Journal of Machine Learning Research ** (2025) 1-33 Submitted MM/DD; Revised MM/DD; Published MM/DD

Optimization-Induced Dynamics of Lipschitz Continuity in Neural Networks

Róisín Luo* ROISINCRTAI@GMAIL.COM
University of Galway, Ireland
Irish National Centre for Research Training in AI (CRT-AI)

James McDermott
University of Galway, Ireland
Irish National Centre for Research Training in AI (CRT-AI)

Christian Gagné
Université Laval, Canada
Canada-CIFAR AI Chair
Mila - Québec AI Institute

Qiang Sun
University of Toronto, Canada
MBZUAI, UAE

Colm O'Riordan
University of Galway, Ireland
Irish National Centre for Research Training in AI (CRT-AI)

Editor: My editor

Abstract

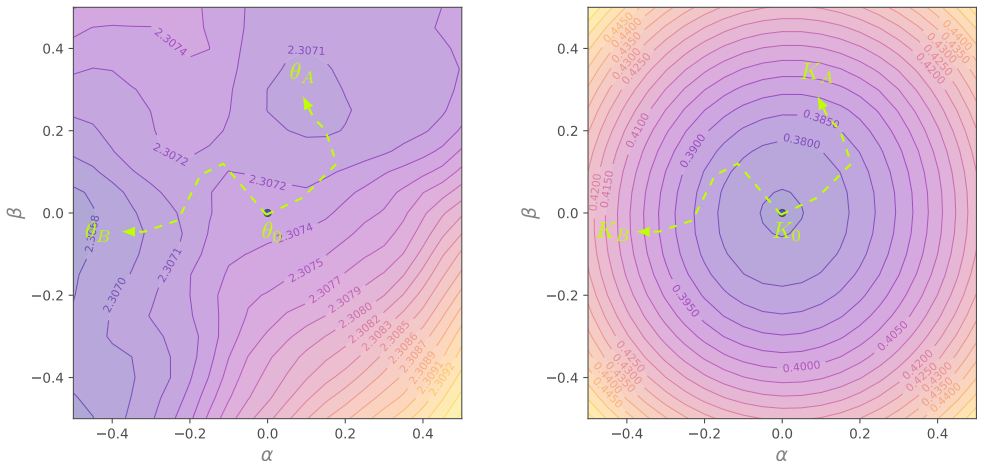
Lipschitz continuity characterizes the worst-case sensitivity of neural networks to small input perturbations; yet its dynamics (*i.e.* temporal evolution) during training remains under-explored. We present a rigorous mathematical framework to model the temporal evolution of Lipschitz continuity during training with stochastic gradient descent (SGD). This framework leverages a system of stochastic differential equations (SDEs) to capture both deterministic force (*i.e.* gradient expectations) and stochastic force (*i.e.* gradient noise). Our theoretical analysis identifies three principal factors driving the evolution: (i) the projection of gradient flows, induced by the optimization dynamics, onto the operator-norm Jacobian of parameter matrices; (ii) the projection of gradient noise, arising from the randomness in mini-batch sampling, onto the operator-norm Jacobian; and (iii) the projection of the gradient noise onto the operator-norm Hessian of parameter matrices. Furthermore, our theoretical framework sheds light on such as how noisy supervision, parameter initialization, batch size, and mini-batch sampling trajectories, among other factors, shape the evolution of the Lipschitz continuity of neural networks. Our experimental results demonstrate strong agreement between the theoretical implications and the observed behaviors.

Keywords: Optimization-Induced Dynamics, Lipschitz Continuity, Optimization Dynamics, Theory for Robustness, Robustness, Trustworthy Deep Learning, Deep Learning

*. Corresponding author. Research code: `https://anonymous.4open.science/r/lipschitz_dynamics_reproducibility-4721`

©2025 Róisín Luo et al..

License: CC-BY 4.0, see `https://creativecommons.org/licenses/by/4.0/`. Attribution requirements are provided at `http://jmlr.org/papers/v**/00-0000.html`.



(a) Optimization trajectories in loss landscape

(b) Induced trajectories in Lipschitz landscape

Figure 1: **Optimization-induced dynamics**. During the training, the network parameters, starting from $\boldsymbol{\theta}_0$, moves towards a solution $\boldsymbol{\theta}_A$ or $\boldsymbol{\theta}_B$ as shown in the loss landscape (a), driven by optimization process. Accordingly, this dynamics, driven by the optimization, induces the evolution of the network Lipschitz continuity, starting from K_0 to K_A or K_B, as shown in the Lipschitz landscape (b). The trajectories in the loss landscape (a) and the Lipschitz landscape (b) are visualized on the same parameter space $\alpha O \beta$. The α and β are two randomly-chosen orthogonal directions in the parameter space.

1 Introduction

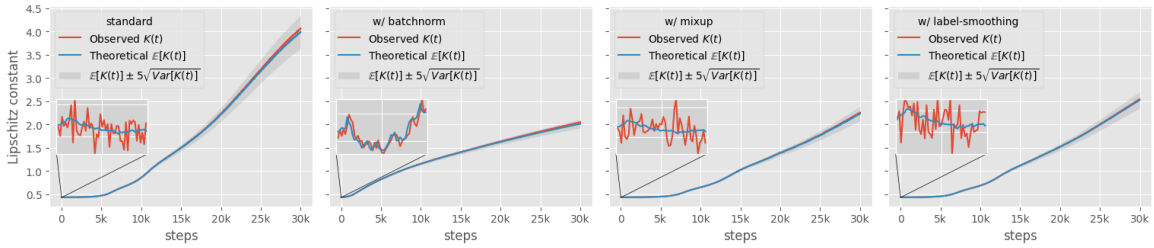
Recent advancements in deep learning have led to models that excel across a broad range of domains, from vision to language. Their vulnerability to input perturbations remains a significant challenge for establishing trustworthy learning systems (Szegedy et al., 2014; Goodfellow et al., 2014; Mądry et al., 2017). Lipschitz continuity (Definition 1) measures the worst-case sensitivity of the output of a network to small input perturbations. Furthermore, several fundamental properties, including robustness to perturbation (Luo et al., 2024) and generalization capability, are closely linked to network Lipschitz continuity (Shalev-Shwartz and Ben-David, 2014; Bartlett et al., 2017; Yin et al., 2019; Zhang et al., 2021). Networks with lower Lipschitz constants tend to be more resilient to input perturbations (*e.g.* adversarial perturbations) and exhibit improved generalization capabilities (Zhang et al., 2022; Fazlyab et al., 2023; Khromov and Singh, 2024).

Definition 1 (Globally K-Lipschitz Continuous (Tao, 2006; Yosida, 2012)) *Let $f : X \mapsto Y$ be a function, where $X \subseteq \mathbb{R}^d$ and $Y \subseteq \mathbb{R}^c$. The function f is said to be* globally K-Lipschitz continuous *if there exists a constant $K > 0$ such that:*

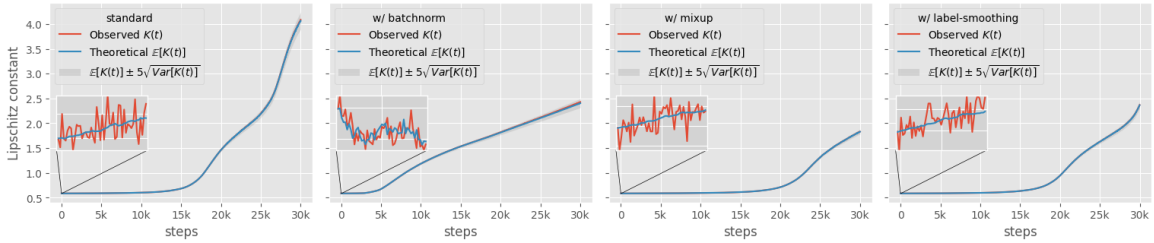
$$\|f(u) - f(v)\|_2 \leq K\|u - v\|_2, \quad \forall\, u, v \in X \tag{1}$$

upper-bounds the function f.

Gradient-based optimization methods, such as stochastic gradient descent (SGD) and its variants (Robbins and Monro, 1951; Hinton et al., 2012; Kingma and Ba, 2017), are funda-



(a) Numerical validation on CIFAR-10



(b) Numerical validation on CIFAR-100

Figure 2: **Numerical validation of our mathematical framework**. The theoretical Lipschitz constants computed using our framework closely agree with empirical observations. To validate our framework, we train a five-layer ConvNet on CIFAR-10 and CIFAR-100 across multiple configurations for $30,000$ steps (200 epochs). We collect the instance-wise gradients over time for all layers. Using Theorem 15, Theorem 16, Theorem 17 and Theorem 18, we are able to theoretically compute the predicted Lipschitz continuity. The inset plots zoom in on the first 50 steps, and demonstrate that **the trends of Lipschitz constants do not necessarily grow monotonically**. Results with more regularization configurations on CIFAR-10 are provided in Appendix C.

mental to training deep learning models by iteratively minimizing the loss function through parameter updates with gradients. The optimization dynamics in a neural network induce the corresponding dynamics of its Lipschitz continuity, which we refer to as **optimization-induced dynamics**. As illustrated in Figure 1, every optimization trajectory in the loss landscape induces a corresponding trajectory in the Lipschitz landscape. Although the dynamics (*i.e.* temporal evolution) of SGD has been extensively explored in the literature, including topics such as: (i) continuous-time/SDE modeling (Li et al., 2019; Malladi et al., 2022; Welling and Teh, 2011; Zhu et al., 2018); (ii) edge-of-stability and implicit bias (Li et al., 2021; Damian et al., 2022; Xing et al., 2018; Jastrzębski et al., 2017); (iii) convergence analysis (Li and Yuan, 2017); (iv) sharp versus flat minima and generalization (Keskar et al., 2016; Chaudhari et al., 2019; Zhang et al., 2021), a comprehensive understanding of how the Lipschitz constant evolves over time during training remains lacking.

This gap motivates us to establish a rigorous theoretical framework for modeling the dynamics of Lipschitz continuity, induced by optimization dynamics. Equipped with the toolkit from stochastic differential equations (SDEs) (Applebaum, 2009; Karatzas and Shreve, 2012; Oksendal, 2013) and operator perturbation theory (Kato, 2012, 2013; Luo et al., 2025), we can rigorously analyze the evolution of the matrix operator norm (*i.e.* matrix Lipschitz constant), induced by the optimization process, within an SDE framework. Our framework

can capture both the deterministic and stochastic forces driving the temporal evolution of Lipschitz constants. The effectiveness of our mathematical framework is empirically verified in Figure 2, where the theoretical implications derived using our framework (Theorem 17 and Theorem 18) are closely aligned with the observations. Furthermore, our theoretical framework sheds light on questions such as how batch size, parameter initialization, mini-batch sampling trajectories, label noise, etc., shape the Lipschitz continuity evolution during optimization.

1.1 Contributions

We highlight the key contributions below:

1. **Theoretical Framework** (Section 2-6). We present a rigorous mathematical framework that models the dynamics of Lipschitz continuity, leveraging an SDE system driven by Wiener processes. This SDE-based framework captures both deterministic and stochastic forces, providing a comprehensive understanding of the dynamics of Lipschitz continuity. To ensure practical applicability, we also develop a low-rank approximation method for modeling gradient noise in Section 4, enabling efficient computation of these dynamics.

2. **Principal Driving Forces** (Section 6.2-6.4). Our theoretical analysis (Theorem 17) identifies three principal forces governing the dynamics: (i) the projection of gradient flows, induced by the optimization dynamics, onto the operator-norm Jacobian of parameter matrices; (ii) the projection of gradient noise, arising from the randomness in mini-batch sampling, onto the operator-norm Jacobian; and (iii) the projection of the gradient noise onto the operator-norm Hessian of parameter matrices. In the evolution of Lipschitz continuity, forces (i) and (iii) act as deterministic forces, while force (ii) modulates the stochasticity of the dynamics.

3. **Framework Validation and Theoretical Implications** (Section 7-8). Firstly, we validate our theoretical framework under multiple configurations, including *batch normalization*, *dropout*, *weight decay*, *mixup*, *auto-augment*, *label smoothing* and *adversarial training*. Furthermore, we test the theoretical implications derived from our framework, as detailed in Section 8, including parameter initialization, noisy gradient regularization, uniform label corruption, batch size, and mini-batch sampling trajectories shape the evolution of Lipschitz continuity. The results show a strong agreement between the theoretical implications and observed behaviors.

2 Preliminaries

We define the notation used in our theoretical analysis. Let $f : \mathbb{R}^d \to \mathbb{R}^c$ be a function. For a time-dependent function g, we use g_k to denote its value at discrete time step k, and $g(t)$ for its value at continuous time t. For a random variable ξ, *e.g.*, representing data sampling, gradient noise, Wiener process, and filtration (Oksendal, 2013), we consistently use subscripts, such as ξ_k or ξ_t, for brevity. A function written as $g^{(\ell)}$ indicates that g is the ℓ-th layer of a neural network. We use I_n to denote a $n \times n$ identity matrix; $\mathbb{1}_n$ to denote an n-dimensional all-ones vector.

Mini-Batch Sampling. Let $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^N \subseteq \mathcal{X} \times \mathcal{Y}$ denote a dataset consisting of N samples, where (x_i, y_i) represents the i-th data point and its corresponding target. Let $\xi_t := \{(x_{t_j}, y_{t_j})\}_{j=1}^M \subset \mathcal{X} \times \mathcal{Y}$ denote a mini-batch sampled from $\mathcal{D}$ at time point t, where M ($M \ll N$) is the size of the mini-batch. The sequence $\{\xi_i\}_{i=0}^t$ up to time t is referred to as a *sampling trajectory* from D. The σ-algebra $\mathcal{F}_t = \sigma\{\xi_0, \xi_1, \ldots, \xi_t\}$, defined on mini-batch sampling, is referred to as the *filtration* generated by the sampling trajectory $\{\xi_i\}_{i=0}^t$ (Oksendal, 2013). In stochastic analysis, $\mathcal{F}_t$ represents that the information accumulates up to time t (*i.e.* history). Any history up to a time point t_k contains the history up to a time point t_l for all $t_k \geq t_l$, *i.e.* $\mathcal{F}_{t_l} \subseteq \mathcal{F}_{t_k}$.

Feed-Forward Network. Let $\boldsymbol{\theta}^{(\ell)} \in \mathbb{R}^{m_\ell \times n_\ell}$ denote the parameter matrix of the ℓ-th layer of an L-layer feed-forward network. Let $\boldsymbol{\theta} := \{\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots, \boldsymbol{\theta}^{(L)}\} \in \Theta$ denote the collection of all L parameter matrices. Let $f_{\boldsymbol{\theta}} : \mathbb{R}^d \mapsto \mathbb{R}^c$ represent a feed-forward network parameterized by $\boldsymbol{\theta}$. Let $\mathrm{vec}(\boldsymbol{\theta}^{(\ell)}) \in \mathbb{R}^{m_\ell n_\ell}$ denote the vectorized $\boldsymbol{\theta}^{(\ell)}$ with *column-major convention* (Horn and Johnson, 2012).

Remark 2 *The explicit use of the* $\mathrm{vec}(\cdot)$ *operator is essential for rigorous spectral analysis of parameter matrices beyond the usual context of parameter updates in SGD optimization.*

Instance and Population Loss. Let $\ell_f(\boldsymbol{\theta}; x, y) : \Theta \times \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$ denote the instance loss for $f_{\boldsymbol{\theta}}$. Let $\mathcal{L}_f(\boldsymbol{\theta}; \xi)$ denote the population loss over a mini-batch ξ:

$$\mathcal{L}_f(\boldsymbol{\theta}; \xi) := \frac{1}{M} \sum_{x_i, y_i \in \xi} \ell_f(\boldsymbol{\theta}; x_i, y_i).$$

Let $\mathcal{L}_f(\boldsymbol{\theta})$ denote the population loss over the dataset $\mathcal{D}$:

$$\mathcal{L}_f(\boldsymbol{\theta}) := \mathbb{E}_{(x_i, y_i) \sim \mathcal{D}} \left[\ell_f(\boldsymbol{\theta}; x_i, y_i) \right].$$

Unbiased Gradient Estimator. For brevity, we use:

$$\nabla^{(\ell)} \mathcal{L}_f(\boldsymbol{\theta}) := \nabla_{\boldsymbol{\theta}^{(\ell)}} \mathcal{L}_f(\boldsymbol{\theta}), \quad \text{and} \quad \nabla \mathcal{L}_f(\boldsymbol{\theta}) := \nabla_{\boldsymbol{\theta}} \mathcal{L}_f(\boldsymbol{\theta}),$$

to denote the gradient with respect to the ℓ-th layer parameter matrix, and the gradient with respect to the collective parameter matrices, respectively. Note that $\nabla \mathcal{L}_f(\boldsymbol{\theta}; \xi)$ is an unbiased gradient estimator for $\nabla \mathcal{L}_f(\boldsymbol{\theta})$:

$$\mathbb{E}\left[\nabla L_t(\boldsymbol{\theta}; \xi)\right] = \mathbb{E}\left[\nabla \mathcal{L}_f(\boldsymbol{\theta})\right] = \mathbb{E}\left[\nabla \ell_f(\boldsymbol{\theta}; x, y)\right].$$

3 Vectorized SDE for Continuous-Time SGD

SGD and its variants (*e.g.* Adam) (Robbins and Monro, 1951; Hinton et al., 2012; Kingma and Ba, 2017) serve as cornerstone optimization algorithms widely used for training deep neural networks. Formally, at a time point k, the SGD-based update regarding the parameter $\boldsymbol{\theta}_k^{(\ell)}$ with a mini-batch ξ_k can be formulated by:

$$\boldsymbol{\theta}_{k+1}^{(\ell)} = \boldsymbol{\theta}_k^{(\ell)} - \eta \nabla^{(\ell)} \mathcal{L}_f(\boldsymbol{\theta}_k; \xi_k), \tag{2}$$

where η is the learning rate. Some studies (Mandt et al., 2015; Su et al., 2016; Stephan et al., 2017) view SGD as a deterministic process with a deterministic ordinary differential equation (ODE) as:

$$\frac{\mathrm{d}\boldsymbol{\theta}^{(\ell)}(t)}{\mathrm{d}t} = -\nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}(t)),$$

where $t \approx k\eta$ and $\mathrm{d}t \approx \eta \to 0$.

However, the inherent stochasticity in the population gradients $\nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}_k; \xi_k)$, arising from the randomness in mini-batch sampling ξ_k, is not accounted for by ODE methods. The *layer-wise batch gradient noise* (*i.e.* batch gradient fluctuations) (Welling and Teh, 2011; Keskar et al., 2016; Chaudhari et al., 2019; Zhang et al., 2021), defined as a positive semi-definite (PSD) matrix:

$$\boldsymbol{\Sigma}_k^{(\ell)} := \mathrm{Var}\left[\mathrm{vec}\left(\nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}_k; \xi_k)\right)\right] = \frac{1}{M}\mathrm{Var}\left[\mathrm{vec}\left(\nabla^{(\ell)}\ell_f(\boldsymbol{\theta}_k; x, y)\right)\right] \in \mathbb{R}^{m_\ell n_\ell \times m_\ell n_\ell}, \quad (3)$$

induced by mini-batch sampling, can influence the optimization trajectory. **It remains under-explored in the literature how such stochasticity affects the evolution of Lipschitz continuity of neural networks over time**.

Assumption 3 (Collective Gradient Noise Structure Assumption) *For tractability, we assume that gradient-noise covariances between different layers are negligible. Equivalently, Σ_t has block-diagonal structure:*

$$\boldsymbol{\Sigma}_t = \mathrm{diag}(\boldsymbol{\Sigma}_t^{(1)}, \boldsymbol{\Sigma}_t^{(2)}, \cdots, \boldsymbol{\Sigma}_t^{(L)}),$$

so that we model batch gradient noise on a per-layer basis only. This layer-wise approximation is common in large-scale SDE analyses, e.g. *(Grosse and Martens, 2016; Malladi et al., 2022; Simsekli et al., 2019).*

To address the limitations of ODE methods, SDE methods (Li and Yuan, 2017; Jastrzębski et al., 2017; Zhu et al., 2018; Chaudhari et al., 2019) extend ODE methods by accounting for gradient noise. SDE-based methods capture the stochasticity inherent in mini-batch updates and its effects on the optimization trajectory. The multivariate Central Limit Theorem (CLT) states that the population gradient estimator:

$$\nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}_k; \xi_k) = \frac{1}{M}\sum_{(x_i, y_i)\in\xi_k} \nabla^{(\ell)}\ell_f(\boldsymbol{\theta}_k; x_i, y_i),$$

distributionally converges to a normal distribution in $\mathbb{R}^{m_\ell n_\ell}$:

$$\mathrm{vec}\left(\nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}_k; \xi_k)\right) \xrightarrow{\mathrm{d}} \mathcal{N}\left(\mathrm{vec}(\nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}_k)), \boldsymbol{\Sigma}_k^{(\ell)}\right),$$

as $M \to \infty$, where the samples (x_i, y_i) are *i.i.d.* Therefore, SGD can be modeled as an Itô process, which provides a more accurate representation of the dynamics by considering the continuous-time SDEs (Mandt et al., 2015; Stephan et al., 2017).

Definition 4 (Vectorized SDE for Continuous-Time SGD) *Under Assumption 3, and the assumption that $\nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}_t)$ and $\boldsymbol{\Sigma}_t^{(\ell)}$ satisfy global Lipschitz and linear-growth conditions (Oksendal, 2013; Karatzas and Shreve, 2012), the SGD update for the parameter $\boldsymbol{\theta}_t^{(\ell)}$ with a mini-batch ξ_t at time t can be formulated as a matrix-valued Itô's SDE by:*

$$\mathrm{dvec}\left(\boldsymbol{\theta}^{(\ell)}(t)\right) = -\mathrm{vec}\left[\nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}(t))\right]\mathrm{d}t + \sqrt{\eta}\left[\boldsymbol{\Sigma}_t^{(\ell)}\right]^{\frac{1}{2}}\mathrm{d}\boldsymbol{B}_t^{(\ell)}, \tag{4}$$

where the batch gradient noise $\left[\boldsymbol{\Sigma}_t^{(\ell)}\right]^{1/2} \in \mathbb{R}^{m_\ell n_\ell \times m_\ell n_\ell}$ satisfies:

$$\left(\boldsymbol{\Sigma}_t^{(\ell)}\right)^{\frac{1}{2}}\left[\left(\boldsymbol{\Sigma}_t^{(\ell)}\right)^{\frac{1}{2}}\right]^\top = \boldsymbol{\Sigma}_t^{(\ell)}, \quad \textit{and} \quad \mathbb{R}^{m_\ell n_\ell} \ni \mathrm{d}\boldsymbol{B}_t^{(\ell)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{m_\ell n_\ell}\mathrm{d}t)$$

represents the infinitesimal increment of a Wiener process (standard Brownian motion) adapted to the filtration $\mathcal{F}_t$ in $\mathbb{R}^{m_\ell n_\ell}$.

4 Estimating batch gradient noise

The covariance of gradient noise structure reflects how neurons interact with each other. Capturing the full structure of gradient noise for numerically analyzing the spectra of network parameters (Stewart, 1990; Horn and Johnson, 2012; Kato, 2013) is shaped by the stochasticity arising from mini batch sampling. However, modeling the full gradient noise covariance is computationally prohibitive: accurately estimating batch gradient noise requires sampling all gradient trajectories over the entire dataset, which leads to intractable storage and computational costs (Mandt et al., 2015; Li et al., 2019; Chaudhari and Soatto, 2018).

SDE-based models often have strong assumptions regarding the structure of gradient noise. For example, in the literature (Welling and Teh, 2011; Stephan et al., 2017; Jastrzębski et al., 2017), the gradient noise is reduced into a constant scalar. Under this assumption, the SDE reduces to an Ornstein-Uhlenbeck process (Oksendal, 2013; Karatzas and Shreve, 2012). However, this simplification overlooks the covariance structures inherent in mini-batch sampling. To address this, some literature assumes a diagonal structure for the gradient noise (Jastrzębski et al., 2018; Simsekli et al., 2020). While this approximation captures varying variances across parameters, it neglects potential correlations between them.

In practice, Mandt et al. compute the exact 2×2 covariance matrix for a low-dimensional logistic-regression problem; Zhu et al. estimate the full noise covariance — solely to extract its leading eigenpairs — in an MLP with several hundred parameters (Zhu et al., 2018). Nonetheless, these methods do not scale to modern deep networks, where the number of parameters can be on the order of millions. In the remaining part of this section, we aim to develop experiment-friendly method for tracking full structure of gradient noise.

4.1 Unbiased batch gradient noise estimator

To capture the complex interactions among neurons, we model the complete structure of the gradient noise. However, by Equation 3, the exact $\mathbf{\Sigma}_t^{(\ell)}$ is given by at time point t:

$$\mathbf{\Sigma}_t^{(\ell)} := \mathrm{Var}\left[\mathrm{vec}\left(\nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}_t;\xi_t)\right)\right] \tag{5}$$

$$= \frac{1}{M}\mathrm{Var}\left[\mathrm{vec}\left(\nabla^{(\ell)}\ell_f(\boldsymbol{\theta}_k;x,y)\right)\right] \tag{6}$$

$$= \frac{1}{M}\mathbb{E}\Bigg[\left\{\mathrm{vec}\left(\nabla^{(\ell)}\ell_f(\boldsymbol{\theta}_k;x,y)\right) - \mathbb{E}\left[\mathrm{vec}\left(\nabla^{(\ell)}\ell_f(\boldsymbol{\theta}_k;x,y)\right)\right]\right\} \left\{\mathrm{vec}\left(\nabla^{(\ell)}\ell_f(\boldsymbol{\theta}_k;x,y)\right) - \mathbb{E}\left[\mathrm{vec}\left(\nabla^{(\ell)}\ell_f(\boldsymbol{\theta}_k;x,y)\right)\right]\right\}^\top\Bigg], \tag{7}$$

which suggests that the $\mathbf{\Sigma}_t^{(\ell)}$ requires the evaluation of

$$\mathrm{vec}\left(\nabla^{(\ell)}\ell_f(\boldsymbol{\theta}_k;x,y)\right) \qquad \text{and} \qquad \mathbb{E}\left[\mathrm{vec}\left(\nabla^{(\ell)}\ell_f(\boldsymbol{\theta}_k;x,y)\right)\right] \tag{8}$$

over the entire dataset for each time point t. Therefore, the computational cost of $\mathbf{\Sigma}_t^{(\ell)}$ poses a challenge to the applicability of our framework. This computation remains highly demanding, even when using state-of-the-art GPUs. To overcome this, we develop a low-rank approximation method. We now seek to estimate $\mathbf{\Sigma}_t^{(\ell)}$ without bias.

Proposition 5 (Unbiased Batch Gradient Noise Estimator) *Starting from Equation 3, the batch gradient noise at time t is estimated by:*

$$\mathbf{\Sigma}_t^{(\ell)} \approx \frac{1}{M}\Bigg[\frac{1}{M-1}\sum_{x_{t_i},y_{t_i}\in\xi_t} \underbrace{\left\{\mathrm{vec}\left[\nabla^{(\ell)}\ell_f(\boldsymbol{\theta}(t);x_{t_i},y_{t_i}) - \nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}(t);\xi_t)\right]\right\}}_{(\mathbf{\Omega}_{t_i}^{(\ell)})^\top} \underbrace{\left\{\mathrm{vec}\left[\nabla^{(\ell)}\ell_f(\boldsymbol{\theta}(t);x_{t_i},y_{t_i}) - \nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}(t);\xi_t)\right]\right\}^\top}_{\mathbf{\Omega}_{t_i}^{(\ell)}}\Bigg]$$

$$= \frac{1}{M}\left(\frac{1}{\sqrt{M-1}}\begin{bmatrix}\mathbf{\Omega}_{t_1}^{(\ell)}\\ \mathbf{\Omega}_{t_2}^{(\ell)}\\ \vdots\\ \mathbf{\Omega}_{t_M}^{(\ell)}\end{bmatrix}\right)^\top\left(\frac{1}{\sqrt{M-1}}\begin{bmatrix}\mathbf{\Omega}_{t_1}^{(\ell)}\\ \mathbf{\Omega}_{t_2}^{(\ell)}\\ \vdots\\ \mathbf{\Omega}_{t_M}^{(\ell)}\end{bmatrix}\right) = \frac{1}{M}(\mathbf{\Omega}_t^{(\ell)})^\top\mathbf{\Omega}_t^{(\ell)}, \tag{9}$$

where $\mathbf{\Omega}_{t_i}^{(\ell)}$ is the point-wise gradient fluctuation:

$$(\mathbf{\Omega}_{t_i}^{(\ell)})^\top = \mathrm{vec}\left[\nabla^{(\ell)}\ell_f(\boldsymbol{\theta}(t);x_{t_i},y_{t_i}) - \nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}(t);\xi_t)\right] \in \mathbb{R}^{m_\ell n_\ell},$$

and $\mathbf{\Omega}_t^{(\ell)}$ *is batch-wise gradient fluctuation:*

$$\mathbf{\Omega}_t^{(\ell)} = \frac{1}{\sqrt{M-1}} \begin{bmatrix} \mathbf{\Omega}_{t_1}^{(\ell)} \\ \mathbf{\Omega}_{t_2}^{(\ell)} \\ \vdots \\ \mathbf{\Omega}_{t_M}^{(\ell)} \end{bmatrix} = \sqrt{\frac{1}{M-1}} \begin{pmatrix} \mathrm{vec}\left[\nabla^{(\ell)} \ell_f(\boldsymbol{\theta}(t); x_{t_1}, y_{t_1}) - \nabla^{(\ell)} \mathcal{L}_f(\boldsymbol{\theta}(t); \xi_t)\right]^\top \\ \mathrm{vec}\left[\nabla^{(\ell)} \ell_f(\boldsymbol{\theta}(t); x_{t_2}, y_{t_2}) - \nabla^{(\ell)} \mathcal{L}_f(\boldsymbol{\theta}(t); \xi_t)\right]^\top \\ \vdots \\ \mathrm{vec}\left[\nabla^{(\ell)} \ell_f(\boldsymbol{\theta}(t); x_{t_M}, y_{t_M}) - \nabla^{(\ell)} \mathcal{L}_f(\boldsymbol{\theta}(t); \xi_t)\right]^\top \end{pmatrix}.$$

4.2 Estimating variance and covariance

To study the contributions of variances (diagonal elements) and covariances (off-diagonal elements) of gradient noise to the dynamics, we decompose the gradient noise into variance and covariance components using Proposition 6.

Proposition 6 (Diagonal and Off-Diagonal Elements of Batch Gradient Noise) *The diagonal variance* $\mathbf{\Sigma}_t^{(\ell)} \mid_{\mathrm{var}}$ *and the off-diagonal covariance* $\mathbf{\Sigma}_t^{(\ell)} \mid_{\mathrm{cov}}$ *are computed, respectively, by:*

$$\mathbf{\Sigma}_t^{(\ell)} \mid_{\mathrm{var}} \approx \frac{1}{M} \left(\mathbf{\Omega}_t^{(\ell)} \odot \mathbf{\Omega}_t^{(\ell)}\right)^\top \mathbb{1}_M, \quad \textit{and} \quad \mathbf{\Sigma}_t^{(\ell)} \mid_{\mathrm{cov}} \approx \mathbf{\Sigma}_t^{(\ell)} - \mathbf{\Sigma}_t^{(\ell)} \mid_{\mathrm{var}},$$

where $\odot$ *denotes Hadamard product. Note that* $\mathbf{\Sigma}_t^{(\ell)} \mid_{\mathrm{cov}}$ *is not necessarily a PSD matrix.*

Proof Starting from Equation 9, this can be obtained by:

$$\begin{aligned} \left(\mathbf{\Sigma}_t^{(\ell)} \mid_{\mathrm{var}}\right)_{i,i} = \left(\Sigma_t^{(\ell)}\right)_{i,i} &\approx \frac{1}{M} \left((\mathbf{\Omega}_t^{(\ell)})^\top \mathbf{\Omega}_t^{(\ell)}\right)_{i,i} = \frac{1}{M} \sum_{j=1}^{M} \left((\mathbf{\Omega}_t^{(\ell)})^\top\right)_{i,j} \mathbf{\Omega}_{j,i}^{(\ell)} \\ &= \frac{1}{M} \sum_{j=1}^{M} \mathbf{\Omega}_{j,i}^{(\ell)} \mathbf{\Omega}_{j,i}^{(\ell)} = \frac{1}{M} \sum_{j=1}^{M} \left(\mathbf{\Omega}_t^{(\ell)} \odot \mathbf{\Omega}_t^{(\ell)}\right)_{j,i} = \frac{1}{M} \left(\left(\mathbf{\Omega}_t^{(\ell)} \odot \mathbf{\Omega}_t^{(\ell)}\right)^\top \mathbb{1}_M\right)_i. \end{aligned}$$

■

4.3 Computing square root

Direct computation of $(\mathbf{\Sigma}_t^{(\ell)})^{1/2}$ by

$$(\mathbf{\Sigma}_t^{(\ell)})^{\frac{1}{2}} \approx \left[\frac{1}{M} (\mathbf{\Omega}_t^{(\ell)})^\top \mathbf{\Omega}_t^{(\ell)}\right]^{\frac{1}{2}} \in \mathbb{R}^{m_\ell n_\ell \times m_\ell n_\ell}$$

is computationally infeasible for large parameter matrices. Note that

$$\frac{1}{M} \mathbf{\Omega}_t^{(\ell)} (\mathbf{\Omega}_t^{(\ell)})^\top \in \mathbb{R}^{M \times M}, \quad M \ll m_\ell n_\ell,$$

has low-rank structure and the same spectrum of $\mathbf{\Sigma}_t^{(\ell)}$. We seek an efficient method by exploiting the low-rank SVD in $\frac{1}{M} \mathbf{\Omega}_t^{(\ell)} (\mathbf{\Omega}_t^{(\ell)})^\top$.

Proposition 7 (Square Root Approximation of Covariance Matrix) *Suppose* $\boldsymbol{\Omega}_t^{(\ell)}/\sqrt{M}$ *admits SVD:*

$$\frac{\boldsymbol{\Omega}_t^{(\ell)}}{\sqrt{M}} = \boldsymbol{U}_t^{(\ell)}(\boldsymbol{\Lambda}_t^{(\ell)})^{\frac{1}{2}}(\boldsymbol{V}_t^{(\ell)})^\top,$$

where $\boldsymbol{\Lambda}_t^{(\ell)}$ *and* $\boldsymbol{U}_t^{(\ell)}$ *are the eigenvalues and eigenvectors of* $\frac{1}{M}\boldsymbol{\Omega}_t^{(\ell)}(\boldsymbol{\Omega}_t^{(\ell)})^\top$. *Then:*

$$(\boldsymbol{\Sigma}_t^{(\ell)})^{\frac{1}{2}} \approx \left[\frac{(\boldsymbol{\Omega}_t^{(\ell)})^\top \boldsymbol{U}_t^{(\ell)}}{\sqrt{M}}\right](\boldsymbol{\Lambda}_t^{(\ell)})^{-\frac{1}{2}}\left[\frac{(\boldsymbol{\Omega}_t^{(\ell)})^\top \boldsymbol{U}_t^{(\ell)}}{\sqrt{M}}\right]^\top.$$

Proof Starting from Equation 9, this can be obtained by:

$$\begin{aligned}(\boldsymbol{\Sigma}_t^{(\ell)})^{\frac{1}{2}} &\approx \left[\frac{1}{M}(\boldsymbol{\Omega}_t^{(\ell)})^\top \boldsymbol{\Omega}_t^{(\ell)}\right]^{\frac{1}{2}} = \left[\frac{(\boldsymbol{\Omega}_t^{(\ell)})^\top}{\sqrt{M}}\boldsymbol{U}_t^{(\ell)}(\boldsymbol{U}_t^{(\ell)})^\top\frac{\boldsymbol{\Omega}_t^{(\ell)}}{\sqrt{M}}\right]^{\frac{1}{2}} \\ &= \left[\frac{(\boldsymbol{\Omega}_t^{(\ell)})^\top \boldsymbol{U}_t^{(\ell)}}{\sqrt{M}}(\boldsymbol{\Lambda}_t^{(\ell)})^{-\frac{1}{2}}(\boldsymbol{\Lambda}_t^{(\ell)})(\boldsymbol{\Lambda}_t^{(\ell)})^{-\frac{1}{2}}\frac{(\boldsymbol{U}_t^{(\ell)})^\top \boldsymbol{\Omega}_t^{(\ell)}}{\sqrt{M}}\right]^{\frac{1}{2}} \end{aligned} \tag{10}$$

Set:

$$\boldsymbol{P} = \frac{(\boldsymbol{\Omega}_t^{(\ell)})^\top \boldsymbol{U}_t^{(\ell)}}{\sqrt{M}}(\boldsymbol{\Lambda}_t^{(\ell)})^{-\frac{1}{2}},$$

and note:

$$\boldsymbol{P} = \frac{(\boldsymbol{\Omega}_t^{(\ell)})^\top \boldsymbol{U}_t^{(\ell)}}{\sqrt{M}}(\boldsymbol{\Lambda}_t^{(\ell)})^{-\frac{1}{2}} = \left[\boldsymbol{U}_t^{(\ell)}(\boldsymbol{\Lambda}_t^{(\ell)})^{\frac{1}{2}}(\boldsymbol{V}_t^{(\ell)})^\top\right]^\top \boldsymbol{U}_t^{(\ell)}(\boldsymbol{\Lambda}_t^{(\ell)})^{-\frac{1}{2}} = \boldsymbol{V}_t^{(\ell)}$$

is orthonormal.

Hence:

$$\begin{aligned}(\boldsymbol{\Sigma}_t^{(\ell)})^{\frac{1}{2}} &\approx \left[\frac{(\boldsymbol{\Omega}_t^{(\ell)})^\top \boldsymbol{U}_t^{(\ell)}}{\sqrt{M}}(\boldsymbol{\Lambda}_t^{(\ell)})^{-\frac{1}{2}}(\boldsymbol{\Lambda}_t^{(\ell)})(\boldsymbol{\Lambda}_t^{(\ell)})^{-\frac{1}{2}}\frac{(\boldsymbol{U}_t^{(\ell)})^\top \boldsymbol{\Omega}_t^{(\ell)}}{\sqrt{M}}\right]^{\frac{1}{2}} \\ &= \left[\frac{(\boldsymbol{\Omega}_t^{(\ell)})^\top \boldsymbol{U}_t^{(\ell)}}{\sqrt{M}}(\boldsymbol{\Lambda}_t^{(\ell)})^{-\frac{1}{2}}\right](\boldsymbol{\Lambda}_t^{(\ell)})^{\frac{1}{2}}\left[(\boldsymbol{\Lambda}_t^{(\ell)})^{-\frac{1}{2}}\frac{(\boldsymbol{U}_t^{(\ell)})^\top \boldsymbol{\Omega}_t^{(\ell)}}{\sqrt{M}}\right]^{\frac{1}{2}} \\ &= \underbrace{\left[\frac{(\boldsymbol{\Omega}_t^{(\ell)})^\top \boldsymbol{U}_t^{(\ell)}}{\sqrt{M}}(\boldsymbol{\Lambda}_t^{(\ell)})^{-\frac{1}{2}}\right]}_{\boldsymbol{P}}(\boldsymbol{\Lambda}_t^{(\ell)})^{\frac{1}{2}}\underbrace{\left[\frac{(\boldsymbol{\Omega}_t^{(\ell)})^\top \boldsymbol{U}_t^{(\ell)}}{\sqrt{M}}(\boldsymbol{\Lambda}_t^{(\ell)})^{-\frac{1}{2}}\right]^\top}_{\boldsymbol{P}^\top} \\ &= \left[\frac{(\boldsymbol{\Omega}_t^{(\ell)})^\top \boldsymbol{U}_t^{(\ell)}}{\sqrt{M}}\right](\boldsymbol{\Lambda}_t^{(\ell)})^{-\frac{1}{2}}\left[\frac{(\boldsymbol{\Omega}_t^{(\ell)})^\top \boldsymbol{U}_t^{(\ell)}}{\sqrt{M}}\right]^\top. \end{aligned}$$

■

5 Temporal evolution of Lipschitz upper bound

Suppose that an L-layer feed-forward neural network f is the composition of L layers, each comprising a linear unit with an activation unit, expressed as:

$$f := \left(\rho^{(L)} \circ \ \phi^{(L)}\right) \circ \cdots \circ \ \left(\rho^{(2)} \circ \ \phi^{(2)}\right) \circ \ \left(\rho^{(1)} \circ \ \phi^{(1)}\right)$$

where $\left(\rho^{(\ell)} \circ \ \phi^{(\ell)}\right)$ represents the ℓ-th layer consisting of the linear unit $\psi^{(\ell)}(\cdot)$ and the activation unit $\rho^{(\ell)}(\cdot)$. Let the Lipschitz constants of $\rho^{(\ell)}$ and $\phi^{(\ell)}$ be $A^{(\ell)}$ and $K^{(\ell)}$, respectively. The upper bound of the network Lipschitz constant of f then is the product of $A^{(\ell)}$ and $K^{(\ell)}$ across all layers, as shown in Proposition 8 (Miyato et al., 2018; Fazlyab et al., 2019; Virmaux and Scaman, 2018; Gouk et al., 2021; Virmaux and Scaman, 2018).

Proposition 8 (Lipschitz Continuity Bound in Feed-Forward Network) *Let $\hat{K}(t)$ be the Lipschitz constant of a feed-forward neural network f without skip connections. The network Lipschitz constant $\hat{K}(t)$ is upper-bounded by:*

$$\hat{K}(t) := \sup_{\forall\ u \neq v} \frac{\|f(u) - f(v)\|_2}{\|u - v\|_2} \leq K(t) := \prod_{l=1}^{L} A^{(\ell)} \cdot \prod_{l=1}^{L} K^{(\ell)}(t).$$

This upper bound is useful in robust certification and theoretical analysis of deep models with complex topologies (Fazlyab et al., 2019; Gouk et al., 2021; Virmaux and Scaman, 2018). We refer to $K(t)$ as Lipschitz continuity or Lipschitz constant for brevity.

Since most activation functions, such as ReLU, Leaky ReLU, Tanh, Sigmoid, etc, have 1-Lipschitz continuity (Virmaux and Scaman, 2018), we set $A^{(\ell)} = 1$ for brevity in theoretical analysis. Note that Proposition 8 does not take into account skip connections.

Proposition 9 (Operator Norm of Linear Unit) *Suppose the $\phi^{(\ell)}(t)$ is a linear unit with matrix multiplication or convolution:*

$$\phi^{(\ell)}(t)(z) := \boldsymbol{\theta}^{(\ell)}(t)(z) + \boldsymbol{b}^{(\ell)}(t), \tag{11}$$

*where $\boldsymbol{\theta}^{(\ell)}(t)$ and $\boldsymbol{b}^{(\ell)}(t)$ are the ℓ-th layer parameter matrix and bias, respectively. Then, the operator norm of $\phi^{(\ell)}(t)$ (*i.e. *spectral norm) admits:*

$$\|\phi^{(\ell)}(t)\|_{op} = \|\boldsymbol{\theta}^{(\ell)}(t)\|_{op} = \sigma_1^{(\ell)}(t),$$

where $\sigma_1^{(\ell)}(t)$ is the largest singular value of $\theta^{(\ell)}(t)$ (Miyato et al., 2018; Sedghi et al., 2018; Yoshida and Miyato, 2017).

Definition 10 (Stochastic Dynamical System of Lipschitz Continuity Bound) *Collect:*

1. *Definition 4: Vectorized SDE for Continuous-Time SGD;*
2. *Proposition 8: Lipschitz Continuity Bound in Feed-Forward Network;*
3. *Proposition 9: Operator Norm of Linear Unit,*

so that the dynamics of Lipschitz continuity bound for a feed-forward neural network is characterized by a system of SDEs:

$$\begin{cases} \mathrm{dvec}(\boldsymbol{\theta}^{(\ell)}(t)) = -\mathrm{vec}\left[\nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}(t))\right]\,\mathrm{d}t + \sqrt{\eta}\left[\boldsymbol{\Sigma}_t^{(\ell)}\right]^{\frac{1}{2}}\,\mathrm{d}\boldsymbol{B}_t^{(\ell)} \\ K^{(\ell)}(t) = \|\boldsymbol{\theta}^{(\ell)}(t)\|_{op} \\ Z(t) = \sum_{l=1}^{L} \log K^{(\ell)}(t) \\ K(t) = \mathrm{e}^{Z(t)} \end{cases}$$

adapted to the filtration $\mathcal{F}_t$, where: (i) $K^{(\ell)}(t)$ governs the ℓ-th layer dynamics of Lipschitz continuity; (ii) $K(t)$ governs the network dynamics of Lipschitz continuity bound; and (iii) $Z(t)$ is the logarithmic Lipschitz constant to facilitate theoretical deductions. ***This stochastic dynamical system characterizes the dynamics of the Lipschitz continuity bound induced by the optimization dynamics****.*

6 Theoretical analysis

Note that the equation system in Definition 10 is driven by a Wiener process, which consists of stochastic differential equations (SDEs), and is adapted to the filtration $\mathcal{F}_t$, with time-dependent parameters $\theta^{(\ell)}(t)$. We leverage Itô's Lemma (Itô, 1951; Oksendal, 2013) to analyze this SDE system. We aim to analyze the dynamics of $Z(t)$ and $K(t)$. The sketch for theoretical analysis is:

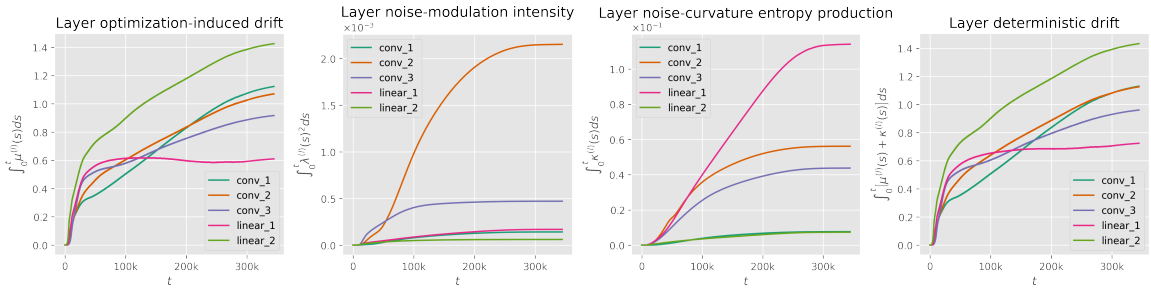
1. **Layer-Specific Dynamics** (Section 6.2). We derive the SDE for $\mathrm{d}K_t^{(\ell)}$ by applying Itô's Lemma to the process $\mathrm{dvec}\left[\theta^{(\ell)}(t)\right]$. This step establishes a direct connection between parameter-level updates driven by the optimization process and the layer-wise operator norm.

2. **Network-Specific Dynamics** (Section 6.3). Building on the layer-specific analysis, we derive the network-level SDE for $\mathrm{d}Z(t)$ by applying Itô's Lemma to the processes $\mathrm{d}K^{(\ell)}(t)$ across L layers. The process $Z(t)$ captures the logarithmic, network-level dynamics of Lipschitz continuity. Finally, we obtain the dynamics of $K(t)$ from $\mathrm{d}Z(t)$ via Itô calculus.

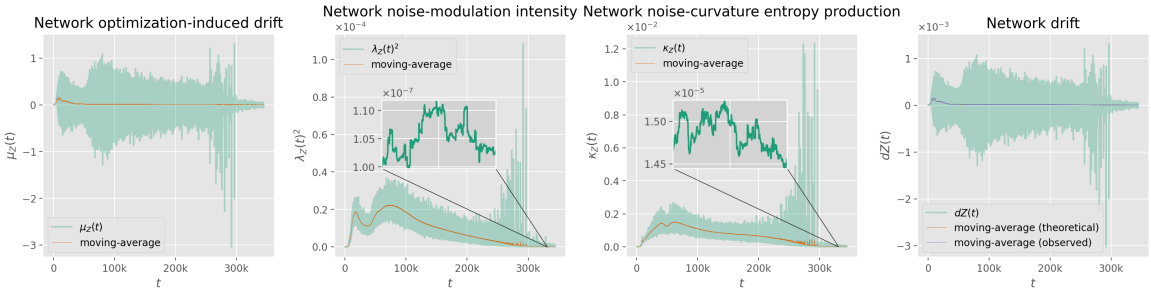
3. **Statistical Characterization** (Section 6.4). We analyze the statistical properties of $Z(t)$ and $K(t)$, including their expectations and variances. These results provide insight into the asymptotic behavior of the system near convergence.

6.1 First- and second-order operator-norm derivatives

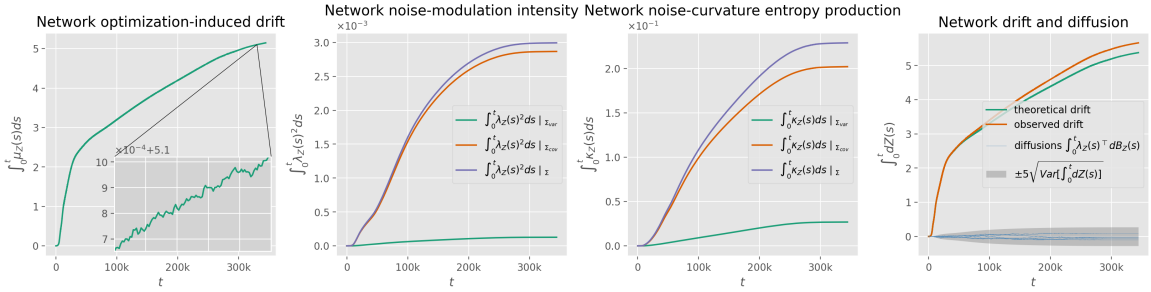
Stochastic spectral analysis regarding the parameter matrices requires the first-order and second-order derivatives of the largest singular values with respect to parameter matrices. The first-order operator-norm derivative is well-known in the literature (Luo et al., 2025; Kato, 2013; Horn and Johnson, 2012; Magnus and Neudecker, 2019) (Lemma 12). The



(a) Layer-specific dynamics



(b) Network-specific dynamics (differential)



(c) Network-specific dynamics (integral)

Figure 3: Dynamics near convergence. We profile both layer-specific and network-specific dynamics over 344,370 steps (1766 epochs) on CIFAR-10. At the end of training, the final training loss and test loss are 9.75×10^{-3} and 2.22, respectively; the final training accuracy and test accuracy are 0.99996 and 0.68540, respectively. To investigate how the variances (*i.e.* diagonal elements) and covariances (*i.e.* off-diagonal elements) of the gradient noise affect the dynamics, the dynamics are computed with respect to variances (Σ_{var}) and covariances (Σ_{cov}) respectively, using Theorem 16. The results indicate that: (i) the optimization-trajectory drift plays the primary role in shaping Lipschitz continuity over time; (ii) the covariances dominate the noise contributions; and (iii) the noise-curvature entropy production $\kappa_Z(t)$ remains significant near convergence, leading to a gradual and steady increase in Lipschitz continuity. The inset plot zooms in on 100 steps at $t = 330,000$. The moving averages are computed over a window of 500 steps.

second-order operator-norm derivative can be derived through perturbation theory (Kato, 2013; Luo et al., 2025). Lemma 13 is deducted in the literature (Luo et al., 2025) based on

perturbation theory for linear operators (Kato, 2012, 2013). Let $\boldsymbol{\theta}^{(\ell)}(t)$ admit a SVD:

$$\boldsymbol{\theta}_t^{(\ell)} = \sum_{i=1}^{r} \sigma_i^{(\ell)}(t)\boldsymbol{u}_i^{(\ell)}(t)\boldsymbol{v}_i^{(\ell)}(t)^\top, \quad \sigma_1^{(\ell)}(t) > \sigma_2^{(\ell)}(t) > \cdots > \sigma_r^{(\ell)}(t), \quad r = \operatorname{rank}(\boldsymbol{\theta}^{(\ell)}(t)),$$

under the Assumption 11.

Assumption 11 (Spectral Differentiability) *We assume the singular values of parameter matrices are* simple*: $\sigma_i^{(\ell)} \neq \sigma_j^{(\ell)}$, for all $i \neq j$. This assumption guarantees $\sigma_i^{(\ell)}, \boldsymbol{u}_i^{(\ell)}, \boldsymbol{v}_i^{(\ell)} \in C^\infty$ (*i.e. *differentiable at arbitrary integer order) (Kato, 2013, 2012).*

Lemma 12 (Operator-Norm Jacobian) *The operator-norm Jacobian of $\|\boldsymbol{\theta}^{(\ell)}(t)\|_{op}$ with respect to* $\operatorname{vec}\left[\boldsymbol{\theta}^{(\ell)}(t)\right]$ *is given by:*

$$\boldsymbol{J}_{op}^{(\ell)}(t) = \frac{\partial \|\boldsymbol{\theta}^{(\ell)}(t)\|_{op}}{\partial \operatorname{vec}\left[\boldsymbol{\theta}^{(\ell)}(t)\right]} = \boldsymbol{v}_1^{(\ell)}(t) \otimes \boldsymbol{u}_1^{(\ell)}(t) \in \mathbb{R}^{m_\ell n_\ell},$$

*where $\otimes$ denotes Kronecker product (*i.e. *tensor product) (Luo et al., 2025; Kato, 2013; Horn and Johnson, 2012; Magnus and Neudecker, 2019).*

Lemma 13 (Operator-Norm Hessian (Luo et al., 2025)) *The operator-norm Hessian of $\|\boldsymbol{\theta}^{(\ell)}(t)\|_{op}$ with respect to* $\operatorname{vec}\left[\boldsymbol{\theta}^{(\ell)}(t)\right]$ *is given by:*

$$\boldsymbol{H}_{op}^{(\ell)}(t) = \frac{\partial}{\partial \operatorname{vec}\left[\boldsymbol{\theta}^{(\ell)}(t)\|_{op}\right]} \operatorname{vec}\left[\frac{\partial \|\boldsymbol{\theta}^{(\ell)}(t)\|_{op}}{\partial \boldsymbol{\theta}^{(\ell)}(t)}\right] = \boldsymbol{H}_L^{(\ell)}(t) + \boldsymbol{H}_R^{(\ell)}(t) + \boldsymbol{H}_C^{(\ell)}(t) \in \mathbb{R}^{m_\ell n_\ell \times m_\ell n_\ell},$$

where:

$$\begin{aligned}
\boldsymbol{e}_{i,j}^{(\ell)}(t) &:= \boldsymbol{v}_i^{(\ell)}(t) \otimes \boldsymbol{u}_j^{(\ell)}(t), \\
\boldsymbol{H}_L^{(\ell)}(t) &= \sum_{i \neq 1, i \leq m_\ell} \frac{\sigma_1^{(\ell)}(t)}{\sigma_1^{(\ell)}(t)^2 - \sigma_i^{(\ell)}(t)^2} \boldsymbol{e}_{1,i}^{(\ell)}(t) \otimes \boldsymbol{e}_{1,i}^{(\ell)}(t)^\top, \\
\boldsymbol{H}_R^{(\ell)}(t) &= \sum_{j \neq 1, j \leq n_\ell} \frac{\sigma_1^{(\ell)}(t)}{\sigma_1^{(\ell)}(t)^2 - \sigma_j^{(\ell)}(t)^2} \boldsymbol{e}_{j,1}^{(\ell)}(t) \otimes \boldsymbol{e}_{j,1}^{(\ell)}(t)^\top, \\
\boldsymbol{H}_C^{(\ell)}(t) &= \sum_{k \neq 1, k \leq r_\ell} \frac{\sigma_k^{(\ell)}(t)}{\sigma_1^{(\ell)}(t)^2 - \sigma_k^{(\ell)}(t)^2} \left[\boldsymbol{e}_{1,k}^{(\ell)}(t) \otimes \boldsymbol{e}_{k,1}^{(\ell)}(t)^\top + \boldsymbol{e}_{k,1}^{(\ell)}(t) \otimes \boldsymbol{e}_{1,k}^{(\ell)}(t)^\top\right],
\end{aligned}$$

and $r_\ell = \operatorname{rank}(\boldsymbol{\theta}^{(\ell)}(t))$.

Remark 14 *Note that the function $\sigma_1^{(\ell)}(t) : \boldsymbol{\theta}^{(\ell)}(t) \mapsto \mathbb{R}$ is convex. Since the Hessian of a convex function is PSD, it follows that $\boldsymbol{H}_{op}^{(\ell)}(t)$ is PSD.*

6.2 Layer-specific dynamics

Theorem 15 (Layer-Specific Dynamics) *The layer-specific dynamics of Lipschitz continuity is given by:*

$$\frac{\mathrm{d}K^{(\ell)}(t)}{K^{(\ell)}(t)} = \left(\mu^{(\ell)}(t) + \kappa^{(\ell)}(t)\right)\,\mathrm{d}t + \boldsymbol{\lambda}^{(\ell)}(t)^\top\,\mathrm{d}\boldsymbol{B}_t^{(\ell)}, \tag{12}$$

where $\mathrm{d}\,\boldsymbol{B}_t^{(\ell)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{m_\ell n_\ell}\mathrm{d}t)$ *represents the increment of a standard Wiener process in* $\mathbb{R}^{m_\ell n_\ell}$*, and:*

$$\mu^{(\ell)}(t) = \frac{1}{\sigma_1^{(\ell)}(t)}\left\langle \boldsymbol{J}_{op}^{(\ell)}(t),\, -\mathrm{vec}\left[\nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}(t))\right]\right\rangle,$$

$$\boldsymbol{\lambda}^{(\ell)}(t) = \frac{\sqrt{\eta}}{\sigma_1^{(\ell)}(t)}\left(\left[\boldsymbol{\Sigma}_t^{(\ell)}\right]^{\frac{1}{2}}\right)^\top \boldsymbol{J}_{op}^{(\ell)}(t), \quad \|\boldsymbol{\lambda}^{(\ell)}(t)\|_2^2 = \boldsymbol{\lambda}^{(\ell)}(t)^\top\boldsymbol{\lambda}^{(\ell)}(t) \geq 0,$$

$$\kappa^{(\ell)}(t) = \frac{\eta}{2\sigma_1^{(\ell)}(t)}\left\langle \boldsymbol{H}_{op}^{(\ell)}(t), \boldsymbol{\Sigma}_t^{(\ell)}\right\rangle \geq 0,$$

where we refer to:

1. $\mu^{(\ell)}(t)$ *as* ***layer optimization-induced drift****, representing the contribution of the gradient flow* $\nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}(t))$ *— induced by the optimization process — to the expectation of Lipschitz continuity. This term corresponds to the projection of the negative gradient expectation onto the principal subspace (*i.e. *largest singular value) of the parameter matrix, and acts as a deterministic drift in the evolution of the Lipschitz constant.*
2. $\boldsymbol{\lambda}^{(\ell)}(t)$ *as* ***layer diffusion-modulation intensity****, representing the contribution of the gradient noise* $\boldsymbol{\Sigma}_t^{(\ell)}$ *— arising from the randomness in mini-batch sampling — to the stochasticity of Lipschitz continuity. This term modulates the stochasticity of Lipschitz continuity and governs the uncertainty of temporal evolution.*
3. $\kappa^{(\ell)}(t)$ *as* ***layer noise-curvature entropy production****, representing the non-negative, irreversible contribution of the gradient noise* $\boldsymbol{\Sigma}_t^{(\ell)}$ *to the deterministicity of Lipschitz continuity. Intuitively, the dynamical system baths in the stochastic gradient fluctuations captured by* $\boldsymbol{\Sigma}_t^{(\ell)}$*, arising from mini-batch sampling. The stochastic fluctuations "dissipate" into the system via the curvature of the operator-norm landscape* $\boldsymbol{H}_{op}^{(\ell)}(t)$*, driving an irreversible increase in the Lipschitz constant and entropy.*

Proof Start with Definition 10:

$$\begin{cases} \mathrm{d}\mathrm{vec}\left(\boldsymbol{\theta}^{(\ell)}(t)\right) = -\mathrm{vec}\left[\nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}(t))\right]\mathrm{d}t + \sqrt{\eta}\left[\boldsymbol{\Sigma}_t^{(\ell)}\right]^{\frac{1}{2}}\mathrm{d}\boldsymbol{B}_t^{(\ell)} & (13) \\ K^{(\ell)}(t) = \|\boldsymbol{\theta}^{(\ell)}(t)\|_{op} = \sigma_1^{(\ell)}(t) & (14) \end{cases}$$

Using Lemma 12 and Lemma 13, we apply Itô's Lemma on Equation 14:

$$\mathrm{d}K^{(\ell)}(t) = \underbrace{\boldsymbol{J}_{op}^{(\ell)}(t)^\top\mathrm{d}\,\mathrm{vec}\left(\boldsymbol{\theta}^{(\ell)}(t)\right)}_{A} + \underbrace{\frac{1}{2}\left(\mathrm{d}\,\mathrm{vec}(\boldsymbol{\theta}^{(\ell)}(t)\right)^\top \boldsymbol{H}_{op}^{(\ell)}(t)\,\mathrm{d}\,\mathrm{vec}(\boldsymbol{\theta}^{(\ell)}(t))}_{B}. \tag{15}$$

Compute A **and** B.

1. Substitute Equation 13 into A:

$$\begin{aligned} A &= \boldsymbol{J}_{op}^{(\ell)}(t)^\top \mathrm{d}\,\mathrm{vec}\left(\boldsymbol{\theta}^{(\ell)}(t)\right) \\ &= \boldsymbol{J}_{op}^{(\ell)}(t)^\top \left[-\mathrm{vec}\left[\nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}(t))\right]\,\mathrm{d}t + \sqrt{\eta}\left[\boldsymbol{\Sigma}_t^{(\ell)}\right]^{\frac{1}{2}}\,\mathrm{d}\boldsymbol{B}_t^{(\ell)}\right] \\ &= \left\langle \boldsymbol{J}_{op}^{(\ell)}(t), -\mathrm{vec}\left[\nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}(t))\right]\right\rangle \mathrm{d}t + \boldsymbol{J}_{op}^{(\ell)}(t)^\top\sqrt{\eta}\left[\boldsymbol{\Sigma}_t^{(\ell)}\right]^{\frac{1}{2}}\,\mathrm{d}\boldsymbol{B}_t^{(\ell)}. \end{aligned} \tag{16}$$

2. Use trace identity $\mathrm{tr}(XYZ) = \mathrm{tr}(YZX)$:

$$\begin{aligned} B &= \frac{1}{2}\left(\mathrm{d}\,\mathrm{vec}(\boldsymbol{\theta}^{(\ell)}(t)\right)^\top \boldsymbol{H}_{op}^{(\ell)}(t)\mathrm{d}\,\mathrm{vec}(\boldsymbol{\theta}^{(\ell)}(t)) \\ &= \frac{1}{2}\,\mathrm{tr}\left[\boldsymbol{H}_{op}^{(\ell)}(t)\,\mathrm{d}\,\mathrm{vec}(\boldsymbol{\theta}^{(\ell)}(t))\left(\mathrm{d}\,\mathrm{vec}(\boldsymbol{\theta}^{(\ell)}(t)\right)^\top\right]. \end{aligned}$$

Substituting Equation 13 into B, we obtain:

$$\begin{aligned} B &= \frac{1}{2}\,\mathrm{tr}\left[\boldsymbol{H}_{op}^{(\ell)}(t)\mathrm{d}\,\mathrm{vec}(\boldsymbol{\theta}^{(\ell)}(t))\left(\mathrm{d}\,\mathrm{vec}(\boldsymbol{\theta}^{(\ell)}(t)\right)^\top\right] \\ &= \frac{1}{2}\,\mathrm{tr}\left[\boldsymbol{H}_{op}^{(\ell)}(t)\eta\left[\boldsymbol{\Sigma}_t^{(\ell)}\right]^{\frac{1}{2}}\left\{\left[\boldsymbol{\Sigma}_t^{(\ell)}\right]^{\frac{1}{2}}\right\}^\top \mathrm{d}\boldsymbol{B}_t^{(\ell)}(\mathrm{d}\boldsymbol{B}_t^{(\ell)})^\top\right] \\ &= \frac{1}{2}\eta\,\mathrm{tr}\left[\boldsymbol{H}_{op}^{(\ell)}(t)\left[\boldsymbol{\Sigma}_t^{(\ell)}\right]^{\frac{1}{2}}\left\{\left[\boldsymbol{\Sigma}_t^{(\ell)}\right]^{\frac{1}{2}}\right\}^\top\right]\mathrm{d}t \\ &= \frac{1}{2}\eta\,\mathrm{tr}\left[\boldsymbol{H}_{op}^{(\ell)}(t)\boldsymbol{\Sigma}_t^{(\ell)}\right]\mathrm{d}t = \frac{1}{2}\eta\left\langle \boldsymbol{H}_{op}^{(\ell)}(t), \boldsymbol{\Sigma}_t^{(\ell)}\right\rangle \mathrm{d}t, \end{aligned} \tag{17}$$

by dropping higher-order infinitesimal terms, as $o\left((\mathrm{d}t)^2\right) \to 0$ and $o\left(\mathrm{d}t\mathrm{d}\boldsymbol{B}_t^{(\ell)}\right) \to \mathbf{0}$.

Combine $A + B$. Combine expanded A and B:

$$\begin{aligned} \frac{\mathrm{d}K^{(\ell)}(t)}{K^{(\ell)}(t)} = &\left[\underbrace{\frac{1}{\sigma_1^{(\ell)}(t)}\left\langle \boldsymbol{J}_{op}^{(\ell)}(t), -\mathrm{vec}\left[\nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta}(t))\right]\right\rangle}_{\mu^{(\ell)}(t)} + \underbrace{\frac{\eta}{2\sigma_1^{(\ell)}(t)}\left\langle \boldsymbol{H}_{op}^{(\ell)}(t), \boldsymbol{\Sigma}_t^{(\ell)}\right\rangle}_{\kappa^{(\ell)}(t)}\right]\mathrm{d}t \\ &+ \underbrace{\frac{\sqrt{\eta}}{\sigma_1^{(\ell)}(t)}\boldsymbol{J}_{op}^{(\ell)}(t)^\top\left[\boldsymbol{\Sigma}_t^{(\ell)}\right]^{\frac{1}{2}}}_{\boldsymbol{\lambda}^{(\ell)}(t)^\top}\,\mathrm{d}\boldsymbol{B}_t^{(\ell)}. \end{aligned} \tag{18}$$

■

6.3 Network-specific dynamics

Theorem 16 (Logarithmic Network-Specific Dynamics) *The logarithmic, network-specific dynamics of Lipschitz continuity is given by:*

$$\mathrm{d}Z(t) = \left(\mu_Z(t) + \kappa_Z(t) - \frac{1}{2}\lambda_Z(t)^2\right)\mathrm{d}t + \lambda_Z(t)\,\mathrm{d}W_t, \tag{19}$$

where $\mathrm{d}W_t \sim \mathcal{N}(0, \mathrm{d}t)$ *represents the increment of a standard Wiener process adapted to the filtration* $\mathcal{F}_t$ *in* $\mathbb{R}$ *by:*

$$\mathrm{d}W_t := \left[\sum_{\ell=1}^{L} \|\boldsymbol{\lambda}^{(\ell)}(t)\|_2^2\right]^{-\frac{1}{2}} \cdot \sum_{\ell=1}^{L} \boldsymbol{\lambda}^{(\ell)}(t)^\top \,\mathrm{d}\boldsymbol{B}_t^{(\ell)} \quad \sim \quad \mathcal{N}(0, \mathrm{d}t),$$

$$\mu_Z(t) = \sum_{\ell=1}^{L} \mu^{(\ell)}(t) \in \mathbb{R}, \quad \kappa_Z(t) = \sum_{\ell=1}^{L} \kappa^{(\ell)}(t) \in \mathbb{R}_+, \quad \lambda_Z(t) = \left[\sum_{\ell=1}^{L} \|\boldsymbol{\lambda}^{(\ell)}(t)\|_2^2\right]^{\frac{1}{2}} \in \mathbb{R}_+,$$

where we refer to: (i) $\mu_Z(t)$ *as* ***network optimization-induced drift****; (ii)* $\lambda_Z(t)$ *as* ***network diffusion-modulation intensity****; and (iii)* $\kappa_Z(t)$ *as* ***network noise-curvature entropy production****, respectively.*

Proof Consider the differentials:

$$\frac{\partial Z(t)}{\partial K^{(\ell)}(t)} = \frac{1}{K^{(\ell)}(t)}, \quad \text{and} \quad \frac{\partial^2 Z(t)}{\partial K^{(\ell)}(t)^2} = -\frac{1}{K^{(\ell)}(t)^2}. \tag{20}$$

We apply Itô's Lemma on $\mathrm{d}K^{(\ell)}$ across all layers to derive the dynamics of:

$$Z(t) = \sum_{l=1}^{L} \log K^{(\ell)}(t),$$

so that:

$$\mathrm{d}Z(t) = \mathrm{d}\left(\sum_{l=1}^{L} \log K^{(\ell)}(t)\right) = \sum_{l=1}^{L} \left(\underbrace{\frac{\partial Z(t)}{\partial K^{(\ell)}(t)} \mathrm{d}K^{(\ell)}(t)}_{C} + \underbrace{\frac{1}{2}\frac{\partial^2 Z(t)}{\partial K^{(\ell)}(t)^2}\left(\mathrm{d}K^{(\ell)}(t)\right)^2}_{D} \right).$$

Derive C **and** D.

1. Derive C:

$$C = \frac{\partial Z(t)}{\partial K^{(\ell)}(t)} \mathrm{d}K^{(\ell)}(t) = \frac{\mathrm{d}K^{(\ell)}(t)}{K^{(\ell)}(t)} = \left(\mu^{(\ell)}(t) + \kappa^{(\ell)}(t)\right)\,\mathrm{d}t + \boldsymbol{\lambda}^{(\ell)}(t)^\top \mathrm{d}\boldsymbol{B}_t^{(\ell)}.$$

2. Derive D:

$$D = \frac{1}{2}\frac{\partial^2 Z(t)}{\partial K^{(\ell)}(t)^2}\left(\mathrm{d}K^{(\ell)}(t)\right)^2 = -\frac{1}{2K^{(\ell)}(t)^2}\left(\mathrm{d}K^{(\ell)}(t)\right)^2 = -\frac{1}{2}\|\boldsymbol{\lambda}^{(\ell)}(t)\|_2^2 \mathrm{d}t,$$

by dropping higher-order infinitesimal terms as $o((\mathrm{d}t)^2) \to 0$ and $o(\mathrm{d}t\mathrm{d}\boldsymbol{B}_t^{(\ell)}) \to \mathbf{0}$.

Combine $C + D$. Hence:

$$\begin{aligned}\mathrm{d}Z(t) &= \left(\sum_{l=1}^{L} \mu^{(\ell)}(t) + \kappa^{(\ell)}(t) - \frac{1}{2}\|\boldsymbol{\lambda}^{(\ell)}(t)\|_2^2\right)\mathrm{d}t + \sum_{\ell=1}^{L} \boldsymbol{\lambda}^{(\ell)}(t)^\top \mathrm{d}\boldsymbol{B}_t^{(\ell)} \\ &= \left(\mu_Z(t) + \kappa_Z(t) - \frac{1}{2}\lambda_Z(t)^2\right)\mathrm{d}t + \lambda_Z(t)\mathrm{d}W_t\end{aligned}$$

where $\mathrm{d}W_t \sim \mathcal{N}(0, \mathrm{d}t)$ represents the increment of a Wiener process in $\mathbb{R}$.

■

Theorem 17 (Integral-Form Dynamics of Lipschitz Continuity) *Using Itô's calculus, the integral-form dynamics of Lipschitz continuity is stated as:*

$$\begin{cases} Z(t) = Z(0) + \displaystyle\int_0^t \left[\mu_Z(s) + \kappa_Z(s) - \frac{1}{2}\lambda_Z(s)^2\right]\mathrm{d}s + \int_0^t \lambda_Z(s)\,\mathrm{d}W_s \\ K(t) = \exp\{Z(t)\} \end{cases}$$

where $Z(0)$ is the initial value of $Z(t)$.

6.4 Statistical characterization

Theorem 18 (Statistics of Lipschitz Continuity) *Let $K(0)$ be the initial Lipschitz continuity bound, and hence $Z(0) = \log K(0)$. The expectation and variance of $K(t)$ are given as:*

$$\mathbb{E}[K(t)] = \mathrm{e}^{\mathbb{E}[Z(t)] + \frac{1}{2}\mathrm{Var}[Z(t)]} = K(0) \cdot \mathrm{e}^{\int_0^t \mu_Z(s)\mathrm{d}s} \cdot \mathrm{e}^{\int_0^t \kappa_Z(s)\mathrm{d}s} \tag{21}$$

$$\mathrm{Var}[K(t)] = \mathbb{E}[K(t)]^2\left(\mathrm{e}^{\mathrm{Var}[Z(t)]} - 1\right) = \mathbb{E}[K(t)]^2\left(\mathrm{e}^{\int_0^t \lambda_Z(s)^2\mathrm{d}s} - 1\right), \tag{22}$$

where:

$$\mathbb{E}[Z(t)] = Z(0) + \int_0^t \left[\mu_Z(s) + \kappa_Z(s) - \frac{1}{2}\lambda_Z(s)^2\right]\mathrm{d}s \tag{23}$$

$$\mathrm{Var}\left[Z(t)\right] = \int_0^t \lambda_Z(s)^2\,\mathrm{d}s. \tag{24}$$

Remark 19 *The expectation of Lipschitz continuity is dominated only by three factors:*

- *i* ***Parameter initialization*** $K(0)$. *The parameter initialization determines the network Lipschitz continuity. We discuss the details in Section 8.2.*
- *ii* ***Optimization-induced drift*** $\mu_Z(t)$. *The projection of gradient expectations on the operator-norm Jacobian, induced by optimization process, drive the evolution of Lipschitz continuity;*
- *iii* ***Noise-curvature entropy production*** $\kappa_Z(t)$. *The gradient fluctuations, arising from mini-batch sampling, has a deterministic, non-negative and irreversible increase in the Lipschitz continuity.*

The uncertainty of Lipschitz continuity is determined by two factors:

i ***Lipschitz continuity expectation*** $\mathbb{E}[K(t)]$. *The uncertainty is proportional to the current Lipschitz constant bound. Larger Lipschitz constant bound leads to larger uncertainty of the evolution;*

ii ***Diffusion-modulation intensity*** $\lambda_Z(t)$. *The projection of the gradient fluctuations on the operator-norm Jacobian modulates the diffusion process of Lipschitz continuity, hence dominates the uncertainty of the evolution.*

Proof

Lemma 20 (Moment-Generating Function of $Z(t)$) *Consider $Z(t)$:*

$$Z(t) \sim \mathcal{N}\left(\mathbb{E}\left[Z(t)\right], \mathrm{Var}\left[Z(t)\right]\right), \tag{25}$$

then the MGF of $Z(t)$ is given as:

$$\mathbb{E}\left[e^{kZ(t)}\right] = e^{k\mathbb{E}[Z(t)]+\frac{1}{2}k^2\,\mathrm{Var}[Z(t)]}. \tag{26}$$

Set $k = 1$:

$$\mathbb{E}\left[K(t)\right] = \mathbb{E}\left[e^{Z(t)}\right] = e^{\mathbb{E}[Z(t)]+\frac{1}{2}\,\mathrm{Var}[Z(t)]} = \mathrm{e}^{Z(0)+\int_0^t[\mu_Z(s)+\kappa_Z(s)]\mathrm{d}s}, \tag{27}$$

so that:

$$\left(\mathbb{E}\left[K(t)\right]\right)^2 = \left(\mathbb{E}\left[e^{Z(t)}\right]\right)^2 = e^{2\mathbb{E}[Z(t)]+\mathrm{Var}[Z(t)]} = \mathrm{e}^{2Z(0)+\int_0^t[2\mu_Z(s)+2\kappa_Z(s)]\mathrm{d}s}. \tag{28}$$

Set $k = 2$:

$$\mathbb{E}\left[K(t)^2\right] = \mathbb{E}\left[e^{2Z(t)}\right] = e^{2\mathbb{E}[Z(t)]+2\,\mathrm{Var}[Z(t)]}. \tag{29}$$

Hence:

$$\mathrm{Var}\left[K(t)\right] = \mathbb{E}\left[K(t)^2\right] - \left(\mathbb{E}\left[K(t)\right]\right)^2 = \mathbb{E}\left[K(t)\right]^2\left(e^{\mathrm{Var}[Z(t)]} - 1\right). \tag{30}$$

■

7 Framework validation

Experimental Settings. We conduct numerical experiments to validate our framework, using a five-layer ConvNet (details in Table 1 in the Appendix), optimized with SGD (without momentum). Experimental details are as follows:

1. **Datasets**. To validate the effectiveness of our framework across various datasets, the ConvNet is trained on CIFAR-10 and CIFAR-100 respectively, for $39,000$ steps (200 epochs) with a learning rate of 10^{-3} and a batch size of 256.

2. **Regularizers**. To validate the effectiveness of our framework under various regularization configurations (see Table 3 in Appendix), the ConvNet is trained with multiple regularizers, including *batch normalization*, *mixup*, and *label smoothing*. Additional results with *dropout*, *weight decay*, *auto-augment*, and *adversarial training* are presented in the Appendix.

Experimental Results. Figure 2a presents the results on CIFAR-10, and Figure 2b shows the corresponding results on CIFAR-100. These experiments demonstrate that our mathematical framework closely captures the dynamics of Lipschitz continuity across a range of regularizers.

8 Theoretical implications

Our theoretical framework provides a unified perspective for analyzing and interpreting the evolution of Lipschitz continuity in neural networks. Specifically, it enables us to answer the following questions:

1. Section 8.1: **Key factors**. What are the key factors governing the temporal evolution of Lipschitz continuity during training?
2. Section 8.2: **Parameter initialization**. How does parameter initialization affect the temporal evolution of Lipschitz continuity?
3. Section 8.3: **Unbounded growth near convergence**. What dynamics emerge as the network approaches convergence, particularly in the regime where the training loss approaches zero?
4. Section 8.4: **Noisy gradient regularization**. How does gradient noise regulate the temporal evolution of Lipschitz continuity?
5. Section 8.5: **Uniform label corruption**. How does uniform label corruption implicitly regularize the dynamics of Lipschitz continuity?
6. Section 8.6: **Batch size**. What role does batch size play in shaping the temporal evolution of Lipschitz continuity?
7. Section 8.7: **Mini-batch sampling trajectory**. What impact does mini-batch sampling have on the temporal evolution of Lipschitz continuity?

8.1 Key factors driving dynamics

Principal driving factors. Theorem 15 and Theorem 16 identify four principal factors driving the evolution of Lipschitz continuity at both the layer and network levels:

(i) **Gradient-principal-direction alignment**. The alignment between the gradient expectations and the principal directions of parameter matrices determines *optimization-induced drift* the $\mu_Z(t)$. As shown in Figure 3, this force dominates the drift of the Lipschitz continuity (see Theorem 15 and Theorem 16).

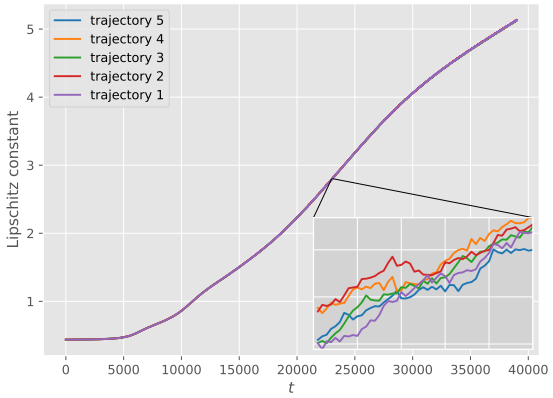


Figure 4: Predicted effect of the trajectory in mini-batch sampling on Lipschitz continuity. The inset plot zooms in on 50 steps at $t = 23000$.

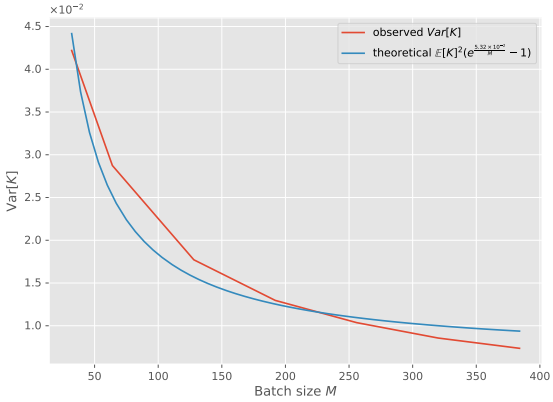


Figure 5: Predicted effect of batch size on the variance of Lipschitz continuity.

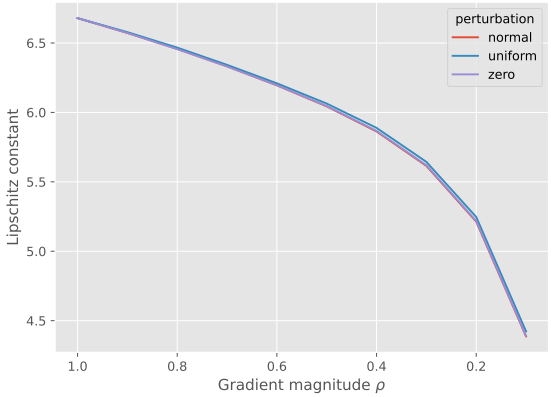


Figure 6: Predicted effect of gradient magnitude and perturbation.

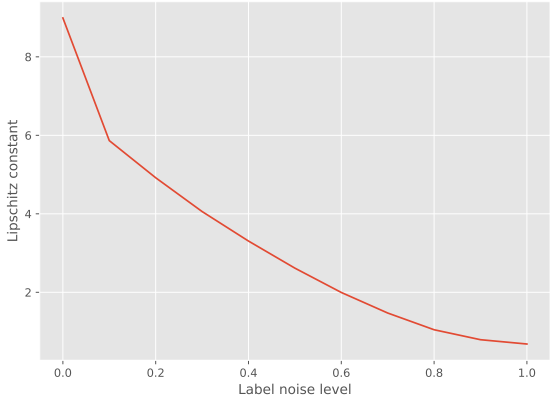


Figure 7: Predicted effect of uniform label corruption.

(ii) **Directional gradient noise**. The directional projection of gradient fluctuations, arising from mini-batch sampling, on the principal directions of parameter matrices determines the *diffusion-modulation intensity* $\lambda_Z(t)$ arises from. As shown in Figure 3, this factor modulates the diffusion of the Lipschitz continuity.

(iii) **Deterministic effect of gradient noise**. The interaction between the gradient fluctuations, arising from mini-batch sampling, and the operator-norm curvatures determines the *noise-curvature entropy production* $\kappa_Z(t)$. Our theoretical framework suggests that this term gives a deterministic, non-negative, and irreversible increase contributing to the drift of the dynamics.

(iv) **Gradient noise amplification effect**. Lemma 13 and Theorem 15 suggest that the inverted principal spectral gaps in parameter matrices, defined as $\left[\sigma_1^{(\ell)}(t)^2 - \sigma_j^{(\ell)}(t)^2\right]^{-1}$, for all $j \neq 1$, can amplify the gradient noise. Thus smaller spectral gaps amplifies gradient noise, contributing to larger *noise-curvature entropy production* $\kappa_Z(t)$.

8.2 Parameter initialization

According to Theorem 18, the network initialization $K(0)$ affects both the expectation and uncertainty of Lipschitz continuity in the evolution. Suppose that the entries of a parameter $\theta^{(\ell)} \in \mathbb{R}^{m_\ell \times n_\ell}$ are sampled from $\mathcal{N}(0, s_\ell^2)$. According to the Tracy-Widom limiting distribution (Tracy and Widom, 1994; Tao, 2012), the largest singular value limit is $\sigma_1^\ell \to (\sqrt{m_\ell} + \sqrt{n_\ell}) \cdot s_\ell$, if $m_\ell \cdot n_\ell$ is sufficiently large. For a network with L layers,

$$K(0) \to \prod_{\ell=1}^{L} (\sqrt{m_\ell} + \sqrt{n_\ell}) \cdot s_\ell,$$

hence a larger network has a larger Lipschitz continuity. For the case of Kaiming initialization, $s_\ell = \sqrt{2/n_\ell}$ (He et al., 2015).

8.3 Unbounded growth near-convergence

Suppose a negative-log-likelihood loss:

$$\ell_f(\boldsymbol{\theta}(t); x, y) = -\log p_{\boldsymbol{\theta}(t)}(y \mid x)$$

near convergence:

$$\ell_f(\boldsymbol{\theta}(t); x, y) \to 0.$$

Under Assumption 3, the Fisher information matrix (FIM) of $p_{\boldsymbol{\theta}(t)}(y \mid x)$ with respect to parameters $\boldsymbol{\theta}^{(\ell)}(t)$ (Martens, 2020) is defined as

$$\boldsymbol{F}^{(\ell)}(t) := \mathbb{E}\left[\mathrm{vec}\left[\nabla^{(\ell)} p_{\boldsymbol{\theta}(t)}(y \mid x)\right] \mathrm{vec}\left[\nabla^{(\ell)} p_{\boldsymbol{\theta}(t)}(y \mid x)\right]^\top\right] \approx M \cdot \boldsymbol{\Sigma}_t^{(\ell)},$$

as $\ell_f(\theta; x, y) \to 0$, where M is the batch size (Jastrzębski et al., 2017; Stephan et al., 2017; Li et al., 2021; Jastrzębski et al., 2017; Martens, 2020). This will serve as a foundation for our later theoretical analysis of the near-convergence dynamics.

Proposition 21 (Unbounded Growth Near-Convergence) *Consider the terms in the dynamics (Theorem 15):*

$$\mu^{(\ell)}(t) = \frac{1}{\sigma_1^{(\ell)}(t)} \left\langle \boldsymbol{J}_{op}^{(\ell)}(t), -\mathrm{vec}\left[\nabla^{(\ell)} \mathcal{L}_f(\boldsymbol{\theta}(t))\right] \right\rangle \approx 0,$$

$$\boldsymbol{\lambda}^{(\ell)}(t) = \frac{\sqrt{\eta}}{\sigma_1^{(\ell)}(t)} \left(\left[\boldsymbol{\Sigma}_t^{(\ell)}\right]^{\frac{1}{2}}\right)^\top \boldsymbol{J}_{op}^{(\ell)}(t) \approx \frac{\sqrt{\eta}}{\sigma_1^{(\ell)}(t)} \left(\left[\frac{1}{M}\boldsymbol{F}^{(\ell)}(t)\right]^{\frac{1}{2}}\right)^\top \boldsymbol{J}_{op}^{(\ell)}(t) \to \mathbf{0},$$

$$\kappa^{(\ell)}(t) = \frac{\eta}{2\sigma_1^{(\ell)}(t)} \left\langle \boldsymbol{H}_{op}^{(\ell)}(t), \boldsymbol{\Sigma}_t^{(\ell)} \right\rangle \approx \frac{\eta}{2\sigma_1^{(\ell)}(t)} \frac{1}{M} \left\langle \boldsymbol{H}_{op}^{(\ell)}(t), \boldsymbol{F}^{(\ell)}(t) \right\rangle > 0,$$

so that Lipschitz continuity is not bounded in expectation:

$$\lim_{t\to 0}\mathbb{E}\left[K(t)\right]=\lim_{t\to 0}\mathrm{e}^{Z(0)+\int_0^t[\mu_Z(s)+\kappa_Z(s)]\mathrm{d}s}=\lim_{t\to 0}\mathrm{e}^{Z(0)+\int_0^t\kappa_Z(s)\mathrm{d}s}\to\infty,$$

and uncertainty:

$$\lim_{t\to 0}\mathrm{Var}\left[K(t)\right]=\lim_{t\to 0}\mathbb{E}\left[K(t)\right]^2\left(e^{\mathrm{Var}[Z(t)]}-1\right)\to\infty,$$

as $t\to\infty$. This result demonstrates that the Lipschitz continuity bound monotonically increases as optimization progresses. This phenomenon has been noted in literature (Yoshida and Miyato, 2017; Bartlett et al., 2017; Sedghi et al., 2018; Gamba et al., 2023).

Experimental Results. We extend the training of the five-layer ConvNet on CIFAR-10 without regularization to 344,370 steps (1766 epochs) for observing the long range dynamics near convergence. Figure 3a shows the layer-specific dynamics; Figure 3c shows the network-specific dynamics. The observations are as follows:

(i) **Unbounded growth**. We train the network for up to 344,370 steps, near convergence. The gradual increase driven by $\kappa_Z(t)$ persists throughout the training, with no observable indication of stopping, suggesting that the dynamics do not admit a finite-time steady state.

(ii) **Non-negligible contribution of gradient noise**. Although the magnitude of $\kappa_Z(t)$ is typically one order smaller than that of $\mu_Z(t)$ in our experiments, its cumulative effect on the growth of Lipschitz continuity near-convergence is non-negligible and persists throughout training.

(iii) **Four Phases**. We observe that $\kappa_Z(t)$ exhibits four distinct phases: (i) phase 1 — a rapid initial increase at the beginning of training, (ii) phase 2 — large fluctuations prior to overfitting, (iii) phase 3 — a steady decline, and (iv) phase 4 — convergence to a non-negative constant.

8.4 Noisy gradient regularization

Prior work shows that injecting noise into gradients during training can improve robustness (Laskey et al., 2017; Welling and Teh, 2011) and help escape local minima (Neelakantan et al., 2015). Our theoretical framework elucidates the mechanism by which gradient noise serves as a regularizer for Lipschitz continuity. Consider a loss ℓ_* with noisy supervision consists of a signal loss ℓ_f and a noisy loss ℓ_n:

$$\ell_*(\boldsymbol{\theta};x,y)=\sqrt{\rho}\cdot\ell_f(\boldsymbol{\theta};x,y)+\sqrt{1-\rho}\cdot\ell_n(\boldsymbol{\theta};x,y),\tag{31}$$

where $\rho\in[0,1]$. The corresponding batch loss is:

$$\nabla^{(\ell)}\mathcal{L}_*(\boldsymbol{\theta};\xi)=\sqrt{\rho}\cdot\nabla^{(\ell)}\mathcal{L}_f(\boldsymbol{\theta};\xi)+\sqrt{1-\rho}\cdot\nabla^{(\ell)}\mathcal{L}_n(\boldsymbol{\theta};\xi),$$

and the gradient noise covariance is:

$$\Sigma^{(\ell),*}=\rho\cdot\Sigma^{(\ell),f}+(1-\rho)\cdot\Sigma^{(\ell),n}.$$

If the noise is sampled from $\mathcal{N}(0, \sigma^2)$, $\Sigma^{(\ell),n} = \sigma^2 \, \mathrm{I}$, the dynamics has the changes:

$$\mu_*^{(\ell)}(t) = \sqrt{\rho}\mu^{(\ell)}(t),$$

and

$$\kappa_*^{(\ell)}(t) = \rho \cdot \kappa^{(\ell)}(t) + (1-\rho) \cdot \frac{\eta}{2\sigma_1^{(\ell)}(t)} \underbrace{\left\langle \boldsymbol{H}_{op}^{(\ell)}(t), \sigma^2 \, \mathrm{I} \right\rangle}_{\text{variance is negligible}} \approx \rho \cdot \kappa^{(\ell)}(t).$$

The variance contribution from $\boldsymbol{H}_{op}^{(\ell)}(t)$ is neglected, as experimental results indicate it is negligible (see Figure 3). According to Theorem 18, by ignoring $\lambda_Z(t)$, we have:

$$\mathbb{E}\left[K(t)\right] \approx \mathrm{e}^{Z(0)+\int_0^t \sqrt{\rho}\cdot\mu_Z(s)\mathrm{d}s+\int_0^t \rho\cdot\kappa_Z(s)\mathrm{d}s} = K(0)\left[\mathrm{e}^{\int_0^t \mu_Z(s)\mathrm{d}s}\right]^{\sqrt{\rho}}\left[\mathrm{e}^{\int_0^t \kappa_Z(s)\mathrm{d}s}\right]^{\rho}. \tag{32}$$

Although the integral $\int_0^t \mu_Z(s)\mathrm{d}s$ and $\int_0^t \kappa_Z(s)\mathrm{d}s$ are unlikely to be computed, it provides insight how gradient magnitude affects the evolution of Lipschitz continuity bound. This result shows higher gradient magnitude leads larger Lipschitz continuity bound. This framework can be used to analyze the uniform label corruption (see Section 8.5).

Experimental Results. We train a MLP (see Table 2) on the MNIST with a batch size of 128 and a learning rate of 0.01, using SGD without momentum. During training, the gradients are scaled by $\sqrt{\rho}$, and additive noise is injected sampled from: (i) a Gaussian distribution $\mathcal{N}(0,1)$; (ii) a uniform distribution $U[-0.5, 0.5]$; and (iii) zero noise, scaled by $\sqrt{1-\rho}$. As shown in Figure 6, the Lipschitz constant decreases monotonically as ρ decreases.

8.5 Uniform label corruption

Suppose a classifier with a negative-log-likelihood loss function $\ell_f(\theta; x, y) := -\log p_\theta(y|x)$. Further suppose that the label corruption probability is $\epsilon \in [0,1]$ with a uniform distribution $U(\mathcal{Y})$, while the label remains intact with a probability $1-\epsilon$. Therefore the gradient with label corruption can be expressed as (Natarajan et al., 2013; Sukhbaatar et al., 2014; Patrini et al., 2017; Ghosh et al., 2017):

$$\mathbb{E}\left[\nabla^{(\ell)}\ell_f(\theta; x, y)\right] := (1-\epsilon)\cdot\mathbb{E}_{x,y}\left[\nabla^{(\ell)}\left[-\log p_\theta(y|x)\right]\right] + \epsilon\cdot\mathbb{E}_x\left[\nabla^{(\ell)}\frac{1}{|\mathcal{Y}|}\sum_{\tilde{y}\sim U(\mathcal{Y})}\left[-\log p_\theta(\tilde{y}|x)\right]\right],$$

where $\tilde{y} \sim U(\mathcal{Y})$. Literature (Natarajan et al., 2013; Sukhbaatar et al., 2014; Ghosh et al., 2017) assume that the loss component with label noise contributes only a constant, hence zero gradient:

$$\langle \boldsymbol{J}_{op}^{(\ell)}(t), \mathrm{vec}\left[-\nabla^{(\ell)}\mathbb{E}_{x,\tilde{y}}\left[-\log p_\theta(\tilde{y}|x)\right]\right]\rangle \to 0.$$

According to Equation 31, we can treat the label noise as ℓ_n. By using Equation 32, note $\rho = (1-\epsilon)^2$, then we have:

$$\mathbb{E}\left[K'(t)\right] \approx K(0)\cdot\left[\mathrm{e}^{\int_0^t \mu_Z(s)\mathrm{d}s}\right]^{1-\epsilon}\cdot\left[\mathrm{e}^{\int_0^t \kappa_Z(s)\mathrm{d}s}\right]^{(1-\epsilon)^2}.$$

Experimental Results. We train a MLP (see Table 2) on the MNIST with a batch size of 128 and a learning rate of 0.01, using SGD without momentum. We inject uniform label noise into the dataset with a level from 0 to 1. As shown in Figure 7, the Lipschitz constant decreases monotonically as label noise level increases.

8.6 Batch size

Suppose that the batch size M in mini-batch sampling affects only the gradient noise $\Sigma(t)$, and does not affect the expectations of gradients $\nabla\mathcal{L}_f(\theta)$. Therefore, the batch size M affects only the diffusion term (Theorem 18). We analyze the effect of mini-batch size.

Proposition 22 (Uncertainty Under Large Batch) *Let the batch size be M and sufficiently large, then:*

$$\mathrm{Var}\left[K(t)\right] \approx \mathbb{E}\left[K(t)\right]^2 \frac{C}{M},$$

where C is a constant given as:

$$C = \int_0^t \sum_{l=1}^{L} \frac{\eta}{\sigma_1^{(\ell)}(t)^2} \boldsymbol{J}_{op}^{(\ell)}(t) \mathrm{Var}\left[\mathrm{vec}\left(\nabla^{(\ell)}\ell_f(\boldsymbol{\theta}(t); x, y)\right)\right] \boldsymbol{J}_{op}^{(\ell)}(t)^\top \,\mathrm{d}s$$

Proof

Substitute Equation 3 into Theorem 18:

$$\begin{aligned}
\mathrm{Var}\left[Z(t)\right] &= \int_0^t \lambda_Z(s)^2 \,\mathrm{d}s = \int_0^t \sum_{\ell=1}^{L} \|\boldsymbol{\lambda}^{(\ell)}(s)\|_2^2 \,\mathrm{d}s = \int_0^t \sum_{l=1}^{L} \frac{\eta}{\sigma_1^{(\ell)}(t)^2} \boldsymbol{J}_{op}^{(\ell)}(t) \boldsymbol{\Sigma}^{(\ell)}(t) \boldsymbol{J}_{op}^{(\ell)}(t)^\top \,\mathrm{d}s \\
&= \int_0^t \sum_{l=1}^{L} \frac{\eta}{\sigma_1^{(\ell)}(t)^2} \boldsymbol{J}_{op}^{(\ell)}(t) \frac{1}{M} \mathrm{Var}\left[\mathrm{vec}\left(\nabla^{(\ell)}\ell_f(\boldsymbol{\theta}(t); x, y)\right)\right] \boldsymbol{J}_{op}^{(\ell)}(t)^\top \,\mathrm{d}s \\
&= \frac{1}{M} \underbrace{\int_0^t \sum_{l=1}^{L} \frac{\eta}{\sigma_1^{(\ell)}(t)^2} \boldsymbol{J}_{op}^{(\ell)}(t) \mathrm{Var}\left[\mathrm{vec}\left(\nabla^{(\ell)}\ell_f(\boldsymbol{\theta}(t); x, y)\right)\right] \boldsymbol{J}_{op}^{(\ell)}(t)^\top \,\mathrm{d}s}_{\text{a constant}} = \frac{1}{M} C.
\end{aligned}$$

Hence:

$$\mathrm{Var}\left[K(t)\right] = \mathbb{E}\left[K(t)\right]^2 \left(\mathrm{e}^{\mathrm{Var}[Z(t)]} - 1\right) = \mathbb{E}\left[K(t)\right]^2 \left(\mathrm{e}^{\frac{C}{M}} - 1\right).$$

Suppose $M \to \infty$:

$$\mathrm{Var}\left[K(t)\right] = \mathbb{E}\left[K(t)\right]^2 \left(\mathrm{e}^{\frac{C}{M}} - 1\right) \approx \mathbb{E}\left[K(t)\right]^2 \left(1 + \frac{C}{M} - 1\right) = \mathbb{E}\left[K(t)\right]^2 \frac{C}{M} \propto\!\!\propto \frac{1}{M}.$$

■

Experimental Results. We train the five-layer ConvNet (Table 1 in the Appendix), optimized with SGD without momentum. The batch size is varied from 32 to 384 in increments

of 32, and each configuration is trained for 39,000 steps. During training, we profile and collect the dynamics to compute the uncertainty of Lipschitz continuity $\mathrm{Var}[K(t)]$. As shown in Figure 5, the observed effect of batch size aligns closely with the theoretical prediction in Proposition 22.

8.7 Mini-batch sampling trajectory

According to Proposition 22, if the batch size is sufficiently large, the effect from various mini-batch sampling trajectories can be neglected since:

$$\mathrm{Var}[K(t)] \approx \mathbb{E}[K(t)]^2 \frac{C}{M} \to 0,$$

as batch size M is sufficiently large.

Experimental Results. We train the five-layer ConvNet (Table 1 in the Appendix), optimized with SGD without momentum. The parameter initialization random seed is fixed to 1, while the random seed for mini-batch sampling is varied from 1 to 5. As shown in Figure 4, the sampling trajectories have a negligible effect on the evolution of Lipschitz continuity.

9 Limitations and future work

While our mathematical framework shows strong agreement with empirical results, it has several limitations:

1. **Layer-wise noise assumption**. The simplification in Assumption 3 neglects inter-layer interactions among neurons; this may pose as a challenging for studying large models.

2. **Distributional assumption**. We assume that the gradient noise follows a time-state-dependent normal distribution varying smoothly. However, this assumption may not hold for small batch sizes, where the noise distribution can deviate significantly from Gaussian.

3. **Continuous–discrete error**. Our framework models gradient dynamics in continuous time, whereas SGD operates in discrete steps. This discrepancy may introduce non-negligible errors, particularly when modeling long-range dynamics.

10 Conclusions

We present a mathematical framework that models the dynamics of Lipschitz continuity in neural networks through a system of SDEs. This theoretical framework not only identifies the key factors governing the evolution of Lipschitz continuity, but also provides insight into how it is implicitly regularized during training. Beyond its analytical value, the framework offers a foundation for future research and development. Our experiments further validate the effectiveness of the proposed framework and the predicted effects of the theoretical implications.

Acknowledgments and Disclosure of Funding

This publication has emanated from research conducted with the financial support of **Taighde Éireann** - Research Ireland under Grant number 18/CRT/6223. We extend heartfelt thank to Prof. Dr. Changjian Shui (University of Ottawa, Canada) for the constructive comments and valuable assistance.

Appendix A. Network configuration

Function Block	# of Layer	Module
Block #1	1	`nn.Conv2d(3, 32, kernel_size=3, padding=1)`
	2	`nn.ReLU()`
	3	`nn.MaxPool2d(kernel_size=2, stride=2)`
Block #2	4	`nn.Conv2d(32, 64, kernel_size=3, padding=1)`
	5 (*)	`nn.Dropout(p=0.2)`
	6	`nn.ReLU()`
	7	`nn.MaxPool2d(kernel_size=2, stride=2)`
Block #3	8	`nn.Conv2d(64, 128, kernel_size=3, padding=1, stride=2)`
	9 (*)	`nn.Dropout(p=0.2)`
	10 (*)	`nn.BatchNorm2d(128, track_running_stats=False)`
	11	`nn.ReLU()`
Classifier	12	`nn.Linear(128 * 4 * 4, 256)`
	13	`nn.ReLU()`
	14 (*)	`nn.Dropout(p=0.3)`
	15	`nn.Linear(256, num_classes)`

Table 1: ConvNet configuration. The layer marked as '(*)' is configurable with respect to experimental needs. There are five parameterized layers (*i.e.* # 1, # 4, # 8, # 12 and # 15).

# of Layer	Module
1	`nn.Linear(28*28, 512)`
2	`nn.ReLU()`
3	`nn.Linear(512, 256)`
4	`nn.ReLU()`
5	`nn.Linear(256, 10)`

Table 2: MLP configuration. There are three parameterized layers (*i.e.* # 1, # 3, and # 5).

Appendix B. Regularization configuration

Regularization	Configuration
mixup	$\alpha = 0.4$
label smoothing	$\epsilon = 0.1$
adversarial training w/ FGSM	$\epsilon = 0.03$
weight-decay	$\lambda = 0.001$
auto-augment	`AutoAugment(policy=AutoAugmentPolicy.CIFAR10)`
	`AutoAugment(policy=AutoAugmentPolicy.CIFAR100)`

Table 3: Regularization configuration.

Appendix C. Full validation experiment

Results of full validation experiments are shown as in Figure 8.

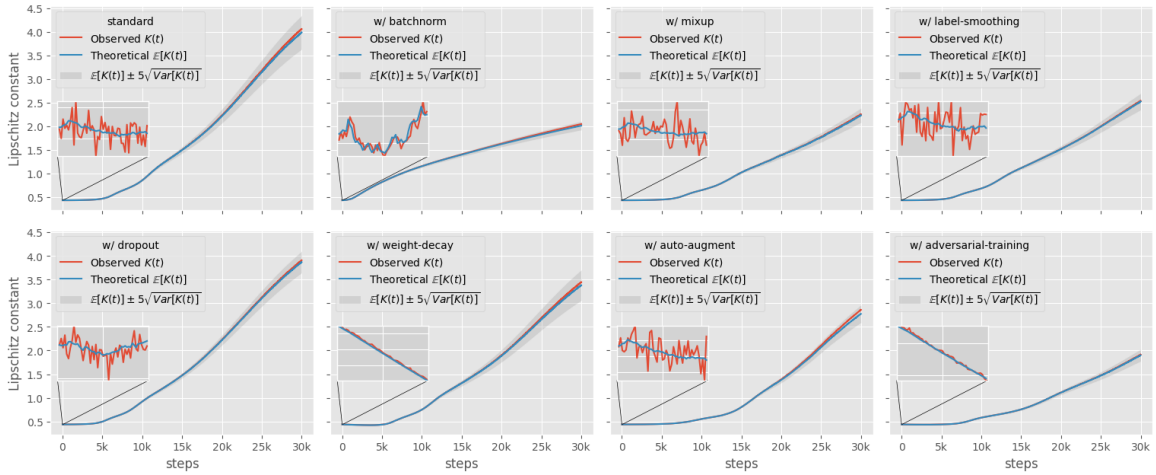


Figure 8: Full numerical validation of our mathematical framework on CIFAR-10.

References

D Applebaum. Levy processes and stochastic calculus. *Cambridge Studies in Advanced Mathematics*, 116, 2009.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. *Advances in neural information processing systems*, 30, 2017.

Pratik Chaudhari and Stefano Soatto. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *2018 Information Theory and Applications Workshop (ITA)*, pages 1–10. IEEE, 2018.

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.

Alex Damian, Eshaan Nichani, and Jason D Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. *arXiv preprint arXiv:2209.15594*, 2022.

Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. *Advances in neural information processing systems*, 32, 2019.

Mahyar Fazlyab, Taha Entesari, Aniket Roy, and Rama Chellappa. Certified robustness via dynamic margin maximization and improved lipschitz regularization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=LDhhi8HBO3`.

Matteo Gamba, Hossein Azizpour, and Mårten Björkman. On the lipschitz constant of deep networks and double descent. *arXiv preprint arXiv:2301.12309*, 2023.

Aritra Ghosh, Himanshu Kumar, and P Shanti Sastry. Robust loss functions under label noise for deep neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110:393–416, 2021.

Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012.

Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

Kiyosi Itô. *On stochastic differential equations*. Number 4. American Mathematical Soc., 1951.

Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Stanisław Jastrzębski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. On the relation between the sharpest directions of dnn loss and the sgd step length. *arXiv preprint arXiv:1807.05031*, 2018.

Ioannis Karatzas and Steven Shreve. *Brownian motion and stochastic calculus*, volume 113. Springer Science & Business Media, 2012.

Tosio Kato. *A short introduction to perturbation theory for linear operators*. Springer Science & Business Media, 2012.

Tosio Kato. *Perturbation theory for linear operators*, volume 132. Springer Science & Business Media, 2013.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Grigory Khromov and Sidak Pal Singh. Some fundamental aspects about lipschitz continuity of neural networks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=5jWsW08zUh.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on robot learning*, pages 143–156. PMLR, 2017.

Qianxiao Li, Cheng Tai, and E Weinan. Stochastic modified equations and dynamics of stochastic gradient algorithms i: Mathematical foundations. *Journal of Machine Learning Research*, 20(40):1–47, 2019.

Yuanzhi Li and Yang Yuan. Convergence analysis of two-layer neural networks with relu activation. *Advances in neural information processing systems*, 30, 2017.

Zhiyuan Li, Tianhao Wang, and Sanjeev Arora. What happens after sgd reaches zero loss?–a mathematical framework. *arXiv preprint arXiv:2110.06914*, 2021.

Roisin Luo, James McDermott, and Colm O'Riordan. Interpreting global perturbation robustness of image models using axiomatic spectral importance decomposition. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=uQYomAuo7M.

Róisín Luo, Colm O'Riordan, and James McDermott. Higher-order singular-value derivatives of rectangular real matrices. *Journal of Mathematical Analysis and Applications (JMAA)*, 2025. ISSN 0022-247X (print); 1096-0813 (web). doi: 10.1016/j.jmaa.2025.130236. URL https://arxiv.org/abs/2506.03764.

Aleksander Mądry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *stat*, 1050(9), 2017.

Jan R Magnus and Heinz Neudecker. *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons, 2019.

Sadhika Malladi, Kaifeng Lyu, Abhishek Panigrahi, and Sanjeev Arora. On the sdes and scaling rules for adaptive gradient algorithms. *Advances in Neural Information Processing Systems*, 35:7697–7711, 2022.

Stephan Mandt, Matthew D Hoffman, David M Blei, et al. Continuous-time limit of stochastic gradient descent revisited. *NIPS-2015*, 2015.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. *Advances in neural information processing systems*, 26, 2013.

Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015.

Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1944–1952, 2017.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

Hanie Sedghi, Vineet Gupta, and Philip M Long. The singular values of convolutional layers. *arXiv preprint arXiv:1805.10408*, 2018.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Umut Simsekli, Levent Sagun, and Mert Gurbuzbalaban. A tail-index analysis of stochastic gradient noise in deep neural networks. In *International Conference on Machine Learning*, pages 5827–5837. PMLR, 2019.

Umut Simsekli, Lingjiong Zhu, Yee Whye Teh, and Mert Gurbuzbalaban. Fractional underdamped langevin dynamics: Retargeting sgd with momentum under heavy-tailed gradient noise. In *International conference on machine learning*, pages 8970–8980. PMLR, 2020.

Mandt Stephan, Matthew D Hoffman, David M Blei, et al. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18(134):1–35, 2017.

GW Stewart. Matrix perturbation theory. *Computer Science and Scientific Computing/Academic Press, Inc*, 1990.

Weijie Su, Stephen Boyd, and Emmanuel J Candes. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. *Journal of Machine Learning Research*, 17(153):1–43, 2016.

Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014. URL `https://arxiv.org/abs/1312.6199`.

Terence Tao. *Analysis*, volume 1. Springer, 2006.

Terence Tao. *Topics in random matrix theory*, volume 132. American Mathematical Soc., 2012.

Craig A Tracy and Harold Widom. Level-spacing distributions and the airy kernel. *Communications in Mathematical Physics*, 159:151–174, 1994.

Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems*, 31, 2018.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

Chen Xing, Devansh Arpit, Christos Tsirigotis, and Yoshua Bengio. A walk with sgd. *arXiv preprint arXiv:1802.08770*, 2018.

Dong Yin, Ramchandran Kannan, and Peter Bartlett. Rademacher complexity for adversarially robust generalization. In *International conference on machine learning*, pages 7085–7094. PMLR, 2019.

Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

Kôsaku Yosida. *Functional analysis*, volume 123. Springer Science & Business Media, 2012.

Bohang Zhang, Du Jiang, Di He, and Liwei Wang. Rethinking lipschitz neural networks and certified robustness: A boolean function perspective. *Advances in neural information processing systems*, 35:19398–19413, 2022.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. *arXiv preprint arXiv:1803.00195*, 2018.