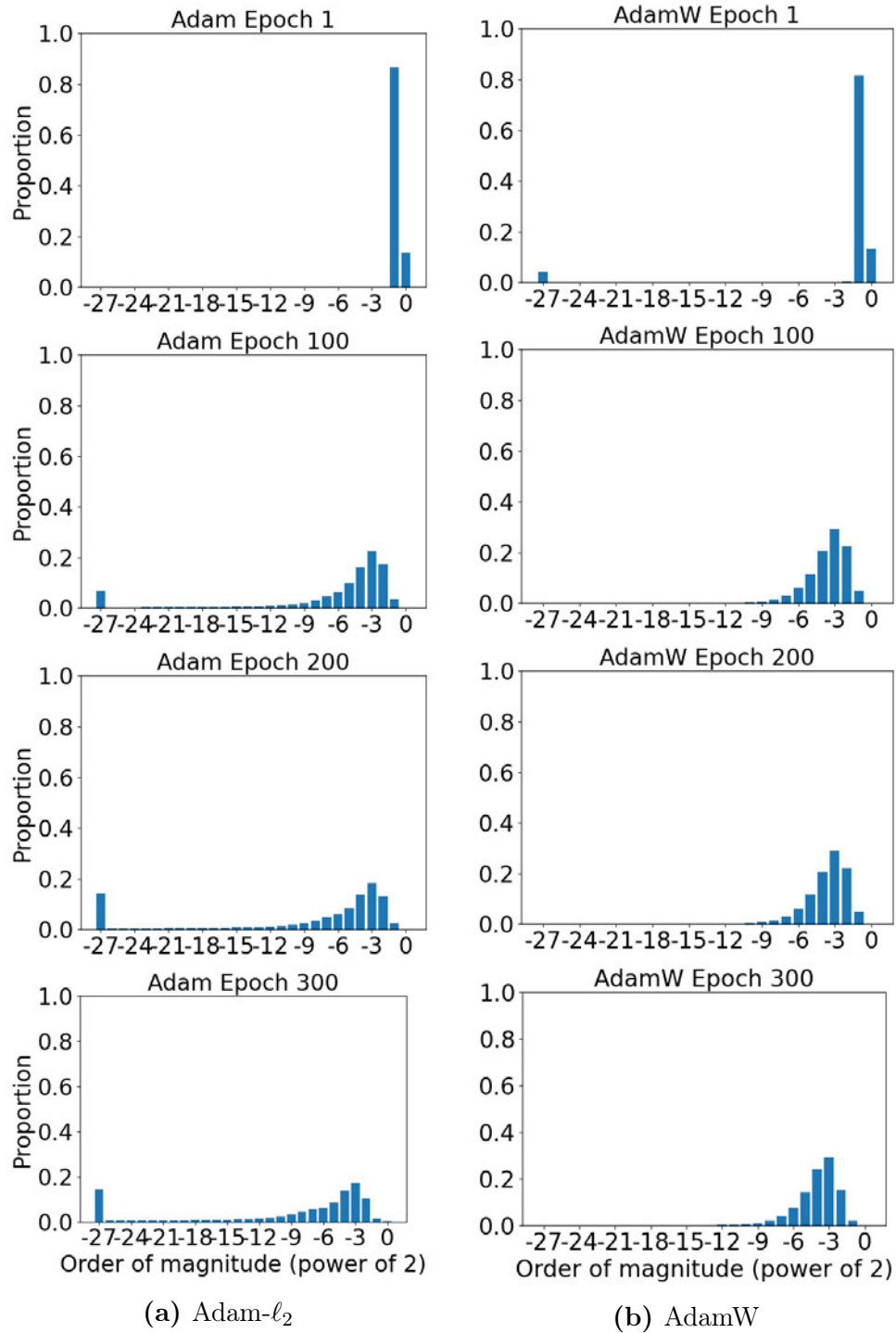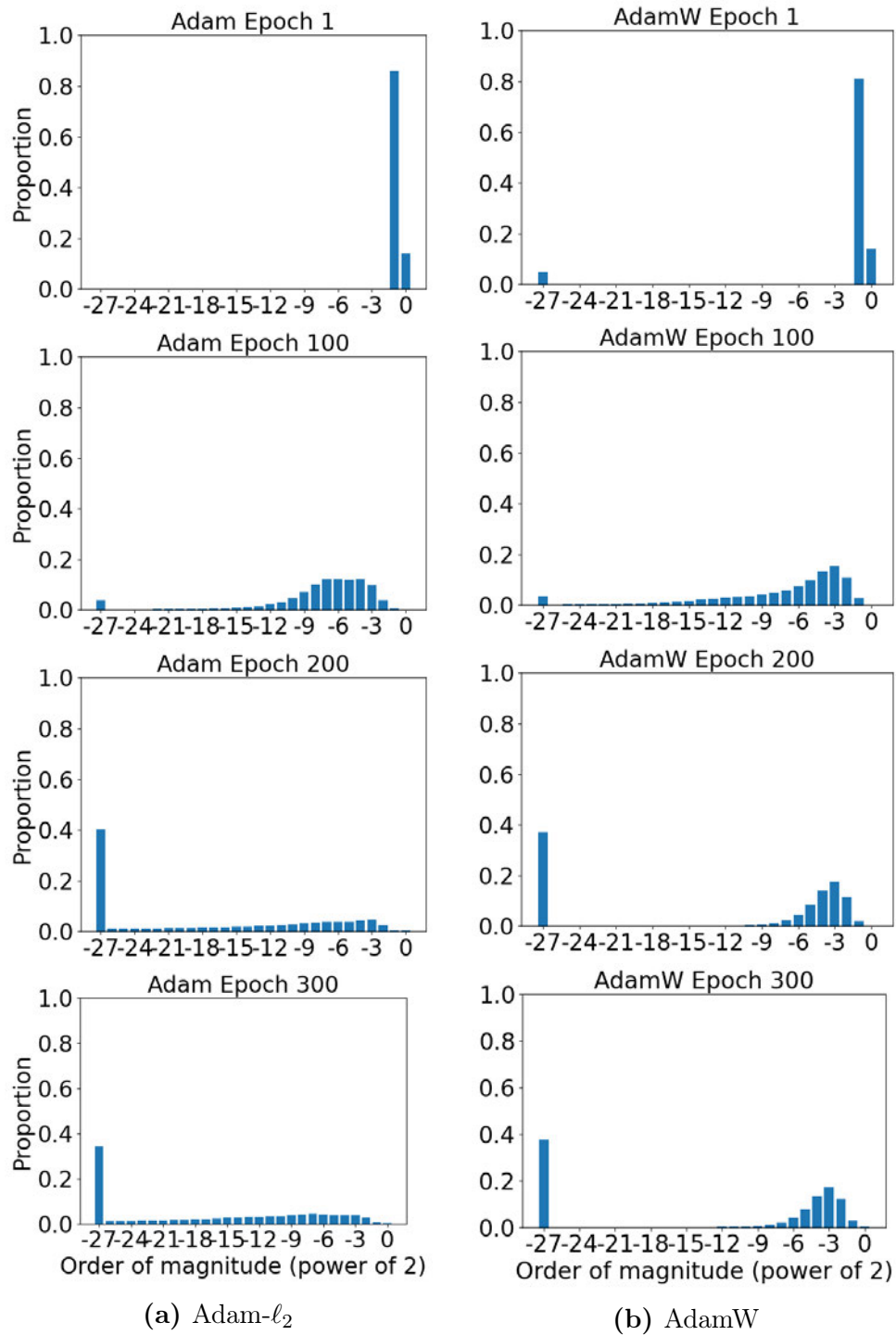
**(a)** Adam-$\ell_2$

**(b)** AdamW

**Figure A·1:** The histograms of the magnitudes of all updates of a 20-layer Resnet with BN disabled trained by AdamW or Adam-$\ell_2$ on CIFAR10.
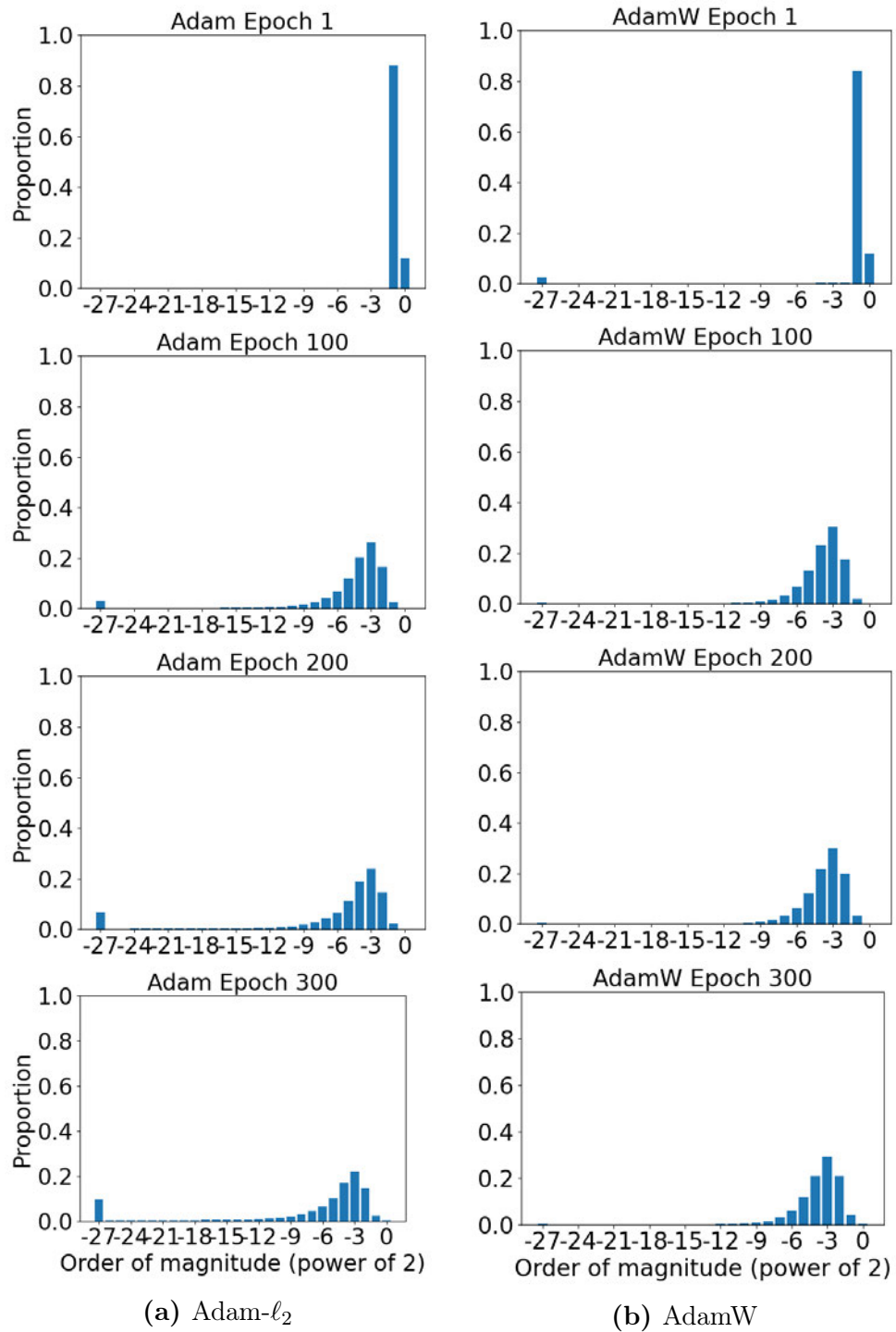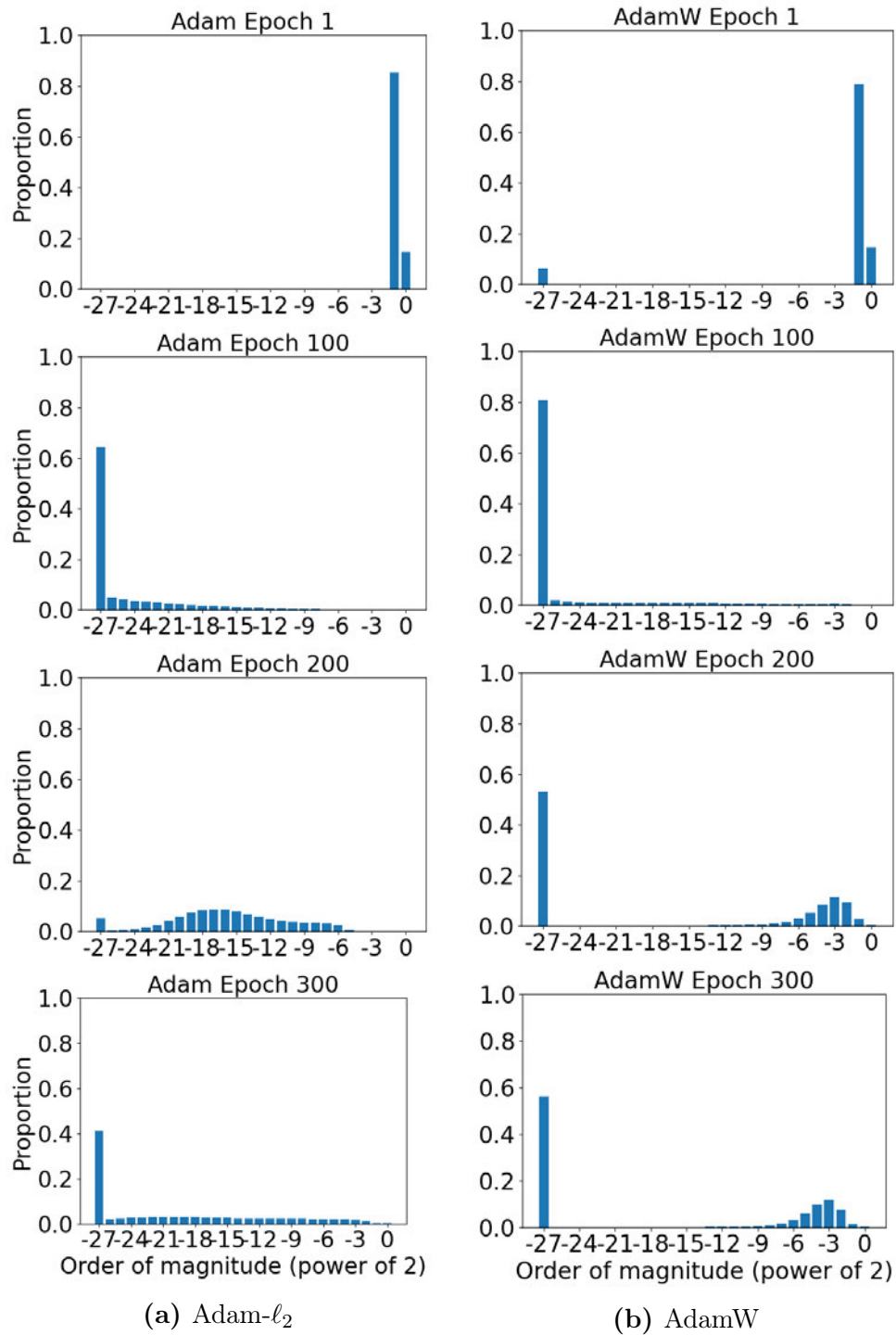
**(a)** Adam-$\ell_2$

**(b)** AdamW

**Figure A·2:** The histograms of the magnitudes of all updates of a 44-layer Resnet with BN disabled trained by AdamW or Adam-$\ell_2$ on CIFAR10.
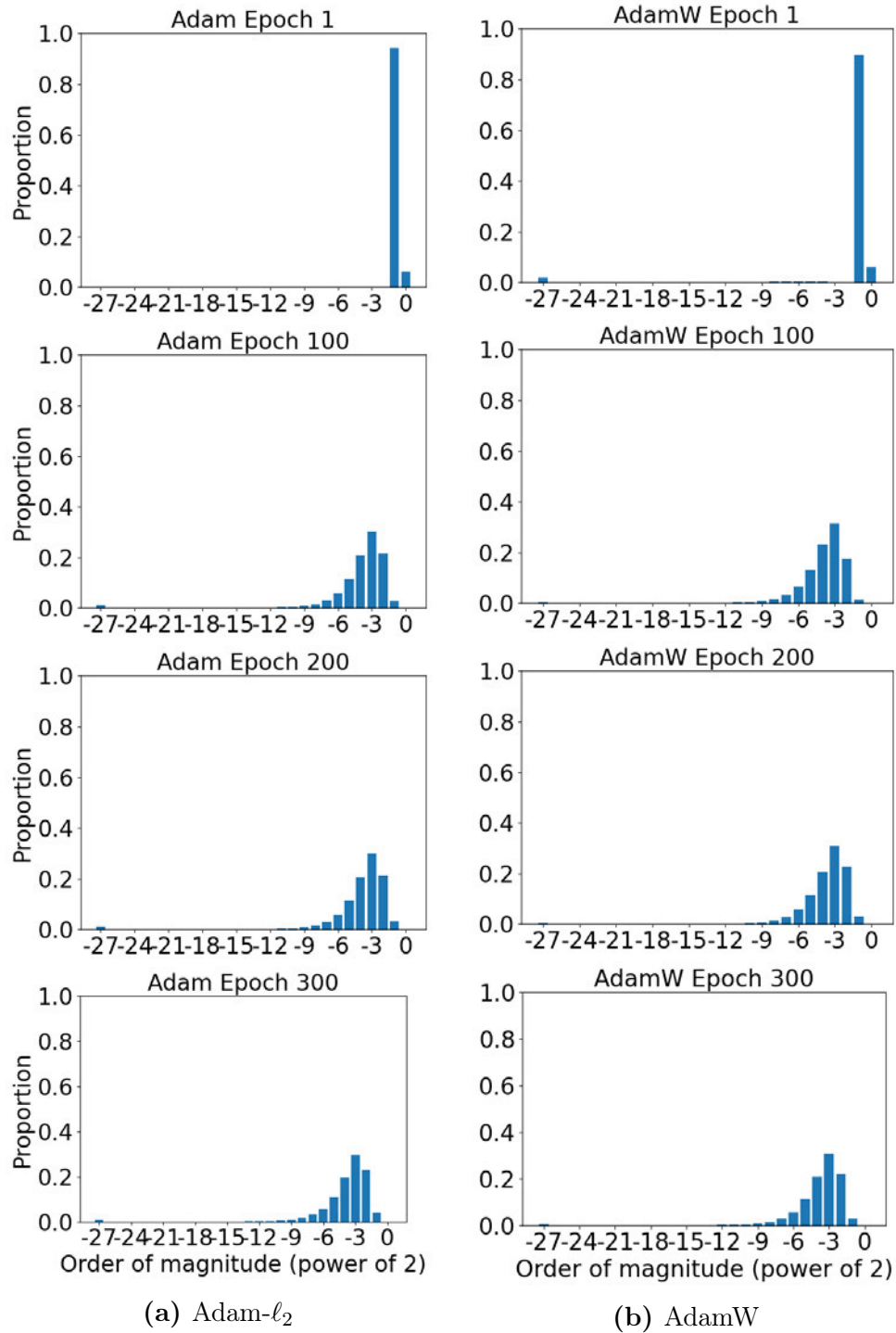
**Figure A·3:** The histograms of the magnitudes of all updates of a 56-layer Resnet with BN disabled trained by AdamW or Adam-$\ell_2$ on CIFAR10.
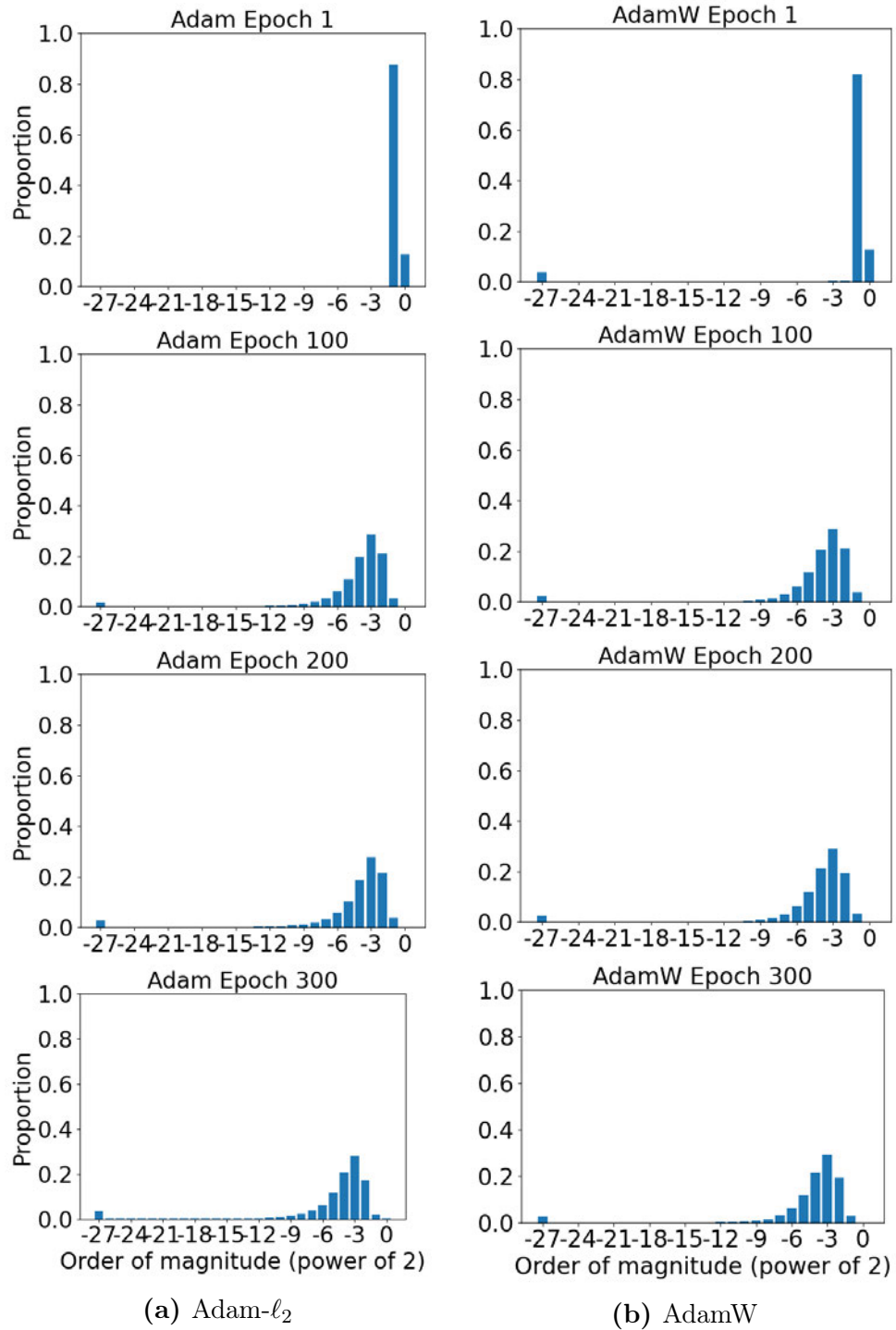
**Figure A·4:** The histograms of the magnitudes of all updates of a 110-layer Resnet with BN disabled trained by AdamW or Adam-$\ell_2$ on CIFAR10.

**Figure A·5:** The histograms of the magnitudes of all updates of a 218-layer Resnet with BN disabled trained by AdamW or Adam-$\ell_2$ on CIFAR10.

**(a)** Adam-$\ell_2$                                           **(b)** AdamW

**Figure A·6:** The histograms of the magnitudes of all updates of a 20-layer Resnet with BN disabled trained by AdamW or Adam-$\ell_2$ on CIFAR100.
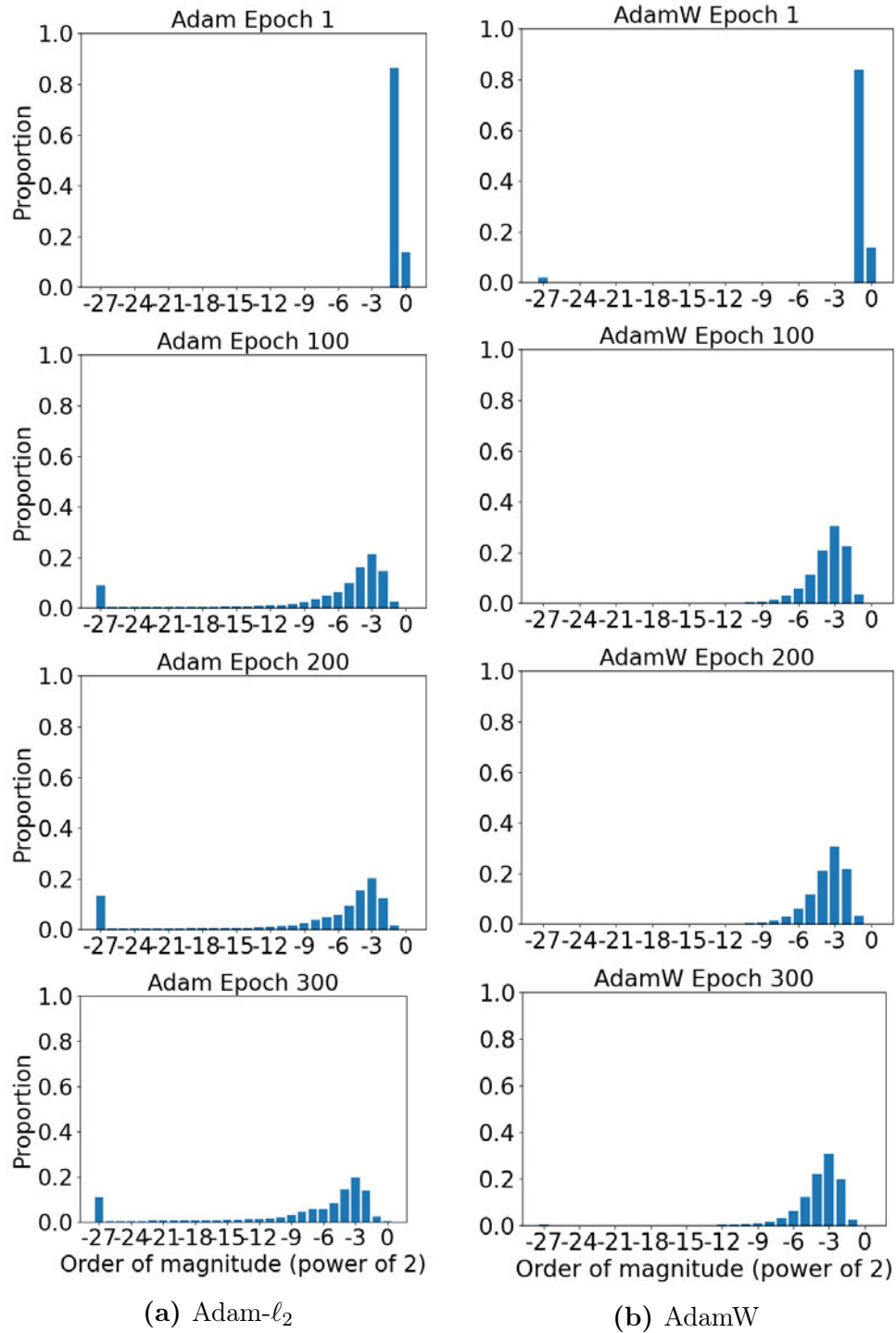
**(a)** Adam-$\ell_2$                    **(b)** AdamW

**Figure A·7:** The histograms of the magnitudes of all updates of a 44-layer Resnet with BN disabled trained by AdamW or Adam-$\ell_2$ on CIFAR100.

**(a)** Adam-$\ell_2$　　　　　　　　　　**(b)** AdamW

**Figure A·8:** The histograms of the magnitudes of all updates of a 56-layer Resnet with BN disabled trained by AdamW or Adam-$\ell_2$ on CIFAR100.

**Figure A·9:** The histograms of the magnitudes of all updates of a 110-layer Resnet with BN disabled trained by AdamW or Adam-$\ell_2$ on CIFAR100.
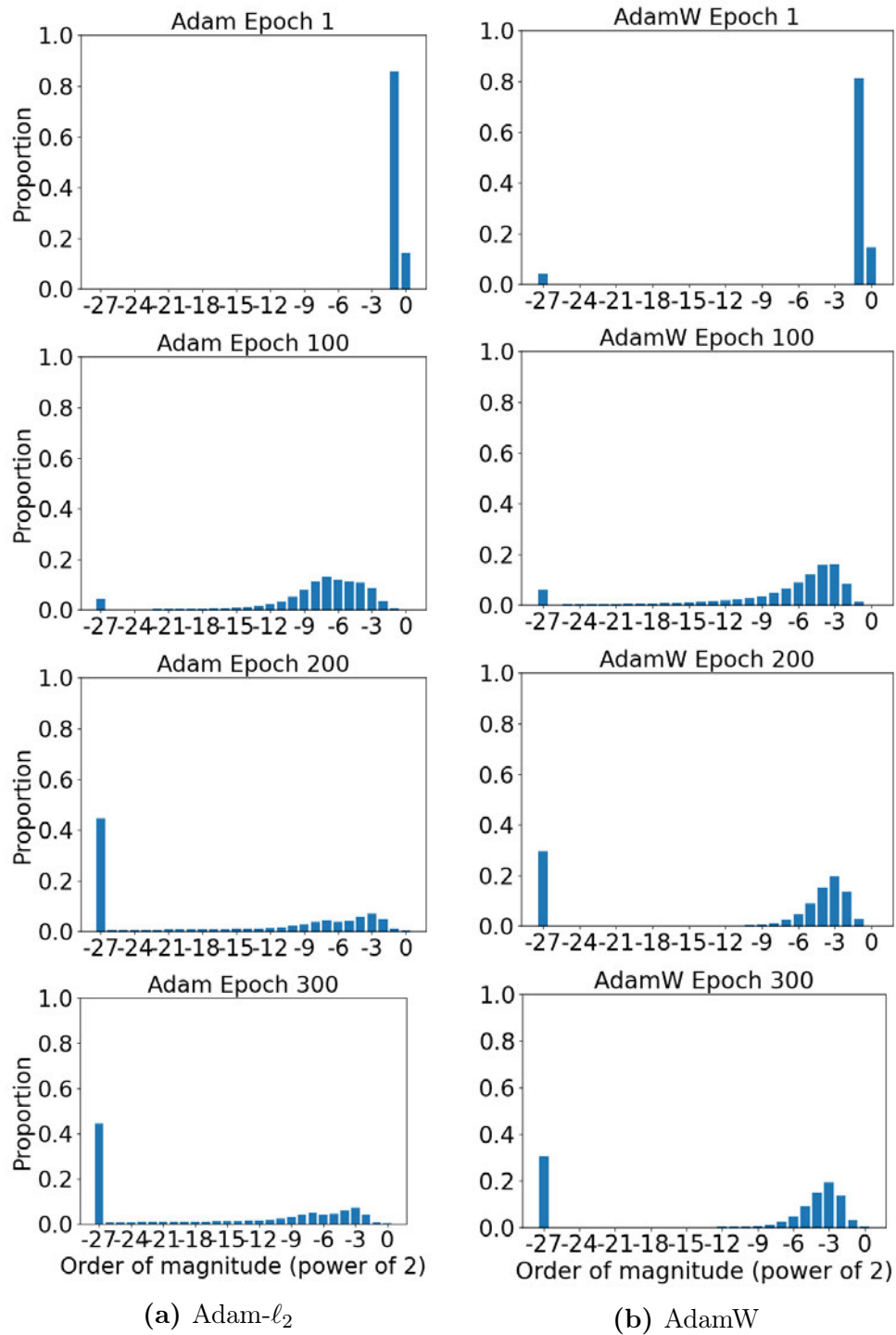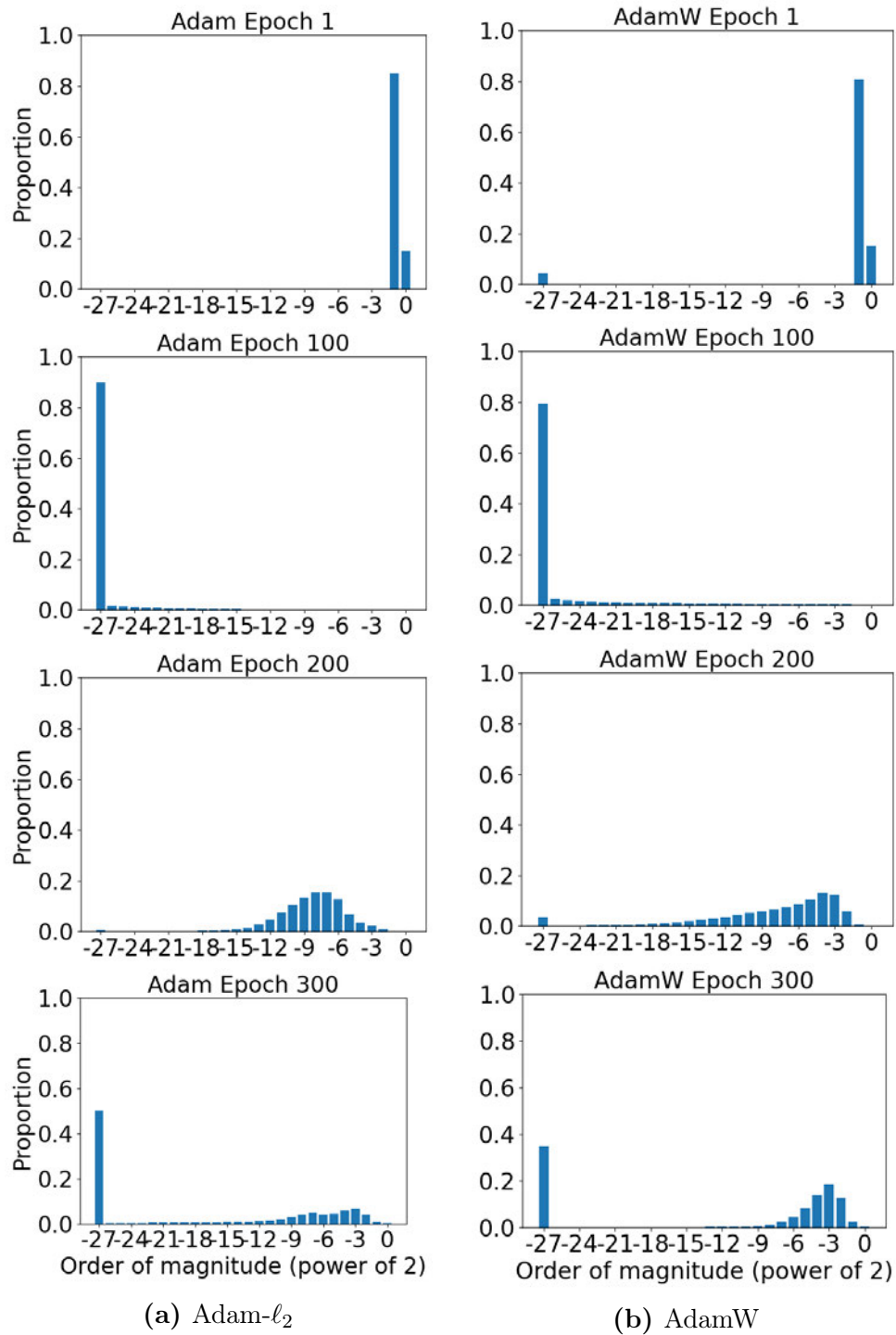
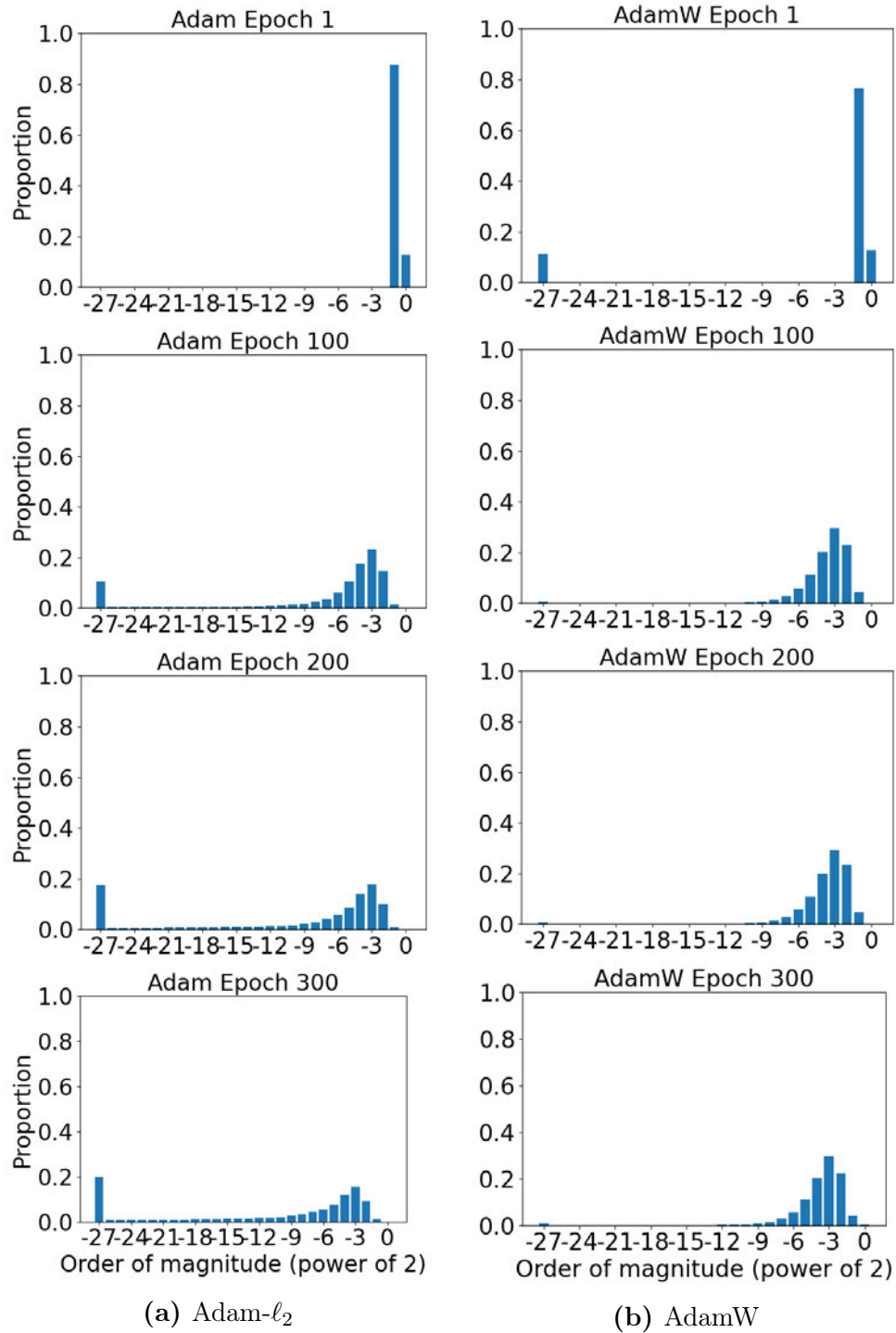**(a)** Adam-$\ell_2$          **(b)** AdamW

**Figure A·10:** The histograms of the magnitudes of all updates of a 218-layer Resnet with BN disabled trained by AdamW or Adam-$\ell_2$ on CIFAR100.

**Figure A·11:** The histograms of the magnitudes of all updates of a 100-layer DenseNet-BC with BN disabled trained by AdamW or Adam-$\ell_2$ on CIFAR100.

# References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Abernethy, J. D., Hazan, E., and Rakhlin, A. (2008). Competing in the dark: An efficient algorithm for bandit linear optimization. In *Proceedings of the 21st Annual Conference on Learning Theory*, pages 263–274. Omnipress.

Abernethy, J. D., Hazan, E., and Rakhlin, A. (2012). Interior-point methods for full-information and bandit online learning. *IEEE Transactions on Information Theory*, 58(7):4164–4175.

Agarwal, N., Anil, R., Hazan, E., Koren, T., and Zhang, C. (2020). Disentangling adaptive gradient methods from learning rates. *arXiv preprint arXiv:2002.11803*.

Alber, Y. I., Iusem, A. N., and Solodov, M. V. (1998). On the projected subgradient method for nonsmooth convex optimization in a Hilbert space. *Mathematical Programming*, 81(1):23–35.

Allen-Zhu, Z., Li, Y., and Song, Z. (2019). A convergence theory for deep learning via over-parameterization. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 242–252. PMLR.

Arjevani, Y., Carmon, Y., Duchi, J., Foster, D. J., Srebro, N., and Woodworth, B. E. (2022). Lower bounds for non-convex stochastic optimization. *Mathematical Programming*, pages 1–50.

Arora, S., Ge, R., Ma, T., and Moitra, A. (2015). Simple, efficient, and neural algorithms for sparse coding. In *Proceedings of The 28th Conference on Learning Theory*, volume 40, pages 113–149. PMLR.

Asi, H. and Duchi, J. C. (2019). Stochastic (approximate) proximal point methods: Convergence, optimality, and adaptivity. *SIAM Journal on Optimization*, 29(3):2257–2290.

Auer, P., Cesa-Bianchi, N., and Gentile, C. (2002). Adaptive and self-confident online learning algorithms. *Journal of Computer and System Sciences*, 64(1):48–75.

Aybat, N. S., Fallah, A., Gurbuzbalaban, M., and Ozdaglar, A. (2019). A universally optimal multistage accelerated stochastic gradient method. In *Advances in Neural Information Processing Systems*, pages 8523–8534.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bach, F. and Moulines, E. (2011). Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems 24*, pages 451–459. Curran Associates, Inc.

Balles, L. and Hennig, P. (2018). Dissecting Adam: The sign, magnitude and variance of stochastic gradients. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 404–413. PMLR.

Beck, A. and Teboulle, M. (2003). Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175.

Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Bernstein, J., Wang, Y., Azizzadenesheli, K., and Anandkumar, A. (2018). SignSGD: Compressed optimisation for non-convex problems. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 560–569. PMLR.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific.

Bjorck, J., Weinberger, K. Q., and Gomes, C. (2021). Understanding decoupled and early weight decay. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6777–6785.

Bos, S. and Chug, E. (1996). Using weight decay to optimize the generalization ability of a perceptron. *Proceedings of International Conference on Neural Networks (ICNN)*, 1:241–246.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Bulò, S. R., Porzi, L., and Kontschieder, P. (2018). In-place activated BatchNorm for memory-optimized training of DNNs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5639–5647.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with Transformers. In *Proceedings of European Conference on Computer Vision*, pages 213–229. Springer.

Carmon, Y., Duchi, J., Hinder, O., and Sidford, A. (2021). Lower bounds for finding stationary points ii: first-order methods. *Mathematical Programming*, 185:315–355.

Cauchy, A. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. *Comptes Rendus de l'Academie des Science*, 25:536–538.

Cesa-Bianchi, N., Mansour, Y., and Stoltz, G. (2007). Improved second-order bounds for prediction with expert advice. *Machine Learning*, 66(2):321–352.

Chang, C.-C. and Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chen, J., Zhou, D., Tang, Y., Yang, Z., and Gu, Q. (2020a). Closing the generalization gap of adaptive gradient methods in training deep neural networks. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 3267–3275.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020b). A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 1597–1607. PMLR.

Chen, Y. and Candes, E. (2015). Solving random quadratic systems of equations is nearly as easy as solving linear systems. In *Advances in Neural Information Processing Systems 28*, pages 739–747. Curran Associates, Inc.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Cramer, J. S. (2002). The origins of logistic regression. *SSRN Electronic Journal*.

Craven, B. D. and Glover, B. M. (1985). Invex functions and duality. *Journal of the Australian Mathematical Society*, 39(1):1–20.

Crawshaw, M., Liu, M., Orabona, F., Zhang, W., and Zhuang, Z. (2022). Robustness to unbounded smoothness of generalized signSGD. In *Advances in Neural Information Processing Systems*.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2019). Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123.

Cutkosky, A. and Mehta, H. (2020). Momentum improves normalized SGD. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 2260–2268. PMLR.

Cutkosky, A. and Mehta, H. (2021). High-probability bounds for non-convex stochastic optimization with heavy tails. In *Advances in Neural Information Processing Systems*, volume 34, pages 4883–4895. Curran Associates, Inc.

Davis, D., Drusvyatskiy, D., and Charisopoulos, V. (2019). Stochastic algorithms with geometric step decay converge linearly on sharp functions. *arXiv preprint arXiv:1907.09547*.

Davis, D., Drusvyatskiy, D., Xiao, L., and Zhang, J. (2021). From low probability to high confidence in stochastic convex optimization. *Journal of Machine Learning Research*, 22(49):1–38.

De, S. and Smith, S. (2020). Batch normalization biases residual blocks towards the identity function in deep networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 19964–19975. Curran Associates, Inc.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional Transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 4171–4186. Association for Computational Linguistics.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations*.

Duchi, J., Hazan, E., and Singer, Y. (2010a). Adaptive subgradient methods for online learning and stochastic optimization. In *Proceedings of the 23rd Annual Conference on Learning Theory*, pages 257–269. Omnipress.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Tewari, A. (2010b). Composite objective mirror descent. In *Proceedings of the 23rd Annual Conference on Learning Theory*, pages 14–26. Omnipress.

Ermoliev, Y. (1988). Stochastic quasigradient methods. In *Numerical techniques for stochastic optimization*, number 10 in Springer Series in Computational Mathematics, pages 141–185. Springer.

Gauss, C. F. (1820). *Theory of the combination of observations least subject to errors, Part One, Part Two, Supplement.* Society for Industrial and Applied Mathematics,. Translated from original manuscript by G. W. Stewart in 1995.

Ge, R., Kakade, S. M., Kidambi, R., and Netrapalli, P. (2019). The step decay schedule: A near optimal, geometrically decaying learning rate procedure for least squares. In *Advances in Neural Information Processing Systems*, volume 32, pages 14951–14962. Curran Associates, Inc.

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

Gitman, I. and Ginsburg, B. (2017). Comparison of batch normalization and weight normalization algorithms for the large-scale image classification. *arXiv preprint arXiv:1709.08145*.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, volume 9, pages 249–256. PMLR.

Goffin, J.-L. (1977). On convergence rates of subgradient optimization methods. *Mathematical programming*, 13(1):329–347.

Gorbunov, E., Danilova, M., and Gasnikov, A. (2020). Stochastic optimization with heavy-tailed noise via accelerated gradient clipping. In *Advances in Neural Information Processing Systems*, volume 33, pages 15042–15053. Curran Associates, Inc.

Goyal, P., Dollár, P., Girshick, R. B., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. (2017). Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*.

Grill, J., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila P., B., Guo, Z., Gheshlaghi Azar, M., Piot, B., kavukcuoglu, K., Munos, R., and Valko, M. (2020). Bootstrap your own latent - a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284. Curran Associates, Inc.

Hanson, M. A. (1981). On sufficiency of the Kuhn-Tucker conditions. *Journal of Mathematical Analysis and Applications*, 80(2):545–550.

Harvey, N. J. A., Liaw, C., Plan, Y., and Randhawa, S. (2019). Tight analyses for non-smooth stochastic gradient descent. In *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99, pages 1579–1613. PMLR.

Hazan, E. and Kale, S. (2014). Beyond the regret minimization barrier: Optimal algorithms for stochastic strongly-convex optimization. *Journal of Machine Learning Research*, 15(71):2489–2512.

Hazan, E., Levy, K. Y., and Shalev-Shwartz, S. (2015). Beyond convexity: Stochastic quasi-convex optimization. In *Advances in Neural Information Processing Systems*, volume 28, pages 1594–1602. Curran Associates, Inc.

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*, pages 770–778.

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., and Li, M. (2019). Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 558–567.

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

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4700–4708.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of International conference on machine learning*, volume 37, pages 448–456. PMLR.

Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260.

Karimi, H., Nutini, J., and Schmidt, M. (2016). Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer.

Khaled, A. and Richtárik, P. (2020). Better theory for SGD in the nonconvex world. *arXiv preprint arXiv:2002.03329*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*.

Kleinberg, B., Li, Y., and Yuan, Y. (2018). An alternative view: When does SGD escape local minima? In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 2698–2707. PMLR.

Koolen, W. M., van Erven, T., and Grünwald, P. (2014). Learning the learning rate for prediction with expert advice. In *Advances in Neural Information Processing Systems*, volume 27, pages 2294–2302. Curran Associates, Inc.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, pages 1106–1114. Curran Associates, Inc.

Krogh, A. and Hertz, J. (1992). A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, volume 4, pages 950–957. Morgan-Kaufmann.

Kuen, J., Perazzi, F., Lin, Z., Zhang, J., and Tan, Y.-P. (2019). Scaling object detection by transferring classification weights. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6044–6053.

Kulunchakov, A. and Mairal, J. (2019). A generic acceleration framework for stochastic composite optimization. In *Advances in Neural Information Processing Systems*, volume 32, pages 12556–12567. Curran Associates, Inc.

Lee, C., Xie, S., Gallagher, P., Zhang, Z., and Tu, Z. (2015). Deeply-supervised nets. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38, pages 562–570. PMLR.

Levy, K. Y., Yurtsever, A., and Cevher, V. (2018). Online adaptive methods, universality and acceleration. In *Advances in Neural Information Processing Systems*, volume 31, pages 6500–6509. Curran Associates, Inc.

Li, M., Andersen, D. G., Park, J. W., Smola, A. J., Ahmed, A., Josifovski, V., Long, J., Shekita, E. J., and Su, B.-Y. (2014). Scaling distributed machine learning with the parameter server. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, volume 14, pages 583–598. USENIX Association.

Li, X. and Orabona, F. (2019). On the convergence of stochastic gradient descent with adaptive stepsizes. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89, pages 983–992. PMLR.

Li, X. and Orabona, F. (2020). A high probability analysis of adaptive SGD with momentum. In *ICML 2020 Workshop on Beyond First Order Methods in ML Systems*.

Li, X., Zhuang, Z., and Orabona, F. (2021). A second look at exponential and cosine step sizes: Simplicity, adaptivity, and performance. In *Proceedings of International Conference on Machine Learning*, volume 139, pages 6553–6564. PMLR.

Li, X.-L. (2018). Preconditioned stochastic gradient descent. *IEEE Transactions on Neural Networks and Learning Systems*, 29(5):1454–1466.

Lifchitz, Y., Avrithis, Y., Picard, S., and Bursuc, A. (2019). Dense classification and implanting for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9258–9267.

Liu, H., Simonyan, K., and Yang, Y. (2019). DARTS: Differentiable architecture search. In *Seventh International Conference on Learning Representations*.

Liu, M., Zhuang, Z., Lei, Y., and Liao, C. (2022). A communication-efficient distributed gradient clipping algorithm for training deep neural networks. In *Advances in Neural Information Processing Systems*.

Łojasiewicz, S. (1963). A topological property of real analytic subsets (in french). *Coll. du CNRS, Les équations aux dérivées partielles*, pages 87–89.

Loshchilov, I. and Hutter, F. (2017). SGDR: Stochastic gradient descent with warm restarts. In *Fifth International Conference on Learning Representations*.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *Seventh International Conference on Learning Representations*.

Mai, V. V. and Johansson, M. (2021). Stability and convergence of stochastic gradient clipping: Beyond lipschitz continuity and smoothness. In *International Conference on Machine Learning*, volume 139, pages 7325–7335. PMLR.

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

Martinet, B. (1970). Brève communication. Régularisation d'inéquations variationnelles par approximations successives. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 4:154–158.

McMahan, H. B. (2017). A survey of algorithms and analysis for adaptive online learning. *Journal of Machine Learning Research*, 18(90):1–50.

McMahan, H. B. and Streeter, M. J. (2010). Adaptive bound optimization for online convex optimization. In *Proceedings of the 23rd Annual Conference on Learning Theory*, pages 244–256. Omnipress.

Meka, R., Jain, P., Caramanis, C., and Dhillon, I. S. (2008). Rank minimization via online learning. In *Proceedings of the 25th International Conference on Machine learning*, pages 656–663. Omnipress.

Merity, S., Keskar, N. S., and Socher, R. (2018). Regularizing and optimizing LSTM language models. In *Sixth International Conference on Learning Representations*.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2017). Pointer sentinel mixture models. In *Fifth International Conference on Learning Representations*.

Mohri, M. and Yang, S. (2016). Accelerating online convex optimization via adaptive prediction. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS*, volume 51, pages 848–856. PMLR.

Moreau, J.-J. (1965). Proximité et dualité dans un espace hilbertien. *Bulletin de la Société Mathématique de France*, 93:273–299.

Nemirovsky, A. S. and Yudin, D. (1983). *Problem complexity and method efficiency in optimization*. John Wiley & Sons.

Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. In *Doklady Akademii Nauk SSSR*, volume 269, pages 543–547.

Nesterov, Y. (2004). *Introductory lectures on convex optimization: A basic course*. Springer.

Nesterov, Y. (2015). Universal gradient methods for convex optimization problems. *Mathematical Programming*, 152(1):381–404.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York.

Orabona, F. and Pál, D. (2015). Scale-free algorithms for online linear optimization. In *Proceedings of the 26th International Conference on Algorithmic Learning Theory*, pages 287–301. Springer.

Orabona, F. and Pál, D. (2018). Scale-free online learning. *Theoretical Computer Science*, 716:50–69. Special Issue on ALT 2015.

Parikh, N. and Boyd, S. (2014). Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of International conference on machine learning*, volume 28, pages 1310–1318. PMLR.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8024–8035. Curran Associates, Inc.

Polyak, B. T. (1963). Gradient methods for minimizing functionals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 3(4):643–653.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*.

Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., and McMahan, H. B. (2021). Adaptive federated optimization. In *Ninth International Conference on Learning Representations*.

Reddi, S. J., Kale, S., and Kumar, S. (2018). On the convergence of Adam and beyond. In *Sixth International Conference on Learning Representations*.

Richtárik, P. and Takác, M. (2014). Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144:1–38.

Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE international conference on neural networks*, pages 586–591. IEEE.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407.

Rockafellar, R. T. (1976). Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898.

Rockafellar, R. T. (1993). Lagrange multipliers and optimality. *SIAM Review*, 35(2):183–238.

Schoenholz, S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. (2017). Deep information propagation. In *Fifth International Conference on Learning Representations*.

Shalev-Shwartz, S. (2007). *Online Learning: Theory, Algorithms, and Applications*. PhD thesis, The Hebrew University.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Shor, N. Z. (2012). *Minimization methods for non-differentiable functions.* Springer Science & Business Media.

Singh, S. and Shrivastava, A. (2019). EvalNorm: Estimating batch normalization statistics for evaluation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3633–3641.

Sun, R. and Luo, Z. (2016). Guaranteed matrix completion via non-convex factorization. *IEEE Transactions on Information Theory*, 62(11):6535–6579.

Sundermeyer, M., Schlüter, R., and Ney, H. (2012). LSTM neural networks for language modeling. In *Proceedings of the 13th Annual Conference of the International Speech Communication Association*, pages 194–197.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566. Association for Computational Linguistics.

Tan, M. and Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 6105–6114. PMLR.

Toulis, P. and Airoldi, E. M. (2017). Asymptotic and finite-sample properties of estimators based on stochastic gradients. *The Annals of Statistics*, 45(4):1694–1727.

van Erven, T. and Koolen, W. M. (2016). MetaGrad: Multiple learning rates in online learning. In *Advances in Neural Information Processing Systems*, volume 29, pages 3666–3674. Curran Associates, Inc.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, volume 30, pages 6000–6010. Curran Associates, Inc.

Vaswani, S., Mishkin, A., Laradji, I., Schmidt, M., Gidel, G., and Lacoste-Julien, S. (2019). Painless stochastic gradient: Interpolation, line-search, and convergence rates. In *Advances in Neural Information Processing Systems*, volume 32, pages 3732–3745. Curran Associates, Inc.

Verhulst, P. F. (1845). Resherches mathématiques sur la loi d'accroissement de la population. *Nouveaux mémoires de l'Académie Royale des Sciences et Belles-Lettres de Bruxelles*, 18:1–41.

Wang, Y., Kang, Y., Qin, C., Wang, H., Xu, Y., Zhang, Y., and Fu, Y. R. (2021). Rethinking Adam: A twofold exponential moving average approach. *arXiv preprint arXiv:2106.11514*.

Ward, R., Wu, X., and Bottou, L. (2019). AdaGrad stepsizes: Sharp convergence over nonconvex landscapes, from any initialization. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 6677–6686. PMLR.

Warmuth, M. K. and Jagota, A. K. (1997). Continuous and discrete-time nonlinear gradient descent: Relative loss bounds and convergence. In *Electronic Proceedings of the 5th International Symposium on Artificial Intelligence and Mathematics*.

Wolf, T. (2019). *Transfer learning in natural language processing*. Available at `https://github.com/huggingface/naacl_transfer_learning_tutorial`.

Wu, Y. and He, K. (2018). Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19.

You, J., Leskovec, J., He, K., and Xie, S. (2020). Graph structure of neural networks. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 10881–10891. PMLR.

Yuan, Z., Yan, Y., Jin, R., and Yang, T. (2019). Stagewise training accelerates convergence of testing error over SGD. In *Advances in Neural Information Processing Systems*, volume 32, pages 2604–2614. Curran Associates, Inc.

Zhang, B., Jin, J., Fang, C., and Wang, L. (2020a). Improved analysis of clipping algorithms for non-convex optimization. In *Advances in Neural Information Processing Systems*, volume 33, pages 15511–15521. Curran Associates, Inc.

Zhang, H., Dauphin, Y. N., and Ma, T. (2019a). Fixup initialization: Residual learning without normalization. In *Seventh International Conference on Learning Representations*.

Zhang, J., He, T., Sra, S., and Jadbabaie, A. (2020b). Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *Eighth International Conference on Learning Representations*.

Zhang, J., Karimireddy, S. P., Veit, A., Kim, S., Reddi, S., Kumar, S., and Sra, S. (2020c). Why are adaptive methods good for attention models? In *Advances in Neural Information Processing Systems*, volume 33, pages 15383–15393. Curran Associates, Inc.

Zhang, X., Wang, Q., Zhang, J., and Zhong, Z. (2019b). Adversarial autoaugment. In *Seventh International Conference on Learning Representations*.

Zhang, Y., Chan, W., and Jaitly, N. (2017). Very deep convolutional networks for end-to-end speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4845–4849.

Zhao, H., Jia, J., and Koltun, V. (2020). Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10076–10085.

Zheng, L., Noroozi, V., and Yu, P. S. (2017). Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, page 425–434. Association for Computing Machinery.

Zhou, J., Cao, Y., Wang, X., Li, P., and Xu, W. (2016). Deep recurrent models with fast-forward connections for neural machine translation. *Transactions of the Association for Computational Linguistics*, 4:371–383.

Zhu, Z. (2018). Natasha 2: Faster non-convex optimization than SGD. In *Advances in Neural Information Processing Systems*, volume 31, pages 2675–2686. Curran Associates, Inc.

Zhuang, Z., Cutkosky, A., and Orabona, F. (2019). Surrogate losses for online learning of stepsizes in stochastic non-convex optimization. In *Proceedings of International Conference on Machine Learning*, volume 97, pages 7664–7672. PMLR.

Zhuang, Z., Liu, M., Cutkosky, A., and Orabona, F. (2022). Understanding AdamW through proximal methods and scale-freeness. *Transactions on Machine Learning Research*.

Zou, D., Cao, Y., Li, Y., and Gu, Q. (2021). Understanding the generalization of Adam in learning neural networks with proper regularization. *arXiv preprint arXiv:2108.11371*.

# CURRICULUM VITAE