Hyperparameter-Free Auto-Scaled Gradient Normalization via Global Standard Deviation Dynamics

Vincent-Daniel Yun JUYOUNG.YUN@USC.EDU
University of Southern California, USA
Open Neural Network Research Lab, MODULABS, Republic of Korea

Abstract

Gradient dynamics play a central role in determining the stability and generalization of deep neural networks. In this work, we provide an empirical analysis of how variance and standard deviation of gradients evolve during training, showing consistent changes across layers and at the global scale in convolutional networks. Motivated by these observations, we propose a hyperparameter-free gradient normalization method that aligns gradient scaling with their natural evolution. This approach prevents unintended amplification, stabilizes optimization, and preserves convergence guarantees. Experiments on the challenging CIFAR-100 benchmark with ResNet-20, ResNet-56, and VGG-16-BN demonstrate that our method maintains or improves test accuracy even under strong generalization. Beyond practical performance, our study highlights the importance of directly tracking gradient dynamics, aiming to bridge the gap between theoretical expectations and empirical behaviors, and to provide insights for future optimization research.

1. Introduction

Gradient-based optimization is the foundation of modern deep learning. While stochastic gradient descent (SGD) [2] and its variants [2, 6, 14, 16, 24, 25] have achieved remarkable success, the dynamics of gradients during training remain a central factor influencing both convergence speed and generalization [5, 10, 28, 29]. Understanding these dynamics is particularly critical, as challenges such as vanishing and exploding gradients [8, 19, 30] are still active research topics and directly impact the stability and efficiency of training.

In this work, we investigate the gradient dynamics of convolutional neural networks (CNNs) [11, 21]. Through empirical observation, we track the variance and standard deviation of gradients over training and find that layer-wise statistics change significantly as optimization progresses. Although prior theory suggests that gradients and their variance decrease during training [7, 13, 20], we directly track both layer-wise and global standard deviation, bridging the gap between theoretical expectation and empirical behavior.

Motivated by these observations, we propose a hyperparameter-free gradient normalization method. Our approach adjusts gradient magnitudes to follow their natural evolution, ensuring they gradually diminish rather than being amplified, which stabilizes optimization and maintains consistent training dynamics. By explicitly visualizing how gradient statistics evolve, we aim to provide insights into their role in convergence and generalization, a direction that has received limited attention in prior work, especially in terms of tracking layer-wise and global gradients over time. This lays groundwork for more reliable optimization strategies and ultimately provides *insights for future research*. The related works is provided in Appendix A.

© V.-D. Yun.

2. Problem Setup

2.1. Layer-wise Gradients Dynamics

Understanding gradient dynamics is essential for optimizing deep neural networks, as shown in Figure 1, which illustrates layer-wise gradient standard deviation across epochs for ResNet-20 [11], ResNet-56 [11], and VGG-16-BN [21] on CIFAR-10 [15]. We observed that gradient std varies across layers, decreasing in some layers while increasing in others, with the pattern depending on the model's architecture.

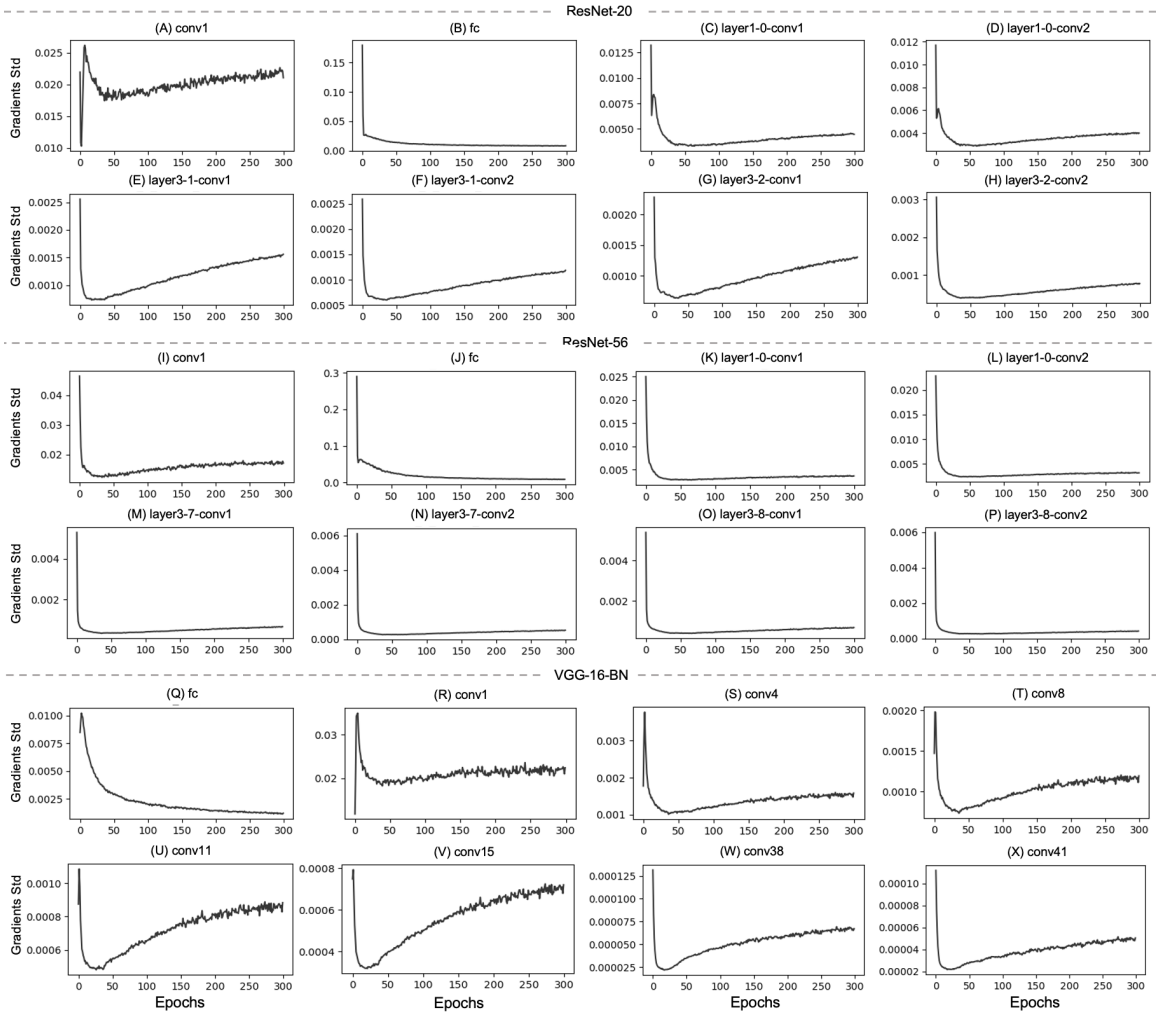

Figure 1: Layer-wise Gradients Standard Deviation for ResNet-20 [11], ResNet-56 [11], and VGG-16-BN [21] Across 300 Epochs on CIFAR-10 [15]. All other settings are same with experimental results section 5

This variability poses a challenge: while some layers' std decreases, others increase or fluctuate irregularly. As a result, when applying layer-wise normalization methods such as ZNorm [25], certain layers experience unwanted amplification while others are suppressed, leading to inconsistent update magnitudes across the network. In particular, applying ZNorm $\frac{\nabla\mathcal{L}(\theta)-\nabla\mathcal{L}(\theta)_{mean}}{\nabla\mathcal{L}(\theta)_{std}}$ amplifies

gradients in layers with low std, especially in architectures such as VGG, where small gradient values cause severe scaling by $1/\text{std}$, resulting in performance degradation or even divergence.

2.2. Global Gradients Dynamics

The global gradient standard deviation provides a comprehensive view of the gradient dynamics across the entire network, as illustrated in Figure 2. This figure presents the evolution of global std over epochs for ResNet-20 [11], ResNet-56 [11], and VGG-16-BN [21] architectures.

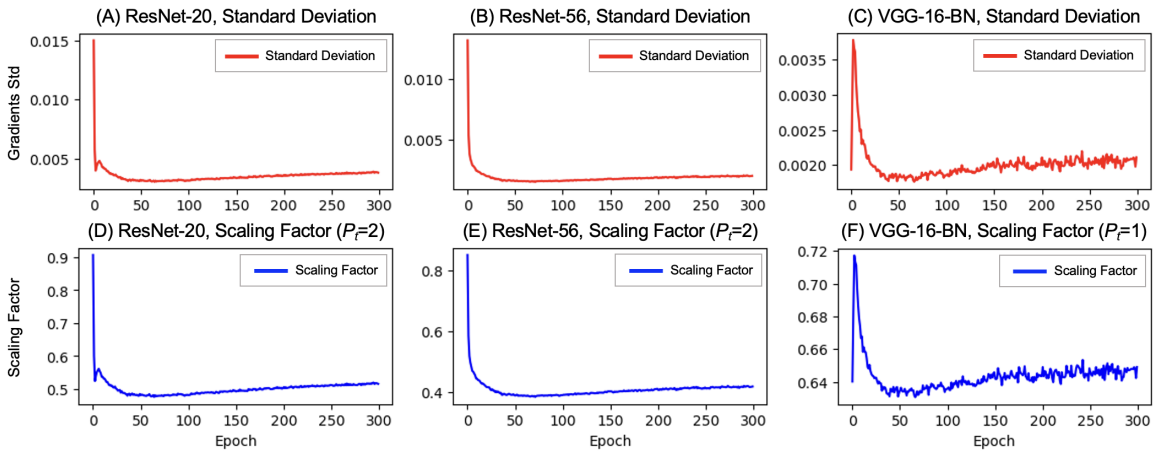

Figure 2: Global gradients standard deviation (top) and corresponding scaling factor (bottom) for ResNet-20 [11], ResNet-56 [11], and VGG-16-BN [21] across 300 epochs on CIFAR-10 [15]. All other settings are same with experimental results section 5 The exponent P_t controls the curvature of the scaling map and is automatically determined in a hyperparameter-free setting, as explained later.

Unlike layer-wise analysis, where the std of individual layers may exhibit increases or irregular variations depending on the model's architecture, the global std demonstrates a consistent decreasing trend as training progresses. As epochs advance, it gradually declines, indicating a stabilization of the overall gradient distribution across all layers. This global perspective highlights a unifying trend that contrasts with the heterogeneous layer-wise behaviors, underscoring the importance of considering `network-wide dynamics` in optimization strategies.

Leveraging the observation that global gradients standard deviation decreases as training progresses, we have devised a method to normalize gradients using a scaling factor that diminishes over time, which will be detailed in the following methodology section.

3. Methodology

3.1. Preliminaries

We consider a deep neural network with L layers and parameters $\{\boldsymbol{\theta}^{(l)}\}_{l=1}^{L}$. For fully connected layers, $\boldsymbol{\theta}^{(l)} \in \mathbb{R}^{D_l \times M_l}$; for convolutional layers, $\boldsymbol{\theta}^{(l)} \in \mathbb{R}^{C_{\text{out}}^{(l)} \times C_{\text{in}}^{(l)} \times k_1^{(l)} \times k_2^{(l)}}$. Let $G_t^{(l)} := \nabla_{\boldsymbol{\theta}^{(l)}} \mathcal{L}(\boldsymbol{\theta}_t)$ denote the gradient tensor of layer l at iteration t. We denote by $\text{vec}(\cdot)$ the vectorization operator and by $\mu_t^{(l)} := \frac{1}{n^{(l)}} \sum_{i=1}^{n^{(l)}} \left(G_t^{(l)}\right)_i$ the (entrywise) mean of $G_t^{(l)}$, where $n^{(l)}$ is the number of elements

in $G_t^{(l)}$. We write $\mathrm{Std}(\cdot)$ for the (unbiased) standard deviation of a vector and use $\epsilon > 0$ as a fixed numerical stabilizer (we use $\epsilon = 10^{-8}$ in all experiments).

3.2. Gradient Autocaled Normalization

Gradient Gradient Autoscaled Normalization performs a two-stage, statistics-driven modification of gradients before the standard gradient descent update [2]: (i) *layer-wise mean removal* (zero-centering) [24], and (ii) a *global autoscale* multiplier shared by all eligible tensors at iteration t. Unlike Z-score normalization [25], Our proposed method does not divide by the (local) standard deviation, thereby avoiding uncontrolled amplification in layers whose gradient variance becomes very small.

Eligible set. At each iteration t, we form the set of tensors whose gradients have at least two dimensions, $\mathcal{S}_t := \{ l \in \{1, \ldots, L\} : \dim(G_t^{(l)}) > 1 \}$, which typically excludes bias vectors and scalar parameters. We aggregate all eligible gradients into a single vector $\mathbf{g}_t := \bigoplus_{l \in \mathcal{S}_t} \mathrm{vec}(G_t^{(l)})$, and define the *global* gradient standard deviation $s_t = \mathrm{Std}(\mathbf{g}_t)$

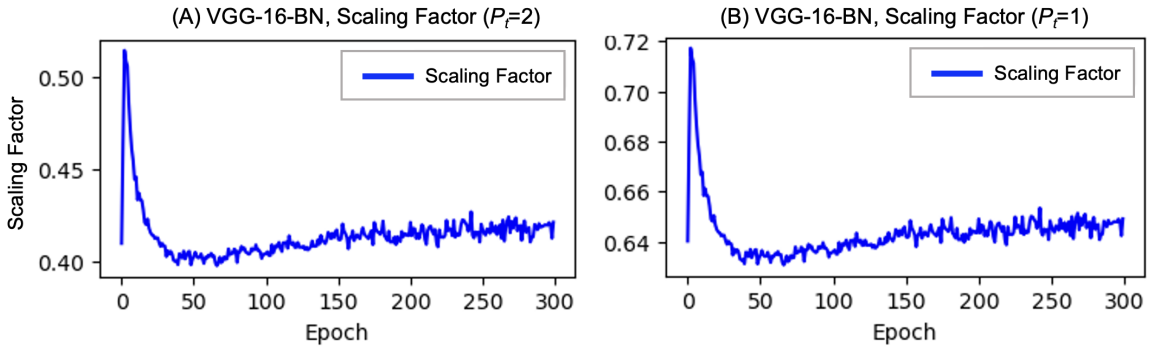

Figure 3: Scaling factor on VGG-16-BN [21] with different settings: (A) $P_t = 2$, (B) $P_t = 1$.

Autoscale via log–std. Our proposed method maps the global scale s_t to a scalar multiplier $a_t \in (0, +\infty)$ through a smooth, hyperparameter-free transform of $|\log s_t|$:

$$a_t = \left(\left(0.5 + \frac{1}{0.5\,(|\log s_t| + \epsilon)} \right) - 0.5 \right) \cdot 2^{p_t}, \quad p_t = \begin{cases} 1, & \text{for all } t \geq 2 \text{ if } a_1 < 0.5, \\ 2, & \text{otherwise.} \end{cases} \tag{1}$$

Algebraically, eq 1 simplifies to $a_t = \left(\frac{4}{|\log s_t| + \epsilon} \right)^{p_t}$ where p_t is an integer *exponent* that controls the curvature of the scaling map. Figure 3 illustrates the behavior of the scaling factor for VGG-16-BN when $p_t = 2$ (left) and $p_t = 1$ (right). When the initial standard deviation s_1 is small, the quadratic mapping ($p_t = 2$) can reduce a_t excessively, leading to overly aggressive downscaling. To prevent this, our proposed method adopts a one-shot safeguard: the exponent p_t is chosen adaptively at the first iteration according to eq 1.

This rule ensures that the scaling factor remains stable and avoids excessive shrinkage in the presence of anomalously small initial gradient variance, while still allowing curvature control in subsequent epochs.

Layer-wise zero-centering and global rescaling. For each eligible layer $l \in \mathcal{S}_t$, we compute the mean-removed gradient $\widetilde{G}_t^{(l)} = G_t^{(l)} - \mu_t^{(l)} \cdot \mathbf{1}$ where $\mathbf{1}$ is the all-ones tensor of the same shape as $G_t^{(l)}$. our proposed method then applies the *same* autoscale a_t to all eligible layers:

$$\widehat{G}_t^{(l)} \;=\; a_t \cdot \widetilde{G}_t^{(l)} \quad \text{for all } l \in \mathcal{S}_t, \qquad \widehat{G}_t^{(l)} \;=\; G_t^{(l)} \quad \text{for } l \notin \mathcal{S}_t. \tag{2}$$

The above equations describe a *zero-mean, globally consistent* gradient transformation that avoids dividing by very small per-layer standard deviations, preventing gradient explosion at the layer level while still adapting to the overall gradient scale of the network.

SGD update. Let $\widehat{\mathbf{g}}_t := \bigoplus_{l=1}^{L} \mathrm{vec}\big(\widehat{G}_t^{(l)}\big)$ denote the transformed gradient concatenated over all parameters. our proposed method with standard stochastic gradient descent (SGD) [2] performs the following update at iteration t: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \cdot \widehat{\mathbf{g}}_t$ where $\eta > 0$ is the learning rate. This formulation highlights that the effect of our proposed method lies entirely in the gradient transformation, while the underlying optimizer step remains identical to standard SGD [2]. This allows easy adaptation to other optimizers such as Adam [14, 16] and RMSProp [22].

4. Theoretical Analysis: Convergence Guarantees

Lemma 1 *Suppose the loss function L is β-smooth and the stochastic gradient is unbiased with bounded variance, i.e., $\mathbb{E}[\nabla L_t(w_t)] = \nabla L(w_t)$ and $\mathbb{E}[\|\nabla L_t(w_t) - \nabla L(w_t)\|^2] \le \frac{\sigma^2}{b}$. If the effective step size satisfies $\eta a_t \le \frac{1}{\beta}$, then the scaled SGD update $w_{t+1} = w_t - \eta a_t \nabla L_t(w_t)$ guarantees:*

$$\mathbb{E}[L(w_{t+1})] \le \mathbb{E}[L(w_t)] - \tfrac{\eta a_t}{2}\mathbb{E}[\|\nabla L(w_t)\|^2] + \tfrac{(\eta a_t)^2 \beta \sigma^2}{2b}. \tag{3}$$

Theorem 2 *Let the loss function L be β-smooth, and assume the stochastic gradient is unbiased with bounded variance, i.e., $\mathbb{E}[\nabla L_t(w_t)] = \nabla L(w_t)$ and $\mathbb{E}[\|\nabla L_t(w_t) - \nabla L(w_t)\|^2] \le \frac{\sigma^2}{b}$. If the effective step size satisfies $\eta a_t \le \frac{1}{\beta}$, then the scaled SGD update $w_{t+1} = w_t - \eta a_t \nabla L_t(w_t)$ guarantees:*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[\|\nabla L(w_t)\|^2] \le \frac{2}{T\eta a_t}\big(L(w_0) - \mathbb{E}[L(w_T)]\big) + \frac{\eta a_t \beta \sigma^2}{b}. \tag{4}$$

The complete proofs follow the standard analysis of SGD [2] and are deferred to Appendix B. Our contribution is the inclusion of the scaling factor a_t in the update rule. Since $a_t \in (0, 1]$, it simply reduces the effective stepsize while leaving the overall convergence guarantees maintained.

5. Experimental Results

Experimental Settings. Experiments are conducted on CIFAR-100 [15], which poses a more challenging benchmark than CIFAR-10. We utilized GitHub repository [18], all models are trained without pretrained weights. We use AdamW [14, 16] (lr=0.001, weight decay 5×10^{-5}) with step decay (0.75 every 30 epochs), batch size 256, and 200 epochs. Backbones include VGG-16-BN [21], ResNet-20 [11], and ResNet-56 [11]. For fair comparison with strong base, label smoothing [17] and CutMix [26] are used. These generalization techniques create a setting where achieving performance gains is inherently difficult, underscoring the robustness of the proposed method.

Network	Method	Top-1 Test Acc.
ResNet-20 [11]	AdamW (Baseline) [16]	0.5932
	AdamW + Gradient Normalization [6]	0.5958
	AdamW + Gradient Centralization [24]	0.6072
	AdamW + Z-Score Normalization [25]	0.5953
	AdamW + Ours	**0.6134**
ResNet-56 [11]	AdamW (Baseline) [16]	0.7001
	AdamW + Gradient Normalization [6]	0.7050
	AdamW + Gradient Centralization [24]	0.6973
	AdamW + Z-Score Normalization [25]	0.6858
	AdamW + Ours	**0.7129**
VGG-16-BN [21]	AdamW (Baseline) [16]	**0.7454**
	AdamW + Gradient Normalization [6]	0.7418
	AdamW + Gradient Centralization [24]	0.7382
	AdamW + Z-Score Normalization [25]	0.7391
	AdamW + Ours	**0.7454**

Table 1: Comparison of performance on CIFAR-100 [15] across ResNet-20 [11], ResNet-56 [11], and VGG-16-BN [21]. Reported results include Top-1 test accuracy

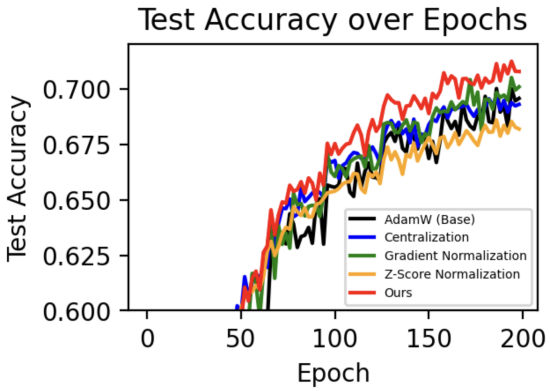

Figure 4: Top-1 Test accuracy comparison on ResNet-56 [11]

Our method consistently improves accuracy on ResNet architectures, while on VGG it achieves performance comparable to the baseline. Notably, alternative normalization techniques often lead to degraded results, underscoring the stability of the proposed approach.

6. Discussion

Our work analyzed gradient dynamics in convolutional networks such as ResNet and VGG, demonstrating the effect of hyperparameter-free normalization. As shown in Figure 4, our method consistently achieves higher test accuracy on ResNet-56 [11] and exhibits smoother convergence compared to baseline AdamW [16] and other normalization approaches. This highlights the benefit of aligning gradient scaling with the natural evolution of gradient statistics. While these results validate the method, they also reveal limitations: the observations are tied to CNNs [11, 21] and experiments were restricted to CIFAR-100 [15]. Future research will extend this analysis to Vision Transformers, whose gradient dynamics differ fundamentally due to attention-based architectures.

7. Conclusion

In this work, we analyzed the gradient dynamics of CNN and proposed a hyperparameter-free normalization method. By aligning gradient scaling with the natural evolution of gradient statistics, our approach mitigates uncontrolled amplification and improves generalization. Experimental results on CIFAR-100 confirm the robustness of our method even under strong generalization settings. We hope this study not only contributes a practical optimization technique but also provides insights into the role of gradient dynamics, paving the way for more robust training strategies.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. 2016. doi: 10.48550/arXiv.1607.06450.

[2] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[3] Leon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

[4] Leon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018. doi: 10.1137/16M1080173.

[5] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019 (12):124018, 2019. doi: 10.1088/1742-5468/ab39d9.

[6] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research (PMLR)*, pages 794–803, Stockholm, Sweden, July 2018. PMLR.

[7] Fartash Faghri, David Duvenaud, David J. Fleet, and Jimmy Ba. A study of gradient variance in deep learning. *arXiv preprint arXiv:2007.04532*, 2020. doi: 10.48550/arXiv.2007.04532.

[8] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[9] Yufei Guo, Yuanpei Chen, Zecheng Hao, Weihang Peng, Zhou Jie, Yuhan Zhang, Xiaode Liu, and Zhe Ma. Take a shortcut back: Mitigating the gradient vanishing for training spiking neural networks. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 24849–24867, 2024.

[10] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 2018. doi: 10.1088/1361-6420/aa9a90.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[12] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. page 448–456, 2015.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[15] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. Technical Report.

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

[17] Rafael Muller, Simon Kornblith, and Geoffrey Hinton. *When does label smoothing help?* 2019.

[18] OmiHub777. ViT-CIFAR: PyTorch implementation for Vision Transformer on CIFAR datasets. https://github.com/omihub777/ViT-CIFAR, 2021. Accessed: 2025-08-15.

[19] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page III–1310–III–1318. JMLR.org, 2013.

[20] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Mkadry. How does batch normalization help optimization? In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, page 2488–2498, 2018.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[22] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5 - RMSProp: Divide the gradient by a running average of its recent magnitude. Coursera: Neural Networks for Machine Learning, 2012. University of Toronto.

[23] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1492–1500, 2017.

[24] Hongwei Yong, Jianqiang Huang, Xiansheng Hua, and Lei Zhang. Gradient centralization: A new optimization technique for deep neural networks. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 635–651, 2020. doi: 10.1007/978-3-030-58568-6_37.

[25] Juyoung Yun. Znorm: Z-score gradient normalization accelerating skip-connected network training without architectural modification. In *AI for Research and Scalable, Efficient Systems*, pages 240–254, Singapore, 2025. Springer Nature Singapore.

[26] Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Seong Joon Oh, Youngjoon Yoo, and Junsuk Choe. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6022–6031, 2019.

[27] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016.

[28] Yang Zhao, Hao Zhang, and Xiuyuan Hu. When will gradient regularization be harmful? In *Forty-first International Conference on Machine Learning*.

[29] Yang Zhao, Hao Zhang, and Xiuyuan Hu. Penalizing gradient norm for efficiently improving generalization in deep learning. In *International Conference on Machine Learning*, pages 26982–26992. PMLR, 2022.

[30] Nicolas Zucchet and Antonio Orvieto. Recurrent neural networks: vanishing and exploding gradients are not the end of the story. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Appendix A. Related Works

Research on gradient-based optimization has mainly focused on how training dynamics influence both convergence and generalization. Early studies showed the importance of stochastic gradient descent [2] and its variants [14, 16, 22] in large-scale learning problems [3, 4], providing both theory and strong results in practice. Later work studied the problems of vanishing and exploding gradients [8, 9, 19, 30], which are still major challenges for stable training, especially in very deep or recurrent networks. These difficulties motivated stabilization techniques such as batch normalization [13, 20], layer normalization [1], and residual connections [11, 12, 23, 27].

A complementary line of research proposed methods that normalize gradients directly. Gradient normalization methods such as GradNorm [6] were first introduced for multitask learning, to keep gradients from different tasks balanced. A common rule is $g_i \leftarrow \frac{g_i}{\|g_i\|} \cdot \alpha$, where g_i is the gradient of task i and α is a scaling factor. Gradient clipping [19] is another widely used method, defined as $g \leftarrow \frac{g}{\max(1, \|g\|/c)}$, where c is a threshold, which prevents exploding gradients by shrinking overly large updates. Gradient centralization [24] improved generalization by simply subtracting the mean, $g \leftarrow g - \mu_g$. More recent work, such as Z-Score Gradient Normalization [25], tried to standardize gradients across layers, $g \leftarrow \frac{g-\mu_g}{\sigma_g+\epsilon}$, but this can become unstable when some layers have very small variance, leading to amplification and worse performance.

While prior methods mainly focus on manipulating gradients, few works have explicitly tracked how their standard deviation evolves across layers or at the global scale. This study directly monitors these dynamics during training, linking theoretical expectations of gradient decay with empirical behavior. Building on this, a hyperparameter-free approach using global statistics is introduced, which avoids unstable amplification and preserves convergence guarantees, while highlighting the importance of studying gradient dynamics themselves as a basis for future optimization research.

Appendix B. Theoretical Analysis: Convergence Guarantees on SGD

Assumption 1 (β-smoothness) *The loss function $L : \mathbb{R}^d \to \mathbb{R}$ is β-smooth, i.e., its gradient is β-Lipschitz continuous $\|\nabla L(u) - \nabla L(v)\| \leq \beta\|u - v\|, \forall u, v \in \mathbb{R}^d$. Equivalently, for any $u, v \in \mathbb{R}^d$, the following descent lemma holds:*

$$L(u) \leq L(v) + \langle \nabla L(v), u - v\rangle + \frac{\beta}{2}\|u - v\|^2. \tag{5}$$

Assumption 2 (Unbiased stochastic gradient) *At each iteration t, the stochastic gradient $\nabla L_t(w_t)$ is an unbiased estimator of the true gradient:*

$$\mathbb{E}[\nabla L_t(w_t)] = \nabla L(w_t). \tag{6}$$

Lemma 1 *Suppose the loss function L is β-smooth and the stochastic gradient is unbiased with bounded variance, i.e., $\mathbb{E}[\nabla L_t(w_t)] = \nabla L(w_t)$ and $\mathbb{E}[\|\nabla L_t(w_t) - \nabla L(w_t)\|^2] \leq \frac{\sigma^2}{b}$. If the effective step size satisfies $\eta a_t \leq \frac{1}{\beta}$, then the scaled SGD update $w_{t+1} = w_t - \eta a_t \nabla L_t(w_t)$ guarantees:*

$$\mathbb{E}[L(w_{t+1})] \leq \mathbb{E}[L(w_t)] - \frac{\eta a_t}{2}\mathbb{E}[\|\nabla L(w_t)\|^2] + \frac{(\eta a_t)^2 \beta \sigma^2}{2b}. \tag{7}$$

Proof By the β-smoothness of L, it holds that

$$L(w_{t+1}) \leq L(w_t) + \langle \nabla L(w_t), w_{t+1} - w_t\rangle + \frac{\beta}{2}\|w_{t+1} - w_t\|^2. \tag{8}$$

Plugging in the update rule $w_{t+1} = w_t - \eta a_t \nabla L_t(w_t)$, we obtain

$$L(w_{t+1}) \leq L(w_t) - \eta a_t \langle \nabla L(w_t), \nabla L_t(w_t)\rangle + \frac{(\eta a_t)^2 \beta}{2}\|\nabla L_t(w_t)\|^2. \tag{9}$$

Taking expectations and using the condition $\mathbb{E}[\nabla L_t(w_t)] = \nabla L(w_t)$, it follows that

$$\begin{aligned}
\mathbb{E}[L(w_{t+1})] &\leq \mathbb{E}[L(w_t)] - \eta a_t \mathbb{E}[\langle \nabla L(w_t), \nabla L_t(w_t)\rangle] + \frac{(\eta a_t)^2 \beta}{2}\mathbb{E}[\|\nabla L_t(w_t)\|^2] && (10)\\
&= \mathbb{E}[L(w_t)] - \eta a_t \mathbb{E}[\|\nabla L(w_t)\|^2] + \frac{(\eta a_t)^2 \beta}{2}\mathbb{E}[\|\nabla L_t(w_t)\|^2]. && (11)
\end{aligned}$$

For the variance term, observe that

$$\begin{aligned}
\mathbb{E}[\|\nabla L_t(w_t)\|^2] &= \mathbb{E}[\|\nabla L_t(w_t) - \nabla L(w_t) + \nabla L(w_t)\|^2] && (12)\\
&= \mathbb{E}[\|\nabla L_t(w_t) - \nabla L(w_t)\|^2 + \|\nabla L(w_t)\|^2 && (13)\\
&\quad + 2\langle \nabla L_t(w_t) - \nabla L(w_t), \nabla L(w_t)\rangle] && (14)\\
&= \mathbb{E}[\|\nabla L_t(w_t) - \nabla L(w_t)\|^2] + \mathbb{E}[\|\nabla L(w_t)\|^2] && (15)\\
&= \mathbb{E}[\|\nabla L_t(w_t) - \nabla L(w_t)\|^2 + \|\nabla L(w_t)\|^2 && (16)\\
&\quad + 2\mathbb{E}[\langle \nabla L_t(w_t) - \nabla L(w_t), \nabla L(w_t)\rangle] && (17)\\
&= \mathbb{E}[\|\nabla L_t(w_t) - \nabla L(w_t)\|^2] + \mathbb{E}[\|\nabla L(w_t)\|^2] && (18)\\
&= \mathbb{E}[\|\nabla L_t(w_t) - \nabla L(w_t)\|^2 + \|\nabla L(w_t)\|^2 && (19)\\
&\quad + 2\mathbb{E}[\langle \nabla L_t(w_t), \nabla L(w_t)\rangle - \langle \nabla L(w_t), \nabla L(w_t)\rangle] && (20)\\
&= \mathbb{E}[\|\nabla L_t(w_t) - \nabla L(w_t)\|^2] + \mathbb{E}[\|\nabla L(w_t)\|^2]. && (21)
\end{aligned}$$

Here the cross term vanishes since $\mathbb{E}[\nabla L_t(w_t) - \nabla L(w_t)] = 0$. By the bounded variance condition $\mathbb{E}[\|\nabla L_t(w_t) - \nabla L(w_t)\|^2] \leq \frac{\sigma^2}{b}$, we have

$$\mathbb{E}[\|\nabla L_t(w_t)\|^2] \leq \mathbb{E}[\|\nabla L(w_t)\|^2] + \tfrac{\sigma^2}{b}. \tag{22}$$

Substituting (21) into (22), we arrive at

$$\begin{aligned}
\mathbb{E}[L(w_{t+1})] &\leq \mathbb{E}[L(w_t)] - \eta a_t \mathbb{E}[\|\nabla L(w_t)\|^2] + \tfrac{(\eta a_t)^2\beta}{2}\left(\mathbb{E}[\|\nabla L(w_t)\|^2] + \tfrac{\sigma^2}{b}\right) \\
&= \mathbb{E}[L(w_t)] - \eta a_t \mathbb{E}[\|\nabla L(w_t)\|^2] + \tfrac{(\eta a_t)^2\beta}{2}\mathbb{E}[\|\nabla L(w_t)\|^2] + \tfrac{(\eta a_t)^2\beta}{2}\cdot\tfrac{\sigma^2}{b} \\
&= \mathbb{E}[L(w_t)] - \eta a_t \mathbb{E}[\|\nabla L(w_t)\|^2] + \tfrac{(\eta a_t)^2\beta}{2}\mathbb{E}[\|\nabla L(w_t)\|^2] + \tfrac{(\eta a_t)^2\beta\sigma^2}{2b} \\
&= \mathbb{E}[L(w_t)] + \left(-\eta a_t + \tfrac{(\eta a_t)^2\beta}{2}\right)\mathbb{E}[\|\nabla L(w_t)\|^2] + \tfrac{(\eta a_t)^2\beta\sigma^2}{2b} \\
&= \mathbb{E}[L(w_t)] + \left(\tfrac{(\eta a_t)^2\beta}{2} - \eta a_t\right)\mathbb{E}[\|\nabla L(w_t)\|^2] + \tfrac{(\eta a_t)^2\beta\sigma^2}{2b} \\
&= \mathbb{E}[L(w_t)] - \left(\eta a_t - \tfrac{(\eta a_t)^2\beta}{2}\right)\mathbb{E}[\|\nabla L(w_t)\|^2] + \tfrac{(\eta a_t)^2\beta\sigma^2}{2b}.
\end{aligned} \tag{23}$$

Finally, using $\eta a_t \leq \frac{1}{\beta}$, we have

$$\eta a_t - \tfrac{(\eta a_t)^2\beta}{2} = \eta a_t\left(1 - \tfrac{\eta a_t\beta}{2}\right) \geq \tfrac{\eta a_t}{2}, \tag{24}$$

since $\eta a_t\beta \leq 1$ implies $1 - \frac{\eta a_t\beta}{2} \geq \frac{1}{2}$. Therefore,

$$\mathbb{E}[L(w_{t+1})] \leq \mathbb{E}[L(w_t)] - \tfrac{\eta a_t}{2}\mathbb{E}[\|\nabla L(w_t)\|^2] + \tfrac{(\eta a_t)^2\beta\sigma^2}{2b}. \tag{25}$$

■

Theorem 2 *Let the loss function L be β-smooth, and assume the stochastic gradient is unbiased with bounded variance, i.e., $\mathbb{E}[\nabla L_t(w_t)] = \nabla L(w_t)$ and $\mathbb{E}[\|\nabla L_t(w_t) - \nabla L(w_t)\|^2] \leq \frac{\sigma^2}{b}$. If the effective step size satisfies $\eta a_t \leq \frac{1}{\beta}$, then the scaled SGD update $w_{t+1} = w_t - \eta a_t \nabla L_t(w_t)$ guarantees:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla L(w_t)\|^2] \leq \frac{2}{T\eta a_t}\big(L(w_0) - \mathbb{E}[L(w_T)]\big) + \frac{\eta a_t\beta\sigma^2}{b}. \tag{26}$$

Proof From the descent lemma applied to scaled SGD, we have

$$\mathbb{E}[L(w_{t+1})] \leq \mathbb{E}[L(w_t)] - \tfrac{\eta a_t}{2}\mathbb{E}[\|\nabla L(w_t)\|^2] + \tfrac{(\eta a_t)^2\beta\sigma^2}{2b}. \tag{27}$$

Averaging inequality (27) over T iterations yields

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[L(w_{t+1})] \leq \frac{1}{T}\sum_{t=0}^{T-1}\left(\mathbb{E}[L(w_t)] - \tfrac{\eta a_t}{2}\mathbb{E}[\|\nabla L(w_t)\|^2] + \tfrac{(\eta a_t)^2\beta\sigma^2}{2b}\right). \tag{28}$$

$$\leq \frac{1}{T}\sum_{t=0}^{T-1}\left(\mathbb{E}[L(w_t)] + \tfrac{(\eta a_t)^2\beta\sigma^2}{2b}\right) - \tfrac{\eta a_t}{2T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla L(w_t)\|^2]. \tag{29}$$

$$\tag{30}$$

Rearranging terms and noting that $\frac{1}{T}\sum_{t=0}^{T-1}\big(\mathbb{E}[L(w_t)] - \mathbb{E}[L(w_{t+1})]\big) = \frac{1}{T}\big(L(w_0) - \mathbb{E}[L(w_T)]\big)$, we obtain

$$\frac{\eta a_t}{2T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla L(w_t)\|^2] \leq \frac{1}{T}\sum_{t=0}^{T-1}\big(\mathbb{E}[L(w_t)] - \mathbb{E}[L(w_{t+1})]\big) + \tfrac{(\eta a_t)^2\beta\sigma^2}{2b}. \tag{31}$$

$$= \tfrac{1}{T}\big(L(w_0) - \mathbb{E}[L(w_T)]\big) + \tfrac{(\eta a_t)^2\beta\sigma^2}{2b}. \tag{32}$$

Dividing both sides by $\frac{\eta a_t}{2}$ gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla L(w_t)\|^2] \leq \frac{2}{T\eta a_t}\big(L(w_0) - \mathbb{E}[L(w_T)]\big) + \frac{\eta a_t\beta\sigma^2}{b}. \tag{33}$$

■