Array 30 (2026) 100807

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/locate/array

A relaxation approach to layerwise determination of learning rates in deep neural networks

Chohong Min [a], Byungjoon Lee [b,*]

[a] Department of Mathematics, Ewha Womans University, Seoul, Republic of Korea
[b] Department of Mathematics, The Catholic University of Korea, Bucheon, Republic of Korea

ARTICLE INFO

Keywords:
Learning rate selection
Line-search minimization
Lipschitz constant
Layerwise training
Neural network

ABSTRACT

Selecting an appropriate learning rate is crucial for training neural networks, but classical line-search methods, while effective, often rely on costly matrix–vector multiplications that limit their practical use. We propose two relaxed line-search formulations that avoid matrix–vector multiplications and can be solved efficiently: a naive relaxation based on a global Lipschitz constant and a more adaptive relaxation using a local Lipschitz constant. Instead of updating all layers simultaneously with a single learning rate, we adopt a layerwise update strategy in which one layer is updated at a time, simplifying the analysis and enabling rigorous convergence results. Numerical experiments further demonstrate that optimal learning rates vary significantly across layers, supporting the effectiveness of the proposed approach.

1. Introduction

In spite of the great success of deep learning, its training process has not been completely understood. There exist numerous methods such as [1–3] that can efficiently train deep neural networks (DNNs) and some interesting convergence results [4–6] for the methods. However, these theoretical results are limited to the cases of smooth and convex problems and cannot be directly applied to DNNs training, which is highly nonconvex and nonsmooth. [7,8] introduced interesting convergence results for nonconvex problems, but the learning rates suggested by their theories turned out to be much smaller than usual learning rate in our empirical tests. There seems to exist a large gap between their estimates and the tight optimal estimate. Given the ubiquitous dominance of deep learning, theoretical investigation of nonconvex stochastic methods is much desired.

In addition to its theoretical significance, understanding and improving the behavior of learning rates in stochastic training contributes to the transparency and reliability of neural network training-key attributes in the development of interpretable and trustworthy machine learning systems. While our focus is not on model-level explainability, we offer interpretability at the training process level, where practitioners can better anticipate, debug, and tune optimization behaviors.

The schedule adjusting the learning rate in stochastic gradient descent (SGD) is an important problem, which is not completely resolved by just empirically tuned. The classical Robbins–Monro theory [9] suggests a series of learning rates that decreases as iteration proceeds. In practice, there is no clear best choice for the learning rate. In many deep learning problems, where the underlying loss function is highly non-convex, one often tests several learning rate schedules with hyperparameters that are empirically decided [4]. One may consider using iteration-dependent learning rate, chosen via line search methods [10,11], but the line search soon becomes too expensive in the computational time as the number of layers and the dimensions of hidden layers increase. From these reasons, the exact line search for training in DNNs by SGD is impractical, and several existing works tried approximations of the exact line search such as Armijo [12] and Wolfe [13] conditions.

It is well known that the Lipschitz constant of the gradient of the loss function is crucial for determining the learning rate in SGD. Despite the importance of the Lipschitz constant, many literature [5,8,14,15] makes use of a quite loose estimate of the Lipschitz constant in the naive sense to their analyses, which may lead to inefficiency caused by selecting smaller learning rate than the expected. This article is aimed at providing an efficient line search for training DNNs with SGD. We introduce a local Lipschitz estimate in forward and backward passes. The Lipschitz estimate is critical to the minimization of loss function from a proper choice of learning rate. Note that there are two types of Lipschitz estimates. One type is the Lipschitz constant in the forward pass [14,15] a measure of sensitivity of DNNs with respect to input perturbation. The first type is used to evaluate the robustness of DNNs and to analyze the stability of closed-loop in deep reinforcement learning. The second type, to which our estimate belongs, is the Lipschitz

* Corresponding author.
E-mail address: blee@catholic.ac.kr (B. Lee).

https://doi.org/10.1016/j.array.2026.100807
Received 20 December 2024; Received in revised form 3 April 2026; Accepted 7 April 2026
Available online 8 April 2026
2590-0056/© 2026 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

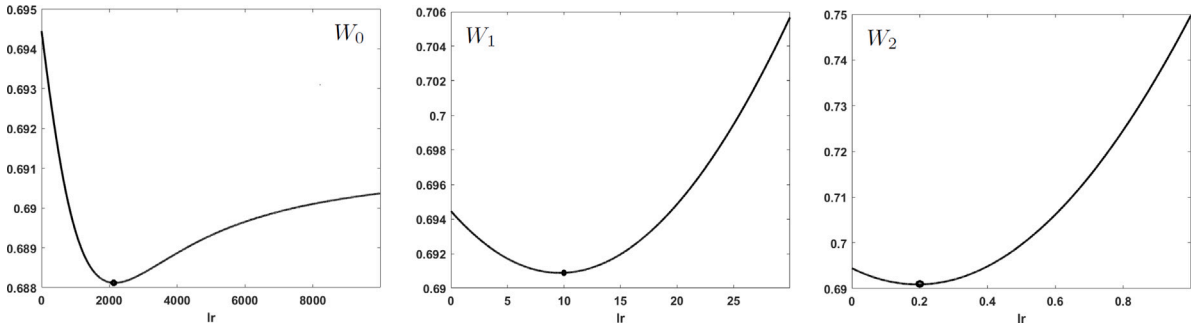


Fig. 1. Three-layered network with SONAR dataset. The plots depict the evolutions of loss functions while one weight is updated, the other weights are fixed. The results indicate that the choice of optimal learning rate, denoted as circles, may differ by large amount from layer to layer.

constant in the backward pass [5,8], and is used to minimize the loss function as much as possible.

By analyzing and making explicit the relationship between local curvature (as captured via the Lipschitz constant) and adaptive learning rate schedules, our approach allows for more interpretable and controllable training dynamics. In this sense, our work contributes to a more systematic understanding of deep learning optimization behavior, which is an essential complement to post hoc interpretability techniques.

Conventionally the gradients of DNNs are backward propagated and their corresponding weights are concurrently updated with the same learning rate. There are two drawbacks in the conventional practice. The first one is that all the weights are updated with the same learning rate, although the optimal learning rate of each layer can be different to each other, as illustrated in Fig. 1. From these reasons, [16] takes different learning rates layerwisely, and suggest specific formula for the learning rate of each layer. The other drawback is the difficulty on perturbation analysis. Each update of weight generates a series of perturbations in forward and backward passes. At each joint node, two perturbations collide and the number of terms to deal with is double by the product rule. The cascade of doublings makes the estimation of perturbations intangible. From these reasons, we take the layerwise update: while one weight is updated, the other weights are fixed.

The objective of this article is to introduce a layerwise determination technique for learning rates in the training of DNNs. For this purpose, we propose relaxations of training progress, which can be viewed as a minimization problem, in Section 2 with local Lipschitz analysis in Section 3. Layerwise applications of suggested approach to DNNs are discussed in Section 4 and numerical results in Section 5 demonstrate the efficiency of the layerwise relaxation approach. In Section 6, we summarize the arguments made in this article and present conclusions.

2. Relaxation approach

The determination of the learning rate is critical in SGD. If the learning rate is chosen larger than the optimal value, SGD with constant learning rate can oscillate and even diverge. The optimal value can be chosen via the line-search minimization. However, the cost of each trial in the line search involves matrix–vector multiplication and is expensive in the computational cost.

In this section, we introduce two relaxations for the line search minimization. The relaxations are based on the global Lipschitz constant and a local Lipschitz constant, which is measure on the line segment that takes the current weight as one endpoint and goes in its gradient direction.

Definition 1 (*Global and Local Lipschitz Constants*)**.** For a function $f(x)$, $L(f)$ denotes its global Lipschitz constant, i.e.

$$L(f) := \sup_{x \neq y} \frac{\|f(x) - f(y)\|}{\|x - y\|},$$

and $L(f, [x, y])$ denotes its local Lipschitz constant over the line segment connecting x and y, i.e.

$$L(f, [x, y]) := \sup_{t \in (0,1]} \frac{\|f(x + t(y - x)) - f(x)\|}{\|t(y - x)\|}.$$

Utilizing the Lipschitz constants, we introduce two relaxations of the line-search minimization. The first relaxation is based on the local Lipschitz constant, and the other one, which is referred to as naive relaxation, is based on the global Lipschitz constant.

Theorem 1 (*Relaxation*)**.** *For a continuously differentiable function $f(W)$, it holds for any $lr \in [0, \infty)$ that*

$$f(W - lr \cdot dW) \leq f(W) + \|dW\|^2 \left(-lr + lr^2 \cdot L(\nabla f, [W, W - lr \cdot dW])\right),$$

where dW denotes the current gradient $\nabla f(W)$.

Proof. Let $g(t) = f(W - t \cdot dW)$. Then, by Mean Value Theorem, there exists an $\epsilon \in (0, lr)$ such that

$$\begin{aligned} g(lr) - g(0) &= lr \cdot g'(\epsilon) \\ &= lr \nabla f(W - \epsilon \cdot dW) \cdot (-dW) \\ &= lr (\nabla f(W - \epsilon \cdot dW) - \nabla f(W) + \nabla f(W)) \cdot (-dW) \\ &= -lr \|dW\|^2 + lr (\nabla f(W - \epsilon \cdot dW) - \nabla f(W)) \cdot (-dW). \end{aligned}$$

Now, by the Cauchy–Schwarz inequality and the definition of local Lipschitz constants, we obtain

$$\begin{aligned} g(lr) - g(0) &= -lr \|dW\|^2 + lr (\nabla f(W - \epsilon \cdot dW) - \nabla f(W)) \cdot (-dW) \\ &\leq -lr \|dW\|^2 + lr \|\nabla f(W - \epsilon \cdot dW) - \nabla f(W)\| \|dW\| \\ &\leq -lr \|dW\|^2 + \epsilon \cdot lr \cdot L(\nabla f, [W, W - lr \cdot dW]) \|dW\|^2 \\ &\leq \|dW\|^2 \left(-lr + lr^2 L(\nabla f, [W, W - lr \cdot dW])\right). \quad \square \end{aligned}$$

Corollary 1 (*Naive Relaxation*)**.** For a continuously differentiable function $f(W)$, it holds for any $lr \in [0, \infty)$ that

$$f(W - lr \cdot dW) \leq f(W) + \|dW\|^2 \left(-lr + lr^2 \cdot L(\nabla f)\right).$$

Proof. It is an immediate consequence of Theorem 1 with the fact that $L(\nabla f, [W, W - lr \cdot dW]) \leq L(\nabla f)$ for any $lr \geq 0$. $\square$

The above results provide relaxations of the line-search condition in terms of Lipschitz constants, either local or global. While the *naive relaxation* yields a simple quadratic bound in lr, the local relaxation involves the more intricate segmental Lipschitz constant. To connect these relaxations with global convergence guarantees, we next establish a sufficient condition under which gradient descent with a fixed step size produces descent and approaches stationarity.

Remark. The following theorem may be viewed as a direct extension of the classical *descent lemma* [17], but stated in terms of a uniform

curvature bound on the gradient. This bridges the local and global perspectives: a global Lipschitz constant is a special case of such a uniform bound.

Theorem 2 (*Uniform Curvature Bound ⇒ Descent and Stationarity*)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be continuously differentiable and bounded below by $f_\star > -\infty$. Assume that for every step segment $[W, W - lr \cdot dW]$ the segmental Lipschitz constant of ∇f satisfies*

$$L\big(\nabla f, [W, W - lr \cdot dW]\big) \le C\big([W, W - lr \cdot dW]\big) \le C_{\max} < \infty.$$

Let the iterates be generated by

$$W_{k+1} = W_k - lr \cdot dW_k,$$

and consider the gradient-descent case $dW_k = \nabla f(W_k)$. Then for any fixed step size

$$0 < lr \le \frac{2}{C_{\max}}$$

the following hold for every k:

$$f(W_{k+1}) \le f(W_k) - \Big(lr - \tfrac{1}{2} lr^2 C_{\max}\Big) \|\nabla f(W_k)\|^2.$$

In particular, choosing the "naive" relaxation

$$lr^G := \frac{1}{2C_{\max}}$$

yields the explicit sufficient decrease

$$f(W_{k+1}) \le f(W_k) - \frac{3}{8C_{\max}} \|\nabla f(W_k)\|^2.$$

Consequently, summing over k gives

$$\sum_{k=0}^{\infty} \|\nabla f(W_k)\|^2 \le \frac{f(W_0) - f_\star}{lr - \frac{1}{2} lr^2 C_{\max}} < \infty,$$

and therefore $\|\nabla f(W_k)\| \to 0$ as $k \to \infty$. Every accumulation point of $\{W_k\}$ is a critical point of f.

Proof. By the descent lemma (Taylor expansion with integral remainder) applied on the segment $[W_k, W_{k+1}]$ we have

$$f(W_{k+1}) \le f(W_k) - lr\langle \nabla f(W_k), \nabla f(W_k)\rangle + \frac{lr^2}{2} L\big(\nabla f, [W_k, W_{k+1}]\big) \|\nabla f(W_k)\|^2.$$

Using the uniform bound $L(\nabla f, [W_k, W_{k+1}]) \le C_{\max}$ gives

$$f(W_{k+1}) \le f(W_k) - \Big(lr - \tfrac{1}{2} lr^2 C_{\max}\Big)\|\nabla f(W_k)\|^2,$$

which proves the first display. If $0 < lr \le 2/C_{\max}$ then the parenthesis is nonnegative, so $f(W_{k+1}) \le f(W_k)$.

Take $lr = lr^G = 1/(2C_{\max})$. Then

$$lr - \tfrac{1}{2} lr^2 C_{\max} = \frac{1}{2C_{\max}} - \frac{1}{2}\frac{1}{4C_{\max}^2} C_{\max} = \frac{3}{8C_{\max}},$$

which yields the explicit decrease stated above.

Because f is bounded below by $f_\star$ and the decrease per step is nonnegative, summing the displayed inequality for $k = 0, \dots, T-1$ yields

$$\Big(lr - \tfrac{1}{2} lr^2 C_{\max}\Big) \sum_{k=0}^{T-1} \|\nabla f(W_k)\|^2 \le f(W_0) - f(W_T) \le f(W_0) - f_\star.$$

Letting $T \to \infty$ shows $\sum_{k=0}^{\infty} \|\nabla f(W_k)\|^2 < \infty$, hence $\|\nabla f(W_k)\| \to 0$. Any accumulation point W_∞ then satisfies $\nabla f(W_\infty) = 0$, so it is a critical point. □

Thus, the global Lipschitz-based relaxation provides a clean convergence guarantee, but it can be overly conservative in practice. In the next section, we turn to the *local Lipschitz constant*, whose estimation leads to a more refined and adaptive step-size strategy.

3. Local Lipschitz constant

Each node in a neural network consists of three elements: input, output, and weight. A perturbation in one element generates perturbations in the other elements during both forward and backward passes. To optimally determine the learning rate, it is essential to estimate the ratio of the generated perturbations relative to the source perturbation. Each node comprises an affine layer, which is linear, and an activation layer, which is non-linear. The ratio of the perturbations is primarily determined by the Lipschitz constant of the activation layer.

The activation layer is implemented using some of the most commonly used conventional functions: sigmoid, hyperbolic tangent, arctangent, or ReLU type activations. We begin our Lipschitz analysis with the sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$ and extend the analysis to the other activation functions.

When the domain of the sigmoid function is restricted to an interval $[a, b]$, the maximum over this interval is referred to as the local Lipschitz constant and is denoted by $L(\sigma, [a, b])$. When $0 \notin [a, b]$, σ' is monotonic on $[a, b]$, and the maximum of the local Lipschitz constant is attained at $\min(|a|, |b|)$, since $\sigma'(x)$ is even and monotonically decreasing on $[0, \infty)$. In the case where $0 \in [a, b]$, the local Lipschitz constant of the sigmoid function is given by $\sigma'(0)$. Taking these arguments into consideration, the local Lipschitz constant of the sigmoid can be succinctly formulated as

$$L(\sigma, [a, b]) \le \sigma'(\operatorname{minmod}(a, b)), \tag{1}$$

where $\operatorname{minmod}(a, b) := \min(|a|, |b|)(ab)^+$ and $x^+ := \max(x, 0)$. Other activation functions, including hyperbolic tangent and arctangent, exhibit similar features to the sigmoid function. Their gradients are also even and monotonically decreasing on $[0, \infty)$. Thus, the formulation above is valid for these activation functions as well:

$$L(\tanh, [a, b]) \le \tanh'(\operatorname{minmod}(a, b)) \text{ and}$$
$$L(\arctan, [a, b]) \le \arctan'(\operatorname{minmod}(a, b)).$$

For piecewise linear activation functions like ReLU and Leaky ReLU (LReLU), the local Lipschitz constants are determined by their slopes. These can be expressed as:

$$L(\mathrm{ReLU}, [a, b]) \le H_0(b) \text{ and}$$
$$L(\mathrm{LReLU}, [a, b]) \le H_\alpha(b),$$

where α is a slope of LReLU for $x \le 0$, and:

$$H_c(x) = \begin{cases} 1 & x > 0, \\ c & x \le 0. \end{cases}$$

The sigmoid function is applied component-wise for vector or matrix-valued inputs, in which case its local Lipschitz constant is estimated as

$$L(\sigma, [\mathbf{a}, \mathbf{b}]) \le \left\|\sigma'(\operatorname{minmod}(\mathbf{a}, \mathbf{b}))\right\|_\infty,$$

and the other activation functions can be treated in a similar manner.

The proposed Lipschitz estimation strategy further extends to modern activation functions and network architectures, and explicit upper bounds can be derived using the same elementary principles. For example, for the Swish activation function $\mathrm{Swish}(x) = x\sigma(x)$, differentiation yields

$$\mathrm{Swish}'(x) = \sigma(x) + x\sigma(x)(1 - \sigma(x)).$$

Using the elementary bounds $0 \le \sigma(x) \le 1$ and $\sigma(x)(1 - \sigma(x)) = \sigma'(x)$, together with the fact that $\sigma'(x)$ is even and monotonically decreasing on $[0, \infty)$, we obtain, for all $x \in [a, b]$,

$$\left|\mathrm{Swish}'(x)\right| \le 1 + |x|\,\sigma'(\operatorname{minmod}(a, b)).$$

Taking the supremum over the interval immediately yields the local Lipschitz bound

$$L(\mathrm{Swish}, [a, b]) \le 1 + \max(|a|, |b|)\,\sigma'(\operatorname{minmod}(a, b)).$$

Similarly, for the GELU activation function defined by

$$\mathrm{GELU}(x) = x\,\Phi(x),$$

where Φ denotes the Gaussian cumulative distribution function, differentiation gives

$$\mathrm{GELU}'(x) = \Phi(x) + x\,\phi(x),$$

with $\phi = \Phi'$ being the standard Gaussian density. Since $0 \le \Phi(x) \le 1$ and $\phi(x)$ is even and monotonically decreasing on $[0,\infty)$, it follows that

$$\left|\mathrm{GELU}'(x)\right| \le 1 + |x|\,\phi(\mathrm{minmod}(a,b)), \qquad x \in [a,b],$$

and hence

$$L(\mathrm{GELU},[a,b]) \le 1 + \max(|a|,|b|)\,\phi(\mathrm{minmod}(a,b)).$$

For more complex architectures such as Transformer networks [18], the same approach applies at the level of individual building blocks. Linear mappings admit Lipschitz constants given by their operator norms, while component-wise activation functions contribute bounds of the form derived above. For the self-attention mechanism, the attention weights depend on bilinear forms of the input through query–key interactions. On a bounded input domain $[\mathbf{A},\mathbf{B}]$, this bilinear dependence implies that the Jacobian of the attention operator can be bounded by a constant proportional to $\|\max(|\mathbf{A}|,|\mathbf{B}|)\|_\infty$. Consequently, a self-attention module admits a local Lipschitz bound of the form

$$L_{\mathrm{att}} \le C_{\mathrm{att}}\,\|\max(|\mathbf{A}|,|\mathbf{B}|)\|_\infty,$$

where C_{att} depends only on the spectral norms of the projection matrices and the attention scaling. By summing the contributions of the identity mapping arising from residual connections, the attention mechanism, and the feedforward subnetwork, one obtains an explicit local Lipschitz bound for a Transformer block. The main theoretical challenge is that the bilinear structure of self-attention precludes a uniform global Lipschitz bound without additional constraints; nevertheless, on bounded domains, the resulting local Lipschitz constants are finite and computable within the proposed framework.

4. Layerwise adaptive learning rates

Conventionally, the gradients of the weights are calculated using backpropagation, and the weights are simultaneously updated through gradient descent. Each update's perturbation generates a series of perturbations during the forward and backward passes. Due to the coincident updates, two propagated perturbations at each joint node collide, resulting in the number of perturbation terms to manage doubling according to the product rule. This cascade of doublings complicates the estimation of the propagated perturbations, making it increasingly challenging as the number of layers grows. This phenomenon is one of the main reasons why there are very few convergence results for deep neural networks (DNNs), whereas numerous results exist for smooth and convex machine learning methods.

Instead, we adopt a layerwise approach, where the weights of one layer are updated alternately while keeping the weights of all other layers fixed. This layerwise update prevents collisions and the doubling of perturbations, allowing us to introduce convergence results. As shown in Fig. 1, the optimal learning rates can vary significantly in scale from one layer to another. In contrast to the conventional approach that uses a uniform learning rate, the learning rate is determined layerwise by solving one of the relaxations discussed in Section 2, which are based on the local Lipschitz constant outlined in Section 3. We begin our discussion with single-layer networks and then progress to double and multi-layer networks.

4.1. Single-layered network

Let us consider a single-layered network for the binary classification of a dataset $\{(x_0, y_0)\}$, where $x_0 \in \mathbb{R}^n$ and $y_0 \in \{0,1\}$. The variables are computed in the forward pass and backward pass as follows:

Forward pass	Backward pass
$p_0 = x_0 W_0$,	$dp_0 = \sigma(p_0) - y_0$,
$x_1 = \sigma(p_0)$,	$dW_0 = x_0^T dp_0$.
$f(W_0) = -x_1 \log(y_0) - (1-x_1)\log(1-y_0)$	

The Lipschitz constant measures the maximum response of the perturbation dW_0 due to a perturbation of W_0. Let $t \in (0, lr]$ and consider the perturbation $\Delta W_0 = -t \cdot dW_0$. This generates a series of perturbations in the forward and backward passes, which can be estimated as follows:

$$\|\Delta p_0\| \le \|x_0\|\|\Delta W_0\|,$$
$$\|\Delta x_1\| \le L(\sigma,[p_0, p_0+\Delta p_0])\|\Delta p_0\|.$$

and

$$\|\Delta(dp_0)\| \le L(\sigma,[p_0, p_0+\Delta p_0])\|\Delta p_0\|,$$
$$\|\Delta(dW_0)\| \le \|x_0\|\|\Delta(dp_0)\|.$$

The local Lipschitz constant of the response is then estimated as:

$$L(\nabla f,[W_0, W_0 - lr\cdot dW_0]) \le L(\sigma,[p_0, p_0 - lr\cdot x_0^T dW_0])\|x_0\|^2,$$

and the global Lipschitz constant is:

$$L(\nabla f) \le L(\sigma)\|x_0\|^2 = \frac{1}{4}\|x_0\|^2.$$

Using the naive relaxation in Corollary 1 with the above global Lipschitz estimate gives a relaxation of the line-search problem $\mathrm{minimize}_{lr\ge 0} f(W_0 - lr\cdot dW_0)$ as follows:

$$f(W_0 - lr\cdot dW_0) \le f(W_0) + \|dW_0\|^2\left(-lr + lr^2\cdot\frac{\|x_0\|^2}{4}\right).$$

This quadratic minimization problem can be readily solved as:

$$lr^G = \frac{2}{\|x_0\|^2}.$$

On the other hand, Theorem 1 provides another relaxation involving $L(\nabla f,[W_0, W_0 - lr\cdot dW_0])$, which is bounded by $\|x_0\|^2 L(\sigma,[p_0, p_0 - lr\cdot x_0^T dW_0])$, equating to $\sigma'(\mathrm{minmod}(p_0, p_0 - lr\cdot x_0^T dW_0))$ as shown in (1). The following theorem states the relaxation and its convergence results.

Theorem 3 (*Single Layer*). *For any W_0 and any $lr \ge 0$, the following holds for a single-layer network:*

$$f\left(W_0 - lr\cdot dW_0\right) \le f\left(W_0\right) + \left\|dW_0\right\|^2 g\left(lr\right),$$

where $g(lr) := -lr + lr^2\cdot\left\|x_0\right\|^2\sigma'\left(minmod\left(p_0, p_0 - lr\cdot x_0^T dW_0\right)\right)$. The function $g(lr)$ is continuous on $[0,\infty)$ and satisfies

$$-lr + lr^2\sigma'\left(p_0\right)\left\|x_0\right\|^2 \le g\left(lr\right) \le -lr + lr^2\sigma'\left(0\right)\left\|x_0\right\|^2.$$

Let lr^* be the minimizer of $g(lr)$ over the interval

$$\left[lr^G = \frac{1}{2\sigma'(0)\|x_0\|^2}, \frac{1}{2\sigma'(p_0)\|x_0\|^2}\right].$$

Then, $g(lr^*) \le 0$, and there is a monotonic decrease in the cost function:

$$f(W_0 - lr^*\cdot dW_0) \le f(W_0) + \|dW_0\|^2 g(lr^*) \le f(W_0).$$

Proof. The function $a^+ = \max(a,0)$ is continuous, as is $\mathrm{minmod}(a,b) = \min(|a|,|b|)\cdot(ab)^+$. Therefore, $g(lr)$, which is a composite function of $\sigma'(x)$ and minmod, is also continuous. Since $\sigma'(x)$ is an even function and decreases monotonically as $|x|$ increases, it holds for any $a, b \in \mathbb{R}$ that

$$\sigma'\left(minmod\left(a,b\right)\right) \le \sigma'\left(0\right) \text{ and}$$

$$\begin{aligned}\sigma'(minmod(a,b)) &= \sigma'\left(\min(|a|,|b|)\cdot(ab)^{+}\right)\\ &\geq \sigma'(\min(|a|,|b|))\\ &\geq \sigma'(a).\end{aligned}$$

From this, we obtain the bounds

$$-lr + lr^2\sigma'(p_0)\|x_0\|^2 \leq g(lr) \leq -lr + lr^2\sigma'(0)\|x_0\|^2.$$

The lower and upper bounds on $g(lr)$ suggest searching for $lr^* = \arg\min g(lr)$ within the interval

$$lr \in \left[1/\left(2\sigma'(0)\|x_0\|^2\right), 1/\left(2\sigma'(p_0)\|x_0\|^2\right)\right].$$

Since $g\left(\frac{1}{2\sigma'(0)\|x_0\|^2}\right) \leq 0$, we have $g(lr^*) \leq 0$. Using the estimate from Theorem 1, we conclude that

$$f(W_0 - lr^*\cdot dW_0) \leq f(W_0) + \|dW_0\|^2 g(lr^*) \leq f(W_0). \quad \square$$

The above theorem addresses a relaxation with a single data point and can be extended to handle multiple data points. Assume we have a dataset $\{(x_0^i, y_0^i)\}_{i=1}^N$ with a large N. Computing the gradient $\nabla f(W_0) = \frac{1}{N}\sum_{i=1}^N \nabla f(W_0, i)$ is expensive and often prohibitive. In stochastic methods, a batch $B \subset \{1,\dots,N\}$ is randomly selected, and the gradient computation is restricted to the batch, yielding

$$\nabla_B f(W_0) = \frac{1}{|B|}\sum_{i\in B}\nabla f(W_0, i) \approx \frac{1}{N}\sum_{i=1}^N \nabla f(W_0, i).$$

Let x_0^B represent the concatenation of the row vectors $\{x_0^i\}_{i\in B}$, and define $f(W_0, B) := \frac{1}{|B|}\sum_{i\in B} f(W_0, i)$. The theorem above can be modified to account for the stochastic gradient computed over the batch.

Corollary 2. For any W_0, any $lr \geq 0$, and any $B \subset \{1,\dots,N\}$, the following holds for a single-layer network:

$$f(W_0 - lr\cdot dW_0, B) \leq f(W_0, B) + \|dW_0\|^2 g(lr),$$

where $g(lr) := -lr + lr^2\left\|x_0^B\right\|^2 \cdot \left\|\sigma'\left(\text{minmod}\left(p_0, p_0 - lr\cdot x_0^i dW_0\right)\right)\right\|_\infty$. The function $g(lr)$ is continuous on $[0,\infty)$ and satisfies

$$-lr + lr^2\left\|\sigma'(p_0)\right\|_\infty\left\|x_0^B\right\|^2 \leq g(lr) \leq -lr + lr^2\sigma'(0)\left\|x_0^B\right\|^2.$$

Let lr^* be the minimizer of $g(lr)$ over the interval

$$\left[lr^G = \frac{1}{2\sigma'(0)\left\|x_0^B\right\|^2}, \frac{1}{2\left\|\sigma'(p_0)\right\|_\infty\left\|x_0^B\right\|^2}\right].$$

Then $g(lr^*) \leq 0$ and there is a monotone decrease in the cost function:

$$f(W_0 - lr^*\cdot dW_0, B) \leq f(W_0, B) + \|dW_0\|^2 g(lr^*) \leq f(W_0, B).$$

Proof. It is an immediate consequence of Theorem 3. $\square$

Remark (*On the Monotonicity of Loss in Practice*)**.** Theorems 2 and Corollary 2 establish that the loss function decreases monotonically, assuming that the same mini-batch is used throughout training. However, in practical settings, including our numerical experiments, mini-batches are resampled at each iteration to simulate stochastic gradient descent. This stochasticity introduces fluctuations in the loss, which can result in non-monotonic behavior, as observed in the figures in Section 5. Despite this, the theoretical results provide important insight into the expected behavior under fixed-batch scenarios.

Remark (*Global Lipschitz vs. Local Lipschitz*)**.** Throughout this manuscript, we introduce two choices of learning rates, both of which ensure a monotonic decrease in the loss function. The first choice is based on the global Lipschitz constant, denoted by lr^G. The second choice is derived from the local Lipschitz constant, denoted by lr^*. These two learning rates are applied to both single-layered and multi-layered networks. In all cases, it holds that $lr^G \leq lr^*$ theoretically. We empirically compare the loss functions corresponding to these two choices.

Remark (*Determination of Batch Size*)**.** In order to apply Corollary 2, it is crucial to determine a batch size $|B|$ such that the batched data is a representative sample of the original dataset. We propose a simple strategy to determine the batch size $|B|$ based on the theory of inference for multivariate normal distributions. Assume that the given D-dimensional data $X_1, X_2, \dots, X_N$ are i.i.d. and follows $N_D(\mu, \Sigma)$. Then, for any sample size n_{sample}, it is well-known that the quantity $n_{\text{sample}}\left(\bar{X}-\mu\right)^T\Sigma^{-1}\left(\bar{X}-\mu\right)$ is distributed as chi-squared χ_D^2 with D degrees of freedom, i.e.

$$n_{\text{sample}}\left(\bar{X}-\mu\right)^T\Sigma^{-1}\left(\bar{X}-\mu\right) \sim \chi_D^2$$

where $\bar{X}$ is the mean of the n_{sample} data. Based on this, we define an effective radius r_{eff} of the confidence region with α confidence as $r_{\text{eff}} = \frac{\lambda_{i_{\text{eff}}}(\Sigma)\chi_D^2(\alpha)}{n_{\text{sample}}}$ where $\lambda_{i_{\text{eff}}}(\Sigma)$ is the i_{eff}th eigenvalue of Σ in decreasing order. In this work, the batch size $|B|$ is selected such that the effective radius r_{eff} is less than or equal to $\frac{1}{2}\|\mu\|$, i.e.,

$$r_{\text{eff}} = \frac{\lambda_{i_{\text{eff}}}(\Sigma)\chi_D^2(\alpha)}{|B|} \leq \frac{1}{2}\|\mu\| \iff |B| = \frac{\lambda_{i_{\text{eff}}}(\Sigma)\chi_D^2(\alpha)}{\frac{1}{2}\|\mu\|} \tag{2}$$

with $i_{\text{eff}} = \lfloor 0.02D \rfloor$.

4.2. Multi-layered networks

As with a single-layer network, we observe a series of perturbations initiated by the weight update $W \leftarrow W - lr\cdot dW$ in SGD. These perturbations propagate through the forward and backward passes, culminating in the gradient $\nabla f(W - lr\cdot dW)$. In this section, we estimate the local Lipschitz constant, which is the ratio of the gradient perturbation to that of the weights.

Consider the two-layer network in Fig. 2. We adopt a layerwise update strategy, where the update of W_1 in the two-layer network is analogous to the update of W_0 in the single-layer network. Thus, it suffices to discuss the layerwise update of W_0 in the two-layer network. The variables and their gradients are updated during the forward and backward passes as follows:

Forward pass	Backward pass
$p_0 = x_0 W_0$	$dp_1 = \sigma(p_1) - y_0$
$x_1 = \sigma(p_0)$	$dx_1 = dp_1\cdot W_1^T$
$p_1 = x_1 W_1$	$dp_0 = \sigma'(p_0)\cdot dx_1$
$x_2 = \sigma(p_1)$	$dW_0 = x_0^T\cdot dp_0$
$f(W_0) = -x_2\log(y_0) - (1-x_2)\log(1-y_0)$	

Consider $t \in [0, lr]$ and a perturbation $\Delta W_0 = -t\cdot dW_0$ on W_0. The series of perturbations generated by ΔW_0 propagates through the forward and backward passes, and we estimate them as follows:

Perturbation estimation	Gradient estimation
$\|\Delta p_0\| \leq \|x_0\|\|\Delta W_0\|$	$\|\Delta(dp_1)\| \leq L(\sigma)\|\Delta p_1\|$
$\|\Delta x_1\| \leq \left\|\sigma'(minmod(p_0, p_0+\Delta p_0))\right\|_\infty\|\Delta p_0\|$	$\|\Delta(dx_1)\| \leq \|\Delta(dp_1)\|\|W_1\|$
$\|\Delta p_1\| \leq \|\Delta x_1\|\|W_1\|$	$\|\Delta(dp_0)\| \leq L(\sigma)\|\Delta(dx_1)\| + L(\sigma')\|dx_1\|_\infty\|\Delta p_0\|$
	$\|\Delta(dW_0)\| \leq \|x_0\|\|\Delta(dp_0)\|$

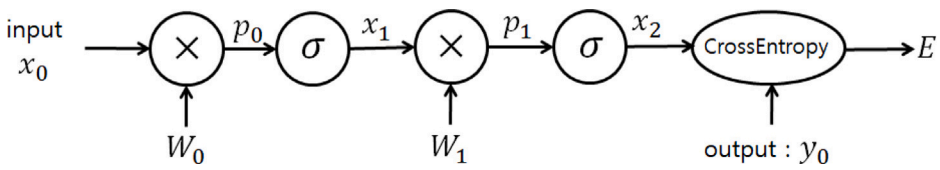


Fig. 2. Two-layered network.

Using the above estimates, the local Lipschitz constant is bounded above as

$$L\left(\nabla f,\left[W_0, W_0 - lr\cdot dW_0\right]\right)$$
$$\le \left\|x_0\right\|^2\left(L\left(\sigma'\right)\left\|dx_1\right\|_\infty + L\left(\sigma\right)^2\left\|W_1\right\|^2\left\|\sigma'\left(minmod\left(p_0, p_0+\Delta p_0\right)\right)\right\|_\infty\right),$$

and the estimation of the global Lipschitz constant is given by

$$L\left(\nabla f\right) \le \left\|x_0\right\|^2\left(L\left(\sigma'\right)\left\|dx_1\right\|_\infty + L\left(\sigma\right)^3\left\|W_1\right\|^2\right).$$

Corollary 1 together with the global Lipschitz constant gives a naive relaxation of the line search problem

$$f\left(W_0 - lr\cdot dW_0\right)$$
$$\le f\left(W_0\right) + \left\|dW_0\right\|^2\left(-lr + lr^2\left\|x_0\right\|^2\left(L\left(\sigma'\right)\left\|dx_1\right\|_\infty + L\left(\sigma\right)^3\left\|W_1\right\|^2\right)\right),$$

and the solution of the naive relaxation is readily given as

$$lr^G = \frac{1}{2\left\|x_0\right\|^2\left(L\left(\sigma'\right)\left\|dx_1\right\|_\infty + L\left(\sigma\right)^3\left\|W_1\right\|^2\right)}.$$

Here $L(\sigma) = 1/4$ and $L(\sigma') = 1/\left(6\sqrt{3}\right)$. On the other hand, Theorem 1 gives another relaxation that involves $L\left(\nabla f,\left[W_0, W_0 - lr\cdot dW_0\right]\right)$ that is bounded by $\left\|x_0\right\|^2\cdot L\left(\sigma,\left[p_0, p_0 - lr\cdot x_0^T dW_0\right]\right)$ which is equal to $\sigma'(\text{minmod}(p_0, p_0 - lr\cdot x_0^T dW_0))$ by (1). The following theorem states the relaxation and its convergence results. Since the proof of the theorem is very analogous to that of Theorem 3, we omit the details.

Theorem 4 (*Two Layers*). *For any W_0 and any $lr \ge 0$, it holds in the two-layered network that*

$$f\left(W_0 - lr\cdot dW_0\right) \le f\left(W_0\right) + \left\|dW_0\right\|^2 g\left(lr\right),$$

where

$$g\left(lr\right) := -lr + lr^2\left\|x_0\right\|^2\left(L\left(\sigma'\right)\left\|dx_1\right\|_\infty + L\left(\sigma\right)^2\left\|W_1\right\|^2\left\|\sigma'\left(minmod\left(p_0, p_0+\Delta p_0\right)\right)\right\|_\infty\right).$$

The function $g(lr)$ is continuous on $[0,\infty)$ and satisfies

$$g\left(lr\right) \ge -lr + lr^2\left\|x_0\right\|^2\left(L\left(\sigma'\right)\left\|dx_1\right\|_\infty + L\left(\sigma\right)^2\left\|\sigma'\left(p_0\right)\right\|_\infty\left\|W_1\right\|^2\right)$$
$$g\left(lr\right) \le -lr + lr^2\left\|x_0\right\|^2\left(L\left(\sigma'\right)\left\|dx_1\right\|_\infty + L\left(\sigma\right)^3\left\|W_1\right\|^2\right).$$

Let lr^* be the argmin of $g(lr)$ on the interval

$$\left[lr^G = \frac{1}{2\left\|x_0\right\|^2\left(L\left(\sigma'\right)\left\|dx_1\right\|_\infty + L\left(\sigma\right)^3\left\|W_1\right\|^2\right)},\right.$$
$$\left.\frac{1}{2\left\|x_0\right\|^2\left(L\left(\sigma'\right)\left\|dx_1\right\|_\infty + L\left(\sigma\right)^2\left\|\sigma'\left(p_0\right)\right\|_\infty\left\|W_1\right\|^2\right)}\right],$$

then $g(lr^*) \le 0$ and we have monotonic decrease on the cost,

$$f\left(W_0 - lr^*\cdot dW_0\right) \le f\left(W_0\right) + \left\|dW_0\right\|^2 g\left(lr^*\right) \le f\left(W_0\right).$$

In a three-layered network, the layerwise updates of weights W_1 and W_2 are simple to state and analyze. The updates are equivalent to the update of W_0's in the single and double layered networks, respectively. Thus, again, it is enough to discuss the update of W_0. In the update $W_0 \leftarrow W_0 - lr\cdot dW_0$, Theorem 1 gives the relaxation problem from which lr is determined . The relaxation requires an estimation of local Lipschitz constant $L\left(\nabla f,\left[W_0, W_0 - lr\cdot dW_0\right]\right)$, the maximum ratio of $\left\|\Delta\left(dW_0\right)\right\|$ to $\left\|\Delta W_0\right\|$. For any $t \in [0, lr]$, the perturbation $\Delta W_0 = -t\cdot dW_0$ generates a series of perturbations in the forward and backward passes as

$$\begin{cases} \left\|\Delta p_0\right\| \le & \left\|x_0\right\|\left\|\Delta W_0\right\| \\ \left\|\Delta x_1\right\| \le & L\left(\sigma,\left[p_0, p_0+\Delta p_0\right]\right)\left\|\Delta p_0\right\| \\ \left\|\Delta p_1\right\| \le & \left\|\Delta x_1\right\|\left\|W_1\right\| \\ \left\|\Delta x_2\right\| \le & L\left(\sigma\right)\left\|\Delta p_1\right\| \\ \left\|\Delta p_2\right\| \le & \left\|\Delta x_2\right\|\left\|W_2\right\| \end{cases}$$

and

$$\begin{cases} \left\|\Delta\left(dp_2\right)\right\| \le & L\left(\sigma\right)\left\|\Delta p_2\right\| \\ \left\|\Delta\left(dx_2\right)\right\| \le & \left\|\Delta\left(dp_2\right)\right\|\left\|W_2\right\| \\ \left\|\Delta\left(dp_1\right)\right\| \le & L\left(\sigma\right)\left\|\Delta\left(dx_2\right)\right\| + L\left(\sigma'\right)\left\|dx_2\right\|_\infty\left\|\Delta p_1\right\| \\ \left\|\Delta\left(dx_1\right)\right\| \le & \left\|\Delta\left(dp_1\right)\right\|\left\|W_1\right\| \\ \left\|\Delta\left(dp_0\right)\right\| \le & L\left(\sigma\right)\left\|\Delta\left(dx_1\right)\right\| + L\left(\sigma'\right)\left\|dx_1\right\|_\infty\left\|\Delta p_0\right\| \\ \left\|\Delta\left(dW_0\right)\right\| \le & \left\|x_0\right\|\left\|\Delta\left(dp_0\right)\right\| \end{cases}.$$

Combining the inequalities from $\left\|\Delta W_0\right\|$ to $\left\|\Delta\left(dW_0\right)\right\|$, we obtain an estimate of the local Lipschitz constant,

$$L\left(\nabla f,\left[W_0, W_0 - lr\cdot dW_0\right]\right) \le \left\|x_0\right\|^2\left(\left\|\sigma'\text{minmod}(p_0, p_0+\Delta p_0)\right\|_\infty L\left(\sigma\right)\left\|W_1\right\|^2 \cdot\left(L\left(\sigma\right)^3\left\|W_2\right\|^2 + L\left(\sigma'\right)\left\|dx_2\right\|_\infty\right) + L\left(\sigma'\right)\left\|dx_1\right\|_\infty\right)$$

from which Theorem 1 leads to a relation

$$f\left(W_0 - lr\cdot dW_0\right) \le f\left(W_0\right) + \left\|dW_0\right\|^2 g\left(lr\right),$$

where

$$g\left(lr\right) := -lr + lr^2\left\|x_0\right\|^2\left(\left\|\sigma'\text{minmod}(p_0, p_0+\Delta p_0)\right\|_\infty L\left(\sigma\right)\left\|W_1\right\|^2 \cdot\left(L\left(\sigma\right)^3\left\|W_2\right\|^2 + L\left(\sigma'\right)\left\|dx_2\right\|_\infty\right) + L\left(\sigma'\right)\left\|dx_1\right\|_\infty\right).$$

As in the single-, two-, and three-layer cases, the relaxation functional $g(lr)$ for multi-layer networks contains no matrix–vector multiplications and can be minimized efficiently by one-dimensional search methods (e.g. golden-section search). The layerwise update rule extends recursively to deeper networks: the final layer reduces to the single-layer case, the second-to-last layer to the two-layer case, and so on. To be precise, index the N-layer network by $n = 0, 1, \dots, N-1$. For each layer define

$$g_n(lr) = -lr + lr^2 S_n,$$

with (see Box I) where the auxiliary sequence R_k is defined backward by

$$R_{N-1} = L(\sigma')\left\|dx_{N-1}\right\|_\infty + L(\sigma)^3\left\|W_{N-1}\right\|^2,$$

and for $2 \le k \le N-2$,

$$R_k = L(\sigma')\left\|dx_k\right\|_\infty + L(\sigma)^2\left\|W_k\right\|^2 R_{k+1}.$$

(When implementing, compute R_{N-1} first and then recursive downward to obtain the needed R_{n+2}.)

$$S_n = \begin{cases} \left\| \sigma'\left(\text{minmod}(p_n, p_n + \Delta p_n)\right) \right\|_\infty, & n = N-1, \\ L(\sigma)^2 \left\| W_{n+1} \right\|^2 \left\| \sigma'\left(\text{minmod}(p_n, p_n + \Delta p_n)\right) \right\|_\infty + L(\sigma') \left\| dx_{n+1} \right\|_\infty, & n = N-2, \\ L(\sigma) \left\| W_{n+1} \right\|^2 \left\| \sigma'\left(\text{minmod}(p_n, p_n + \Delta p_n)\right) \right\|_\infty R_{n+2} + L(\sigma') \left\| dx_{n+1} \right\|_\infty, & 0 \le n \le N-3, \end{cases}$$

Box I.

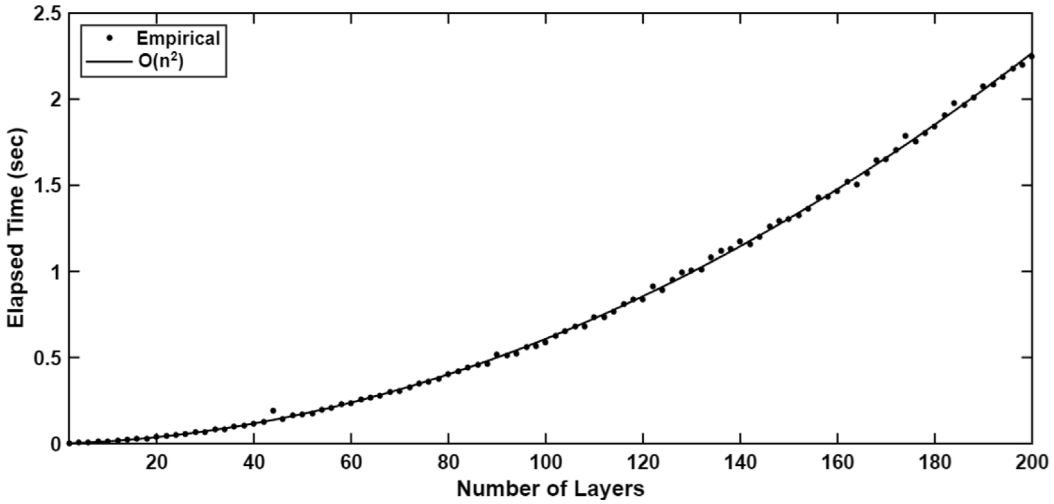


Fig. 3. Elapsed time of the procedure to determine lr versus number of layers. Black circles indicate empirical measurements, and the line shows a quadratic fit, illustrating the observed $O(n^2)$.

In terms of computational cost, the procedure scales favorably with network depth. Since evaluating each $g_n(lr)$ requires only simple scalar operations and norms, and the one-dimensional minimization is performed exactly once per layer, the overall complexity grows essentially with the number of layers. Fig. 3 reports the empirical runtime of the procedure for determining lr in networks with up to 200 layers. The measured cost exhibits a clear quadratic trend with respect to the number of layers, closely matching the fitted $O(n^2)$ curve. This behavior is consistent with the recursive structure of the backward construction of the quantities R_k, which induces a cumulative cost scaling proportional to $1 + 2 + \cdots + N = O(N^2)$. Despite this quadratic dependence, the absolute times remain small (around two seconds even for 200 layers), confirming that the method is computationally lightweight in practice.

4.3. Overshooting, catapults, and generalization dynamics

While the preceding subsections have focused on the mathematical formulation of layerwise adaptive learning rates, it is also important to place our approach in the broader context of empirical optimization phenomena observed in deep learning. In particular, several works have emphasized the counter-intuitive role of "overshooting" — that is, taking learning rates large enough to temporarily increase the loss. Such behavior has been associated with the so-called *catapult effect* [19,20], where transient spikes in the training loss can accelerate convergence and, more importantly, lead to better generalization by escaping narrow, non-generalizing valleys. Related practices such as one-cycle scheduling [21] exploit a similar principle by rapidly ramping learning rates towards the brink of divergence. Furthermore, architectural mechanisms such as batch normalization also alter learning rate dynamics by introducing scale-invariance and affecting gradient sensitivity [22]. In this subsection, we discuss how these phenomena relate to our proposed method, clarifying its behavior in the presence of overshooting, scheduling strategies, and normalization effects.

4.3.1. Catapult effect and Lipschitz-based learning rates

It has been observed that excessively large learning rates can trigger the *catapult effect*, characterized by temporary divergence of the training loss followed by rapid re-stabilization and improved generalization [19,20]. At first sight, one might suspect that controlling learning rates through Lipschitz constants could suppress this phenomenon. The next lemma shows that the relevant Lipschitz constants are themselves controlled by the largest eigenvalue of the Hessian along the update segment, clarifying that our scheme does not, in principle, exclude the catapult regime.

Lemma 1 (*Hessian Bound for the Local Lipschitz Constant of* ∇f)**.** Let $f : \mathbb{R}^d \to \mathbb{R}$ be twice continuously differentiable and let $x, y \in \mathbb{R}^d$. Along the segment $[x, y]$,

$$\|\nabla f(y) - \nabla f(x)\| \le \left(\sup_{z \in [x,y]} \|H_f(z)\|_2 \right) \|y - x\|.$$

Consequently,

$$L\big(\nabla f, [x, y]\big) \le \sup_{z \in [x,y]} \|H_f(z)\|_2.$$

If f is convex, then $\|H_f(z)\|_2 = \lambda_{\max}(H_f(z))$, hence

$$L\big(\nabla f, [x, y]\big) \le \sup_{z \in [x,y]} \lambda_{\max}\big(H_f(z)\big).$$

Proof. Let $\gamma(t) = x + t(y - x)$. By the fundamental theorem of calculus, $\nabla f(y) - \nabla f(x) = \int_0^1 H_f(\gamma(t))(y - x)\, dt$. Taking norms and using submultiplicativity gives $\|\nabla f(y) - \nabla f(x)\| \le \int_0^1 \|H_f(\gamma(t))\|_2 \|y - x\|\, dt \le \big(\sup_{z \in [x,y]} \|H_f(z)\|_2\big)\|y - x\|$. The stated consequences follow, and in the convex case $\|H_f(z)\|_2 = \lambda_{\max}(H_f(z))$. □

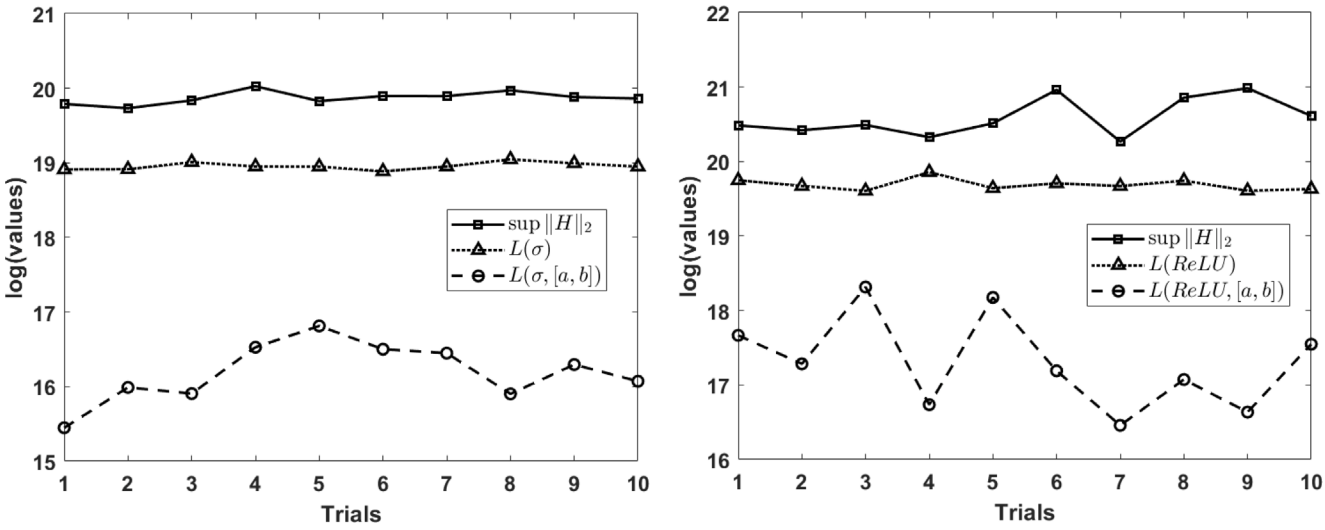


Fig. 4. $\sup\|H\|_2$, local, and global Lipschitz constants for sigmoid (left) and ReLU (right) from a simple two-layered network ($d_{\text{in}} = 2$, $d_h = 5$, $d_{\text{out}} = 2$) with randomly initialized weights, where local values are computed along 10 random line segments in parameter space.

Specializing Lemma 1 to the update segment $[W,\ W - lr \cdot dW]$ yields

$$L\big(\nabla f,\ [W,\ W - lr \cdot dW]\big) \ \le \sup_{Z \in [W,\ W - lr \cdot dW]} \|H_f(Z)\|_2$$
$$(\text{convex:}\ \le \sup_Z \lambda_{\max}(H_f(Z))).$$

Thus any Lipschitz-based learning-rate rule in our framework is governed by the largest curvature encountered along the step. In particular, fixed-step descent guarantees (e.g., Theorem 2) follow by upper-bounding this quantity, while learning rates chosen near the threshold $1/\|H_f\|_2$ (convex: $1/\lambda_{\max}(H_f)$) remain within reach. Hence, our approach does not preclude training in the near-divergent regime where catapult phenomena have been observed [19,20]. Fig. 4 illustrates the relationship between the local Lipschitz constants, the spectral norm of the Hessian along each segment, and the global Lipschitz estimate for a simple network. As predicted by Lemma 1, the local Lipschitz constant is always bounded by the maximum Hessian eigenvalue along the segment, and both remain below the global Lipschitz estimate. This visualization confirms that our theoretical bound accurately captures the curvature behavior of the network.

4.3.2. Connection to cyclical learning rates

Our layerwise learning rate is chosen inversely to a local curvature proxy on the update segment, so (suppressing layer indices)

$$\eta_t \ \propto \ \frac{1}{\widehat{L}_t}, \qquad \Rightarrow \qquad \log_{10}\eta_t \ = \ C - \log_{10}\widehat{L}_t,$$

with $\widehat{L}_t \le \sup_{Z \in [W_t,\ W_t - \eta_t dW_t]} \|H_f(Z)\|_2$ by Lemma 1. Consequently, any oscillation in the encountered curvature $\widehat{L}_t$ (driven by batch-to-batch gradient variability, normalization re-scalings, or traversing alternating sharp/flat regions) appears as an opposite-phase oscillation in $\log_{10}\eta_t$. This explains the ripples in Fig. 5: when curvature temporarily increases, $\log_{10}\eta_t$ dips; when curvature relaxes, $\log_{10}\eta_t$ rises. In other words, our rule induces *endogenous cyclical learning rates* (CLR) without an explicit schedule.

Fig. 5 shows two contrasting behaviors. Under the naive relaxation, the envelope of $\log_{10}\eta_t$ remains nearly flat: the global curvature bound does not track segment-wise variations, so the learning rate oscillates around a roughly constant mean, resembling standard CLR without drift. In contrast, the proposed relaxation captures segment-wise curvature changes, producing both oscillations and a slowly rising envelope. This upward drift reflects the gradual reduction of local curvature along the optimization path: as training escapes sharper regions and moves toward flatter basins, $\sup_Z \|H_f(Z)\|_2$ decreases, allowing inverse-curvature step sizes to increase.

Classical CLR prescribes exogenous up- and down-ramps of the learning rate to probe the brink of instability and then anneal. Our method achieves a similar effect *implicitly*: the locally estimated curvature naturally produces short cycles (fast oscillations from batch-level curvature fluctuations) within a slowly varying envelope (longer-term curvature drift), placing the trajectory intermittently near the $1/\|H_f\|$ threshold discussed above. Thus, the observed oscillatory $\log_{10}\eta_t$ with a moving envelope can be interpreted as a data-driven analogue of CLR, no extra hyperparameters or scheduling are required, yet the training dynamics still benefit from periodic excursions toward the near-divergent regime associated with improved exploration.

The oscillations and their rising envelope in Fig. 5 are not artifacts but a direct consequence of $\eta_t \ \propto \ 1/\widehat{L}_t$: they mirror the local curvature cycles encountered during training and provide an endogenous, curvature-aware counterpart to cyclical learning rate strategies.

4.3.3. Normalization-induced scale invariance and its interaction with step-size rules

Normalization layers such as batch normalization (BN) have long been known to alter the effective learning rate dynamics by rescaling activations and gradients. The authors in [22] provided a systematic account of how BN and related mechanisms can induce exploding gradients, showing that seemingly benign rescaling operations may, in fact, amplify sensitivity during training. More recent theoretical models further clarify that layer-wise sensitivity is not constant but can fluctuate significantly throughout optimization, complicating the task of prescribing globally stable learning rates.

From the perspective of our framework, these phenomena are naturally absorbed into the local Lipschitz estimates: if a BN layer temporarily amplifies the effective curvature, then the segment-wise Lipschitz constant increases and our rule automatically prescribes a smaller step. Conversely, when normalization induces scale-invariance, the Lipschitz constant decreases, allowing larger steps without risking divergence. In this way, our method adapts to the layer-wise fluctuations induced by normalization without the need for explicit reparameterization or delicate scheduling.

It is worth noting that scale-invariance also changes the qualitative form of "overshooting." Rather than pushing weights towards divergence, a large step along a scale-invariant direction simply rescales the parameters without increasing the loss. In practice, this means that overshooting under BN does not lead to the catastrophic loss spikes

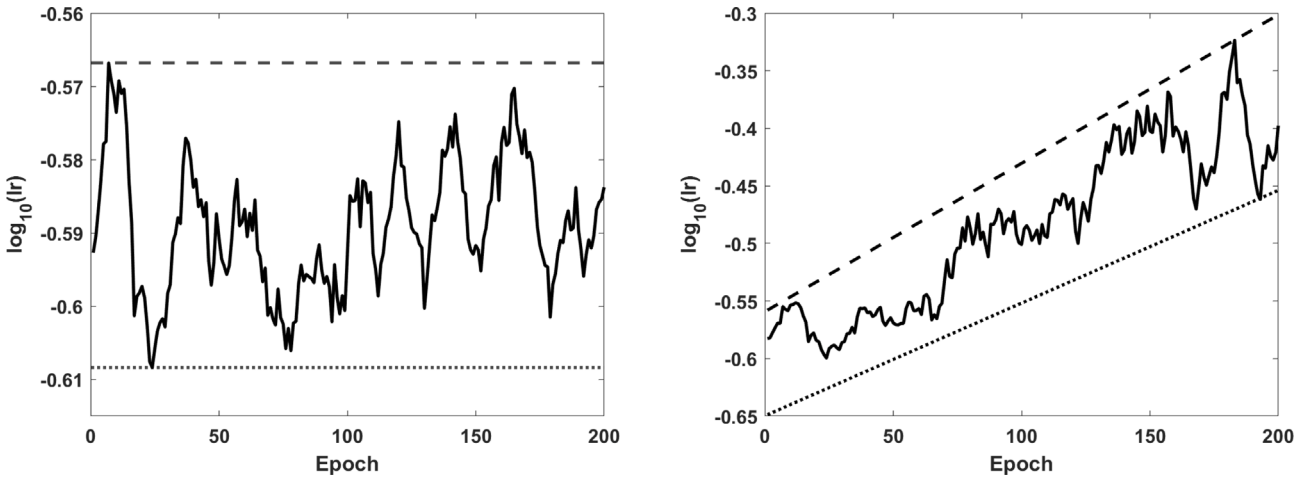


Fig. 5. Evolution of learning rates for the naive (left) and proposed (right) relaxations using a single-layered network on the SONAR dataset.

associated with the catapult effect, but rather to a freezing of progress along the invariant directions. Our local Lipschitz rule is consistent with this behavior: the estimated curvature remains small along invariant subspaces, so the method does not waste effort in suppressing harmless rescalings, while still controlling instability in the sensitive directions. This highlights a potential strength of our approach: it can flexibly accommodate normalization-induced fluctuations while retaining a principled safeguard against genuine divergence.

5. Numerical experiments

In this section, several numerical experiments are carried out to verify the mathematical analyses in Sections 2 and 4, and validate the outperformance of our proposed method in training the well-known SONAR [23] and MNIST [24] datasets. Each relaxation to determine the optimal learning rate lr^* is directly solved by golden-section search, which begins with the closed interval $[a, b]$ that Theorems 3 and 4 provide and iteratively reduces the interval length until $(b - a) < 10^{-5}$. To validate the effectiveness of the relaxations, the resulting loss values are compared with those obtained from well-known optimization techniques, including momentum-based methods such as AdaGrad [2] and ADAM [1], and a second-order natural-gradient method, K-FAC [25]. All the computations were implemented in C++, and conducted on a regular personal computer with 3.60 GHz CPU and 16.0 GB memory.

5.1. Relaxation

We begin with numerical validation on the relaxation, stated in Theorem 1, which is the main ingredient of the proposed method. As a simple but motivating example, let us consider a three-layered neural network for the binary classification, whose activation functions are all sigmoid functions. Then, from Theorem 1 for given weights W_0, W_1, W_2, the estimations for the relaxation

$$\begin{aligned} f\left(W_0 - lr \cdot dW_0, W_1, W_2\right) &\le f\left(W_0, W_1, W_2\right) + \left\|dW_0\right\|^2 g_0\left(lr\right) \\ f\left(W_0, W_1 - lr \cdot dW_1, W_2\right) &\le f\left(W_0, W_1, W_2\right) + \left\|dW_1\right\|^2 g_1\left(lr\right) \\ f\left(W_0, W_1, W_2 - lr \cdot dW_2\right) &\le f\left(W_0, W_1, W_2\right) + \left\|dW_2\right\|^2 g_2\left(lr\right) \end{aligned} \tag{3}$$

holds for any $lr \ge 0$ where g_0, g_1, g_2 are defined as in Section 4.

Fig. 6 describes the evolutions of the loss function with one weight update while others fixed, naive relaxation, and proposed relaxation, which verify theorems and corollaries in Section 2. The plots reveal that the proposed relaxation expects more gains on the learning rate for updating W_2 than the naive one, but almost same, visually indistinguishable, performance for W_0 and W_1. In particular, the plot corresponding to W_2 demonstrates a visibly larger gap between the proposed (solid line) and naive (dashed line) relaxation curves, where the proposed method consistently achieves lower loss values over the range of learning rates. This implies that our proposed relaxation yields a more advantageous update direction and expects greater loss reduction, especially when the curvature of the loss surface varies more significantly, as is the case for W_2.

In order to check the computational cost of the relaxations, the operation counts for evaluating the loss function, naive relaxation, and proposed relaxation in training SONAR data are presented in Table 1. For multi-layered networks, the size of hidden layers are selected to be $H = 80$. As we expected, evaluating the relaxations are computationally much cheaper than the loss function, which arise from the nature of the relaxations, that is free of matrix–vector multiplications.

5.2. Single-layered network

In this example, a single-layered neural network is considered for training the SONAR dataset. In order to validate the efficiency of lr taken from Theorem 3 in Section 4.1, we present a result of training with an SGD optimizer. Due to the limitation in the performance of single-layered network, the whole dataset for training is confined to be the first 80 data of the SONAR dataset. Since $\|\mu\| = 2.749517$ and $\lambda_{i_{\text{eff}}}(\Sigma) = \lambda_{\lfloor 0.02\times 60\rfloor}(\Sigma) = \lambda_1(\Sigma) = 0.5428$ for the first 80 SONAR data, a proper batch size $|B|$ from (2) with $\alpha = 0.95$ is $|B| = \lambda_{i_{\text{eff}}}(\Sigma)\,\chi^2_D(\alpha) \,/\, \left(\frac{1}{2}\|\mu\|\right) = 17.0536 \approx 17$.

The evolutions of learning rates determined from the naive and proposed relaxation approaches are plotted in the left of Fig. 7. The result shows that lr from Theorem 3 tends to increase as the training progresses while the other is not. From this observation, a faster decay in the loss function can be expected when lr is computed from the local Lipschitz analysis. The evolutions of loss functions on the right of Fig. 7 demonstrate that this expectation is met, and that the proposed relaxation achieves comparable or faster convergence than well-known optimization methods such as AdaGrad, ADAM, and K-FAC. We also note that ADAM performs surprisingly well even in this simple setting, often yielding competitive long-term convergence behavior. Nevertheless, the proposed relaxation offers a clear advantage in the early stages of training, where its theoretically motivated step-size selection leads to faster initial descent of the loss.

5.3. Multi-layered network

In order to verify the scalability of the proposed relaxation approach to the multi-layered network, as introduced in Section 4.2, we construct two- and three-layered networks to train SONAR and binary MNIST

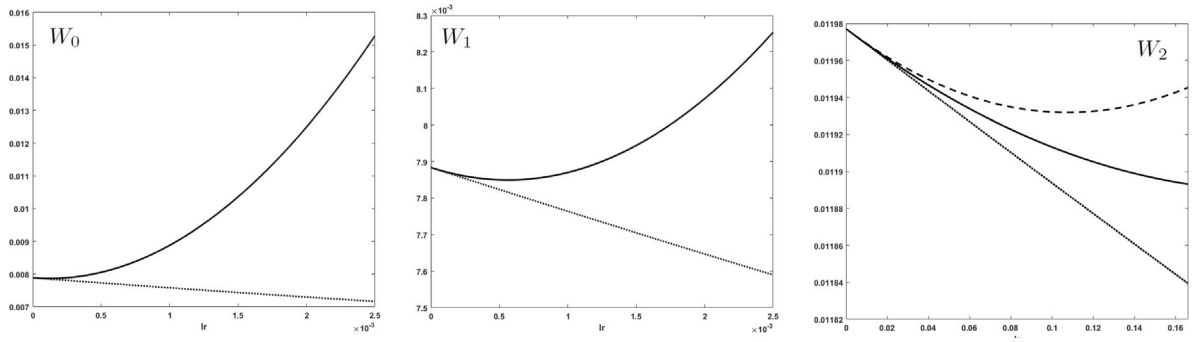


Fig. 6. Three-layered network with the SONAR dataset. Each plot shows the loss evolution with respect to a learning rate (lr) for a single weight update: W_0 (left), W_1 (middle), and W_2 (right). The dotted line (:) indicates the loss when updating only the target weight while keeping others fixed. The solid line (-) represents our proposed relaxation method, and the dashed line (–) corresponds to a naive relaxation approach. In all cases, the proposed relaxation achieves lower loss values than the naive relaxation, particularly evident in the W_2 plot, indicating greater expected gains in loss reduction.

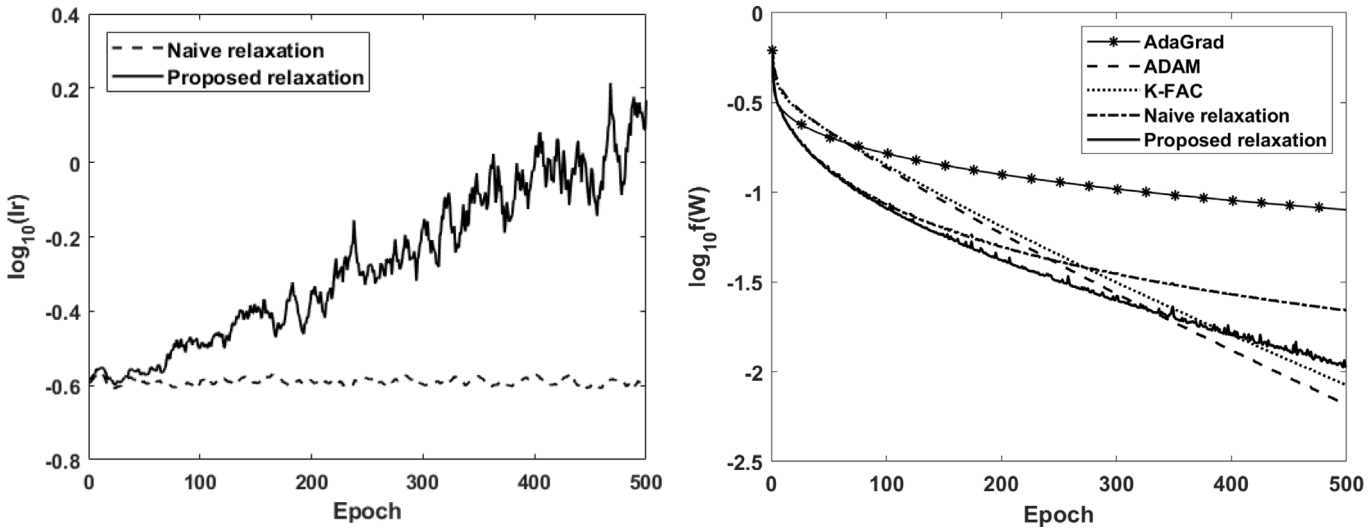


Fig. 7. Single-layered network with SONAR dataset. The evolution of learning rates from the naive and proposed relaxation (left) and its corresponding loss functions compared with those of AdaGrad, ADAM, and K-FAC (right). The results show that an adaptive choice of lr from corollary 2 enhances the performance of SGD in the training of a single-layered network.

Table 1
The operation counts for the loss function, naive relaxation, and the proposed relaxation.

	Operation counts		
	Naive relaxation	Proposed relaxation	Loss function
Single layer	3	8	66
Two layer	7	12	5046
Three layer	11	16	11 606

dataset, respectively. A binary MNIST dataset (N = 12,665) consists of the data labeled $\{0, 1\}$ from the MNIST dataset. From the universal approximation theorem [26], a proper choice of the size of hidden layers in the network with more than one hidden layer allows us to utilize whole data for training. In this example, we set the size of hidden layers as $H = 80$ for both two- and three-layered nets, and the batch size of SGD is determined by Eq. (2) ($\left|B_{\text{SONAR}}\right| = 17$, $\left|B_{\text{binary MNIST}}\right| = 133$).

The evolutions of loss functions in Fig. 8 indicate that the proposed layerwise adaptation techniques (naive and proposed relaxations) achieve convergence behavior comparable to established optimization methods such as AdaGrad, ADAM, and K-FAC. In the two-layered SONAR case (left), our method converges slightly more slowly than ADAM and K-FAC, but still reaches a stable solution, while in the three-layered MNIST case (right), it performs on par with these optimizers. These results suggest that while ADAM often excels in terms of long-run convergence speed, the proposed approach provides a competitive and robust alternative across different architectures and datasets, offering a theoretically grounded mechanism for stable convergence.

To further evaluate the robustness and depth scalability of the proposed relaxation framework, we additionally consider networks with up to 15 layers on the MNIST dataset, which corresponds to the deepest architecture without batch normalization for which any method remains trainable before severe gradient-vanishing effects cause complete failure. While ADAM typically performs very well in shallow and moderately deep networks, the results in Fig. 9 show that ADAM and K-FAC begin to suffer from stagnant or highly unstable training behavior once the network exceeds roughly 10 layers, whereas both of our relaxation-based strategies continue to make steady progress and successfully optimize the objective up to 15 layers. This improved stability arises from the layerwise relaxation mechanism, which adaptively adjusts the effective step size based on per-layer Lipschitz constants. As a result, the proposed method avoids the overly conservative learning rates that traditional optimizers fall into in deep settings, thereby mitigating gradient-vanishing issues and maintaining meaningful updates even when gradients are extremely small. These findings demonstrate that the relaxation framework is not only competitive in shallow

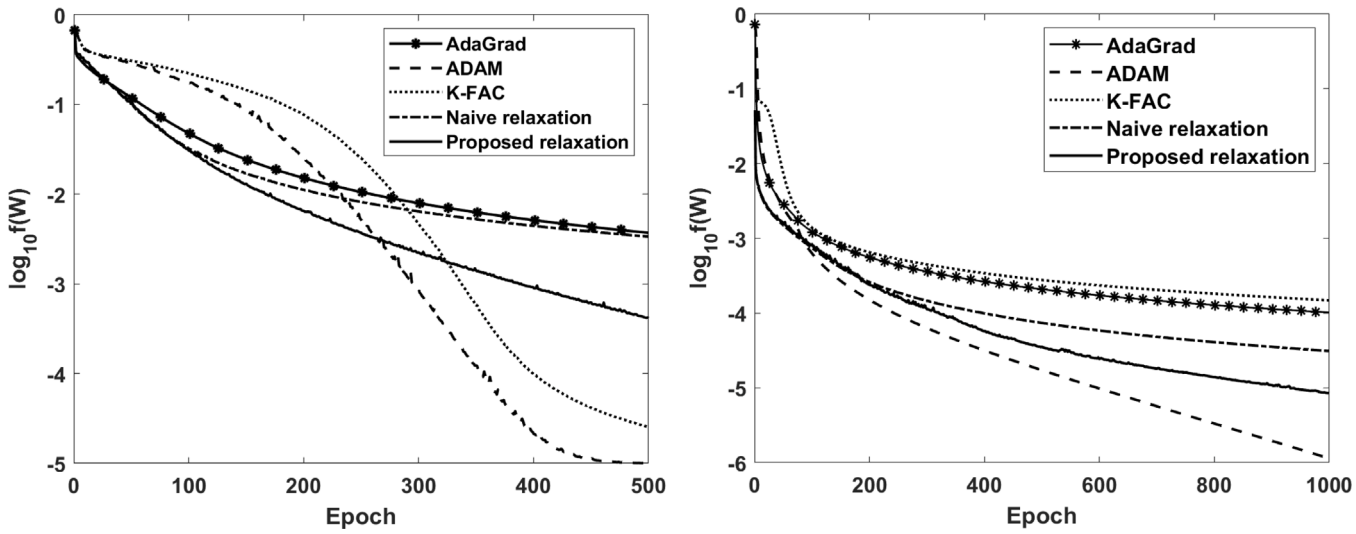


Fig. 8. Evolutions of loss functions of a two-layered network with SONAR (left) and a three-layered network with binary MNIST (right), both compared with those of AdaGrad, ADAM, and K-FAC. These results validate that the proposed relaxation is more effective than the naive relaxation.

and moderate-depth networks but also substantially more resilient to depth-induced degradation than widely used first- and second-order optimization methods.

6. Conclusion and future work

We introduced a layerwise method for training DNNs, and provided rigorous analyses that assure the monotonic convergence of the loss function. There are few training methods that can assure convergence in multi-layered networks, but the demand for such methods with convergence assured keeps increasing, especially when training is regularly carried out without human supervision. Another field in urgent need for such methods is training encrypted data without a decryption. Our proposed method supported by convergence analysis is expected to be utilized in the formentioned fields.

The layerwise update is implemented by solving one of two relaxations of the line-search minimization. While the line search involves matrix–vector multiplications that are costly in the computation, the two relaxations are free of the multiplications and can be directly solved. One relaxation is based on global Lipschitz constant and can be readily solved, and the other is based on local Lipschitz constant and is solved by golden-section search within a closed interval that our theory provides.

To compute the Lipschitz bound, we estimated the perturbation propagation from ΔW_0 to $\Delta(dW_0)$ throughout the forward and backward passes. Each nonlinear perturbation allows a choice between a global Lipschitz constant, which is simpler but potentially conservative, and a local Lipschitz constant, which is tighter but requires more careful evaluation. In pursuit of matrix–vector-free relaxations, we adopted local Lipschitz constants only for the term $\|\Delta\sigma(p_0)\|$ and used global constants elsewhere. While incorporating additional local constants can yield a tighter relaxation, it would reintroduce matrix–vector multiplications. Our numerical tests on the SONAR dataset illustrate that, depending on the layer, the resulting relaxation can closely resemble the naive relaxation, indicating that the optimal balance between global and local constants is application-dependent.

Beyond the theoretical analysis, we expanded the empirical study to include networks up to 15 layers, which is the maximum depth that remains trainable without batch normalization before severe gradient-vanishing effects cause all standard methods to fail. These experiments show that while optimizers such as ADAM and K-FAC begin to stagnate or destabilize beyond roughly 10 layers, the proposed relaxation-based method continues to make consistent progress and successfully trains networks of depth 15. This demonstrates both the robustness and the practical scope of the method in realistic deep-learning settings.

We further observe that, for substantially deeper networks, the proposed method exhibits limitations similar to those encountered by standard optimization techniques. In particular, as the network depth increases beyond this regime, training may fail to converge due to severe gradient-vanishing effects. This behavior is not specific to the proposed relaxation-based framework, but rather reflects a fundamental difficulty in gradient-based training of very deep architectures without additional stabilization mechanisms.

A related issue arises in more complex network structures involving residual connections or attention mechanisms. While the Lipschitz-based analysis developed in this work remains applicable at a local level, the recursive estimation of Lipschitz bounds may become increasingly conservative as depth and architectural complexity increase. In particular, the bilinear structure of self-attention complicates the derivation of sharp bounds and may exacerbate gradient attenuation when such estimates are applied repeatedly across many layers.

From a scalability perspective, an important limitation of the current framework lies in the recursive nature of the Lipschitz-bound estimation. While this recursion is essential for guaranteeing monotone descent, it can lead to increasingly conservative step-size bounds as network depth or architectural complexity grows. As a result, although the method scales well to shallow and moderately deep feedforward networks, its direct application to substantially deeper models or highly composite architectures may become impractical without further refinement. Addressing this trade-off between theoretical rigor and scalability is a key challenge for extending the proposed approach to modern large-scale neural networks.

Future work includes extending the relaxation framework with a more refined use of local Lipschitz constants, analyzing the interaction between batch normalization and the Lipschitz-based bounds, and exploring broader applications such as encrypted training and large-scale unsupervised learning, where guaranteed convergence is particularly valuable. In addition, addressing the conservativeness of recursive Lipschitz estimates in very deep or architecturally complex networks, including those with residual connections or attention mechanisms, remains an important direction for future investigation. Developing tighter yet computationally tractable bounds may help mitigate gradient-vanishing effects and broaden the applicability of the proposed framework to modern deep-learning architectures.

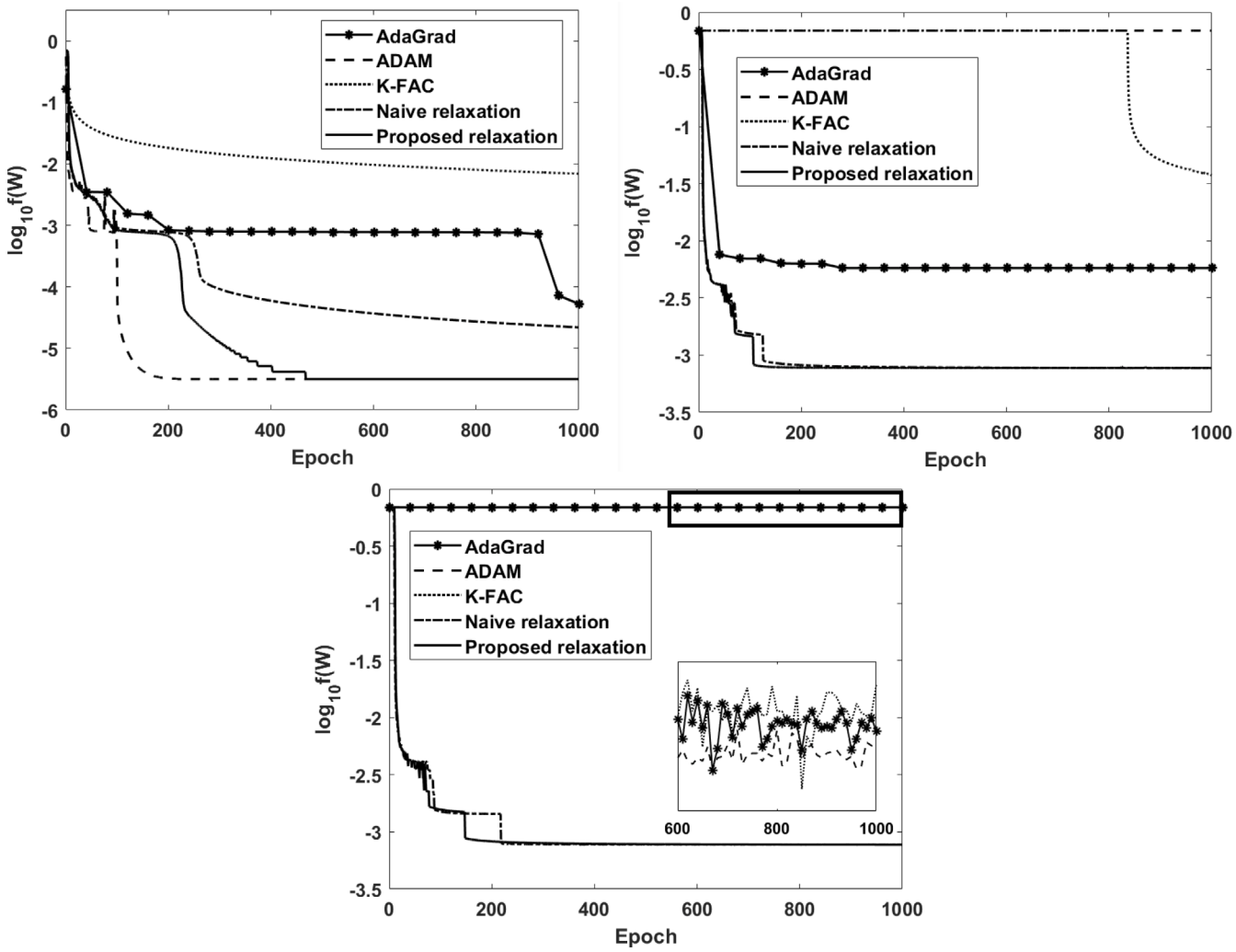


Fig. 9. Evolutions of the objective function for N–layer networks on binary MNIST: $N = 5$ (top left), $N = 10$ (top right), and $N = 15$ (bottom). All cases are compared with AdaGrad, ADAM, and K-FAC. The results show that the proposed relaxation-based methods effectively mitigate the gradient-vanishing problem by allowing larger learning rates derived from Lipschitz constants, whereas the other methods fail as the networks become deeper.

CRediT authorship contribution statement

Chohong Min: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Funding acquisition, Formal analysis, Conceptualization. **Byungjoon Lee:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Project administration, Methodology, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Byungjoon Lee reports financial support was provided by National Research Foundation of Korea. Chohong Min reports financial support was provided by National Research Foundation of Korea.

Acknowledgments

The research of Chohong Min was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education of Korea (Grant No. 2019R1A6A1A11051177 and 2021R1A2C1095703). The research of Byungjoon Lee was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) grant RS-2025-24523778. The authors are grateful to anonymous reviewers for their careful reading and valuable comments.

Data availability

Data will be made available on request.

References

[1] P Diederik. Kingma and jimmy ba. In: International conference on learning representations. 2015.
[2] Duchi John, Hazan Elad, Singer Yoram. Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res 2011;12(7).
[3] Ruder Sebastian. An overview of gradient descent optimization algorithms. 2016, arXiv preprint arXiv:1609.04747.
[4] Wu Xiaoxia, Ward Rachel, Bottou Léon. Wngrad: Learn the learning rate in gradient descent. 2018, arXiv preprint arXiv:1803.02865.
[5] Zeng Jinshan, Ouyang Shikang, Lau Tim Tsz-Kit, Lin Shaobo, Yao Yuan. Global convergence in deep learning with variable splitting via the kurdyka-łojasiewicz property. 2018, arXiv preprint arXiv:1803.00225.
[6] Johnson Rie, Zhang Tong. Accelerating stochastic gradient descent using predictive variance reduction. Adv Neural Inf Process Syst 2013;26:315–23.
[7] Reddi Sashank J, Hefny Ahmed, Sra Suvrit, Poczos Barnabas, Smola Alex. Stochastic variance reduction for nonconvex optimization. In: International conference on machine learning. PMLR; 2016, p. 314–23.

[8] Huang Feihu, Chen Songcan, Lu Zhaosong. Stochastic alternating direction method of multipliers with variance reduction for nonconvex optimization. 2016, arXiv preprint arXiv:1610.02758.
[9] Robbins Herbert, Monro Sutton. A stochastic approximation method. Ann Math Stat 1951;400–7.
[10] Nocedal Jorge, Wright Stephen. Numerical optimization. Springer Science & Business Media; 2006.
[11] Bertsekas Dimitri P. Nonlinear programming. J Oper Res Soc 1997;48(3):334.
[12] Vaswani Sharan, Mishkin Aaron, Laradji Issam, Schmidt Mark, Gidel Gauthier, Lacoste-Julien Simon. Painless stochastic gradient: Interpolation, line-search, and convergence rates. Adv Neural Inf Process Syst 2019;32:3732–45.
[13] Mahsereci Maren, Hennig Philipp. Probabilistic line searches for stochastic optimization. J Mach Learn Res 2017;18(1):4262–320.
[14] Scaman Kevin, Bach Francis, Bubeck Sébastien, Lee Yin, Massoulié Laurent. Optimal convergence rates for convex distributed optimization in networks. J Mach Learn Res 2019;20:1–31.
[15] Goodfellow Ian J, Shlens Jonathon, Szegedy Christian. Explaining and harnessing adversarial examples. 2014, arXiv preprint arXiv:1412.6572.
[16] Singh Bharat, De Soham, Zhang Yangmuzi, Goldstein Thomas, Taylor Gavin. Layer-specific adaptive learning rates for deep networks. In: 2015 IEEE 14th international conference on machine learning and applications. IEEE; 2015, p. 364–8.
[17] Nesterov Yurii. Introductory lectures on convex optimization: a basic course. Springer Science & Business Media; 2013.
[18] Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N, Kaiser Łukasz, Polosukhin Illia. Attention is all you need. Adv Neural Inf Process Syst 2017;30.
[19] Lewkowycz Aitor, Bahri Yasaman, Dyer Ethan, Sohl-Dickstein Jascha, Gur-Ari Guy. The large learning rate phase of deep learning: the catapult mechanism. 2020, arXiv preprint arXiv:2003.02218.
[20] Zhu Libin, Liu Chaoyue, Radhakrishnan Adityanarayanan, Belkin Mikhail. Catapults in SGD: spikes in the training loss and their impact on generalization through feature learning. In: Proceedings of the 41st international conference on machine learning, vol. 235, 2024, p. 62476–509.
[21] Smith Leslie N. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. 2018, arXiv preprint arXiv:1803.09820.
[22] Yang Greg, Hu Edward J, Gouk Henry, Chen Shuxiao, Chen Zixiang. Tensor programs V: Tuning large neural networks via zero-shot hyperparameter transfer. Adv Neural Inf Process Syst (NeurIPS) 2022;35:35368–83.
[23] Dua Dheeru, Graff Casey. UCI machine learning repository. 2017.
[24] LeCun Yann, Bottou Léon, Bengio Yoshua, Haffner Patrick. Gradient-based learning applied to document recognition. Proc IEEE 1998;86(11):2278–324.
[25] Martens J, Grosse R. Optimizing neural networks with Kronecker-factored approximate curvature. In: Proceedings of the 32nd international conference on machine learning. 2015, p. 2408–17.
[26] Hornik Kurt, Stinchcombe Maxwell, White Halbert. Multilayer feedforward networks are universal approximators. Neural Netw 1989;2(5):359–66.