明治大学大学院　理工学研究科

２０２４年度

修士学位請求論文

論文題名：

Implicit Graduated Optimization with Noise in Stochastic Gradient Descent

指導教員名：飯塚　秀明

情報科学専攻

請求者氏名：佐藤　尚樹

Master's Thesis

Implicit Graduated Optimization with Noise in Stochastic Gradient Descent

明治大学理工学研究科情報科学専攻 数理最適化研究室
2 年 55 組 17 番 佐藤 尚樹

The graduated optimization approach is used to find global optimal solutions for nonconvex functions by using a function smoothing operation with stochastic noise. We found that stochastic noise in stochastic gradient descent (SGD) has the effect of smoothing the objective function, the degree of which is determined by the learning rate, batch size, and variance of the stochastic gradient. On the basis of this finding, we propose and analyze a graduated optimization algorithm that varies the degree of smoothing by varying the learning rate and batch size and provide experimental results on image classification tasks with ResNets that support our theoretical findings. We further demonstrate an intriguing relationship between the degree of smoothing by SGD's stochastic noise, the well-studied "sharpness" indicator, and the model's generalization performance.

Contents

Section 1. Introduction

1.1 Background

1.1.1 Machine Learning and Optimization

Machine learning is a technique used to classify and predict unknown data by learning rules and patterns from given data. When training a deep learning model, which is a specific machine learning method, the objective function is defined as the average error between the model's predictions and the correct values. The goal is to minimize this error by adjusting the parameters of the deep learning model. Therefore, solving the optimization problem defined below is referred to as training the model.

$$\text{Minimize} \quad f(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{x} \in \mathbb{R}^d, \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ is parameter of the model and $f_i(\boldsymbol{x})\colon \mathbb{R}^d \to \mathbb{R}$ is the loss function for the i-th training data point (d is the dimension of the model, and n is the number of training data points). The objective function f in (1) is called an empirical loss function because it is computed from a limited set of training data, and Problem (1) is called an empirical risk minimization problem. In general, an empirical loss function is a differentiable nonconvex function. The optimal solution to Problem (1), $\boldsymbol{x}^\star \in \mathbb{R}^d$, represents the most suitable parameter for the deep learning model derived from the training data. Therefore, to successfully train the model, it is essential to solve Problem (1) in order to approximate this optimal solution, $\boldsymbol{x}^\star \in \mathbb{R}^d$.

1.1.2 Optimization Methods for Empirical Risk Minimization

An optimization method called the gradient method, which uses the gradient (the differential information of a function) is often used to solve Problem (1). The gradient method is used for updating the current point $\boldsymbol{x}_t \in \mathbb{R}^d$ to the next point $\boldsymbol{x}_{t+1} \in \mathbb{R}^d$ by stepping along vector $\boldsymbol{d}_t \in \mathbb{R}^d$ with step size $\eta_t > 0$:

$$\boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \eta_t \boldsymbol{d}_t \tag{2}$$

That is, a point sequence $(\boldsymbol{x}_t)_{t\in\mathbb{N}} \in \mathbb{R}^d$ defined as in Equation (2) is generated so that $\boldsymbol{x}_t$ approximates the optimal solution of Problem 1 when the time or the number of iterations t is sufficiently large. Here, $\eta_t > 0$ is referred to as the learning rate or step size, and vector $\boldsymbol{d}_t$ is referred to as the search direction at time t. The simplest gradient method is gradient descent (GD) using the full gradient $\nabla f(\boldsymbol{x}_t)$ of objective function f as the search direction:

$$\boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \eta_t \nabla f(\boldsymbol{x}_t).$$

However, since the total number of model parameters d ranges from hundreds of thousands to trillions, and the total number of training data points n ranges from tens of thousands to tens of millions, GD training, which requires the calculation of the

full gradient $\nabla f(\boldsymbol{x}_t)$ at each time step, is not practical. Stochastic gradient descent (SGD) [1] (defined below) uses mini-batch stochastic gradients $\nabla f_{\mathcal{S}_t}$ computed on b ($\leq n$) randomly selected training data points as the search direction instead of using all n training data points at each time step. It can be performed by setting batch size b appropriately:

$$\boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \eta_t \nabla f_{\mathcal{S}_t}(\boldsymbol{x}_t).$$

SGD is the simplest of the optimization methods used to train deep learning models. The amazing success of deep neural networks (DNNs) in recent years has been based on optimization by SGD and its variants, such as Adam [2]. These methods have been widely studied for their convergence [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14] and stability [15, 16, 17, 18] in nonconvex optimization.

1.1.3 Empirical Loss Function and Model Generalizability

The most important aspect of machine learning is the generalization performance of the model. Generalization performance refers to the ability to accurately predict and classify data beyond the training data, i.e., the test data. By applying an optimization algorithm such as SGD to Problem (1), we can obtain an approximate solution that minimizes the empirical loss function value $f(\boldsymbol{x})$. In this scenario, the training accuracy, which is the prediction accuracy for the training data, approaches 100% while the test accuracy, which is the prediction accuracy for the test data, typically ranges between 60% and 90%, depending on the training dataset and model type. Although achieving 100% test accuracy is the ultimate goal of machine learning, only the empirical loss function can be theoretically analyzed. Numerous studies have explored the relationship between the empirical loss function and generalization performance. Most of these studies focused on the shape of the neighborhood around the optimal solution of the empirical loss function, with the prevailing hypothesis being that an optimal solution with a flat neighborhood offers better generalizability than one with a steep neighborhood."

As shown in Figure 1, there is generally a difference between the empirical loss function calculated from training data only and the expected loss function calculated from an infinite amount of data, including the test data. While we can optimize only the empirical loss function, we must also consider the expected loss function, since if we can minimize the expected loss function value, the test accuracy should be 100%. Analysis of the discrepancy between the two loss functions suggests that the optimal solution for the empirical loss function in a flatter neighborhood is likely to achieve a lower expected loss function value, as illustrated in Figure 1. This hypothesis originates from the work of Hochreiter and Schmidhuber [19]. There have been numerous studies on this hypothesis. For example, several studies have proposed using "sharpness" as a measure of the smoothness of the function in the neighborhood of the approximate solution [20, 21, 22, 23, 24], and experimental results have shown that there is a correlation between sharpness and a model's generalization performance.

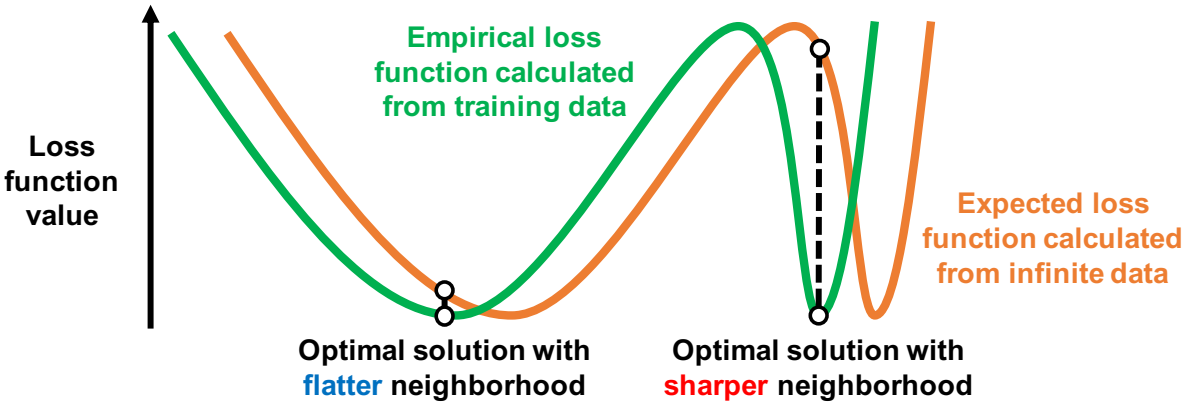

Fig. 1 Conceptual diagram of empirical and expected loss functions. Minimizing the expected loss function is essential for achieving high generalization performance, but we can only deal with the empirical loss function. Therefore, near the optimal solution of the empirical loss function, where the optimization methods converge, a flat optimal solution with a smaller error between the two functions (dotted line) is considered to offer high generalization performance.

1.1.4 Smoothing by Stochastic Noise in SGD

Unlike GD, which processes all training data points simultaneously, SGD processes only b data points simultaneously, so stochastic noise is introduced depending on the data selected at each iteration. It is reasonable to represent this stochastic noise in terms of the error $\nabla f_{\mathcal{S}_t}(\boldsymbol{x}_t) - \nabla f(\boldsymbol{x}_t) =: \boldsymbol{\omega}_t$ in each search direction at each iteration. Some studies claim that it is crucial in nonconvex optimization. For example, it has been proven that noise helps the algorithm to escape local minima [25, 26, 27, 28], achieve better generalization [15, 17], and find a local minimum with a small loss value in polynomial time under some assumptions [29]. Several studies have also shown that performance can be improved by adding artificial noise to GD [25, 30, 31, 32].

[33] also suggests that noise smoothes the objective function. Here, at time t, let $\boldsymbol{y}_t$ be the parameter updated by GD and $\boldsymbol{x}_{t+1}$ be the parameter updated by SGD, i.e.,

$$\boldsymbol{y}_t := \boldsymbol{x}_t - \eta \nabla f(\boldsymbol{x}_t), \ \boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \eta \nabla f_{\mathcal{S}_t}(\boldsymbol{x}_t).$$

Then, the update rule for the sequence $\{\boldsymbol{y}_t\}$ is as follows:

$$\mathbb{E}_{\boldsymbol{\omega}_t}\left[\boldsymbol{y}_{t+1}\right] = \mathbb{E}_{\boldsymbol{\omega}_t}\left[\boldsymbol{y}_t\right] - \eta \nabla \mathbb{E}_{\boldsymbol{\omega}_t}\left[f(\boldsymbol{y}_t - \eta \boldsymbol{\omega}_t)\right], \tag{3}$$

where f is Lipschitz continuous and differentiable (The derivation of Equation (3) is in Section A). Therefore, if we define a new function $\hat{f}(\boldsymbol{y}_t) := \mathbb{E}_{\boldsymbol{\omega}_t}[f(\boldsymbol{y}_t - \eta \boldsymbol{\omega}_t)]$, $\hat{f}$ can be smoothed by convolving f with noise (see Definition 2.1, also [34]), and its parameters $\boldsymbol{y}_t$ can approximately be viewed as being updated by using the GD to minimize $\hat{f}$. In other words, simply using SGD with a mini-batch smoothes the function to some extent and may enable escapes from local minima (see also Figure 2).

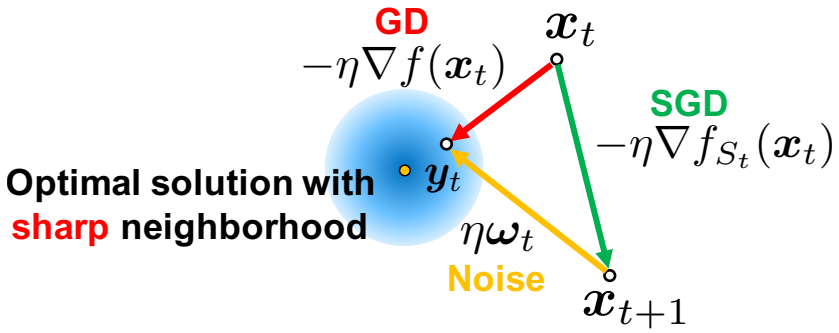

Fig. 2 Conceptual diagram of how stochastic noise in SGD prevents finding of local optimal solution. The center of the blue region represents a local optimal solution with a sharp neighborhood and poor generalization performance. The steepest descent direction $-\nabla f(\boldsymbol{x}_t)$ (red arrow) is straight towards the local optimal solution. In contrast, the SGD direction $-\nabla f_{\mathcal{S}_t}(\boldsymbol{x}_t)$ (green arrow) is away from the local optimal solution due to the presence of stochastic noise $\boldsymbol{\omega}_t$ (yellow arrow).

1.1.5 Graduated Optimization

Graduated optimization is one of the global optimization methods that search for the global optimal solution of difficult multimodal optimization problems. The method generates a sequence of simplified optimization problems that gradually approach the original problem through different levels of local smoothing operations. It solves the easiest simplified problem first, as the easiest simplification should have nice properties such as convexity or strong convexity; after that, it uses that solution as the initial point for solving the second-simplest problem, then the second solution as the initial point for solving the third-simplest problem and so on, as it attempts to escape from local optimal solutions of the original problem and reach a global optimal solution (see Figure 3 for conceptual diagram of graduated optimization).

This idea first appeared in the form of graduated non-convexity by [35] and has since been studied in the field of computer vision for many years. Similar early approaches can be found in [36] and [37]. Moreover, the same concept has appeared in the fields of numerical analysis [38] and optimization [39, 34]. Over the past 35 years, graduated optimization has been successfully applied to many tasks in computer vision, such as early vision [40], image denoising [41], optical flow [42, 43], dense correspondence of images [44], and robust estimation [45, 46, 47]. In addition, it has been applied to certain tasks in machine learning, such as semi-supervised learning [48, 49, 50], unsupervised learning [51], and ranking [52]. Moreover, score-based generative models [53, 54] and diffusion models [55, 56, 57, 58], which are currently state-of-the-art generative models, implicitly use the techniques of graduated optimization. A comprehensive survey on the graduated optimization approach can be found in [59].

Several previous studies have theoretically analyzed the graduated optimization algorithm. [60] performed the first theoretical analysis, but they did not provide a practical algorithm. [61] defined a family of nonconvex functions satisfying certain

conditions, called σ-nice, and proposed a first-order algorithm based on graduated optimization. In addition, they studied the convergence and convergence rate of their algorithm to a global optimal solution for σ-nice functions. [62] analyzed graduated optimization based on a special smoothing operation. Note that [63] pioneered the theoretical analysis of optimizers using Gaussian smoothing operations for nonsmooth convex optimization problems.

1.2 Motivation

Equation (3) indicates that SGD smoothes the objective function [33], but it is not clear to what extent the function is smoothed or what factors are involved in the smoothing. Therefore, we decided to clarify these aspects and identify what parameters contribute to the smoothing. Also, once it is known what parameters of SGD contribute to smoothing, an implicit graduated optimization can be achieved by varying the parameters so that the noise level is reduced gradually. Our goal was thus to construct an implicit graduated optimization framework using the smoothing properties of SGD to achieve global optimization of DNNs.

1.3 Contributions

1. SGD's Smoothing Property (Section 3). We show that the degree of smoothing δ provided by SGD's stochastic noise depends on the quantity $\delta = \frac{\eta C}{\sqrt{b}}$, where η is the learning rate, b is the batch size, and C^2 is the variance of the stochastic gradient (see Assumption 2.1). Accordingly, the smaller the batch size b is and the larger the learning rate η is, the smoother the function becomes (see Figure 3). This finding provides a theoretical explanation for several experimental observations. For example, as is well known, training with a large batch size leads to poor generalization performance, as evidenced by the fact that several prior studies [64, 65, 66] provided techniques that do not impair generalization performance even with large batch sizes. This is because, if we use a large batch size, the degree of smoothing $\delta = \frac{\eta C}{\sqrt{b}}$ becomes smaller and the original nonconvex function is not smoothed enough, so the sharp local minima do not disappear and the optimizer is more likely to fall into one. [20] showed this experimentally, and our results provide theoretical support for it.

2. New theoretical findings on σ-nice function (Section 4.1). It has been unclear to what extent the σ-nice function is a special function and whether the empirical loss function used in practical applications satisfies the σ-nice property. We propose a σ_m-nice function which slightly extends the definition of the σ-nice function to make it more practical. We also show that the cross entropy loss and mean squared error, which are often used in machine learning, are σ_m-nice functions under a certain assumption (Proposition 4.1). This allows all previous theoretical and experimental findings on graduated optimization to be incorporated in machine learning through the σ_m-nice function.

3. Implicit Graduated Optimization (Sections 4.2 and 4.3). Since the de-

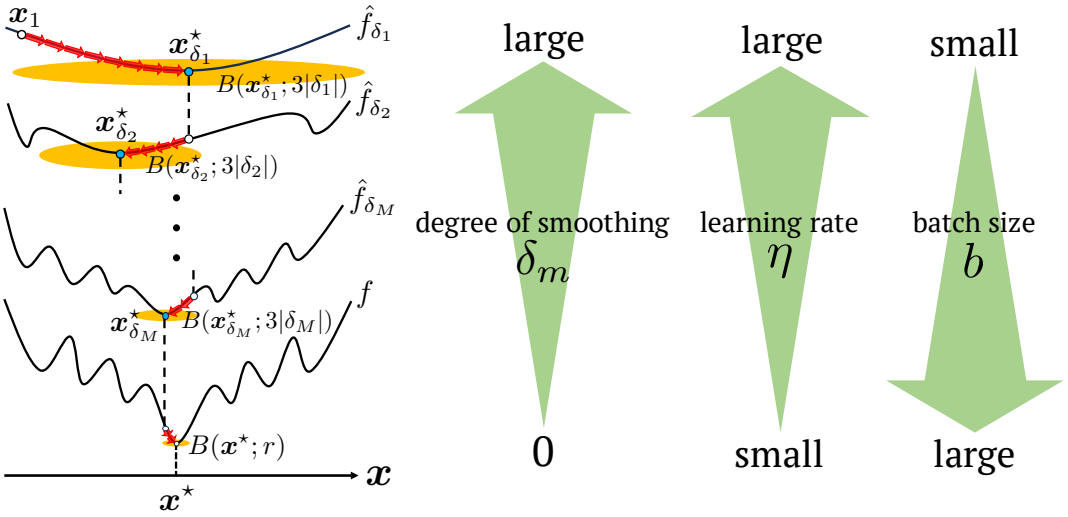

Fig. 3 Conceptual diagram of implicit graduated optimization for σ_m-nice function.

gree of smoothing of the objective function by stochastic noise in SGD is determined by $\delta = \frac{\eta C}{\sqrt{b}}$, it should be possible to construct an implicit graduated optimization algorithm by decreasing the learning rate and/or increasing the batch size during training. Building on this theoretical intuition, we propose a new implicit graduated optimization algorithm. We also show that the algorithm for the σ_m-nice function converges to an ϵ-neighborhood of the global optimal solution in $\mathcal{O}\left(1/\epsilon^2\right)$ rounds. In Section 4.3, we show experimentally that our implicit graduated algorithm outperforms SGD using a constant learning rate and constant batch size. We also find that methods which increase the batch size outperform those which decrease the learning rate when the decay rate of the degree of smoothing is set at $1/\sqrt{2}$.

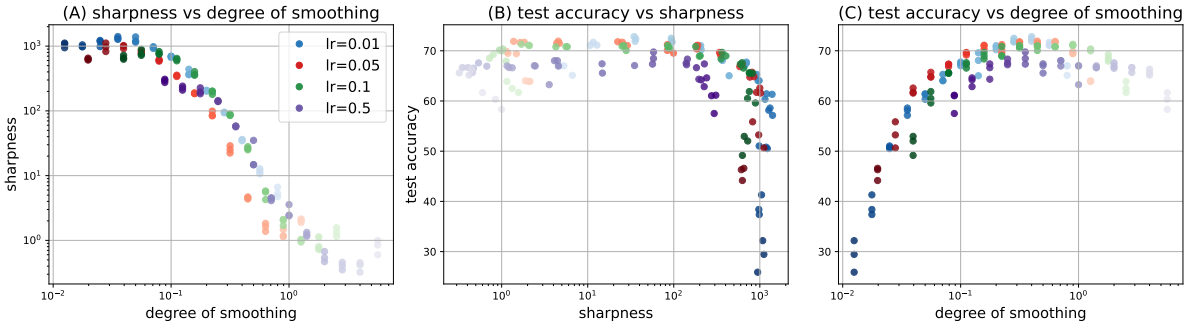

Fig. 4 **(A)** Sharpness versus degree of smoothing calculated from learning rate, batch size, and the estimated variance of the stochastic gradient. **(B)** Test accuracy after 200 epochs ResNet18 training on the CIFAR100 dataset versus sharpness. **(C)** Test accuracy versus degree of smoothing. The color shading in the scatter plots represents the batch size: the larger the batch size, the darker the color of the plotted points. "lr" means learning rate.

4. Relationship between degree of smoothing, sharpness, and generalizability (Section 5). To support our theory that simply using SGD for optimization smoothes the objective function and that the degree of smoothing is determined by $\delta = \eta C/\sqrt{b}$, we experimentally confirmed the relationship between the sharpness of the function around the approximate solution to which the optimizer converges and the degree of smoothing. We showed that the degree of smoothing is clearly able to express the smoothness/sharpness of the function as well as the well-studied "sharpness" indicator (Figure 4 (A)), and that there is a clear relationship between generalization performance and the degree of smoothing; generalization performance is clearly a concave function with respect to the degree of smoothing (Figure 4 (C)). Our results follow up on a previous study [67] that found, through extensive experiments, correlations between generalization performance and hyperparameters such as the learning rate, but no correlation between it and sharpness.

Section 2. Preliminaries

2.1 Notations and Definitions

Let $\mathbb{N}$ be the set of nonnegative integers. For $m \in \mathbb{N} \setminus \{0\}$, define $[m] := \{1, 2, \dots, m\}$. Let $\mathbb{R}^d$ be a d-dimensional Euclidean space with inner product $\langle \cdot, \cdot \rangle$, which induces the norm $\| \cdot \|$. I_d denotes a $d \times d$ identity matrix. $B(\boldsymbol{y}; r)$ is the Euclidean closed ball of radius r centered at $\boldsymbol{y}$, i.e., $B(\boldsymbol{y}; r) := \left\{ \boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x} - \boldsymbol{y}\| \leq r \right\}$. Let $\mathcal{N}(\boldsymbol{\mu}; \Sigma)$ be a d-dimensional Gaussian distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and variance $\Sigma \in \mathbb{R}^{d \times d}$. The DNN is parameterized by a vector $\boldsymbol{x} \in \mathbb{R}^d$, which is optimized by minimizing the empirical loss function $f(\boldsymbol{x}) := \frac{1}{n} \sum_{i \in [n]} f_i(\boldsymbol{x})$, where $f_i(\boldsymbol{x})$ is the loss function for $\boldsymbol{x} \in \mathbb{R}^d$ and the i-th training data $\boldsymbol{z}_i$ ($i \in [n]$). Let ξ be a random variable that does not depend on $\boldsymbol{x} \in \mathbb{R}^d$, and let $\mathbb{E}_\xi[X]$ denote the expectation with respect to ξ of a random variable X. $\xi_{t,i}$ is a random variable generated from the i-th sampling at time t, and $\boldsymbol{\xi}_t := (\xi_{t,1}, \xi_{t,2}, \dots, \xi_{t,b})$ is independent of $(\boldsymbol{x}_k)_{k=0}^t \subset \mathbb{R}$, where b ($\leq n$) is the batch size. The independence of $\boldsymbol{\xi}_0, \boldsymbol{\xi}_1, \dots$ allows us to define the total expectation $\mathbb{E}$ as $\mathbb{E} = \mathbb{E}_{\boldsymbol{\xi}_0} \mathbb{E}_{\boldsymbol{\xi}_1} \cdots \mathbb{E}_{\boldsymbol{\xi}_t}$. Let $\mathsf{G}_{\boldsymbol{\xi}_t}(\boldsymbol{x})$ be the stochastic gradient of $f(\cdot)$ at $\boldsymbol{x} \in \mathbb{R}^d$. The mini-batch $\mathcal{S}_t$ consists of b samples at time t, and the mini-batch stochastic gradient of $f(\boldsymbol{x}_t)$ for $\mathcal{S}_t$ is defined as $\nabla f_{\mathcal{S}_t}(\boldsymbol{x}_t) := \frac{1}{b} \sum_{i \in [b]} \mathsf{G}_{\xi_{t,i}}(\boldsymbol{x}_t)$.

2.2 Assumptions and Lemma

We make the following assumptions:

Assumption 2.1

(A1) $f \colon \mathbb{R}^d \to \mathbb{R}$ is continuously differentiable and L_g-smooth, i.e., for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$,

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq L_g \|\boldsymbol{x} - \boldsymbol{y}\|.$$

(A2) $f\colon \mathbb{R}^d \to \mathbb{R}$ is an L_f-Lipschitz function, i.e., for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$,

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq L_f \|\boldsymbol{x} - \boldsymbol{y}\|.$$

(A3) Let $(\boldsymbol{x}_t)_{t\in\mathbb{N}} \subset \mathbb{R}^d$ be the sequence generated by SGD.
(i) For each iteration t,

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

(ii) There exists a nonnegative constant C^2 such that

$$\mathbb{E}_{\xi_t}\left[\|\mathsf{G}_{\xi_t}(\boldsymbol{x}_t) - \nabla f(\boldsymbol{x}_t)\|^2\right] \leq C^2.$$

(A4) For each iteration t, SGD samples a mini-batch $\mathcal{S}_t \subset \mathcal{S}$ and estimates the full gradient ∇f as

$$\nabla f_{\mathcal{S}_t}(\boldsymbol{x}_t) := \frac{1}{b}\sum_{i\in[b]} \mathsf{G}_{\xi_{t,i}}(\boldsymbol{x}_t) = \frac{1}{b}\sum_{\{i\colon \boldsymbol{z}_i\in\mathcal{S}_t\}} \nabla f_i(\boldsymbol{x}_t).$$

The following lemma is very important to our theory. The proof of Lemma 2.1 can be found in Appendix C.

Lemma 2.1 Suppose that (A3)(ii) and (A4) hold for all $t \in \mathbb{N}$; then,

$$\mathbb{E}_{\xi_t}\left[\|\nabla f_{\mathcal{S}_t}(\boldsymbol{x}_t) - \nabla f(\boldsymbol{x}_t)\|^2\right] \leq \frac{C^2}{b}.$$

2.3 Function Smoothing

Definition 2.1 (Smoothed function) Given a function $f\colon \mathbb{R}^d \to \mathbb{R}$, define $\hat{f}_\delta\colon \mathbb{R}^d \to \mathbb{R}$ to be the function obtained by smoothing f as

$$\hat{f}_\delta(\boldsymbol{x}) := \mathbb{E}_{\boldsymbol{u}\sim\mathcal{L}}\left[f(\boldsymbol{x} - \delta\boldsymbol{u})\right],$$

where $\delta > 0$ represents the degree of smoothing and $\boldsymbol{u}$ is a random variable from a any light-tailed distribution $\mathcal{L}$ with $\mathbb{E}_{\boldsymbol{u}\sim\mathcal{L}}\left[\|\boldsymbol{u}\|\right] \leq 1$. Also,

$$\boldsymbol{x}^\star := \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}) \quad \text{and} \quad \boldsymbol{x}_\delta^\star := \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^d} \hat{f}_\delta(\boldsymbol{x}).$$

Note that, in the definition of the smoothed function $\hat{f}_\delta$, [61] defined that the random variable $\boldsymbol{u}$ follows a uniform distribution from the unit Euclidean ball. In contrast, from experimental results in Section H, we define the random variable $\boldsymbol{u}$ to follow any light-tailed distribution. Since the uniform distribution is a light-tailed distribution (see Section I.3), our Definition 2.1 contains Definition 4.1 of [61] and does not conflict with it. The graduated optimization algorithm uses several smoothed functions with

different noise levels. There are a total of M noise levels $(\delta_m)_{m\in[M]}$ and smoothed functions $(\hat{f}_{\delta_m})_{m\in[M]}$ in this paper. The largest noise level is δ_1 and the smallest is δ_M(see also Figure 3). For all $m \in [M]$, $(\hat{\boldsymbol{x}}_t^{(m)})_{t\in\mathbb{N}}$ is the sequence generated by an optimizer to minimize $\hat{f}_{\delta_m}$. Here, this paper refers to the graduated optimization approach with explicit smoothing operations (Definition 2.1) as "explicit graduated optimization" and to the graduated optimization approach with implicit smoothing operations as "implicit graduated optimization". All previous studies (see Section 1.1) have considered explicit graduated optimization, and we consider implicit graduated optimization for the first time.

Finally, we present an example of function smoothing. Figure 5 plots the single-variable Rastrigin function [68, 69] defined as in Equation (4) and its smoothed version, computed according to definition 2.1 with Gaussian noise $u \sim \mathcal{N}(0; 1)$

$$\text{(Rastrigin's function)} \quad f(x) := x^2 - 10\cos(2\pi x) + 10 \tag{4}$$

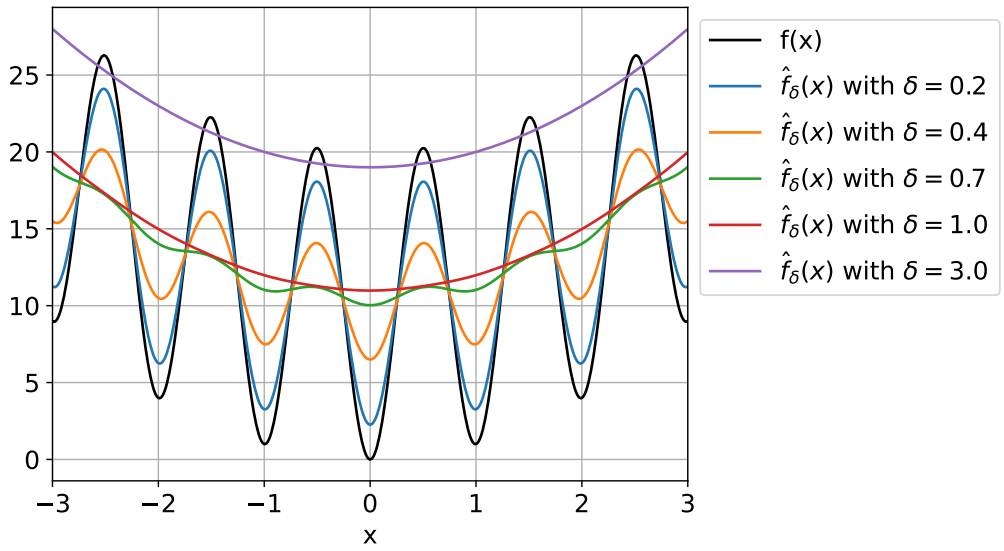

Fig. 5 Original objective function f and smoothed function $\hat{f}_\delta$ computed using degree of smoothing $\delta \in \{0.2, 0.4, 0.7, 1.0, 3.0\}$.

Figure 5 shows that as δ increases, the function becomes smoother and the valleys of the function gradually become less and less smooth. At $\delta = 1.0$, function $\hat{f}_\delta$ is completely convex.

Section 3. SGD's smoothing property

This section discusses the smoothing effect of using stochastic gradients. From Lemma 2.1, we have

$$\mathbb{E}_{\xi_t}\left[\|\boldsymbol{\omega}_t\|\right] \leq \frac{C}{\sqrt{b}},$$

due to $\boldsymbol{\omega}_t := \nabla f_{\mathcal{S}_t}(\boldsymbol{x}_t) - \nabla f(\boldsymbol{x}_t)$. The $\boldsymbol{\omega}_t$ for which this equation is satisfied can be expressed as $\boldsymbol{\omega}_t = \frac{C}{\sqrt{b}}\boldsymbol{u}_t$, where $\mathbb{E}_{\xi_t}\left[\|\boldsymbol{u}_t\|\right] \leq 1$. Here, we assume that $\boldsymbol{\omega}_t$ in image classification tasks with CNN-based models follows a light-tailed distribution in accordance with experimental observations in several previous studies [70, 71] and our experimental results (see Section I.4). Therefore, $\boldsymbol{\omega}_t \sim \hat{\mathcal{L}}$ and thereby $\boldsymbol{u}_t \sim \mathcal{L}$, where $\hat{\mathcal{L}}$ and $\mathcal{L}$ are light-tailed distributions and $\mathcal{L}$ is a scaled version of $\hat{\mathcal{L}}$. Then, using Definition 2.1, we further transform Equation (3) as follows:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\omega}_t}\left[\boldsymbol{y}_{t+1}\right] &= \mathbb{E}_{\boldsymbol{\omega}_t}\left[\boldsymbol{y}_t\right] - \eta\nabla\mathbb{E}_{\boldsymbol{\omega}_t}\left[f(\boldsymbol{y}_t - \eta\boldsymbol{\omega}_t)\right] \\
&= \mathbb{E}_{\boldsymbol{\omega}_t}\left[\boldsymbol{y}_t\right] - \eta\nabla\mathbb{E}_{\boldsymbol{u}_t\sim\mathcal{L}}\left[f\left(\boldsymbol{y}_t - \frac{\eta C}{\sqrt{b}}\boldsymbol{u}_t\right)\right] \\
&= \mathbb{E}_{\boldsymbol{\omega}_t}\left[\boldsymbol{y}_t\right] - \eta\nabla\hat{f}_{\frac{\eta C}{\sqrt{b}}}(\boldsymbol{y}_t). \qquad (5)
\end{aligned}$$

This shows that $\mathbb{E}_{\boldsymbol{\omega}_t}\left[f(\boldsymbol{y}_t - \eta\boldsymbol{\omega}_t)\right]$ is a smoothed version of f with a noise level $\eta C/\sqrt{b}$ and its parameter $\boldsymbol{y}_t$ can be approximately updated by using the GD to minimize $\hat{f}_{\frac{\eta C}{\sqrt{b}}}$. Therefore, we can say that the degree of smoothing δ by the stochastic noise $\boldsymbol{\omega}_t$ in SGD is determined by the learning rate η, the batch size b, and the variance of the stochastic gradient C^2 and that optimizing the function f with SGD and optimizing the smoothed function $\hat{f}_{\frac{\eta C}{\sqrt{b}}}$ with GD are equivalent in the sense of expectation.

There are still more discoveries that can be made from the finding that simply by using SGD for optimization, the objective function is smoothed and the degree of smoothing is determined by $\delta = \eta C/\sqrt{b}$.

Why the Use of Large Batch Sizes Leads to Solutions Falling into Sharp Local Minima. It is known that training with large batch sizes leads to a persistent degradation of model generalization performance. In particular, [20] showed experimentally that learning with large batch sizes leads to sharp local minima and worsens generalization performance. According to Equation (5), using a large learning rate and/or a small batch size will make the function smoother. Thus, in using a small batch size, the sharp local minima will disappear through extensive smoothing, and SGD can reach a flat local minimum. Conversely, when using a large batch size, the smoothing is weak and the function is close to the original multimodal function, so it is easy for the solution to fall into a sharp local minimum. Thus, we have theoretical support for what [20] showed experimentally, and our experiments have yielded similar results (see Figure 7 (a)).

Why Decaying Learning Rates and Increasing Batch Sizes are Superior to Fixed Learning Rates and Batch Sizes. From Equation (5), the use of a decaying learning rate or increasing batch size during training is equivalent to decreasing the noise level of the smoothed function, so using a decaying learning rate or increasing the batch size is an implicit graduated optimization. Thus, we can say that using a decaying learning rate [72, 73, 74, 75] or increasing batch size [76, 77, 78, 79, 6, 80] makes sense in terms of avoiding local minima and provides theoretical support for their experimental superiority.

Section 4. Implicit Graduated Optimization

In this section, we construct an implicit graduated optimization algorithm that varies the learning rate η and batch size b so that the degree of smoothing $\delta = \eta C/\sqrt{b}$ by stochastic noise in SGD gradually decreases and then analyze its convergence.

4.1 New theoretical findings on σ-nice function

In order to analyze the graduated optimization algorithm, Hazan et al. defined σ-nice functions (see Definition I.1), a family of nonconvex functions that has favorable conditions for a graduated optimization algorithm to converge to a global optimal solution [61]. We define the following function, which is a slight extension of the σ-nice function. See Appendix I for details on its extension.

Definition 4.1 (σ_m-nice function) Let $M \in \mathbb{N}$, $m \in [M]$, and $\gamma \in [0.5, 1)$. A function $f\colon \mathbb{R}^d \to \mathbb{R}$ is said to be σ_m-nice if the following two conditions hold:

(i) For all $\delta_m > 0$ and all $\boldsymbol{x}^\star_{\delta_m}$, there exists $\boldsymbol{x}^\star_{\delta_{m+1}}$ such that:

$$\left\| \boldsymbol{x}^\star_{\delta_m} - \boldsymbol{x}^\star_{\delta_{m+1}} \right\| \leq \delta_{m+1} := \gamma \delta_m.$$

(ii) For all $\delta_m > 0$, the function $\hat{f}_{\delta_m}(\boldsymbol{x})$ over $N(\boldsymbol{x}^\star_{\delta_m}; 3\delta_m)$ is σ_m-strongly convex.

The σ-nice property implies that optimizing the smoothed function $\hat{f}_{\delta_m}$ is a good start for optimizing the next smoothed function $\hat{f}_{\delta_{m+1}}$, which has been shown to be sufficient for graduated optimization [61]. We will take this route and consider an implicit graduated optimization algorithm for σ_m-nice functions. One might still think that the definition of the σ_m-nice function is not practical and that there is a gap between theory and application. However, we can show that the commonly used empirical loss function is a σ_m-nice function under certain assumptions (The proof of Proposition 4.1 is in Appendix I.2).

Proposition 4.1 In the definition of the smoothed function (Definition 2.1), suppose that $\boldsymbol{u}$ follows a standard Gaussian distribution. Then, the cross entropy loss is a 1-nice function and the mean squared error is a 2-nice function.

The cross entropy loss and mean squared error are the most common empirical loss functions in classification and regression tasks, respectively, and Proposition 4.1 im-

plies that they are σ_m-nice functions. This discovery opens up an unlimited range of applications of graduated optimization in machine learning, and not only the theoretical results that we will present, but all previous theoretical and experimental findings on graduated optimization can be incorporated into machine learning through σ_m-nice functions.

4.2 Analysis of Implicit Graduated Optimization

Algorithm 4.1 embodies the framework of implicit graduated optimization with SGD for σ_m-nice functions, while Algorithm 4.2 is used to optimize each smoothed function; it should be GD (see (5)). Note that our implicit graduated optimization (Algorithm 4.1) is achieved by SGD with decaying learning rate and/or increasing batch size.

Algorithm 4.1 Implicit Graduated Optimization

Require: $\epsilon, \boldsymbol{x}_1 \in B(\boldsymbol{x}^\star_{\delta_1}; 3\delta_1), \eta_1 > 0, b_1 \in [n], \gamma \geq 0.5$
  $\delta_1 = \frac{\eta_1 C}{\sqrt{b_1}}, \alpha_0 = \min\left\{\frac{1}{16 L_f \delta_1}, \frac{1}{\sqrt{2\sigma}\delta_1}\right\}, M = \log_\gamma \alpha_0 \epsilon$
  for $m = 1$ to $M + 1$ **do**
    if $m \neq M + 1$ **then**
      $\epsilon_m := \sigma_m \delta_m^2 / 2,\ T_m := H_m / \epsilon_m$
      $\kappa_m / \sqrt{\lambda_m} = \gamma$ $(\kappa_m \in (0, 1], \lambda_m \geq 1)$
    end if
    $\boldsymbol{x}_{m+1} := \mathrm{GD}(T_m, \boldsymbol{x}_m, \hat{f}_{\delta_m}, \eta_m)$
    $\eta_{m+1} := \kappa_m \eta_m, b_{m+1} := \lambda_m b_m$
    $\delta_{m+1} := \frac{\eta_{m+1} C}{\sqrt{b_{m+1}}}$
  end for
  return $\boldsymbol{x}_{M+2}$

Algorithm 4.2 Gradient Descent

Require: $T_m, \hat{\boldsymbol{x}}_1^{(m)}, \hat{f}_{\delta_m}, \eta > 0$
  for $t = 1$ to T_m **do**
    $\hat{\boldsymbol{x}}_{t+1}^{(m)} := \hat{\boldsymbol{x}}_t^{(m)} - \eta \nabla \hat{f}_{\delta_m}(\boldsymbol{x}_t)$
  end for
  return $\hat{\boldsymbol{x}}_{T_m+1}^{(m)}$

From the definition of σ_m-nice function, the smoothed function $\hat{f}_{\delta_m}$ is σ_m-strongly convex in $B(\boldsymbol{x}^\star_{\delta_m}; 3\delta_m)$. Also, the learning rate used by Algorithm 4.2 to optimize $\hat{f}_{\delta_m}$ should always be constant. Therefore, let us now consider the convergence of GD with a constant learning rate for a σ_m-strongly convex function $\hat{f}_{\delta_m}$. The proof of Theorem 4.1 is in Appendix F.1.

Theorem 4.1 (Convergence analysis of Algorithm 4.2) Suppose that $\hat{f}_{\delta_m}\colon \mathbb{R}^d \to \mathbb{R}$ is a σ_m-strongly convex and L_g-smooth and $\eta < \min\left\{\frac{1}{\sigma_m}, \frac{2}{L_g}\right\}$. Then, the sequence $(\hat{\boldsymbol{x}}_t^{(m)})_{t\in\mathbb{N}}$ generated by Algorithm 4.2 satisfies

$$\min_{t\in[T]} \hat{f}_{\delta_m}\left(\hat{\boldsymbol{x}}_t^{(m)}\right) - \hat{f}_{\delta_m}(\boldsymbol{x}_{\delta_m}^\star) \leq \frac{H_m}{T} = \mathcal{O}\left(\frac{1}{T}\right), \tag{6}$$

where $H_m := \frac{9(1-\sigma_m\eta)\delta_m^2}{2\eta} + \frac{3L_f\delta_m}{\eta(2-L_g\eta)}$ is a nonnegative constant.

Theorem 4.1 shows that Algorithm 4.2 can reach an ϵ_m-neighborhood of the optimal solution $\boldsymbol{x}_{\delta_m}^\star$ of $\hat{f}_{\delta_m}$ in approximately $T_m := H_m/\epsilon_m$ iterations. The next proposition is crucial to the success of Algorithm 4.1 and guarantees the soundness of the σ_m-nice function (The proof is in Appendix F.2).

Proposition 4.2 Let f be a σ_m-nice function and $\delta_{m+1} := \gamma\delta_m$. Suppose that $\gamma \in [0.5, 1)$ and $\boldsymbol{x}_1 \in B(\boldsymbol{x}_{\delta_1}^\star; 3\delta_1)$. Then for all $m \in [M]$, $\|\boldsymbol{x}_m - \boldsymbol{x}_{\delta_m}^\star\| < 3\delta_m$.

$\boldsymbol{x}_m$ is the approximate solution obtained by optimization of the smoothed function $\hat{f}_{\delta_{m-1}}$ with Algorithm 4.2 and is the initial point of optimization of the next smoothed function $\hat{f}_{\delta_m}$. Therefore, Proposition 4.2 implies that $\gamma \in [0.5, 1)$ must hold for the initial point of optimization of $\hat{f}_{\delta_m}$ to be contained in the strongly convex region of $\hat{f}_{\delta_m}$. Therefore, from Theorem 4.1 and Proposition 4.2, if f is a σ_m-nice function and $\boldsymbol{x}_1 \in B(\boldsymbol{x}_{\delta_1}^\star; 3\delta_1)$ holds, the sequence $(\boldsymbol{x}_m)_{m\in[M]}$ generated by Algorithm 4.1 never goes outside of the σ_m-strongly convex region $B(\boldsymbol{x}_{\delta_m}^\star; 3\delta_m)$ of each smoothed function $\hat{f}_{\delta_m}$ $(m \in [M])$.

The next theorem guarantees the convergence of Algorithm 4.1 with the σ_m-nice function (The proof of Theorem 4.2 is in Appendix F.3). Note that Theorem 4.2 provides a total complexity including those of Algorithm 4.1 and Algorithm 4.2, because Algorithm 4.1 uses Algorithm 4.2 at each $m \in [M]$.

Theorem 4.2 (Convergence analysis of Algorithm 4.1) Let $\epsilon \in (0, 1)$ and $f\colon \mathbb{R}^d \to \mathbb{R}$ be an L_f-Lipschitz σ_m-nice function. Suppose that we run Algorithm 4.1; then after $\mathcal{O}\left(1/\epsilon^2\right)$ rounds, the algorithm reaches an ϵ-neighborhood of the global optimal solution $\boldsymbol{x}^\star$.

4.3 Numerical Results

Full experimental results and our code are in Appendix G. We compared four types of SGD for image classification: *1.* constant learning rate and constant batch size, *2.* decaying learning rate and constant batch size, *3.* constant learning rate and increasing batch size, *4.* decaying learning rate and increasing batch size, in training ResNet34 [81] on the ImageNet dataset [82] (Figure 6), ResNet18 on the CIFAR100 dataset (Figure 8 in Appendix G), and WideResNet-28-10 [83] on the CIFAR100 dataset (Figure 9 in Appendix G). Therefore, methods 2, 3, and 4 are our Algorithm 4.1. All experiments were run for 200 epochs. In methods 2, 3, and 4,

the noise decreased every 40 epochs, with a common decay rate of $1/\sqrt{2}$. That is, every 40 epochs, the learning rate of method 2 was multiplied by $1/\sqrt{2}$, the batch size of method 3 was doubled, and the learning rate and batch size of method 4 were respectively multiplied by $\sqrt{3}/2$ and 1.5. Note that this $1/\sqrt{2}$ decay rate is γ in Algorithm 4.1 and it satisfies the condition in Proposition 4.2. The initial learning rate was 0.1 for all methods, which was determined by performing a grid search among $[0.01, 0.1, 1.0, 10]$. The noise reduction interval was every 40 epochs, which was determined by performing a grid search among $[10, 20, 25, 40, 50, 100]$. A history of the learning rate or batch size for each method is provided in the caption of each figure.

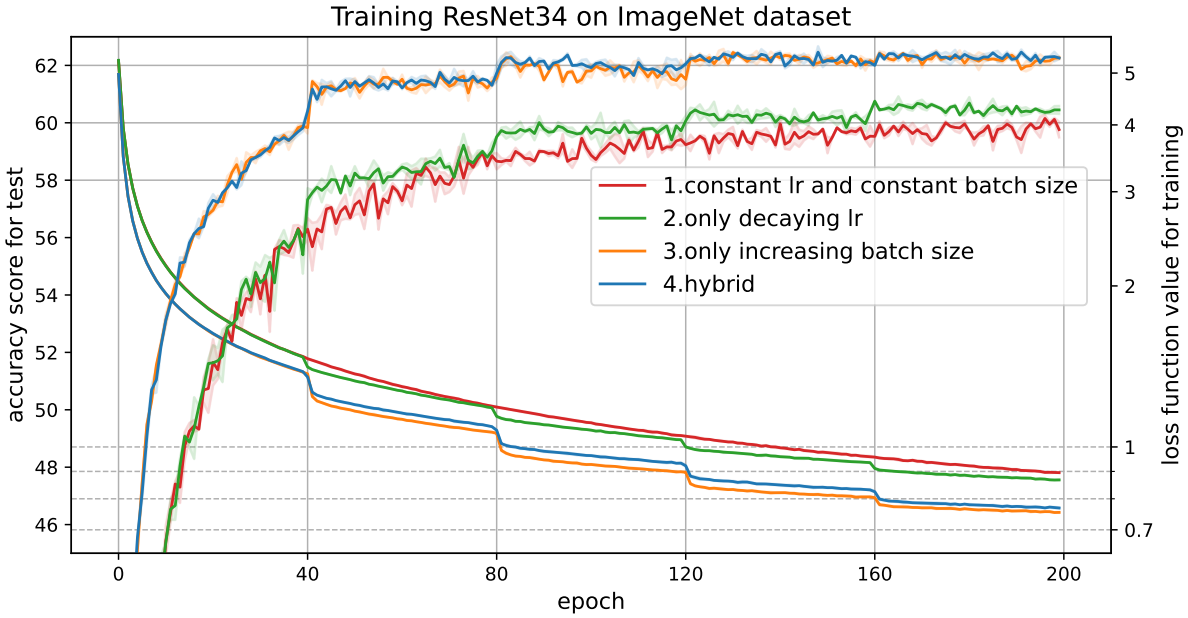

Fig. 6 Accuracy score for the testing and loss function value for training versus the number of epochs in training ResNet34 on the ImageNet dataset. The solid line represents the mean value, and the shaded area represents the maximum and minimum over three runs. In method 1, the learning rate and batch size were fixed at 0.1 and 256, respectively. In method 2, the learning rate was decreased every 40 epochs as $\left[0.1, \frac{1}{10\sqrt{2}}, 0.05, \frac{1}{20\sqrt{2}}, 0.025\right]$ and the batch size was fixed at 256. In method 3, the learning rate was fixed at 0.1, and the batch size was increased as $[32, 64, 128, 256, 512]$. In method 4, the learning rate was decreased as $\left[0.1, \frac{\sqrt{3}}{20}, 0.075, \frac{3\sqrt{3}}{80}, 0.05625\right]$ and the batch size was increased as $[32, 48, 72, 108, 162]$.

For methods 2, 3, and 4, the decay rates are all $1/\sqrt{2}$, and the decay intervals are all 40 epochs, so throughout the training, the three methods should theoretically be optimizing the exact same five smoothed functions in sequence. Nevertheless, the local solutions reached by each of the three methods are not exactly the same. All results indicate that method 3 is superior to method 2 and that method 4 is superior

to method 3 in both test accuracy and training loss function values. This difference can be attributed to the different learning rates used to optimize each smoothing function. Among methods 2, 3, and 4, method 3, which does not decay the learning rate, maintains the highest learning rate 0.1, followed by method 4 and method 2. In all graphs, the loss function values are always small in that order; i.e., the larger the learning rate is, the lower loss function values become. Therefore, we can say that the noise level δ, expressed as $\frac{\eta C}{\sqrt{b}}$, needs to be reduced, while the learning rate η needs to remain as large as possible. Alternatively, if the learning rate is small, then a large number of iterations are required. Thus, for the same rate of change and the same number of epochs, an increasing batch size is superior to a decreasing learning rate because it can maintain a large learning rate and can be made to iterate a lot when the batch size is small.

Theoretically, the noise level δ_m should gradually decrease and become zero at the end, so in our Algorithm 4.1, the learning rate η_m should be zero at the end or the batch size b_m should match the number of data sets at the end. However, if the learning rate is 0, training cannot proceed, and if the batch size is close to a full batch, it is not feasible from a computational point of view. For this reason, the experiments described in this paper are not fully graduated optimizations; i.e., full global optimization is not achieved. In fact, the last batch size used by method 2 is around 128 to 512, which is far from a full batch. Therefore, the solution reached in this experiment is the optimal one for a function that has been smoothed to some extent, and to achieve a global optimization of the DNN, it is necessary to increase only the batch size to eventually reach a full batch, or increase the number of iterations accordingly while increasing the batch size and decaying the learning rate.

Section 5. Relationship between degree of smoothing, sharpness, and generalizability

The graduated optimization algorithm is a method in which the degree of smoothing δ is gradually decreased. Let us consider the case where the degree of smoothing δ is constant throughout the training. The following lemma shows the relationship between the error of the original function value and that of the smoothed function value.

Lemma 5.1 Let $\hat{f}_\delta$ be the smoothed version of f; then, for all $\boldsymbol{x} \in \mathbb{R}^d$,

$$\left| \hat{f}_\delta(\boldsymbol{x}) - f(\boldsymbol{x}) \right| \leq \delta L_f.$$

Here, a larger degree of smoothing should be necessary to make many local optimal solutions of the objective function f disappear and lead the optimizer to the global optimal solution. On the other hand, Lemma 5.1 implies that the larger the degree of smoothing is, the further away the smoothed function will be from the original function. Therefore, there should be an optimal value for the degree of smoothing

that balances the tradeoffs, because if the degree of smoothing is too large, the original function is too damaged and thus cannot be optimized properly, and if it is too small, the function is not smoothed enough and the optimizer falls into a local optimal solution. This knowledge is useful because the degree of smoothing due to stochastic noise in SGD is determined by the learning rate and batch size (see Section 3), so when a constant learning rate and constant batch size are used, the degree of smoothing is constant throughout the training.

The smoothness of the function, and in particular the sharpness of the function around the approximate solution to which the optimizer converged, has been well studied because it has been thought to be related to the generalizability of the model. In this section, we reinforce our theory by experimentally observing the relationship between the degree of smoothing and the sharpness of the function.

Several previous studies [19, 20, 84, 85, 67] have addressed the relationship between the sharpness of the function around the approximate solution to which the optimizer converges and the generalization performance of the model. In particular, the hypothesis that flat local solutions have better generalizability than sharp local solutions is at the core of a series of discussions, and several previous studies [20, 21, 22, 23, 24] have developed measures of sharpness to confirm this. In this paper, we use "adaptive sharpness" [24, 67] as a measure of the sharpness of the function that is invariant to network reparametrization, highly correlated with generalization, and generalizes several existing sharpness definitions. In accordance with [67], let $\mathcal{S}$ be a set of training data; for arbitrary model weights $\boldsymbol{w} \in \mathbb{R}^d$, the worst-case adaptive sharpness with radius $\rho \in \mathbb{R}$ and with respect to a vector $\boldsymbol{c} \in \mathbb{R}^d$ is defined as

$$S^{\rho}_{\max}(\boldsymbol{w}, \boldsymbol{c}) := \mathbb{E}_{\mathcal{S}} \left[\max_{\|\boldsymbol{\delta} \odot \boldsymbol{c}^{-1}\|_p \leq \rho} f(\boldsymbol{w} + \boldsymbol{\delta}) - f(\boldsymbol{w}) \right],$$

where $\odot/^{-1}$ denotes elementwise multiplication/inversion. Thus, the larger the sharpness value is, the sharper the function around the model weight $\boldsymbol{w}$ becomes, with a smaller sharpness leading to higher generalizability.

We trained ResNet18 [81] with the learning rate $\eta \in \{0.01, 0.05, 0.1, 0.1\}$ and batch size $b \in \{2^1, \ldots, 2^{13}\}$ for 200 epochs on the CIFAR100 dataset [86] and then measured the worst-case l_∞ adaptive sharpness of the obtained approximate solution with radius $\rho = 0.0002$ and $\boldsymbol{c} = (1, 1, \ldots, 1)^\top \in \mathbb{R}^d$. Our implementation was based on [67] and the code used is available on our anonymous GitHub. Figure 7 plots the relationship between measured sharpness and the batch size b and the learning rate η used for training as well as the degree of smoothing δ calculated from them. Figure 7 also plots the relationship between test accuracy, sharpness, and degree of smoothing. Three experiments were conducted per combination of learning rate and batch size, with a total of 156 data plots. The variance of the stochastic gradient C^2 included in the degree of smoothing $\delta = \eta C/\sqrt{b}$ used values estimated from theory and experiment (see Appendix B for details).

Figure 7 (a) shows that the larger the batch size is, the larger the sharpness value becomes, whereas (b) shows that the larger the learning rate is, the smaller the

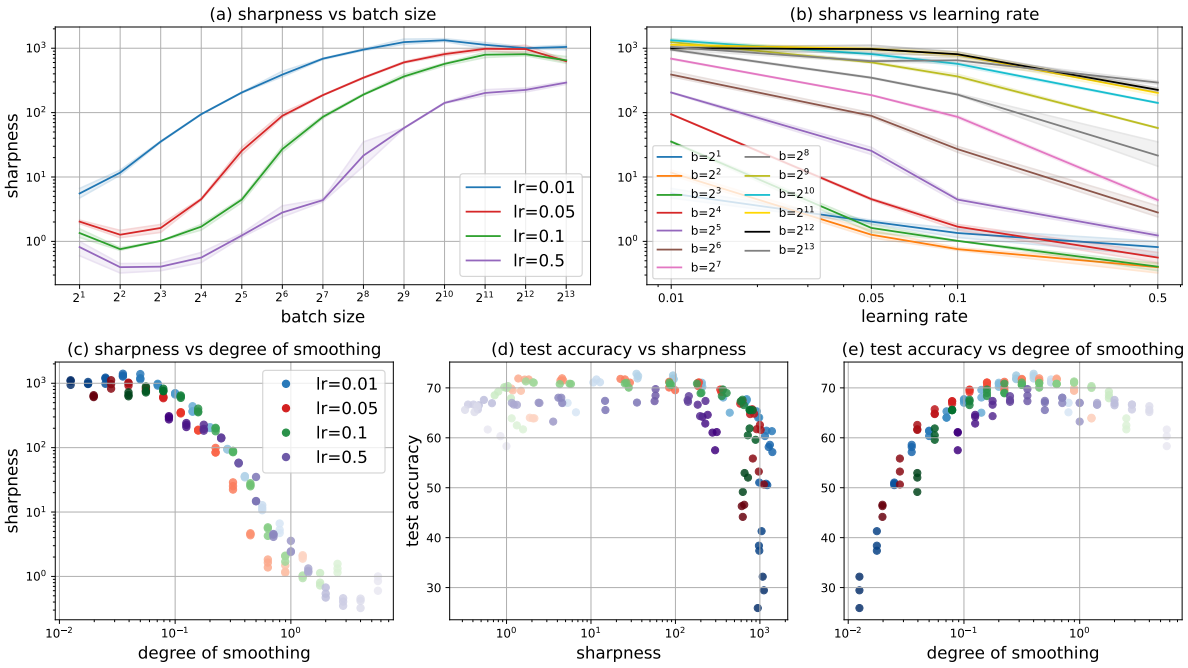

Fig. 7 **(a)** Sharpness around the approximate solution after 200 epochs of ResNet18 training on the CIFAR100 dataset versus batch size used. **(b)** Sharpness versus learning rate used. **(c)** Sharpness versus degree of smoothing calculated from learning rate, batch size, and estimated variance of the stochastic gradient. **(d)** Test accuracy after 200 epochs training versus sharpness. **(e)** Test accuracy versus degree of smoothing. The solid line represents the mean value, and the shaded area represents the maximum and minimum over three runs. The color shade in the scatter plots represents the batch size; the larger the batch size, the darker the color of the plotted points. "lr" means learning rate. The experimental results that make up the all graphs are all identical.

sharpness becomes, and (c) shows a greater the degree of smoothing for a smaller sharpness. These experimental results guarantee the our theoretical result that the degree of smoothing δ is proportional to the learning rate η and inversely proportional to the batch size b, and they reinforce our theory that the quantity $\eta C/\sqrt{b}$ is the degree of smoothing of the function. Figure 7 (d) also shows that there is no clear correlation between the generalization performance of the model and the sharpness around the approximate solution. This result is also consistent with previous study [67]. On the other hand, Figure 7 (e) shows an excellent correlation between generalization performance and the degree of smoothing; generalization performance is clearly a concave function with respect to the degree of smoothing. Thus, a degree of smoothing that is neither too large nor too small leads to high generalization performance. This experimental observation can be supported theoretically (see Lemma 5.1). That is, if the degree of smoothing is a constant throughout the training, then there should be an optimal value for the loss function value or test accuracy; for the training of ResNet18 on the CIFAR100 dataset, for example, 0.1 to 1 was the desired value (see Figure 7

(e)). For degrees of smoothing smaller than 0.1, the generalization performance is not good because the function is not sufficiently smoothed so that locally optimal solutions with sharp neighborhoods do not disappear, and the optimizer falls into this trap. On the other hand, a degree of smoothing greater than 1 leads to excessive smoothing and smoothed function becomes too far away from the original function to be properly optimized; the generalization performance is not considered excellent. In addition, the optimal combination of learning rate and batch size that practitioners search for by using grid searches or other methods when training models can be said to be a search for the optimal degree of smoothing. If the optimal degree of smoothing can be better understood, the huge computational cost of the search could be reduced.

[67] observed the relationship between sharpness and generalization performance in extensive experiments and found that they were largely uncorrelated, suggesting that the sharpnesss may not be a good indicator of generalization performance and that one should avoid blanket statements like "flatter minima generalize better". Figure 7 (d) and (e) show that there is no correlation between sharpness and generalization performance, as in previous study, while there is a correlation between degree of smoothing and generalization performance. Therefore, we can say that degree of smoothing may be a good indicator to theoretically evaluate generalization performance, and it may be too early to say that "flatter minima generalize better" is invalid.

Section 6. Conclusion

We proved that SGD with a mini-batch stochastic gradient has the effect of smoothing the function, and the degree of smoothing is greater with larger learning rates and smaller batch sizes. This shows theoretically that smoothing with large batch sizes is makes it easy to fall into sharp local minima and that using a decaying learning rate and/or increasing batch size is implicitly graduated optimization, which makes sense in the sense that it avoids local optimal solutions. Building on these findings, we proposed a new graduated optimization algorithm for a σ_m-nice function that uses a decaying learning rate and increasing batch size and analyzed it. Our finding that the commonly used empirical loss functions, i.e., the cross entropy loss and mean squared error, are σ_m-nice functions will dramatically stimulate the application of graduated optimization in machine learning. We also conducted experiments whose results showed the superiority of our recommended framework for image classification tasks on CIFAR100 and ImageNet. In addition, we observed that the degree of smoothing of the function due to stochastic noise in SGD can express the degree of smoothness of the function as well as sharpness does, and that the degree of smoothing is a good indicator of the generalization performance of the model.

Acknowledgments

This research was partly supported by the computational resources of the DGX A100 computing system, TAIHO, at Meiji University and by the Tsuchiya Cultural Foundation. I am deeply grateful to Professor Hideaki Iiduka, Department of Computer Science, Faculty of Science and Technology, Meiji University, for his support and encouragement. I gratefully acknowledge the contributions of past and present members of our laboratory, especially Hiroyuki Sakai, who created a very helpful LaTeX template for composing papers. I extend my heartfelt thanks to everyone who has supported this research.

References

[1] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 1951.

[2] D. P. Kingma and J. L. Ba, "A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, 2015, pp. 1–15.

[3] E. Moulines and F. Bach, "Non-asymptotic analysis of stochastic approximation algorithms for machine learning," in *Proceedings of the 25th Annual Conference on Neural Information Processing Systems*, vol. 24, 2011.

[4] D. Needell, R. A. Ward, and N. Srebro, "Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm," in *Proceedings of the 28th Annual Conference on Neural Information Processing Systems*, vol. 27, 2014, pp. 1017–1025.

[5] B. Fehrman, B. Gess, and A. Jentzen, "Convergence rates for the stochastic gradient descent method for non-convex objective functions," *Journal of Machine Learning Research*, vol. 21, pp. 1–48, 2020.

[6] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[7] K. Scaman and C. Malherbe, "Robustness analysis of non-convex stochastic gradient descent using biased expectations," in *Proceedings of the 34th Conference on Neural Information Processing Systems*, vol. 33, 2020, pp. 16 377–16 387.

[8] N. Loizou, S. Vaswani, I. Laradji, and S. Lacoste-Julien, "Stochastic polyak step-size for SGD: An adaptive learning rate for fast convergence: An adaptive learning rate for fast convergence," in *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 130, 2021.

[9] M. Zaheer, S. J. Reddi, D. Sachan, S. Kale, and S. Kumar, "Adaptive methods for nonconvex optimization," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, vol. 31, 2018.

[10] F. Zou, L. Shen, Z. Jie, W. Zhang, and W. Liu, "A sufficient condition for convergences of adam and rmsprop," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11 119–11 127, 2019.

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

[12] D. Zhou, J. Chen, Y. Cao, Y. Tang, Z. Yang, and Q. Gu, "On the convergence of adaptive gradient methods for nonconvex optimization," *12th Annual Workshop on Optimization for Machine Learning*, 2020.

[13] J. Chen, D. Zhou, Y. Tang, Z. Yang, Y. Cao, and Q. Gu, "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*, vol. 452, 2021, pp. 3267–3275.

[14] H. Iiduka, "Appropriate learning rates of adaptive learning rate optimization algorithms for training deep neural networks." *IEEE Transactions on Cybernetics*, vol. 52, no. 12, pp. 13 250–13 261, 2022.

[15] M. Hardt, B. Recht, and Y. Singer, "Train faster, generalize better: Stability of stochastic gradient descent," in *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48, 2016, pp. 1225–1234.

[16] J. Lin, R. Camoriano, and L. Rosasco, "Generalization properties and implicit regularization for multiple passes SGM," in *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48, 2016, pp. 2340–2348.

[17] W. Mou, L. Wang, X. Zhai, and K. Zheng, "Generalization bounds of SGLD for non-convex learning: Two theoretical viewpoints," in *Proceedings of the 31st Annual Conference on Learning Theory*, vol. 75, 2018, pp. 605–638.

[18] F. He, T. Liu, and D. Tao, "Control batch size and learning rate to generalize well: Theoretical and empirical evidence," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2019, pp. 1141–1150.

[19] S. Hochreiter and J. Schmidhuber, "Flat minima," *MIT Press*, vol. 9, no. 1, pp. 1–42, 1997.

[20] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," in *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[21] T. Liang, T. A. Poggio, A. Rakhlin, and J. Stokes, "Fisher-rao metric, geometry, and complexity of neural networks," in *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, vol. 89, 2019, pp. 888–896.

[22] Y. Tsuzuku, I. Sato, and M. Sugiyama, "Normalized flat minima: Exploring scale invariant definition of flat minima for neural networks using pac-bayesian analysis," in *Proceedings of the 37th International Conference on Machine Learning*, vol. 119, 2020, pp. 9636–9647.

[23] H. Petzka, M. Kamp, L. Adilova, C. Sminchisescu, and M. Boley, "Relative flatness and generalization," in *Proceedings of the 34th Conference on Neural Information Processing Systems*, 2021, pp. 18 420–18 432.

[24] J. Kwon, J. Kim, H. Park, and I. K. Choi, "ASAM: adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks," in *Proceedings of the 38th International Conference on Machine Learning*, vol. 139, 2021, pp.

5905–5914.

[25] R. Ge, F. Huang, C. Jin, and Y. Yuan, "Escaping from saddle points - online stochastic gradient for tensor decomposition," in *Proceedings of the 28th Conference on Learning Theory*, vol. 40, 2015, pp. 797–842.

[26] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordam, "How to escape saddle points efficiently," vol. http://arxiv.org/abs/1703.00887, 2017.

[27] H. Daneshmand, J. M. Kohler, A. Lucchi, and T. Hofmann, "Escaping saddles with stochastic gradients," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018, pp. 1163–1172.

[28] H. Vardhan and S. U. Stich, "Escaping local minima with stochastic noise," in *the 13th International OPT Workshop on Optimization for Machine Learning in NeurIPS 2021*, 2021.

[29] Y. Zhang, P. Liang, and M. Charikar, "A hitting time analysis of stochastic gradient langevin dynamics," in *Proceedings of the 30th Conference on Learning Theory*, vol. 65, 2017, pp. 1980–2022.

[30] M. Zhou, T. Liu, Y. Li, D. Lin, E. Zhou, and T. Zhao, "Toward understanding the importance of noise in training neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, 2019, pp. 7594–7602.

[31] C. Jin, P. Netrapalli, R. Ge, S. M. Kakade, and M. I. Jordan, "On nonconvex optimization for machine learning: Gradients, stochasticity, and saddle points," *Journal of the ACM*, vol. 68, no. 2, pp. 1–29, 2021.

[32] A. Orvieto, H. Kersting, F. Proske, F. R. Bach, and A. Lucchi, "Anticorrelated noise injection for improved generalization," in *Proceedings of the 39th International Conference on Machine Learning*, vol. 162, 2022, pp. 17 094–17 116.

[33] R. Kleinberg, Y. Li, and Y. Yuan, "An alternative view: When does SGD escape local minima?" in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018, pp. 2703–2712.

[34] Z. Wu, "The effective energy transformation scheme as a special continuation approach to global optimization with application to molecular conformation," *SIAM Journal on Optimization*, vol. 6, no. 3, pp. 748–768, 1996.

[35] A. Blake and A. Zisserman, *Visual Reconstruction*. MIT Press, 1987.

[36] A. P. Witkin, D. Terzopoulos, and M. Kass, "Signal matching through scale space," *Intenational Jounral of Computer Vision*, vol. 1, no. 2, pp. 133–144, 1987.

[37] A. L. Yuille, "Energy functions for early vision and analog networks," *Biological Cybernetics*, vol. 61, no. 2, pp. 115–123, 1989.

[38] E. L. Allgower and K. Georg, *Numerical continuation methods - an introduction*. Springer, 1990, vol. 13.

[39] K. Rose, E. Gurewitz, and G. Fox, "A deterministic annealing approach to clustering," *Pattern Recognition Letters*, vol. 11, no. 9, pp. 589–594, 1990.

[40] M. J. Black and A. Rangarajan, "On the unification of line processes, outlier rejection, and robust statistics with applications in early vision," *International Journal of Computer Vision*, vol. 19, no. 1, pp. 57–91, 1996.

[41] M. Nikolova, M. K. Ng, and C.-P. Tam, "Fast nonconvex nonsmooth minimiza-

tion methods for image restoration and reconstruction," *IEEE Transactions on Image Processing*, vol. 19, no. 12, 2010.

[42] D. Sun, S. Roth, and M. J. Black, "Secrets of optical flow estimation and their principles," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2432–2439.

[43] T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," *IEEE Transactions on Pattern Analysis and Machine Learning*, vol. 33, no. 3, pp. 500–513, 2011.

[44] J. Kim, C. Liu, F. Sha, and K. Grauman, "Deformable spatial pyramid matching for fast dense correspondences," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2307–2314.

[45] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone, "Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1127–1134, 2020.

[46] P. Antonante, V. Tzoumas, H. Yang, and L. Carlone, "Outlier-robust estimation: Hardness, minimally tuned algorithms, and applications," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 281–301, 2022.

[47] L. Peng, C. Kümmerle, and R. Vidal, "On the convergence of IRLS and its variants in outlier-robust estimation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 808–17 818.

[48] O. Chapelle, M. Chi, and A. Zien, "A continuation method for semi-supervised SVMs," in *Proceedings of the 23rd International Conference on Machine Learning*, vol. 148, 2006, pp. 185–192.

[49] V. Sindhwani, S. S. Keerthi, and O. Chapelle, "Deterministic annealing for semi-supervised kernel machines," in *Proceedings of the 23rd International Conference on Machine Learning*, vol. 148, 2006, pp. 841–848.

[50] O. Chapelle, V. Sindhwani, and S. S. Keerthi, "Optimization techniques for semi-supervised support vector machines," *Journal of Machine Learning Research*, vol. 9, pp. 203–233, 2008.

[51] N. A. Smith and J. Eisner, "Annealing techniques for unsupervised statistical language learning," in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, 2004, pp. 486–493.

[52] O. Chapelle and M. Wu, "Gradient descent optimization of smoothed information retrieval metrics," *Information retrieval*, vol. 13, no. 3, pp. 216–235, 2010.

[53] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019, pp. 11 895–11 907.

[54] Y. Song, J. Sohl-Dickstein, D. P. kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *Proceedings of the 9th International Conference on Learning Represantations*, 2021.

[55] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37, 2015, pp. 2256–

2265.

[56] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proceedings of the 34th Conference on Neural Information Processing Systems*, 2020.

[57] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *Proceedings of the 9th International Conference on Learning Represantations*, 2021.

[58] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[59] H. Mobahi and J. W. Fisher III, "On the link between gaussian homotopy continuation and convex envelopes," in *Proceedings of the 10th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, vol. 8932, 2015, pp. 43–56.

[60] ——, "A theoretical analysis of optimization by gaussian continuation," in *Proceedings of the 39th AAAI Conference on Artificial Intelligence*, 2015, pp. 1205–1211.

[61] E. Hazan, K. Yehuda, and S. Shalev-Shwartz, "On graduated optimization for stochastic non-convex problems," in *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48, 2016, pp. 1833–1841.

[62] D. Li, J. Wu, and Q. Zhang, "Stochastic gradient descent in the viewpoint of graduated optimization," vol. https://arxiv.org/abs/2308.06775, 2023.

[63] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright, "Randomized smoothing for stochastic optimization," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 674–701, 2012.

[64] E. Hoffer, I. Hubara, and D. Soudry, "Train longer, generalize better: closing the generalization gap in large batch training of neural networks," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 1731–1741.

[65] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch SGD: training imagenet in 1 hour," vol. https://arxiv.org/abs/1706.02677, 2017.

[66] Y. You, J. Li, S. J. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C. Hsieh, "Large batch optimization for deep learning: Training BERT in 76 minutes," in *Proceedings of the 8th International Conference on Learning Representations*, 2020.

[67] M. Andriushchenko, F. Croce, M. Müller, M. Hein, and N. Flammarion, "A modern look at the relationship between sharpness and generalization," in *Proceedings of the 40th International Conference on Machine Learning*, vol. 202, 2023, pp. 840–902.

[68] A. A. Törn and A. Zilinskas, *Global Optimization*, ser. Lecture Notes in Computer Science. Springer, 1989, vol. 350.

[69] G. Rudolph, "Globale optimierung mit parallelen evolutionsstrategien," Ph.D. dissertation, Universität Dortmund Fachbereich Informatik, 7 1990.

[70] J. Zhang, S. P. Karimireddy, A. Veit, S. Kim, S. Kumar, and S. Sra, "Why are

adaptive methods good for attention models?" in *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, 2020.

[71] F. Kunstner, J. Chen, J. W. Lavington, and M. Schmidt, "Noise is not the main factor behind the gap between SGD and adam on transformers, but sign descent might be," in *Proceedings of the 8th International Conference on Learning Representations*, 2023.

[72] I. Loshchilov and F. Hutter, "SGDR: stochastic gradient descent with warm restarts," in *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[73] A. Hundt, V. Jain, and G. D. Hager, "sharpDARTS: faster and more accurate differentiable architecture search," vol. https://arxiv.org/abs/1903.09900, 2019.

[74] K. You, M. Long, J. Wang, and M. I. Jordam, "How does learning rate decay help modern neural networks?" vol. https://arxiv.org/abs/1908.01878, 2019.

[75] A. Lewkowycz, "How to decay your learning rate," vol. https://arxiv.org/abs/2103.12682, 2021.

[76] R. H. Byrd, G. M. Chin, J. Nocedal, and Y. Wu, "Sample size selection in optimization methods for machine learning," *Mathematical Programming*, vol. 134, no. 1, pp. 127–155, 2012.

[77] M. P. Friedlander and M. Schmidt, "Hybrid deterministic-stochastic methods for data fitting," *SIAM Journal on Scientific Computing*, vol. 34, no. 3, 2012.

[78] L. Balles, J. Romero, and P. Hennig, "Coupling adaptive batch sizes with learning rates," in *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence*, 2017.

[79] S. De, A. K. Yadav, D. W. Jacobs, and T. Goldstein, "Automated inference with adaptive batches," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, vol. 54, 2017, pp. 1504–1513.

[80] S. L. Smith, P. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," in *Proceedings of the 6th International Conference on Learning Representations*, 2018.

[81] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[82] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database." in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[83] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proceedings of the British Machine Vision Conference*, 2016.

[84] P. Izmailov, D. Podoprikhin, T. Garipov, D. P. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," in *Proceedings of the 34th Conference on Uncertainly in Artificial Intelligence*, 2018, pp. 876–885.

[85] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, 2018, pp. 6391–6401.

[86] A. Krizhevsky, "Learning multiple layers of features from tiny images," vol. https:

//www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf, 2009.

[87] A. Shapiro, D. Dentcheva, and A. Ruszczynski, *Lectures on Stochastic Programming - Modeling and Theory*, ser. MOS-SIAM Series on Optimization. SIAM, 2009.

[88] K. Imaizumi and H. Iiduka, "Iteration and stochastic first-order oracle complexities of stochastic gradient descent using constant and decaying learning rates," *Optimization*, vol. 0, no. 0, pp. 1–24, 2024.

[89] N. Sato and H. Iiduka, "Role of momentum in smoothing objective function and generalizability of deep neural networks," vol. https://arxiv.org/abs/2402.02325, 2024.

[90] H. Iwakiri, Y. Wang, S. Ito, and A. Takeda, "Single loop gaussian homotopy method for non-convex optimization," in *Proceedings of the 36th Conference on Neural Information Processing Systems*, 2022.

[91] C. S. Marcin Molga, "Test functions for optimization needs," https://robertmarks.org/Classes/ENGR5358/Papers/functions.pdf, 4 2005.

[92] P. L.A., K. Jagannathan, and R. Kolla, "Concentration bounds for CVaR estimation: The cases of light-tailed and heavy-tailed distributions," in *Proceedings of the 37th International Conference on Machine Learning*, vol. 119, 2020, pp. 5577–5586.

A Derivation of Equation (3)

Let $\boldsymbol{y}_t$ be the parameter updated by gradient descent (GD) and $\boldsymbol{x}_{t+1}$ be the parameter updated by SGD at time t, i.e.,

$$\begin{aligned}\boldsymbol{y}_t &:= \boldsymbol{x}_t - \eta \nabla f(\boldsymbol{x}_t),\\ \boldsymbol{x}_{t+1} &:= \boldsymbol{x}_t - \eta \nabla f_{\mathcal{S}_t}(\boldsymbol{x}_t)\\ &= \boldsymbol{x}_t - \eta(\nabla f(\boldsymbol{x}_t) + \boldsymbol{\omega}_t).\end{aligned}$$

Then, we have

$$\begin{aligned}\boldsymbol{x}_{t+1} &:= \boldsymbol{x}_t - \eta \nabla f_{\mathcal{S}_t}(\boldsymbol{x}_t)\\ &= (\boldsymbol{y}_t + \eta \nabla f(\boldsymbol{x}_t)) - \eta \nabla f_{\mathcal{S}_t}(\boldsymbol{x}_t)\\ &= \boldsymbol{y}_t - \eta \boldsymbol{\omega}_t, \end{aligned} \tag{7}$$

from $\boldsymbol{\omega}_t := \nabla f_{\mathcal{S}_t}(\boldsymbol{x}_t) - \nabla f(\boldsymbol{x}_t)$. Hence,

$$\begin{aligned}\boldsymbol{y}_{t+1} &= \boldsymbol{x}_{t+1} - \eta \nabla f(\boldsymbol{x}_{t+1})\\ &= \boldsymbol{y}_t - \eta \boldsymbol{\omega}_t - \eta \nabla f(\boldsymbol{y}_t - \eta \boldsymbol{\omega}_t).\end{aligned}$$

By taking the expectation with respect to $\boldsymbol{\omega}_t$ on both sides, we have, from $\mathbb{E}_{\boldsymbol{\omega}_t}[\boldsymbol{\omega}_t] = \mathbf{0}$,

$$\mathbb{E}_{\boldsymbol{\omega}_t}[\boldsymbol{y}_{t+1}] = \mathbb{E}_{\boldsymbol{\omega}_t}[\boldsymbol{y}_t] - \eta \nabla \mathbb{E}_{\boldsymbol{\omega}_t}[f(\boldsymbol{y}_t - \eta \boldsymbol{\omega}_t)],$$

where we have used $\mathbb{E}_{\boldsymbol{\omega}_t}[\nabla f(\boldsymbol{y}_t - \eta \boldsymbol{\omega}_t)] = \nabla \mathbb{E}_{\boldsymbol{\omega}_t}[f(\boldsymbol{y}_t - \eta \boldsymbol{\omega}_t)]$, which holds for a Lipschitz continuous and differentiable f [87, Theorem 7.49]. In addition, from (7) and $\mathbb{E}_{\boldsymbol{\omega}_t}[\boldsymbol{\omega}_t] = \mathbf{0}$, we obtain

$$\mathbb{E}_{\boldsymbol{\omega}_t}[\boldsymbol{x}_{t+1}] = \boldsymbol{y}_t.$$

Therefore, on average, the parameter $\boldsymbol{x}_{t+1}$ of the function f arrived at by SGD coincides with the parameter $\boldsymbol{y}_t$ of the smoothed function $\hat{f}(\boldsymbol{y}_t) := \mathbb{E}_{\boldsymbol{\omega}_t}[f(\boldsymbol{y}_t - \eta \boldsymbol{\omega}_t)]$ arrived at by GD.

B Estimation of variance of stochastic gradient

In Section 5, we need to estimate the variance C^2 of the stochastic gradient in order to plot the degree of smoothing $\delta = \eta C/\sqrt{b}$. In general, this is difficult to measure, but several previous studies [88, 89] have provided the following estimating formula. For some $\epsilon > 0$, when training until $\frac{1}{T}\sum_{k=1}^{T} \mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] \leq \epsilon^2$, the variance of the stochastic gradient can be estimated as

$$C^2 < \frac{b^\star \epsilon^2}{\eta},$$

where $b^\star$ is the batch size that minimizes the amount of computation required for training and η is learning rate used in training. We determined the stopping condition ϵ for each learning rate, measured the batch size that minimized the computational complexity required for the gradient norm of the preceding t steps at time t to average less than ϵ in training ResNet18 on the CIFAR100 dataset, and estimated the variance of the stochastic gradient by using an estimation formula (see Table 1). Table 2 shows the results of a similar experiment for the training WideResNet(WRN)-28-10 on the CIFAR100 dataset.

Table 1 Learning rate η and threshold ϵ used for training, measured optimal batch size $b^\star$ and estimated variance of the stochastic gradient C^2 in training ResNet18 on the CIFAR100 dataset.

η	ϵ	$b^\star$	C^2
0.01	1.0	2^7	12800
0.05	0.5	2^9	1280
0.1	0.5	2^{10}	1280
0.5	0.5	2^{10}	256

Table 2 Learning rate η and threshold ϵ used for training, measured optimal batch size $b^\star$ and estimated variance of the stochastic gradient C^2 in training WRN-28-10 on the CIFAR100 dataset.

η	ϵ	$b^\star$	C^2
0.01	1.0	2^2	400
0.05	0.5	2^2	20
0.1	0.5	2^2	10
0.5	0.5	2^2	2

C Proofs of the Lemmas

C.1 Proof of Lemma 2.1

(Proof) (A3)(ii) and (A4) guarantee that

$$\begin{aligned}
\mathbb{E}_{\xi_t}\left[\|\nabla f_{\mathcal{S}_t}(\boldsymbol{x}_t)-\nabla f(\boldsymbol{x}_t)\|^2\right] &= \mathbb{E}_{\xi_t}\left[\left\|\frac{1}{b}\sum_{i=1}^{b}\mathsf{G}_{\xi_{t,i}}(\boldsymbol{x}_t)-\nabla f(\boldsymbol{x}_t)\right\|^2\right]\\
&= \mathbb{E}_{\xi_t}\left[\left\|\frac{1}{b}\sum_{i=1}^{b}\mathsf{G}_{\xi_{t,i}}(\boldsymbol{x}_t)-\frac{1}{b}\sum_{i=1}^{b}\nabla f(\boldsymbol{x}_t)\right\|^2\right]\\
&= \mathbb{E}_{\xi_t}\left[\left\|\frac{1}{b}\sum_{i=1}^{b}\left(\mathsf{G}_{\xi_{t,i}}(\boldsymbol{x}_t)-\nabla f(\boldsymbol{x}_t)\right)\right\|^2\right]\\
&= \frac{1}{b^2}\mathbb{E}_{\xi_t}\left[\left\|\sum_{i=1}^{b}\left(\mathsf{G}_{\xi_{t,i}}(\boldsymbol{x}_t)-\nabla f(\boldsymbol{x}_t)\right)\right\|^2\right]\\
&= \frac{1}{b^2}\mathbb{E}_{\xi_t}\left[\sum_{i=1}^{b}\left\|\mathsf{G}_{\xi_{t,i}}(\boldsymbol{x}_t)-\nabla f(\boldsymbol{x}_t)\right\|^2\right]\\
&\leq \frac{C^2}{b}.
\end{aligned}$$

This completes the proof. **(Q.E.D.)**

C.2 Proof of Lemma 5.1

(Proof) From Definition 2.1 and (A2), we have, for all $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^d$,

$$\begin{aligned}
\left|\hat{f}_\delta(\boldsymbol{x})-f(\boldsymbol{x})\right| &= \left|\mathbb{E}_{\boldsymbol{u}}\left[f(\boldsymbol{x}-\delta\boldsymbol{u})\right]-f(\boldsymbol{x})\right|\\
&= \left|\mathbb{E}_{\boldsymbol{u}}\left[f(\boldsymbol{x}-\delta\boldsymbol{u})-f(\boldsymbol{x})\right]\right|\\
&\leq \mathbb{E}_{\boldsymbol{u}}\left[\left|f(\boldsymbol{x}-\delta\boldsymbol{u})-f(\boldsymbol{x})\right|\right]\\
&\leq \mathbb{E}_{\boldsymbol{u}}\left[L_f\|(\boldsymbol{x}-\delta\boldsymbol{u})-\boldsymbol{x}\|\right]\\
&= \delta L_f\mathbb{E}_{\boldsymbol{u}}\left[\|\boldsymbol{u}\|\right]\\
&\leq \delta L_f.
\end{aligned}$$

This completes the proof. **(Q.E.D.)**

D Lemmas on smoothed function

The following Lemmas concern the properties of smoothed functions $\hat{f}_\delta$.

Lemma D.1 Suppose that (A1) holds; then, $\hat{f}_\delta$ defined by Definition 2.1 is also L_g-smooth; i.e., for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$,

$$\left\| \nabla \hat{f}_\delta(\boldsymbol{x}) - \nabla \hat{f}_\delta(\boldsymbol{y}) \right\| \leq L_g \|\boldsymbol{x} - \boldsymbol{y}\|.$$

(Proof) From Definition 2.1 and (A1), we have, for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$,

$$\begin{aligned}
\left\| \nabla \hat{f}_\delta(\boldsymbol{x}) - \nabla \hat{f}_\delta(\boldsymbol{y}) \right\| &= \|\nabla \mathbb{E}_{\boldsymbol{u}} \left[f(\boldsymbol{x} - \delta \boldsymbol{u}) \right] - \nabla \mathbb{E}_{\boldsymbol{u}} \left[f(\boldsymbol{y} - \delta \boldsymbol{u}) \right] \| \\
&= \|\mathbb{E}_{\boldsymbol{u}} \left[\nabla f(\boldsymbol{x} - \delta \boldsymbol{u}) \right] - \mathbb{E}_{\boldsymbol{u}} \left[\nabla f(\boldsymbol{y} - \delta \boldsymbol{u}) \right] \| \\
&= \|\mathbb{E}_{\boldsymbol{u}} \left[\nabla f(\boldsymbol{x} - \delta \boldsymbol{u}) - \nabla f(\boldsymbol{y} - \delta \boldsymbol{u}) \right] \| \\
&\leq \mathbb{E}_{\boldsymbol{u}} \left[\| \nabla f(\boldsymbol{x} - \delta \boldsymbol{u}) - \nabla f(\boldsymbol{y} - \delta \boldsymbol{u}) \| \right] \\
&\leq \mathbb{E}_{\boldsymbol{u}} \left[L_g \, \| (\boldsymbol{x} - \delta \boldsymbol{u}) - (\boldsymbol{y} - \delta \boldsymbol{u}) \| \right] \\
&= \mathbb{E}_{\boldsymbol{u}} \left[L_g \, \| \boldsymbol{x} - \boldsymbol{y} \| \right] \\
&= L_g \| \boldsymbol{x} - \boldsymbol{y} \|.
\end{aligned}$$

This completes the proof. **(Q.E.D.)**

Lemma D.2 Suppose that (A2) holds; then $\hat{f}_\delta$ is also an L_f-Lipschitz function; i.e., for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$,

$$\left| \hat{f}_\delta(\boldsymbol{x}) - \hat{f}_\delta(\boldsymbol{y}) \right| \leq L_f \|\boldsymbol{x} - \boldsymbol{y}\|.$$

(Proof) From Definition 2.1 and (A2), we have, for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$,

$$\begin{aligned}
\left| \hat{f}_\delta(\boldsymbol{x}) - \hat{f}_\delta(\boldsymbol{y}) \right| &= |\mathbb{E}_{\boldsymbol{u}} \left[f(\boldsymbol{x} - \delta \boldsymbol{u}) \right] - \mathbb{E}_{\boldsymbol{u}} \left[f(\boldsymbol{y} - \delta \boldsymbol{u}) \right]| \\
&= |\mathbb{E}_{\boldsymbol{u}} \left[f(\boldsymbol{x} - \delta \boldsymbol{u}) - f(\boldsymbol{y} - \delta \boldsymbol{u}) \right]| \\
&\leq \mathbb{E}_{\boldsymbol{u}} \left[| f(\boldsymbol{x} - \delta \boldsymbol{u}) - f(\boldsymbol{y} - \delta \boldsymbol{u}) | \right] \\
&\leq \mathbb{E}_{\boldsymbol{u}} \left[L_f \| (\boldsymbol{x} - \delta \boldsymbol{u}) - (\boldsymbol{y} - \delta \boldsymbol{u}) \| \right] \\
&= \mathbb{E}_{\boldsymbol{u}} \left[L_f \, \| \boldsymbol{x} - \boldsymbol{y} \| \right] \\
&= L_f \| \boldsymbol{x} - \boldsymbol{y} \|.
\end{aligned}$$

This completes the proof. **(Q.E.D.)**

Lemmas D.1 and D.2 imply that the Lipschitz constants L_f of the original function f and L_g of ∇f are taken over by the smoothed function $\hat{f}_\delta$ and its gradient $\nabla \hat{f}_\delta$ for all $\delta \in \mathbb{R}$.

E Lemmas used in the proofs of the theorems

Lemma E.1 Suppose that $\hat{f}_{\delta_m}\colon \mathbb{R}^d \to \mathbb{R}$ is σ_m-strongly convex and $\hat{\boldsymbol{x}}_{t+1}^{(m)} := \hat{\boldsymbol{x}}_t^{(m)} - \eta_t \boldsymbol{g}_t$. Then, for all $t \in \mathbb{N}$,

$$\hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)}) - \hat{f}_{\delta_m}(\boldsymbol{x}^\star) \leq \frac{1-\sigma_m\eta_t}{2\eta_t} X_t - \frac{1}{2\eta_t} X_{t+1} + \frac{\eta_t}{2}\|\boldsymbol{g}_t\|^2,$$

where $\boldsymbol{g}_t := \nabla \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)})$, $X_t := \|\hat{\boldsymbol{x}}_t^{(m)} - \boldsymbol{x}_{\delta_m}^\star\|^2$, and $\boldsymbol{x}_{\delta_m}^\star$ is the global minimizer of $\hat{f}_{\delta_m}$. **(Proof)** Let $t \in \mathbb{N}$. The definition of $\hat{\boldsymbol{x}}_{t+1}^{(m)}$ guarantees that

$$\begin{aligned}\|\hat{\boldsymbol{x}}_{t+1}^{(m)} - \boldsymbol{x}^\star\|^2 &= \|(\hat{\boldsymbol{x}}_t^{(m)} - \eta_t \boldsymbol{g}_t) - \boldsymbol{x}^\star\|^2 \\ &= \|\hat{\boldsymbol{x}}_t^{(m)} - \boldsymbol{x}^\star\|^2 - 2\eta_t \langle \hat{\boldsymbol{x}}_t^{(m)} - \boldsymbol{x}_{\delta_m}^\star, \boldsymbol{g}_t\rangle + \eta_t^2\|\boldsymbol{g}_t\|^2.\end{aligned}$$

From the σ_m-strong convexity of $\hat{f}_{\delta_m}$,

$$\|\hat{\boldsymbol{x}}_{t+1}^{(m)} - \boldsymbol{x}_{\delta_m}^\star\|^2 \leq \|\hat{\boldsymbol{x}}_t^{(m)} - \boldsymbol{x}_{\delta_m}^\star\|^2 + 2\eta_t\left(\hat{f}_{\delta_m}(\boldsymbol{x}_{\delta_m}^\star) - \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)}) - \frac{\sigma_m}{2}\|\hat{\boldsymbol{x}}_t^{(m)} - \boldsymbol{x}_{\delta_m}^\star\|^2\right) + \eta_t^2\|\boldsymbol{g}_t\|^2.$$

Hence,

$$\hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)}) - \hat{f}_{\delta_m}(\boldsymbol{x}_{\delta_m}^\star) \leq \frac{1-\sigma_m\eta_t}{2\eta_t}\|\hat{\boldsymbol{x}}_t^{(m)} - \boldsymbol{x}_{\delta_m}^\star\|^2 - \frac{1}{2\eta_t}\|\hat{\boldsymbol{x}}_{t+1}^{(m)} - \boldsymbol{x}_{\delta_m}^\star\|^2 + \frac{\eta_t}{2}\|\boldsymbol{g}_t\|^2.$$

This completes the proof. **(Q.E.D.)**

Lemma E.2 Suppose that $\hat{f}_{\delta_m}\colon \mathbb{R}^d \to \mathbb{R}$ is L_g-smooth and $\hat{\boldsymbol{x}}_{t+1}^{(m)} := \hat{\boldsymbol{x}}_t^{(m)} - \eta_t \boldsymbol{g}_t$. Then, for all $t \in \mathbb{N}$,

$$\eta_t\left(1 - \frac{L_g\eta_t}{2}\right)\|\nabla \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)})\|^2 \leq \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)}) - \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_{t+1}^{(m)}).$$

where $\boldsymbol{g}_t := \nabla \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)})$ and $\boldsymbol{x}_{\delta_m}^\star$ is the global minimizer of $\hat{f}_{\delta_m}$. **(Proof)** From the L_g-smoothness of the $\hat{f}_{\delta_m}$ and the definition of $\hat{\boldsymbol{x}}_{t+1}^{(m)}$, we have, for all $t \in \mathbb{N}$,

$$\begin{aligned}\hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_{t+1}^{(m)}) &\leq \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)}) + \langle \nabla \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)}), \hat{\boldsymbol{x}}_{t+1}^{(m)} - \hat{\boldsymbol{x}}_t^{(m)}\rangle + \frac{L_g}{2}\|\hat{\boldsymbol{x}}_{t+1}^{(m)} - \hat{\boldsymbol{x}}_t^{(m)}\|^2 \\ &= \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)}) - \eta_t\langle \nabla \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)}), \boldsymbol{g}_t\rangle + \frac{L_g\eta_t^2}{2}\|\boldsymbol{g}_t\|^2 \\ &\leq \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)}) - \eta_t\left(1 - \frac{L_g\eta_t}{2}\right)\|\nabla \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)})\|^2.\end{aligned}$$

Therefore, we have

$$\eta_t\left(1 - \frac{L_g\eta_t}{2}\right)\|\nabla \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)})\|^2 \leq \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)}) - \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_{t+1}^{(m)}).$$

This completes the proof. **(Q.E.D.)**

Lemma E.3 Suppose that $\hat{f}_{\delta_m}\colon \mathbb{R}^d \to \mathbb{R}$ is L_g-smooth, $\hat{\boldsymbol{x}}_{t+1}^{(m)} := \hat{\boldsymbol{x}}_t^{(m)} - \eta_t \boldsymbol{g}_t$, and $\eta_t := \eta < \frac{2}{L_g}$. Then, for all $t \in \mathbb{N}$,

$$\frac{1}{T}\sum_{t=1}^{T} \|\boldsymbol{g}_t\|^2 \leq \frac{6L_f\delta_m}{\eta\left(2 - L_g\eta\right)T},$$

where $\boldsymbol{g}_t := \nabla \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)})$ and $\boldsymbol{x}_{\delta_m}^{\star}$ is the global minimizer of $\hat{f}_{\delta_m}$. **(Proof)** According to Lemma E.2, we have

$$\eta\left(1 - \frac{L_g\eta}{2}\right)\|\nabla F(\boldsymbol{x}_t^{(m)})\|^2 \leq \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)}) - \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_{t+1}^{(m)}).$$

Summing over t, we find that

$$\eta\left(1 - \frac{L_g\eta}{2}\right)\frac{1}{T}\sum_{t=1}^{T}\|\nabla \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)})\|^2 \leq \frac{\hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_1^{(m)}) - \hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_{T+1}^{(m)})}{T}.$$

Hence, from $\eta < \frac{2}{L_g}$,

$$\frac{1}{T}\sum_{t=1}^{T}\|\boldsymbol{g}_t\|^2 = \frac{2\left(\hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_1^{(m)}) - \hat{f}_{\delta_m}(\boldsymbol{x}_{\delta_m}^{\star})\right)}{\eta\left(2 - L_g\eta\right)T}.$$

Here, from the L_f-Lipschitz continuity of $\hat{f}_{\delta_m}$,

$$\begin{aligned}\hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_1^{(m)}) - \hat{f}_{\delta_m}(\boldsymbol{x}_{\delta_m}^{\star}) &\leq L_f\|\hat{\boldsymbol{x}}_1^{(m)} - \boldsymbol{x}_{\delta_m}^{\star}\| \\ &\leq 3L_f\delta_m,\end{aligned}$$

where we have used $\boldsymbol{x}_1^{(m)} \in B(\boldsymbol{x}_{\delta_m}^{\star}; 3\delta_m)$. Therefore, we have

$$\frac{1}{T}\sum_{t=1}^{T}\|\boldsymbol{g}_t\|^2 = \frac{6L_f\delta_m}{\eta\left(2 - L_g\eta\right)T}.$$

This completes the proof. **(Q.E.D.)**

F Proof of the Theorems and Propositions

F.1 Proof of Theorem 4.1

(Proof) Lemma E.1 guarantees that

$$\begin{aligned}\hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)}) - \hat{f}_{\delta_m}(\boldsymbol{x}_{\delta_m}^{\star}) &\leq \frac{1 - \sigma_m\eta_t}{2\eta_t}X_t - \frac{1}{2\eta_t}X_{t+1} + \frac{\eta_t}{2}\|\boldsymbol{g}_t\|^2 \\ &= \frac{1 - \sigma_m\eta}{2\eta}\left(X_t - X_{t+1}\right) - \frac{\sigma_m}{2}X_{t+1} + \frac{\eta}{2}\|\boldsymbol{g}_t\|^2.\end{aligned}$$

From $\eta < \min\left\{\frac{1}{\sigma_m}, \frac{2}{L_g}\right\}$ and Lemma E.3, by summing over t we find that

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T}\left(\hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)}) - \hat{f}_{\delta_m}(\boldsymbol{x}_{\delta_m}^{\star})\right) &\leq \frac{1-\sigma_m\eta}{2\eta T}\left(X_1 - X_{T+1}\right) - \frac{\sigma_m}{2T}\sum_{t=1}^{T} X_{t+1} \\
&\quad + \frac{\eta}{2T}\sum_{t=1}^{T}\|\boldsymbol{g}_t\|^2 \\
&\leq \frac{1-\sigma_m\eta}{2\eta T} X_1 + \frac{\eta}{2T}\sum_{t=1}^{T}\|\boldsymbol{g}_t\|^2 \\
&\leq \underbrace{\frac{9\left(1-\sigma_m\eta\right)\delta_m^2}{2\eta}}_{=:H_1}\frac{1}{T} + \underbrace{\frac{3L_f\delta_m}{\eta\left(2-L_g\eta\right)}}_{=:H_2}\frac{1}{T} \\
&= \underbrace{\left(H_1 + H_2\right)}_{=:H_m}\frac{1}{T} \\
&= \frac{H_m}{T},
\end{aligned}$$

where we have used $X_1 := \|\hat{\boldsymbol{x}}_1^{(m)} - \boldsymbol{x}_{\delta_m}^{\star}\|^2 \leq 9\delta_m^2$ and $H_m > 0$ is a nonnegative constant. From the convexity of F,

$$\hat{f}_{\delta_m}\left(\frac{1}{T}\sum_{t=1}^{T}\hat{\boldsymbol{x}}_t^{(m)}\right) \leq \frac{1}{T}\sum_{t=1}^{T}\hat{f}_{\delta_m}(\hat{\boldsymbol{x}}_t^{(m)}).$$

Hence,

$$\hat{f}_{\delta_m}\left(\frac{1}{T}\sum_{t=1}^{T}\hat{\boldsymbol{x}}_t^{(m)}\right) - \hat{f}_{\delta_m}(\boldsymbol{x}_{\delta_m}^{\star}) \leq \frac{H_m}{T} = \mathcal{O}\left(\frac{1}{T}\right).$$

In addition, since the minimum value is smaller than the mean, we have

$$\min_{t\in[T]}\left(\hat{f}_{\delta_m}\left(\hat{\boldsymbol{x}}_t^{(m)}\right) - \hat{f}_{\delta_m}(\boldsymbol{x}_{\delta_m}^{\star})\right) \leq \frac{H_m}{T} = \mathcal{O}\left(\frac{1}{T}\right).$$

This completes the proof. **(Q.E.D.)**

F.2 Proof of Proposition 4.2

(Proof) This proposition can be proved by induction. Since we assume $\boldsymbol{x}_1 \in N(\boldsymbol{x}_{\delta_1}^{\star}; 3\delta_1)$, we have

$$\|\boldsymbol{x}_1 - \boldsymbol{x}_{\delta_1}^{\star}\| < 3\delta_1,$$

which establishes the case of $m = 1$. Now let us assume that the proposition holds for any $m > 1$. Accordingly, the initial point $\boldsymbol{x}_m$ for the optimization of the m-th smoothed function $\hat{f}_{\delta_m}$ and its global optimal solution $\boldsymbol{x}^\star_{\delta_m}$ are both contained in the its σ_m-strongly convex region $N(\boldsymbol{x}^\star_{\delta_m}; 3\delta_m)$. Thus, after $T_m := H_m/\epsilon_m$ iterations, Algorithm 4.2 (GD) returns an approximate solution $\hat{\boldsymbol{x}}^{(m)}_{T_m+1} =: \boldsymbol{x}_{m+1}$, and the following holds from Theorem 4.1:

$$\hat{f}_{\delta_m}(\boldsymbol{x}_{m+1}) - \hat{f}_{\delta_m}(\boldsymbol{x}^\star_{\delta_m}) \leq \frac{H_m}{T_m} = \epsilon_m := \frac{\sigma_m \delta_m^2}{2} = \frac{\sigma_m \delta_{m+1}^2}{2\gamma^2}.$$

Hence, from the σ_m-strongly convexity of $\hat{f}_{\delta_m}$,

$$\frac{\sigma_m}{2}\|\boldsymbol{x}_{m+1} - \boldsymbol{x}^\star_{\delta_m}\|^2 \leq \frac{\sigma_m \delta_{m+1}^2}{2\gamma^2}, \text{ i.e., } \|\boldsymbol{x}_{m+1} - \boldsymbol{x}^\star_{\delta_m}\| \leq \frac{\delta_{m+1}}{\gamma}$$

Therefore, from the σ_m-niceness of f and $\gamma \in [0.5, 1)$,

$$\begin{aligned}
\|\boldsymbol{x}_{m+1} - \boldsymbol{x}^\star_{\delta_{m+1}}\| &\leq \|\boldsymbol{x}_{m+1} - \boldsymbol{x}^\star_{\delta_m}\| + \|\boldsymbol{x}^\star_{\delta_m} - \boldsymbol{x}^\star_{\delta_m+1}\| \\
&\leq \frac{\delta_{m+1}}{\gamma} + (|\delta_m| - \delta_{m+1}) \\
&= \frac{\delta_{m+1}}{\gamma} + \left(\frac{\delta_{m+1}}{\gamma} - \delta_{m+1}\right) \\
&= \left(\frac{2}{\gamma} - 1\right)\delta_{m+1} \\
&\leq 3\delta_{m+1}.
\end{aligned}$$

This completes the proof. **(Q.E.D.)**

F.3 Proof of Theorem 4.2

The following proof uses the technique presented in [61]. **(Proof)** According to $\delta_{m+1} := \frac{\eta_{m+1} C}{\sqrt{b_{m+1}}}$ and $\frac{\kappa_m}{\sqrt{\lambda_m}} = \gamma$, we have

$$\begin{aligned}
\delta_{m+1} &:= \frac{\eta_{m+1} C}{\sqrt{b_{m+1}}} \\
&= \frac{\kappa_m \eta_m C}{\sqrt{\lambda_m}\sqrt{b_m}} \\
&= \frac{\kappa_m}{\sqrt{\lambda_m}}\delta_m \\
&= \gamma \delta_m.
\end{aligned}$$

Therefore, from $M := \log_\gamma(\alpha_0\epsilon) + 1$ and $\delta_1 := \frac{\eta_1 C}{\sqrt{b_1}}$

$$\begin{aligned}\delta_M &= \delta_1 \gamma^{M-1} \\ &= \delta_1 \alpha_0 \epsilon \\ &= \frac{\eta_1 C \alpha_0 \epsilon}{\sqrt{b_1}}.\end{aligned}$$

According to Theorem 4.1,

$$\begin{aligned}\mathbb{E}\left[\hat{f}_{\delta_M}(\boldsymbol{x}_{M+1}) - \hat{f}_{\delta_M}(\boldsymbol{x}^\star_{\delta_M})\right] &\leq \epsilon_M \\ &= \sigma_M \delta_M^2 \\ &= \left(\frac{\sqrt{\sigma_M}\eta_1 C \alpha_0 \epsilon}{\sqrt{b_1}}\right)^2\end{aligned}$$

From Lemmas D.2 and 5.1,

$$\begin{aligned}f(\boldsymbol{x}_{M+2}) - f(\boldsymbol{x}^\star) &= \left\{f(\boldsymbol{x}_{M+2}) - \hat{f}_{\delta_M}(\boldsymbol{x}_{M+2})\right\} + \left\{\hat{f}_{\delta_M}(\boldsymbol{x}^\star) - f(\boldsymbol{x}^\star)\right\} \\ &\quad + \left\{\hat{f}_{\delta_M}(\boldsymbol{x}_{M+2}) - \hat{f}_{\delta_M}(\boldsymbol{x}^\star)\right\} \\ &\leq \left\{f(\boldsymbol{x}_{M+2}) - \hat{f}_{\delta_M}(\boldsymbol{x}_{M+2})\right\} + \left\{\hat{f}_{\delta_M}(\boldsymbol{x}^\star) - f(\boldsymbol{x}^\star)\right\} \\ &\quad + \left\{\hat{f}_{\delta_M}(\boldsymbol{x}_{M+2}) - \hat{f}_{\delta_M}(\boldsymbol{x}^\star_{\delta_M})\right\} \\ &\leq \delta_M L_f + \delta_M L_f + \left\{\hat{f}_{\delta_M}(\boldsymbol{x}_{M+2}) - \hat{f}_{\delta_M}(\boldsymbol{x}^\star_{\delta_M})\right\} \\ &= 2\delta_M L_f + \left\{\hat{f}_{\delta_M}(\boldsymbol{x}_{M+2}) - \hat{f}_{\delta_M}(\boldsymbol{x}_{M+1})\right\} + \left\{\hat{f}_{\delta_M}(\boldsymbol{x}_{M+1}) - \hat{f}_{\delta_M}(\boldsymbol{x}^\star_{\delta_M})\right\} \\ &\leq 2\delta_M L_f + L_f \left\|\boldsymbol{x}_{M+2} - \boldsymbol{x}_{M+1}\right\| + \left\{\hat{f}_{\delta_M}(\boldsymbol{x}_{M+1}) - \hat{f}_{\delta_M}(\boldsymbol{x}^\star_{\delta_M})\right\}.\end{aligned}$$

Then, we have

$$\begin{aligned}f(\boldsymbol{x}_{M+2}) - f(\boldsymbol{x}^\star) &\leq 2\delta_M L_f + 6 L_f \delta_M + \epsilon_M \\ &= 8 L_f \delta_M + \epsilon_M,\end{aligned}$$

where we have used $\|\boldsymbol{x}_{M+2} - \boldsymbol{x}_{M+1}\| \leq 6\delta_M$ since $\boldsymbol{x}_{M+2}, \boldsymbol{x}_{M+1} \in N(\boldsymbol{x}^\star; 3\delta_M)$. Therefore,

$$\begin{aligned}f(\boldsymbol{x}_{M+2}) - f(\boldsymbol{x}^\star) &\leq \frac{8 L_f \eta_1 C \alpha_0 \epsilon}{\sqrt{b_1}} + \left(\frac{\sqrt{\sigma_M}\eta_1 C \alpha_0 \epsilon}{\sqrt{b_1}}\right)^2 \\ &\leq \frac{8 L_f \eta_1 C \alpha_0 \epsilon}{\sqrt{b_1}} + \left(\frac{\sqrt{\sigma}\eta_1 C \alpha_0 \epsilon}{\sqrt{b_1}}\right)^2 \\ &\leq \epsilon,\end{aligned}$$

where we have used $\alpha_0 := \min\left\{\frac{\sqrt{b_1}}{16L_f\eta_1 C}, \frac{\sqrt{b_1}}{\sqrt{2\sigma}\eta_1 C}\right\}$.
Let T_{total} be the total number of queries made by Algorithm 4.1; then,

$$T_{\text{total}} = \sum_{m=1}^{M+1} \frac{H_m}{\epsilon_m} = \sum_{m=1}^{M+1} \frac{H_m}{\sigma\delta_m^2}.$$

Here, from the proof of Theorem 4.1 (see Section F.1), we define $H_4 > 0$ as follows:

$$H_m := \frac{9(1-\sigma_m\eta)\delta_m^2}{2\eta} + \frac{3L_f\delta_m}{\eta(2-L_g\eta)} \leq \frac{9(1-\sigma_1\eta)\delta_1^2}{2\eta} + \frac{3L_f\delta_1}{\eta(2-L_g\eta)} =: H_4$$

Thus, from $\delta_M = \delta_1\alpha_0\epsilon$,

$$\begin{aligned} T_{\text{total}} &= \sum_{m=1}^{M+1} \frac{H_m}{\sigma_m\delta_m^2} \leq H_4 \sum_{m=1}^{M+1} \frac{1}{\sigma_m\delta_m^2} \leq H_4 \sum_{m=1}^{M+1} \frac{1}{\sigma_1\delta_M^2} = \frac{H_4(M+1)}{\sigma_1\delta_M^2} \\ &= \frac{H_4(M+1)}{\sigma_1\delta_1^2\alpha_0^2\epsilon^2} = \mathcal{O}\left(\frac{1}{\epsilon^2}\right). \end{aligned}$$

This completes the proof. **(Q.E.D.)**

G Full Experimental Results

The experimental environment was as follows: NVIDIA GeForce RTX 4090×2GPU and Intel Core i9 13900KF CPU. The software environment was Python 3.10.12, PyTorch 2.1.0 and CUDA 12.2. The code is available at `https://anonymous.4open.science/r/new-sigma-nice`.

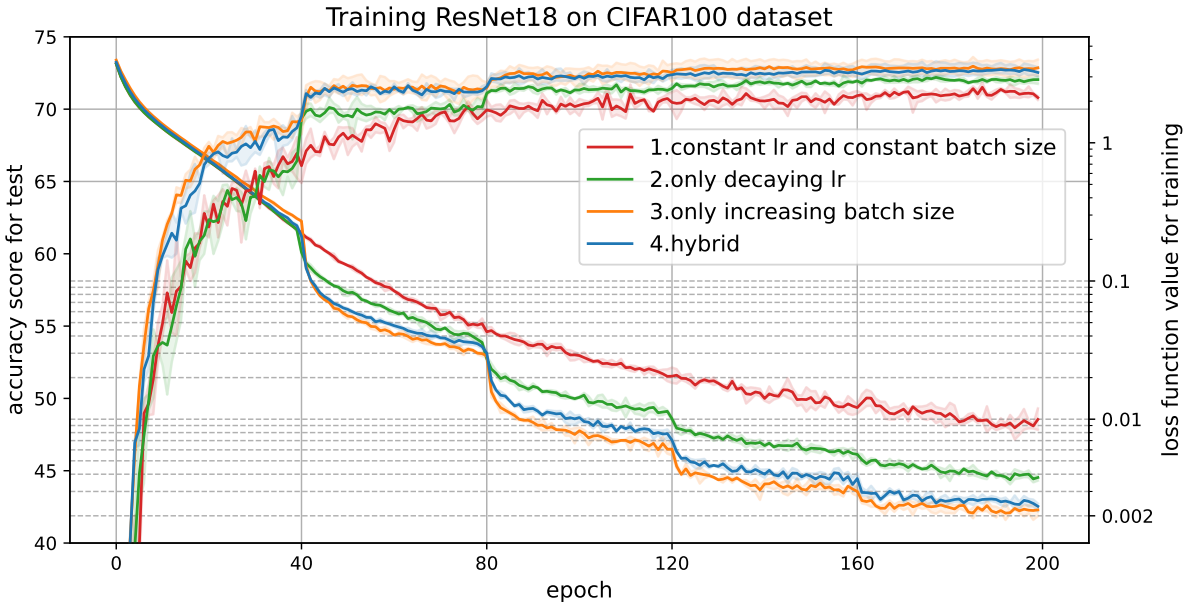

Fig. 8 Accuracy score for testing and loss function value for training versus the number of epochs (**left**) and the number of parameter updates (**right**) in training ResNet18 on the CIFAR100 dataset. The solid line represents the mean value, and the shaded area represents the maximum and minimum over three runs. In method 1, the learning rate and the batch size were fixed at 0.1 and 128, respectively. In method 2, the learning rate decreased every 40 epochs as $\left[0.1, \frac{1}{10\sqrt{2}}, 0.05, \frac{1}{20\sqrt{2}}, 0.025\right]$ and the batch size was fixed at 128. In method 3, the learning rate was fixed at 0.1, and the batch size was increased as $[16, 32, 64, 128, 256]$. In method 4, the learning rate was decreased as $\left[0.1, \frac{\sqrt{3}}{20}, 0.075, \frac{3\sqrt{3}}{80}, 0.05625\right]$ and the batch size was increased as $[32, 48, 72, 108, 162]$.

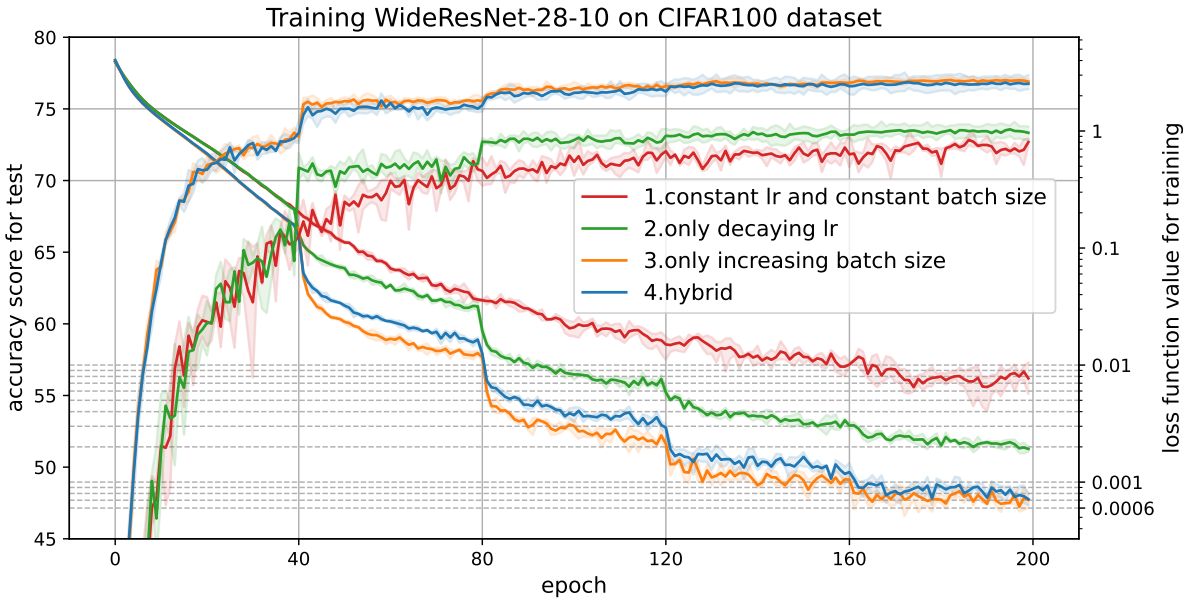

Fig. 9 Accuracy score for testing and loss function value for training versus the number of epochs (**left**) and the number of parameter updates (**right**) in training WideResNet-28-10 on the CIFAR100 dataset. The solid line represents the mean value, and the shaded area represents the maximum and minimum over three runs. In method 1, the learning rate and batch size were fixed at 0.1 and 128, respectively. In method 2, the learning rate was decreased every 40 epochs as $\left[0.1, \frac{1}{10\sqrt{2}}, 0.05, \frac{1}{20\sqrt{2}}, 0.025\right]$ and the batch size was fixed at 128. In method 3, the learning rate was fixed at 0.1, and the batch size was increased as $[8, 16, 32, 64, 128]$. In method 4, the learning rate was decreased as $\left[0.1, \frac{\sqrt{3}}{20}, 0.075, \frac{3\sqrt{3}}{80}, 0.05625\right]$ and the batch size was increased as $[8, 12, 18, 27, 40]$.

H Discussion on the definition of the smoothed function

Recall the general definition of the smoothing of the function.

Definition H.1 Given a function $f\colon \mathbb{R}^d \to \mathbb{R}$, define $\hat{f}_\delta\colon \mathbb{R}^d \to \mathbb{R}$ to be the function obtained by smoothing f as

$$\hat{f}_\delta(\boldsymbol{x}) := \mathbb{E}_{\boldsymbol{u}\sim\mathcal{N}\left(\mathbf{0};\frac{1}{\sqrt{d}}I_d\right)}\left[f(\boldsymbol{x}-\delta\boldsymbol{u})\right],$$

where $\delta > 0$ represents the degree of smoothing and $\boldsymbol{u}$ is a random variable from a Gaussian distribution.

What probability distribution the random variable $\boldsymbol{u} \in \mathbb{R}^d$ follows in the definition of smoothed function varies in the literature. [34, 59, 90] assumes a Gaussian distribution, while [61] assumes a uniform distribution for the sake of theoretical analysis. So what probability distribution the random variable $\boldsymbol{u}$ should follow in order to smooth the function? This has never been discussed.

It is difficult to confirm from a strictly theoretical point of view whether the function $\hat{f}_\delta$ obtained by using a random variable $\boldsymbol{u}$ that follows a certain probability distribution is smoother than the original function f (more precisely, it is possible with a Gaussian distribution). Therefore, we smoothed a very simple nonconvex function with random variables following several major probability distributions and compared it with the original function. We deal with one-dimensional Rastrigin's function [68, 69] and Drop-Wave function [91] defined as follows:

$$\text{(Rastrigin's function)}\quad f(x) := x^2 - 10\cos(2\pi x) + 10, \tag{8}$$

$$\text{(Drop-Wave function)}\quad f(x) := -\frac{1+\cos(12\pi x)}{0.5x^2+2}. \tag{9}$$

We smooth the above functions in accordance with Definition 2.1 using random variables following light-tailed distributions: Gaussian, uniform, exponential, and Rayleigh, and heavy-tailed distributions: Pareto, Cauchy, and Levy. We have added the code for this smoothing experiment to our anonymous GitHub. For more information on the parameters of each probability distribution, please see there. First, Figure 10 plots the Rastrigin's function and its smoothed version with a degree of smoothing of $\delta = 0.5$, using a random variable that follows several probability distributions.

Figures 10 shows that smoothing using random variable from light-tailed distributions works, while smoothing using random variable from heavy-tailed distributions does not. The reason for this is thought to be that extremely large values tend to appear in heavy-tailed distributions, and the function values are not stable.

Next, Figure 11 plots the Drop-Wave function and its smoothed version with a degree of smoothing of $\delta = 0.5$, using a random variable that follows several probability distributions.

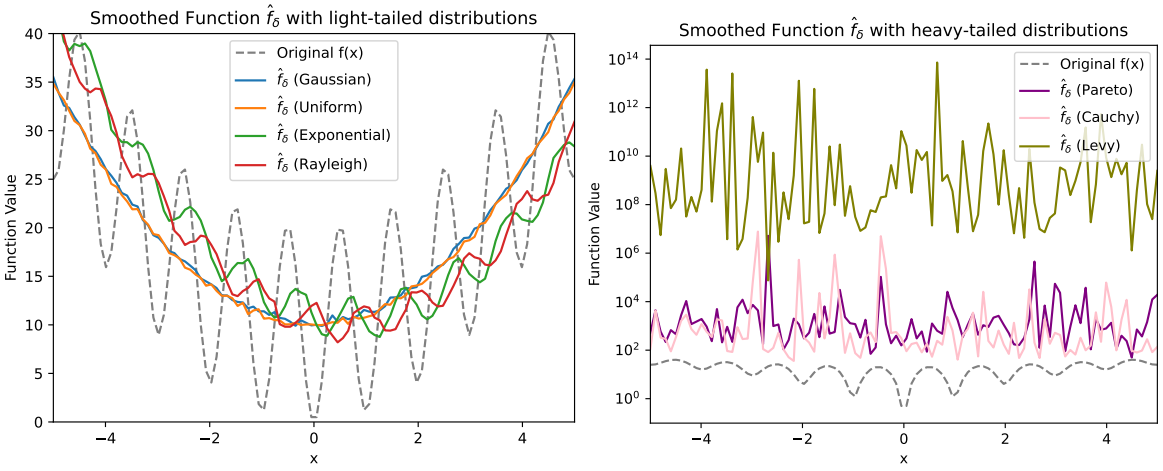

Fig. 10 Rastrigin's function (8) and its smoothed version using random variables following a light-tailed distribution (**left**) and heavy-tailed distribution (**right**). The degree of smoothing is set to 0.5. Note that right graph has the logarithmic vertical axis with a base of 10.

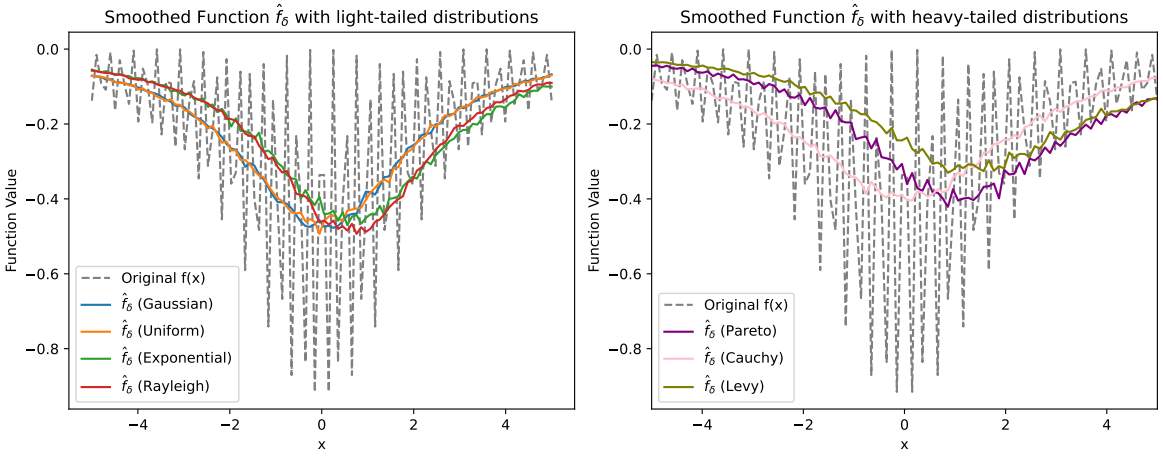

Fig. 11 Drop-Wave function (9) and its smoothed version using random variables following a light-tailed distribution (**left**) and heavy-tailed distribution (**right**). The degree of smoothing is set to 0.5.

Figure 11 shows that, in contrast to Figure 10, the heavy-tailed distribution successfully smooths the function as well as the light-tailed distribution. The reason for this lies in the definition of the Drop-Wave function. The Drop-Wave function has an x^2 term in its denominator, which prevents the function value from exploding even when the heavy-tailed distribution provides extremely large values, and thus the smoothing works.

In smoothing of the function, random variables have been defined to follow primarily a Gaussian distribution (see Definition H.1), but these experimental results motivate us to extend it from a Gaussian distribution to a light-tailed distribution (see Definition 2.1). Note that we also provide the interesting finding that, depend-

ing on the definition of the original function, random variables from heavy-tailed distributions can also be useful for smoothing.

I Discussion on σ_m-nice function

I.1 Extension from σ-nice function to σ_m-nice function

Hazan et al., proposed σ-nice function to analyze graduated optimization algorithm.

Definition I.1 (σ-nice function [61]) Let $M \in \mathbb{N}$ and $m \in [M]$. A function $f\colon \mathbb{R}^d \to \mathbb{R}$ is said to be σ-nice if the following two conditions hold:
(i) For all $\delta_m > 0$ and all $\boldsymbol{x}^\star_{\delta_m}$, there exists $\boldsymbol{x}^\star_{\delta_{m+1}}$ such that:

$$\left\|\boldsymbol{x}^\star_{\delta_m} - \boldsymbol{x}^\star_{\delta_{m+1}}\right\| \leq \delta_{m+1} := \frac{\delta_m}{2}.$$

(ii) For all $\delta_m > 0$, the function $\hat{f}_{\delta_m}(\boldsymbol{x})$ over $N(\boldsymbol{x}^\star_{\delta_m}; 3\delta_m)$ is σ-strongly convex.

Recall our σ_m-nice function (Definition 4.1).

Definition I.2 (σ_m-nice function) Let $M \in \mathbb{N}$, $m \in [M]$, and $\gamma \in [0.5, 1)$. A function $f\colon \mathbb{R}^d \to \mathbb{R}$ is said to be σ_m-nice if the following two conditions hold:
(i) For all $\delta_m > 0$ and all $\boldsymbol{x}^\star_{\delta_m}$, there exists $\boldsymbol{x}^\star_{\delta_{m+1}}$ such that:

$$\left\|\boldsymbol{x}^\star_{\delta_m} - \boldsymbol{x}^\star_{\delta_{m+1}}\right\| \leq \delta_{m+1} := \gamma\delta_m.$$

(ii) For all $\delta_m > 0$, the function $\hat{f}_{\delta_m}(\boldsymbol{x})$ over $N(\boldsymbol{x}^\star_{\delta_m}; 3\delta_m)$ is σ_m-strongly convex.

In condition (i), we extended the decay rate of the degree of smoothing from a constant 0.5 to a constant $\gamma \in [0.5, 1)$. See Proposition 4.2 for the soundness of this extension. In condition (ii), $\hat{f}_{\delta_m}$ was always defined to be σ-strongly convex in the definition of σ-nice function, which is a rather strong assumption. In fact, the greater the degree of smoothing δ_m is, the smoother the smoothed function $\hat{f}_{\delta_m}$ becomes and the smaller the strong convexity parameter may be. Here, let σ_{small} be a strongly convexity parameter of f and $0 < \sigma_{\text{small}} < \sigma_{\text{big}}$, then the function f is not σ_{big}-strongly convex function. Therefore, the strongly convexity parameter should depend on the degree of smoothing δ_m and we extend the strongly convexity parameter to σ_m from σ.

I.2 Proof of Proposition 4.1

(Proof) First, we consider the cross entropy loss. Let $\boldsymbol{x}_i \in \mathbb{R}^d$ $(i \in [n])$ be the i-th training data, $\boldsymbol{y}_i \in \mathbb{R}^c$ be the i-th label (one-hot vector), and $f(\boldsymbol{x}_i)$ be the i-th output of the model, where d is the number of model parameters, c is the number of classes, and n is the number of training data. Assume that the output elements $f(\boldsymbol{x}_i)^{(j)} \in \mathbb{R}$

($j \in [c]$) are normalized to $(0, 1]$ by using a softmax function. In this case, the cross entropy loss can be expressed as

$$L^{\mathrm{CEL}} := \frac{1}{n} \sum_{i \in [n]} L_i^{\mathrm{CEL}}, \text{ where } L_i^{\mathrm{CEL}} := -\log f(\boldsymbol{x}_i)^{(y_i^{\mathrm{hot}})},$$

where y_i^{hot} is an index with element 1 of label $\boldsymbol{y}_i$, and $f(\boldsymbol{x}_i)^{(y_i^{\mathrm{hot}})} \in \mathbb{R}$ is an element of $f(\boldsymbol{x}_i)$ corresponding to that index y_i^{hot}. Therefore, we can consider the following function:

$$g(x) := -\log x \ (0 < x \leq 1).$$

Recall the definition of the smoothed function with a Gaussian u,

$$\hat{g}_\delta(x) := \mathbb{E}_{u \sim N(0;1)}[g(x - \delta u)].$$

From the Taylor expansion, we have

$$-\log(x - \delta u) \approx -\log x + \frac{\delta u}{x} + \frac{\delta^2 u^2}{2x^2}.$$

Hence,

$$\hat{g}_\delta(x) := \mathbb{E}\left[-\log(x - \delta u)\right] = -\log x + \frac{\delta^2}{2x^2},$$

where we use $\mathbb{E}[u] = 0, \mathbb{E}[u^2] = 1$.

First, both g and $\hat{g}_\delta$ are functions have a global minimum at $x = 1$, i.e., $x^\star = 1$ and $x_\delta^\star = 1$ for all δ. Thus, the first σ_m-nice condition is satisfied. Next, since $g''(x) = \frac{1}{x^2} \geq 1$ and $\hat{g}_\delta''(x) = \frac{1}{x^2} + \frac{3\delta^2}{x^4} \geq 1$ hold, both g and $\hat{g}_\delta$ are 1-strongly convex. Thus, the σ_m-nice second condition is also satisfied. Therefore, g i.e., L_i^{CEL} is a 1-nice function. Since cross entropy loss L^{CEL} is the average of n functions L_i^{CEL}, L^{CEL} is also a 1-nice function. This completes the proof for the cross entropy loss.

Next, we consider the mean squared error. Let $\hat{y}_i$ be the i-th output of the model, y_i be the i-th true value, and n be the number of training data. Then, the mean squared error can be expressed as

$$L_{\mathrm{MSE}} := \frac{1}{n} \sum_{i \in [n]} L_i^{\mathrm{MSE}}, \text{ where } L_i^{\mathrm{MSE}} := (\hat{y}_i - y_i)^2.$$

Therefore, we can consider the following function:

$$h(x) := x^2.$$

From the definition of the smoothed function with Gaussian u, we have

$$\begin{aligned}\hat{h}_\delta(x) &:= \mathbb{E}_{u\sim\mathcal{N}(0;1)}[h(x-\delta u)] \\ &= \mathbb{E}_{u\sim\mathcal{N}(0;1)}[(x-\delta u)^2] \\ &= x^2 - 2\delta\mathbb{E}_u[u] + \delta^2\mathbb{E}_u[u^2] \\ &= x^2 + \delta^2,\end{aligned}$$

where we use $\mathbb{E}[u] = 0, \mathbb{E}[u^2] = 1$.

First, both h and $\hat{h}_\delta$ are functions that have a global minimum at $x = 0$, i.e., $x^\star = 0$ and $x^\star_\delta = 0$ for all δ. Thus, the first σ_m-nice condition is satisfied. Next, since $h''(x) = 2$ and $\hat{h}''_\delta(x) = 2$ hold, both h and $\hat{h}_\delta$ are 2-strongly convex. Thus, the second σ_m-nice condition is also satisfied. Therefore, the function h i.e., L_i^{MSE} is a 1-nice function. Since the mean squared error L^{MSE} is the average of n functions L_i^{MSE}, L^{MSE} is also a 2-nice function. This completes the proof for the mean squared error. **(Q.E.D.)**

I.3 Discussion on the light-tailed distribution

According to [92, Definition 3.1], the light-tailed distribution is defined as follows:

Definition I.3 (light-tailed distribution) A random variable X is said to be light-tailed if there exists a $c_0 > 0$ such that $\mathbb{E}\left[\exp(\lambda X)\right] < \infty$ for all $|\lambda| < c_0$.

This definition implies that the probability density function of the light-tailed distribution decreases exponentially at the tail. The definition of function smoothing in the previous study [61, Definition 4.1] used a random variable that follows a uniform distribution. We can show that the uniform distribution is light-tailed distribution.

Proposition I.1 The uniform distribution is light-tailed distribution.

(Proof) Let X be a random variable which follows uniform distribution over the interval $[a, b]$, where $a, b \in \mathbb{R}$ $(a \leq b)$. Then, the probability density function $f_X(x)$ is defined as follows:

$$f_X(x) = \begin{cases} \frac{1}{b-a}, & \text{if } x \in [a,b], \\ 0, & \text{otherwise.} \end{cases}$$

For all $\lambda \in \mathbb{R} \setminus \{0\}$, we have

$$\mathbb{E}\left[e^{\lambda X}\right] = \int_a^b e^{\lambda x} f_X(x)dx = \frac{1}{b-a}\int_a^b e^{\lambda x}dx = \frac{1}{b-a}\cdot\frac{1}{\lambda}\left[e^{\lambda x}\right]_a^b = \frac{1}{\lambda(b-a)}\left(e^{\lambda b} - e^{\lambda a}\right).$$

When $\lambda = 0$, we have

$$\mathbb{E}\left[e^{\lambda X}\right] = \mathbb{E}\left[e^0\right] = 1.$$

Therefore, for all $\lambda \in \mathbb{R}$, we obtain $\mathbb{E}\left[e^{\lambda X}\right] < \infty$. **(Q.E.D.)**

I.4 Distribution of SGD's stochastic noise

We collected 1000 each of stochastic noise $\boldsymbol{\omega}_t := \nabla f_{\mathcal{S}_t}(\boldsymbol{x}_t) - \nabla f(\boldsymbol{x}_t)$ and tested whether each element follows a light-tailed distribution. They were collected at the point where ResNet18 had been trained on the CIFAR100 dataset (10,000 steps). The code used in this experiment is available on our anonymous GitHub. ResNet18 has about 11M parameters, so $\boldsymbol{\omega}_t$ form an 11M-dimentional vector. Figure 12 plots the results for the $\boldsymbol{\omega}_t$ elements from dimension 0 to dimension 100,000. Figure 13 present the results for all elements. These results demonstrate that the SGD's stochastic noise $\boldsymbol{\omega}_t$ follows a light-tailed distribution.

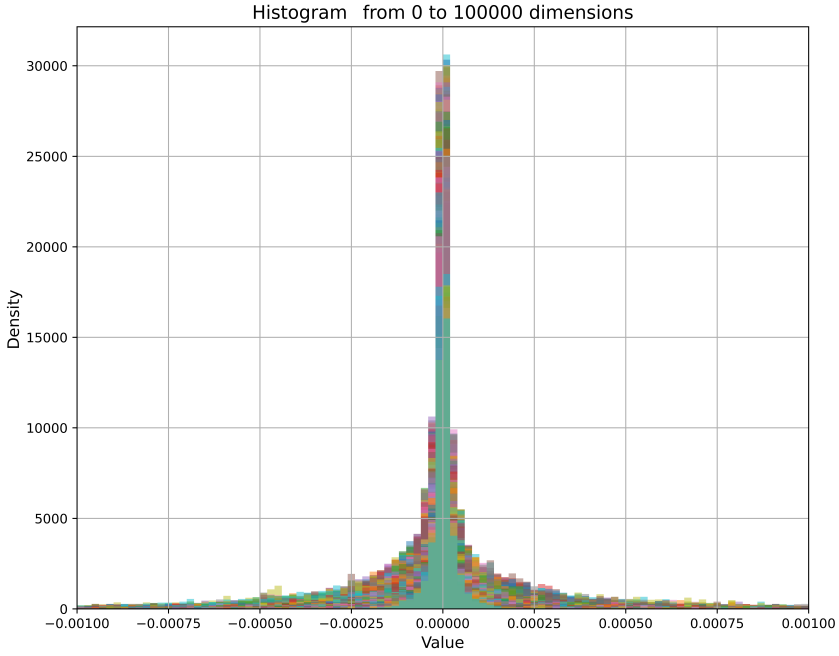

Fig. 12 Distribution of 1000 $\boldsymbol{\omega}_t$ elements from 0 to 100,000 dimensions.

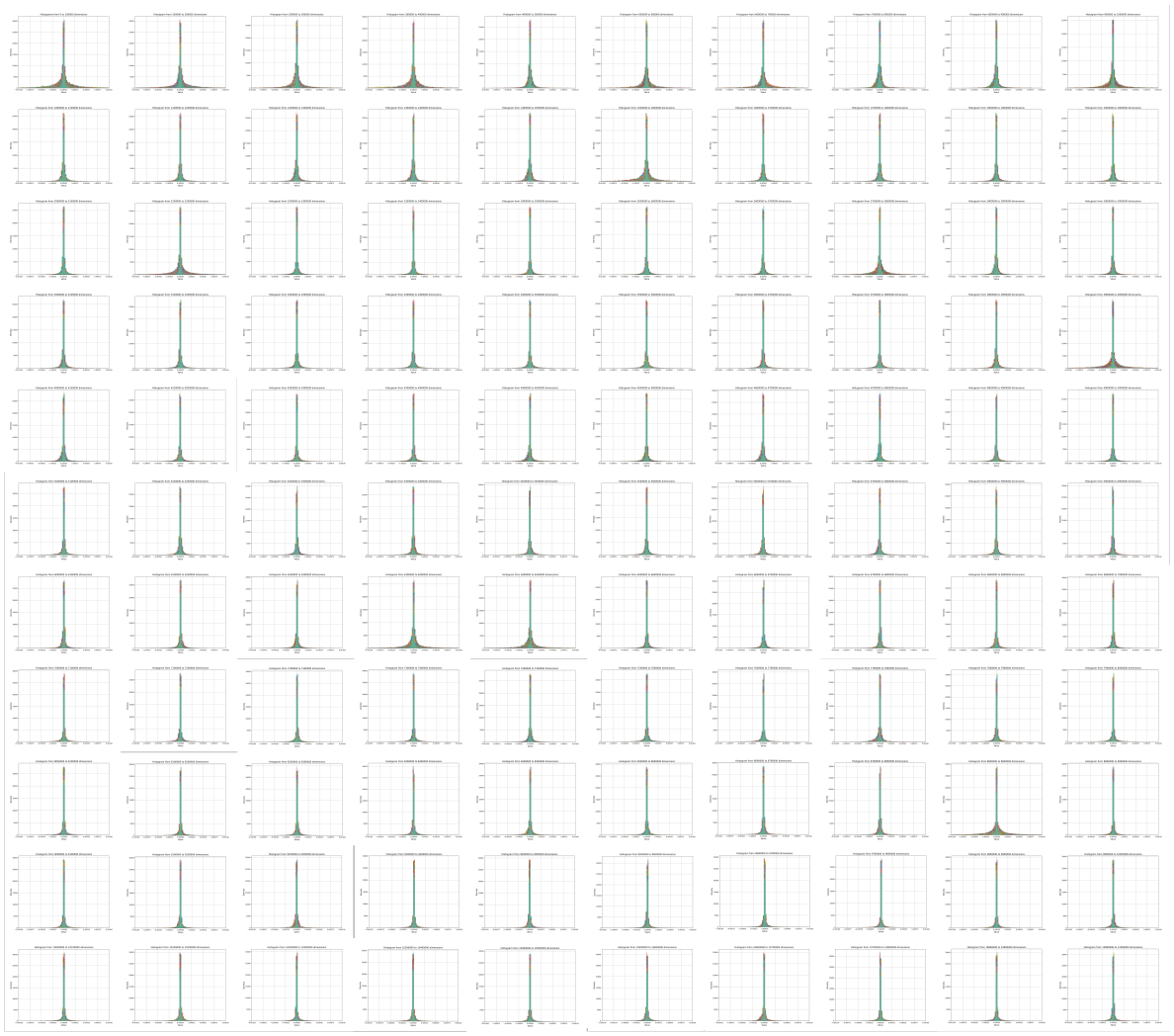

Fig. 13 Complete results for distribution of 1000 $\boldsymbol{\omega}_t$ elements. The distribution is plotted separately for each 100,000 dimensions.