On the Lipschitz Constant of Deep Networks and Double Descent

Matteo Gamba
mgamba@kth.se
Hossein Azizpour
azizpour@kth.se
Mårten Björkman
celle@kth.se

KTH Royal Institute of Technology
Stockholm

Abstract

Existing bounds on the generalization error of deep networks assume some form of smooth or bounded dependence on the input variable, falling short of investigating the mechanisms controlling such factors in practice. In this work, we present an extensive experimental study of the empirical Lipschitz constant of deep networks undergoing double descent, and highlight non-monotonic trends strongly correlating with the test error. Building a connection between parameter-space and input-space gradients for SGD around a critical point, we isolate two important factors – namely loss landscape curvature and distance of parameters from initialization – respectively controlling optimization dynamics around a critical point and bounding model function complexity, even beyond the training data. Our study presents novel insights on implicit regularization via overparameterization, and effective model complexity for networks trained in practice.

1 Introduction

A longstanding question towards understanding the remarkable generalization ability of deep networks is characterizing the hypothesis class of models *trained in practice*, thus isolating properties of the networks' model function that capture generalization [19, 36]. Chiefly, a central problem is understanding the role played by overparameterization [1, 37, 48] – a key design choice of state of the art models – in promoting regularization of the model function.

Modern overparameterized networks can achieve good generalization while perfectly interpolating the training set [35]. This phenomenon is described by the *double descent* curve of the test error [4, 15]: as model size increases, the error follows the classical bias-variance trade-off curve [16], peaks when a model is large enough to interpolate the training data, and then decreases again as model size grows further. Thus, a promising direction for capturing model complexity is studying regularity w.r.t. smooth interpolation of training data.

Interestingly, many existing bounds on the generalization error of deep networks *postulate* bounded dependence of the model function on the input variable, via bounded Lipschitz constant [2, 25, 29, 34, 46], falling short of investigating the mechanisms controlling Lipschitz continuity and input-space smoothness in relation to interpolation.

Thus, a natural question to ask is *whether bounded Lipschitzness – a key theoretical assumption for representing well-behaved model functions for fixed-size architectures – provides a faithful representation of the hypothesis class of networks trained in practice, when model size varies in the overparameterized setting*. Specifically, any notion of regularity of model functions capturing generalization should mirror the non-monotonic trend of the test error.

Contributions In this work[1], (1) we present an empirical investigation of input-space smoothness of deep networks through a lower bound on their Lipschitz constant, capturing smoothness of interpolation of the training data, as model size varies; (2) we observe non-monotonic trends for the empirical Lipschitz lower bound, showing strong correlation with double descent; (3) we provide an upper bound on the true Lipschitz constant, also mirroring double descent; (4) we establish a theoretical connection between the observed trends and parameter-space dynamics of SGD in terms of sharpness of the loss landscape; (5) we present several correlates of double descent, providing insights on the hypothesis class of networks trained in practice and their effective complexity.

Experimental Setup We study deep networks under double descent, when model size is controlled by network width. We reproduce the double descent curves of the test error [4] by training a family of ConvNets and ResNet18s [21] on the CIFAR datasets [26] with up to 20% training labels randomly perturbed. Following Nakkiran et al. [35], we control model size by increasing the number ω of learned feature maps of each convolutional stage in both model families, following the progression $[\omega, 2\omega, 4\omega, 8\omega]$, for $\omega = 1, \dots 64$. To isolate the role of overparameterization, we remove potential confounders from the optimization process by training all networks with crossentropy loss and SGD with momentum and fixed learning rate, without any explicit regularization (e.g. batch norm, weight decay. Full details in appendix B).

Figure 1 (top) shows the double descent curve for the test error for our experimental setting, with the test error showing the classic U-shaped curve for small models, and a second descent as the degree of parameterization grows further.

Hereafter, we denote with *interpolation threshold* the smallest model width perfectly classifying the training data. Furthermore, we refer to the Lipschitz lower bound (introduced in section 2) as the *empirical Lipschitz constant*. We emphasize that our study focuses on the trends presented by Lipschitz smoothness on the training data, rather than on precisely estimating the true Lipschitz constant of deep networks (which is NP-hard [24, 45]).

Outline of the Paper Section 2 presents our main results, connecting input-smoothness with parameter-space curvature of the loss landscape and model function. Section 3 discusses broader implications of our results. Finally, section 4 discusses related works.

2 Input-Smoothness Follows Double Descent

We begin by introducing the empirical Lipschitz constant for piece-wise linear networks. We consider feed-forward networks $\mathbf{f}(\mathbf{x}, \boldsymbol{\theta}) : \Omega \times \mathbb{R}^p \to \mathbb{R}^K$, composing L affine layers with the continuous piece-wise linear activation ReLU $\phi(x) = \max\{0, x\}$, interpreted as functions

[1] Source code available at https://github.com/magamba/overparameterization

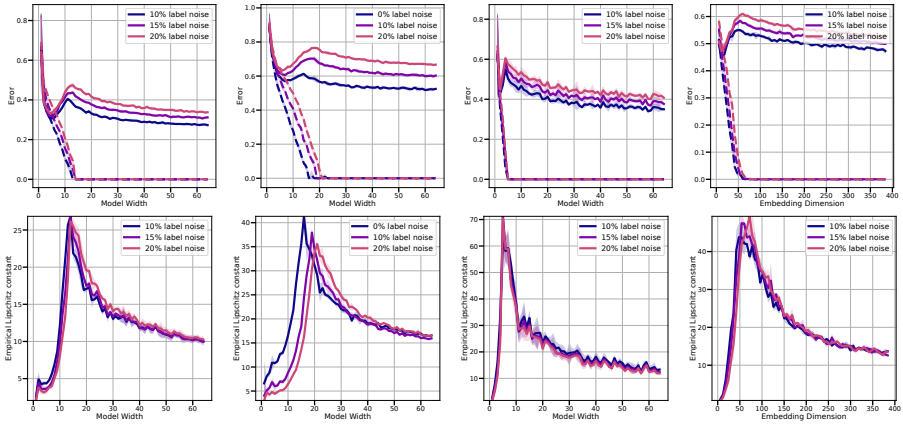

Figure 1: (Top) **Train error** (dashed) and **test error** (solid) for our experimental setting, with the test error undergoing double descent as model size increases. (Left to right) ConvNets trained on CIFAR-10 (left) and CIFAR-100 (mid-left), ResNets trained on CIFAR-10 (mid-right) and Vision Transformers on CIFAR-10 (right). (Bottom) **Empirical Lipschitz constant** for the same models. The Lipschitz lower bound depends non-monotonically on model size, strongly correlating with double descent, showing that overparameterization promotes regularization of the learned model functions via increased local Lipschitz continuity.

$\mathbf{f}(\mathbf{x},\boldsymbol{\theta}) = \boldsymbol{\theta}^L\phi(\boldsymbol{\theta}^{L-1}\phi(\cdots\phi(\boldsymbol{\theta}^1\mathbf{x}+\mathbf{b}^1))+\mathbf{b}^{L-1})+\mathbf{b}^L$, with $\boldsymbol{\theta} = (\mathrm{vec}(\boldsymbol{\theta}^1),\mathbf{b}^1,\ldots,\mathrm{vec}(\boldsymbol{\theta}^L),\mathbf{b}^L)$ representing the vectorized model parameter, and $\mathbf{x}\in\Omega\subseteq\mathbb{R}^d$ the input to the network [2].

For each fixed value of $\boldsymbol{\theta}$, $\mathbf{f}_{\boldsymbol{\theta}}:\mathbb{R}^d\to\mathbb{R}^K$ corresponds to a fixed hypothesis in the space $\mathcal{H}$ of all functions expressible by the network architecture. Each model function $\mathbf{f}_{\boldsymbol{\theta}}$ is itself continuous piece-wise linear, and partitions its input space Ω into disjoint convex polytopes P_ε known as activation regions [31, 40], on each of which a linear function is computed, with $\cup_\varepsilon P_\varepsilon = \Omega$. By piece-wise linearity, one can write $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_\varepsilon \mathbb{1}_{P_\varepsilon(\mathbf{x})}\left[\boldsymbol{\theta}_\varepsilon\mathbf{x}+\mathbf{b}_\varepsilon\right]$, where the indicator function selects the activation region P_ε according to $\mathbf{x}$, and $\boldsymbol{\theta}_\varepsilon$ represents conditioning the factorization $\boldsymbol{\theta}_\varepsilon := \prod_{\ell=1}^L \mathrm{diag}(S_{\mathbf{x}}^\ell)\boldsymbol{\theta}^\ell$ by the binary activation pattern $(S_{\mathbf{x}}^1,\ldots,S_{\mathbf{x}}^L)$ associated with P_ε according to each ReLU activation, dependent on the input $\mathbf{x}$ to the network [3]. Formally, $(S_{\mathbf{x}}^\ell)_i = \mathbb{1}[\boldsymbol{\theta}_i^\ell\mathbf{x}^{\ell-1}+\mathbf{b}_i^\ell > 0]$, where $\boldsymbol{\theta}_i^\ell$ denotes the i:th row of $\boldsymbol{\theta}^\ell$, and $\mathbf{x}^{\ell-1}$ is the input to layer ℓ.

Particularly, for any input $\bar{\mathbf{x}}\in\Omega$, evaluating the Jacobian $\nabla_{\mathbf{x}}\mathbf{f}_{\boldsymbol{\theta}}$ at $\bar{\mathbf{x}}$ yields $\boldsymbol{\theta}_\varepsilon$, i.e. the linear function computed by $\mathbf{f}_{\boldsymbol{\theta}}$ on the activation region ε containing $\bar{\mathbf{x}}$. Hence, given a dataset $\mathcal{D} = \{(\mathbf{x}_n,y_n)\}_{n=1}^N$, and denoting $\varepsilon_n := \varepsilon(\mathbf{x}_n)$, the empirical Lipschitz constant of $\mathbf{f}_{\boldsymbol{\theta}}$ on $\mathcal{D}$ can be estimated by computing the expected operator norm

$$(\mathbb{E}_{\mathcal{D}}\|\nabla_{\mathbf{x}}\mathbf{f}_{\boldsymbol{\theta}}\|_2^2)^{\frac{1}{2}} := \left(\frac{1}{N}\sum_{n=1}^N \sup_{\mathbf{x}:\|\mathbf{x}\|\neq 0}\frac{\|\boldsymbol{\theta}_{\varepsilon_n}\mathbf{x}\|_2^2}{\|\mathbf{x}\|_2^2}\right)^{\frac{1}{2}} \tag{1}$$

representing the expected largest change propagated by the function on activation regions covering $\mathcal{D}$, and can be thought of as a measure of scale of $\mathbf{f}_{\boldsymbol{\theta}}$. Appendix C outlines a procedure for estimating the operator norm in practice via a power method.

[2] typically Ω is a bounded domain, e.g. RGB pixels.

[3] A similar conditioning is applied to compute the bias term $\mathbf{b}_\varepsilon$.

2.1 Input Smoothness of Piece-wise Linear Networks

The empirical Lipschitz constant measures sensitivity of the model function around each training point. In the interpolating regime, it captures smoothness of interpolation.

In Figure 1 (bottom), we compute Equation 1 for deep networks trained in practice and present our main result: *the empirical Lipschitz constant of deep networks is non-monotonic in model size, increasing until the interpolation threshold, and then decreasing afterward, strongly correlating with the test error. The trend is consistent across all architectures, datasets, and noise settings considered.*

Figure 1 (bottom right) extends the finding beyond piece-wise linear networks to Vision Transformers [12, 44] trained on CIFAR-10, whereupon model size is controlled by changing the embedding dimension, as well as the width of MLP layers (see appendix B for details). Analogous observations on machine translation tasks are reported in appendix D.4.

This finding sheds light on the effective complexity of *trained networks* in relation to model size, complementing existing notions of Lipschitz continuity assumed in many theoretical works [2, 25, 29, 34, 46] which miss the observed non-monotonicity, and extending to the double descent setting the relevance of local Lipschitz continuity for generalization. The observed trends highlight a strong correlation between increased relative smoothness of $\mathbf{f}_{\boldsymbol{\theta}}$ and its generalization ability, as well as dependency of the phenomenon on model size.

With the main message of this work established, in the following sections we draw formal connections between the empirical Lipschitz constant and parameter-space regularity (Section 2.2); we discuss implications for the true Lipschitz constant (Figure 2); finally, we present further experiments that offer broader insights on model complexity and double descent (Section 3).

2.2 Connection to Parameter-Space Dynamics

In this section, we connect the empirical Lipschitz constant to parameter-space dynamics of SGD, by studying the relationship between input-space and parameter-space gradients of $\mathbf{f}_{\boldsymbol{\theta}}$. We defer all proofs to appendix F. Let $\mathbf{x}^{\ell} := \phi(\boldsymbol{\theta}^{\ell}\mathbf{x}^{\ell-1} + \mathbf{b}^{\ell})$ denote the output of the ℓ-th layer, for $\ell = 1, \ldots, L$, with $\mathbf{x}^0 := \mathbf{x} \in \Omega$. We begin by noting that linear layers, when composed hierarchically, enjoy a duality between their input and parameters, which ties parameter-space gradients at each layer to gradients w.r.t. its input.

Formally, during backpropagation, computing the gradient $\frac{\partial \mathbf{f}(\mathbf{x},\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^{\ell}} = \frac{\partial \mathbf{f}(\mathbf{x},\boldsymbol{\theta})}{\partial(\boldsymbol{\theta}^{\ell}\mathbf{x}^{\ell-1}+\mathbf{b}^{\ell})}\mathbf{x}^{\ell-1^T}$ entails calculating the upstream gradient $\frac{\partial \mathbf{f}(\mathbf{x},\boldsymbol{\theta})}{\partial(\boldsymbol{\theta}^{\ell}\mathbf{x}^{\ell-1}+\mathbf{b}^{\ell})}$, which also appears in the computation of the partial derivative w.r.t. the ℓ:th layer's input $\frac{\partial \mathbf{f}(\mathbf{x},\boldsymbol{\theta})}{\partial \mathbf{x}^{\ell-1}} = \boldsymbol{\theta}^{\ell^T}\frac{\partial \mathbf{f}(\mathbf{x},\boldsymbol{\theta})}{\partial(\boldsymbol{\theta}^{\ell}\mathbf{x}^{\ell-1}+\mathbf{b}^{\ell})}$. The relationship ties the two gradients, providing the following statement.

Theorem 1. *Let* $\mathbf{f}$ *denote a neural network with a least one hidden layer, with* $\|\boldsymbol{\theta}^1\| > 0$ *and arbitrary weights* $\boldsymbol{\theta}^2, \ldots, \boldsymbol{\theta}^L$. *Let* $x_{\min} := \min_{\mathbf{x}_n \in \mathcal{D}} \|\mathbf{x}_n\|_2$. *Then, parameter-space gradients bound input-space gradients of* $\mathbf{f}$ *from above:*

$$\frac{x_{\min}^2}{\|\boldsymbol{\theta}^1\|_2^2}\mathbb{E}_{\mathcal{D}}\|\nabla_{\mathbf{x}}\mathbf{f}\|_2^2 \leq \mathbb{E}_{\mathcal{D}}\|\nabla_{\boldsymbol{\theta}}\mathbf{f}\|_2^2 . \tag{2}$$

Crucially, the bound highlights an implicit regularization mechanism arising from hierarchical representations, whereby parameter-space gradients control input-space sensitivity

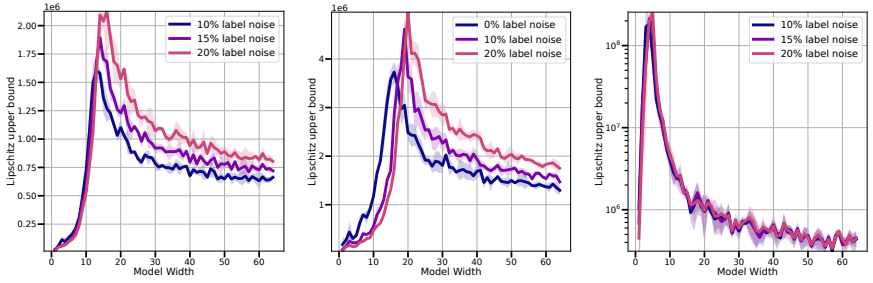

Figure 2: **Upper bound on the true Lipschitz constant**, undergoing double descent as model size increases. From left to right: ConvNets trained on CIFAR-10 (left), CIFAR-100 (middle) and ResNets trained on CIFAR-10 (right).

by bounding the expected norm of input-space gradients $\mathbb{E}_{\mathcal{D}}\|\nabla_{\mathbf{x}}\mathbf{f}\|$, thus regularizing the empirical Lipschitz constant. We note that, while an analogous bound was first observed by Ma and Ying [29] (Theorem 3) for the first layer's preactivation, the authors propose a uniform bound $\mathbb{E}_{\mathcal{D}}\|\nabla_{\boldsymbol{\theta}}\mathbf{f}\| \leq \alpha p$ that linearly increases with the number of model parameters p, with constant α depending on learning rate and batch size. In contrast, we generalize the bound to any layer beyond the first, and study it in connection to double descent, as p varies with network width. Specifically, in section 2.3 we provide an upper bound to Theorem 1 that captures double descent in practical settings.

Interestingly, by recalling that $\nabla_{\mathbf{x}}\mathbf{f}_{\boldsymbol{\theta}} = \prod_{\ell=1}^{L} \operatorname{diag}(S_{\mathbf{x}}^{\ell})\boldsymbol{\theta}^{\ell}$ for $\mathbf{x} \in \Omega$, we note that the empirical Lipschitz constant is intimately tied to the model's parameters, and thus the bound in Theorem 1 controls the expected growth of all layers save for $\boldsymbol{\theta}^1$. Additionally, by noting that the operator norm $\|\operatorname{diag}(S_{\mathbf{x}}^{\ell})\|_2 = 1$, for $\ell = 1, \ldots, L$, the factorization $\nabla_{\mathbf{x}}\mathbf{f}_{\boldsymbol{\theta}} = \prod_{\ell=1}^{L} \operatorname{diag}(S_{\mathbf{x}}^{\ell})\boldsymbol{\theta}^{\ell}$ allows to derive an upper bound on the *true Lipschitz constant* $\operatorname{Lip}(\mathbf{f})$ of $\mathbf{f}_{\boldsymbol{\theta}}$ on the whole domain Ω.

$$\operatorname{Lip}(\mathbf{f}) := \sup_{\mathbf{x}\in\Omega} \|\nabla_{\mathbf{x}}\mathbf{f}_{\boldsymbol{\theta}}\| \leq \sup_{\mathbf{x}\in\Omega} \prod_{\ell=1}^{L} \|\operatorname{diag}(S_{\mathbf{x}}^{\ell})\boldsymbol{\theta}^{\ell}\| \leq \sup_{\mathbf{x}\in\Omega} \prod_{\ell=1}^{L} \|\boldsymbol{\theta}^{\ell}\| = \prod_{\ell=1}^{L} \|\boldsymbol{\theta}^{\ell}\|_2 \tag{3}$$

Figure 2 presents the upper bound on the true Lipschitz constant for ConvNets trained on CIFAR-10, CIFAR-100, and ResNets trained on CIFAR-10. Similarly to the empirical Lipschitz lower bound, the upper bound closely follows double descent for the test error, peaking near the interpolation threshold. We note that, since the upper bound is independent of the binary activation pattern of ReLU, it bounds global worst-case sensitivity of the network on the whole domain Ω of $\mathbf{f}$, suggesting that the non-monotonic dependency of Lipschitz continuity on model size holds also beyond the training set $\mathcal{D}$. This observation is substantiated experimentally in section 3.

We conclude this section by extending Theorem 1 to exponential losses $\mathcal{L}(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y})$, such as crossentropy and Mean Squared Error (MSE).

Corollary 1. *Consider the composition of a loss function $\mathcal{L}$ with a neural network $\mathbf{f}$ with a least one hidden layer, with $\|\boldsymbol{\theta}^1\| > 0$ and arbitrary weights $\boldsymbol{\theta}^2, \ldots, \boldsymbol{\theta}^L$. Then,*

$$\frac{x_{\min}^2}{\|\boldsymbol{\theta}^1\|_2^2} \mathbb{E}_{\mathcal{D}}\|\nabla_{\mathbf{x}}\mathcal{L}\|_2^2 \leq \mathbb{E}_{\mathcal{D}}\|\nabla_{\boldsymbol{\theta}}\mathcal{L}\|_2^2 . \tag{4}$$

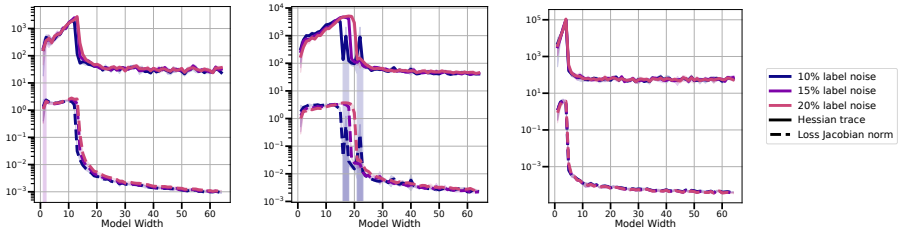

Figure 3: (Top) **Loss Jacobian norm** $\|\nabla_{\mathbf{x}}\mathcal{L}\|$ in *input space* and **Mean loss curvature** (Hessian trace) in *parameter space*. (From left to right) ConvNets trained on CIFAR-10 (left), CIFAR-100 (middle) and ResNets trained on CIFAR-10 (right). In all settings, mean parameter-space curvature strongly correlate with double descent, peaking at the interpolation threshold, and highlighting a nonlinear dependence on model size.

Importantly, the proof of Corollary 1 is based on the factorizations $\nabla_{\mathbf{x}}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial \mathbf{f}} \nabla_{\mathbf{x}} \mathbf{f}_{\boldsymbol{\theta}}$, and $\nabla_{\boldsymbol{\theta}}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial \mathbf{f}} \nabla_{\boldsymbol{\theta}} \mathbf{f}_{\boldsymbol{\theta}}$, with the common factor $\frac{\partial \mathcal{L}}{\partial \mathbf{f}}$ contributing a bounded monotonic rescaling of the model function gradients (visualized in Figure 11 in the appendix). Thus, the double descent trend of the empirical Lipschitz constant can be observed also in the loss landscape, by tracking the input-space loss Jacobian $\nabla_{\mathbf{x}}\mathcal{L}$. Figure 3 presents non-monotonic trends for input-space loss Jacobian norm, as model size increases.

In the following, building on Corollary 1, we draw an explicit connection between $\|\nabla_{\mathbf{x}}\mathcal{L}\|_2$ and the parameter-space geometry of the loss landscape.

2.3 Connection to Parameter-Space Curvature

To study the implications of Theorem 1 and Equation 4, we consider the dynamics of SGD in proximity of a minimum $\boldsymbol{\theta}^* \in \mathbb{R}^p$ of the loss $\mathcal{L}$. We adopt a linear stability perspective [22, 47], and approximate the loss in a neighbourhood of $\boldsymbol{\theta}^*$ via a second-order Taylor expansion in $\boldsymbol{\theta}$

$$\mathbb{E}_{\mathcal{D}}\mathcal{L}(\boldsymbol{\theta}, \mathbf{x}, y) = \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T H (\boldsymbol{\theta} - \boldsymbol{\theta}^*) + o(\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|^2) \tag{5}$$

where the first order term vanishes at the critical point $\boldsymbol{\theta}^*$, as does the zeroth order term for interpolating models, and H represents the expected Hessian of the training loss.

In the next result, we derive upper bounds on *input-space* smoothness of the loss (Equation 4), in connection to *parameter-space* geometry, focusing on the mean squared error $\mathcal{L} = \frac{1}{2N}\sum_{n=1}^{N}(f_{\boldsymbol{\theta}}(\mathbf{x}_n) - y_n)^2$. In the following, let $\mathcal{L}(\boldsymbol{\theta}) := \mathbb{E}_{\mathcal{D}}[\mathcal{L}(\boldsymbol{\theta}, x, y)]$.

Theorem 2. *Let $\boldsymbol{\theta}^*$ be a critical point for the loss $\mathcal{L}(\boldsymbol{\theta}, \mathbf{x}, y)$ on $\mathcal{D}$. Let $\mathbf{f}_{\boldsymbol{\theta}}$ denote a neural network with at least one hidden layer, with $\|\boldsymbol{\theta}^1\| > 0$. Then,*

$$\frac{x_{\min}^2}{\|\boldsymbol{\theta}^1\|_2^2}\mathbb{E}_{\mathcal{D}}\|\nabla_{\mathbf{x}}\mathcal{L}\|_2^2 \leq 2\mathcal{L}_{\max}(\boldsymbol{\theta})\,\Delta(\mathcal{L}(\boldsymbol{\theta})) + o(\mathcal{L}(\boldsymbol{\theta})) \tag{6}$$

with $\Delta(\mathcal{L}(\boldsymbol{\theta})) := \mathrm{tr}\,(H)$ denoting the Laplace operator, $H := \mathbb{E}_{\mathcal{D}}[\frac{\partial^2 \mathcal{L}}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T}]$ denoting the expected parameter-space Hessian of $\mathcal{L}$, and $\mathcal{L}_{\max}(\boldsymbol{\theta}) := \max_{(\mathbf{x}_n, y_n) \in \mathcal{D}} \mathcal{L}(\boldsymbol{\theta}, \mathbf{x}_n, y_n)$.

Theorem 2 links input-space smoothness of the loss to the geometry of the loss landscape in parameter space, via mean curvature $\Delta(\mathcal{L}(\boldsymbol{\theta}))$ in a neighbourhood of $\boldsymbol{\theta}^*$.

Figure 3 shows mean curvature of the loss landscape (solid line) in parameter space for our experimental setup (see appendix C for algorithmic details), as well as input-space smoothness of the loss (dashed line). Mean curvature mirrors the *input-space* loss Jacobian as model width increases, peaking near the interpolation threshold and decreasing afterward. This substantiates the bound in Equation 6, and provides a characterization of the empirical Lipschitz constant in the loss landscape in terms of fundamental quantities in parameter space (Hessian trace). Figure 6 complements our observations, by tracking the largest and smallest non-zero Hessian eigenvalues in parameter space, and showing that both quantities track double descent.

Connection to Stochastic Noise SGD is known to fluctuate around critical points due to stochastic noise arising from the discretization of the dynamic (finite learning rate) [33], as well as the use of mini-batches to estimate model gradients [43, 50]. At iteration t, an estimate of the noise $\boldsymbol{\varepsilon}_t$ is given by $\boldsymbol{\varepsilon}_t = \frac{1}{B}\sum_{b=1}^{B} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t, \mathbf{x}_{\xi_b}, y_{\xi_b}) - \mathbb{E}_{\boldsymbol{\xi}} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t)$, where B denotes the batch size, and the indices $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_B)$ represent sampling of mini-batches.

For models trained without weight decay, the parameter-space gradient covariance $C = \mathbb{E}_{\boldsymbol{\xi}}[\boldsymbol{\varepsilon}_t \boldsymbol{\varepsilon}_t^T]$ is closely related to the mean Hessian H [50], producing the following corollary.

Corollary 2. *Let $\boldsymbol{\theta}^*$ be a critical point for the loss $\mathcal{L}(\boldsymbol{\theta}, \mathbf{x}, y)$ on $\mathcal{D}$. Let $\mathbf{f}_{\boldsymbol{\theta}}$ denote a neural network with at least one hidden layer, with $\|\boldsymbol{\theta}^1\| > 0$. Then,*

$$\frac{x_{\min}^2}{\|\boldsymbol{\theta}^1\|_2^2} \mathbb{E}_{\mathcal{D}} \|\nabla_{\mathbf{x}} \mathcal{L}\|_2^2 \leq \operatorname{tr}(S) + o(\mathcal{L}(\boldsymbol{\theta})) \tag{7}$$

with $S = C + \frac{1}{B} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})^T$ denoting the gradient uncentered covariance.

Figure 7 tracks the largest principal component of C for increasing model size, showing that stochastic noise peaks near the interpolation threshold, and then decreases in the overparameterized regime. In section D.3 we conclude our theoretical analysis by discussing the role of hyperparameters in controlling mean curvature and, in turn, input-space smoothness.

Summary By relating the empirical Lipschitz constant to the mean curvature of parameter space, we highlight a mechanism by which optimization implicitly controls sensitivity of $\mathbf{f}_{\boldsymbol{\theta}}$ on the training data $\mathcal{D}$ in proximity of a critical point $\boldsymbol{\theta}^*$.

3 Implications for Implicit Regularization

We conclude our study by exploring broader implications of the trends observed in section 2. We begin by studying the empirical Lipschitz constant throughout epochs, then explore implications of our findings for understanding effective complexity of trained networks.

Overparameterization Accelerates Interpolation Figure 4 (top) shows the empirical Lipschitz constant of ConvNets (left) and ResNet18s (middle) trained on CIFAR-10 with 20% noisy training labels, for representative model widths, together with the respective training error (bottom). Heatmaps for all model widths are presented in appendix D.5, connecting to

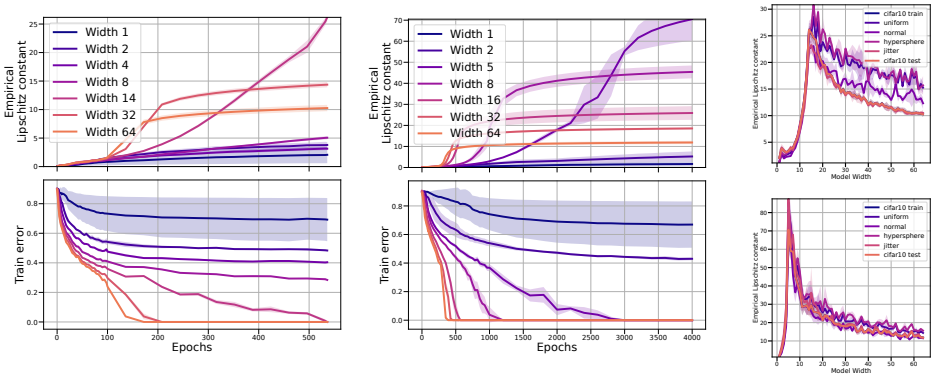

Figure 4: (Left and middle panels) **Empirical Lipschitz constant over epochs** (top) and **Train error** (bottom) for ConvNets (left) and ResNets (middle) trained on CIFAR-10. (Right) Empirical Lipschitz constant for the same models on test data and random noise, for ConvNets (right, top panel) and ResNets (right, bottom panel).

the test error in Figure 9. We recall that the model-wise interpolation threshold [4] denotes the smallest model width ω_0 that perfectly classifies the training set, in our experiment corresponding to $\omega_0 = 14$ for ConvNets, and $\omega_0 = 5$ for ResNets.

During training, we observe three distinct behaviours. Small models ($\omega \ll \omega_0$) are unable to interpolate the entire training set, and their training error as well as empirical Lipschitz constant quickly plateau, remaining stable therefrom. Increasing size among small models reduces their training error, and correspondingly increases the empirical Lipschitz constant.

At the same time, models near the interpolation threshold ω_0 – peaking in test error and empirical Lipschitz constant (cfr. Figure 1) – are able to achieve interpolation, *only when given considerable training budget*. Correspondingly, the empirical Lipschitz constant monotonically increases over training as the training error is reduced, resulting in models achieving worst sensitivity and worst test error. In contrast, consistently with the double descent trends reported in section 2, large models ($\omega \gg \omega_0$) are able to quickly interpolate the training set, with the largest models requiring fewer epochs to achieve interpolation.

The seemingly unbounded empirical Lipschitz constant of models near ω_0 suggests that the observations reported in Hardt et al. [20] – for which prolonged training budgets may hurt generalization performance – are pertaining only to models near the threshold. In fact, larger models can be trained for considerably long without a comparable increase in complexity.

Overparameterization Constrains Complexity Referring again to Figure 4, we now consider implications for effective complexity of trained networks. First, since model weights are typically initialized to small values around zero [17, 21], the empirical Lipschitz constant of all models is close to zero at the beginning of training. This corresponds to each model expressing a very simple function (low empirical Lipschitz constant), albeit with low generalization performance (typically close to random chance). Second, during training, fitting the dataset requires all models' Lipschitz constant to grow, with corresponding increase in model complexity (as measured by Equation 2). When zero error is reached ($\omega \geq \omega_0$), the empirical Lipschitz constant approximately plateaus, thereafter only slowly increasing over epochs. Recalling that large models interpolate faster, this finding suggest that large models may achieve interpolation via least meaningful deviation from initialization, realizing an

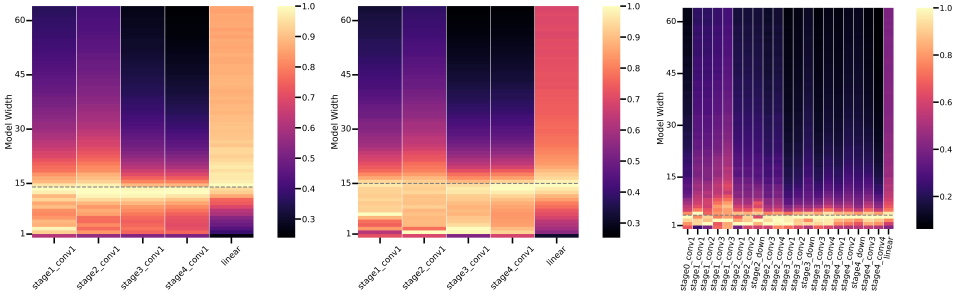

Figure 5: **Distance from initialization for each layer** of ConvNets trained on noisy CIFAR-10 (left), CIFAR-100 (middle), and ResNet18s trained on noisy CIFAR-10 (right). For each ConvNet and most ResNet layers, distance from initialization follows double descent, peaking at the interpolation threshold (dashed), suggesting global boundedness of the model function beyond training data for large models.

overall smooth function even beyond the training set.

To assess our hypothesis, we study the normalized distance from initialization $\frac{\|\boldsymbol{\theta}_T^\ell - \boldsymbol{\theta}_0^\ell\|_F}{\|\boldsymbol{\theta}_0^\ell\|_F}$ of each layer ℓ, with $\boldsymbol{\theta}_0^\ell$ and $\boldsymbol{\theta}_T^\ell$ respectively denoting the layers' weights at initialization and convergence. Figure 5 presents distance from initialization (colour) as model width increases (y-axis), for each layer (x-axis), for ConvNets (left) and ResNets (right) trained on CIFAR-10 with 20% label noise, and ConvNets on CIFAR-100 with no label noise (middle). For almost all layers, the quantity follows double descent as model width increases, peaking near the interpolation threshold (dashed line), and matching the epoch-wise trend reported in Figure 4.

This exciting finding supports our interpretation that faster interpolation, as promoted by overparameterization, results in model functions which are overall low-complexity, due to least (but meaningful) deviation from initialization. Our findings extend Neyshabur et al. [37], who initially reported that distance from initialization decreases for overparameterized models. Importantly, we show that the statistic is non-monotonic in model size, and that it strongly correlates with double descent for the test error. Together with the observed low mean curvature of large models shown in section 2.3, this finding shares potential connection to the linear mode connectivity phenomenon [14], by which low-loss paths that connect solutions obtained by optimization of the same model and task have been found in practice. Indeed, deeper layers of large models can be rewound to their value at initialization without considerably affecting model performance [8, 49], supporting our observations.

Bounded Complexity Beyond Training Data To conclude, in Figure 4 (right panel) we estimate the empirical Lipschitz constant of ConvNets (top) and ResNets (bottom) trained on CIFAR-10, probing the networks by computing Equation 1 on unseen test data as well as on random noise lying far from the support of the data distribution (experimental details in appendix E). Intriguingly, the empirical Lipschitz constant remains bounded even far from $\mathcal{D}$, and the model-wise trend follows double descent, peaking at the interpolation threshold. This finding further strengthens the view that reduced distance from initialization via acceleration may essentially control complexity of the *whole* model function.

4 Related Work and Discussion

Deep networks are able to express a rich family of functions as their model size increases [9, 42, 48]. However, the complexity of generalizing models appears to be constrained in practice [36, 37, 49]. Developing a formal characterization of the phenomenon is still a challenging open problem. Theoretical studies hinge upon finding a parameterization of the hypothesis class of trained networks that meaningfully constrains their expressivity. Importantly, several works rely on uniform bounds on the Lipschitz constant to constrain model function variation [2, 25, 29, 34, 46]. Moreover, in practical settings, explicitly regularizing the Lipschitz constant yields improved performance [18, 32, 38].

Recently, the study of the Lipschitz constant has received renewed attention, with Bubeck and Sellke [6] prescribing overparameterization as a necessary condition for smooth interpolation, for a generic class of learners. Our work corroborates their findings, by also presenting an upper bound on the constant in relation to the geometry of the loss landscape.

While tightly estimating the Lipschitz constant is NP-hard for deep networks [24, 45], we focus on complexity w.r.t. training data. Crucially, our Theorem 1 extends a uniform bound on input-space sensitivity [29] with a novel one that experimentally captures double descent.

Interestingly, a concurrent work uses Sobolev seminorms of ReLU networks on the training set to propose a complexity measure [11]. In line with our findings, their proposed measure mirrors the test error. Our works differ in that they focus on studying regularization of the metric, while instead we build a theoretical connection to several fundamental quantities capturing double descent in connection to the geometry of the loss landscape.

5 Conclusions

We carry out an extensive study of the empirical Lipschitz constant of deep networks undergoing double descent, presenting implications for Lipschitz continuity and its implicit regularization via overparameterization. By building a theoretical connection with the loss landscape geometry, we present several correlates of double descent in terms of fundamental notions, that we hope will inspire further theoretical studies. We isolate two important quantities – namely loss landscape curvature and distance of parameters from initialization – respectively controlling optimization dynamics around a critical point and bounding model function complexity beyond training data. We believe understanding the structure and singularity of the overparameterized mapping from parameters to model functions is a fundamental open problem, which might reveal the true latent factors driving generalization.

Acknowledgments

The authors thank David Lopez-Paz for many fruitful discussions on smoothness priors of deep networks and Kevin Scaman for technical feedback on an earlier draft of the paper. The work was partially funded by Swedish Research Council project 2017-04609. Scientific computation was enabled by the supercomputing resource Berzelius provided by National Supercomputer Centre at Linköping University and the Knut and Alice Wallenberg foundation, as well as by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at Alvis partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

References

[1] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 244–253. PMLR, 10–15 Jul 2018.

[2] 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.

[3] Peter L. Bartlett, Philip M. Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117 (48):30063–30070, 2020.

[4] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[5] Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950.

[6] Sébastien Bubeck and Mark Sellke. A universal law of robustness via isoperimetry. *Advances in Neural Information Processing Systems*, 34, 2021.

[7] Mauro Cettolo, Christian Girardi, and Marcello Federico. Wit3: Web inventory of transcribed and translated talks. In *Proceedings of the Conference of European Association for Machine Translation (EAMT)*, pages 261–268, 2012.

[8] Niladri S Chatterji, Behnam Neyshabur, and Hanie Sedghi. The intriguing role of module criticality in the generalization of deep networks. In *International Conference on Learning Representations*, 2020.

[9] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[10] Alexandre De Brebisson and Pascal Vincent. An exploration of softmax alternatives belonging to the spherical loss family. *International Conference on Learning Representations*, 2016.

[11] Benoit Dherin, Michael Munn, Mihaela Rosca, and David GT Barrett. Why neural networks find simple solutions: the many regularizers of geometric complexity. *arXiv preprint arXiv:2209.13083*, 2022.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[13] Matteo Gamba, Adrian Chmielewski-Anders, Josephine Sullivan, Hossein Azizpour, and Mårten Björkman. Are all linear regions created equal? In *International Conference on Artificial Intelligence and Statistics*, pages 6573–6590. PMLR, 2022.

[14] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018.

[15] Mario Geiger, Stefano Spigler, Stéphane d'Ascoli, Levent Sagun, Marco Baity-Jesi, Giulio Biroli, and Matthieu Wyart. Jamming transition as a paradigm to understand the loss landscape of deep neural networks. *Physical Review E*, 100(1):012115, 2019.

[16] Stuart Geman, Elie Bienenstock, and René Doursat. Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, 4(1):1–58, 01 1992. ISSN 0899-7667.

[17] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[18] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021.

[19] Boris Hanin and David Rolnick. Deep relu networks have surprisingly few activation patterns. In *Advances in Neural Information Processing Systems*, pages 359–368, 2019.

[20] Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1225–1234, New York, New York, USA, 20–22 Jun 2016. PMLR.

[21] 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.

[22] Yohei Hosoe and Tomomichi Hagiwara. On second-moment stability of discrete-time linear systems with general stochastic dynamics. *IEEE Transactions on Automatic Control*, 67(2):795–809, 2022.

[23] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

[24] Matt Jordan and Alexandros G Dimakis. Exactly computing the local lipschitz constant of relu networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7344–7353. Curran Associates, Inc., 2020.

[25] Kenji Kawaguchi, Zhun Deng, Kyle Luh, and Jiaoyang Huang. Robustness implies generalization via data-dependent generalization bounds. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 10866–10894. PMLR, 17–23 Jul 2022.

[26] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

[27] Daniel LeJeune, Randall Balestriero, Hamid Javadi, and Richard G Baraniuk. Implicit rugosity regularization via data augmentation. *arXiv preprint arXiv:1905.11639*, 2019.

[28] Kangqiao Liu, Liu Ziyin, and Masahito Ueda. Noise and fluctuation of finite learning rate stochastic gradient descent. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7045–7056. PMLR, 18–24 Jul 2021.

[29] Chao Ma and Lexing Ying. On linear stability of sgd and input-smoothness of neural networks. *Advances in Neural Information Processing Systems*, 34:16805–16817, 2021.

[30] Matouš Macháček and Ondřej Bojar. Results of the wmt14 metrics shared task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 293–301, 2014.

[31] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2924–2932, 2014.

[32] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[33] Takashi Mori, Liu Ziyin, Kangqiao Liu, and Masahito Ueda. Power-law escape rate of SGD. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15959–15975. PMLR, 17–23 Jul 2022.

[34] Vaishnavh Nagarajan and Zico Kolter. Deterministic pac-bayesian generalization bounds for deep networks via generalizing noise-resilience. In *International Conference on Learning Representations*, 2018.

[35] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2019.

[36] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. In *International Conference on Learning Representations Workshop Track*, 2015.

[37] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. The role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations*, 2018.

[38] Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018.

[39] Barak A Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6 (1):147–160, 1994.

[40] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *International Conference on Machine Learning*, pages 2847–2854, 2017.

[41] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. Adversarial training is a form of data-dependent operator norm regularization. *Advances in Neural Information Processing Systems*, 33:14973–14985, 2020.

[42] Matus Telgarsky. Benefits of depth in neural networks. In *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 1517–1539, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.

[43] Valentin Thomas, Fabian Pedregosa, Bart van Merriënboer, Pierre-Antoine Manzagol, Yoshua Bengio, and Nicolas Le Roux. On the interplay between noise and curvature and its effect on optimization and generalization. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3503–3513. PMLR, 26–28 Aug 2020.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

[46] Colin Wei and Tengyu Ma. Data-dependent sample complexity of deep neural networks via lipschitz augmentation. *Advances in Neural Information Processing Systems*, 32, 2019.

[47] Lei Wu and Chao Ma. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. *Advances in Neural Information Processing Systems*, 31, 2018.

[48] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations*, 2018.

[49] Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal? *ICML Workshop Deep Phenomena*, 2019.

[50] Liu Ziyin, Kangqiao Liu, Takashi Mori, and Masahito Ueda. Strength of minibatch noise in SGD. In *International Conference on Learning Representations*, 2022.

A Organization of the Appendix

- Section B fully details our experimental setup, as well as the hardware infrastructure used for our experiments.
- Section C presents the algorithms used for estimating the empirical Lipschitz constant and to measure parameter space curvature.
- Section D presents additional figures supporting the experiments in sections 2 and 3.
  - Section D.1 discusses an upper bound on the true Lipschitz constant of piece-wise linear networks, undergoing double descent as mode size increases.
  - Section D.2 presents additional results on parameter-space curvature of the loss landscape under double descent.
  - Section D.3 discusses our results in relationship to training hyperparameters.
  - Section D.4 extends our results to Transformers trained on machine translation tasks.
  - Section D.5 extends the epoch-wise trends reported for selected models in Figure 4 to all model widths considered in our study.
  - Section D.6 validates Theorem 2 in the interpolating regime for the models considered.
- Section E describes the distributions used for generating random validation data for Figure 4.
- Finally, section F presents proofs of the formal statements appearing in section 2.

B Experimental Setup

We train a family of ConvNets composed of 4 convolutional stages – each corresponding to a `[Conv, ReLU]` block followed by maxpooling with stride 2 – and 1 dense classification layer. We also train a family of ResNet18s [21] without batch normalization layers. Both network architectures are composed of 4 convolutional stages, in which each spatial dimension is reduced by factor of 2 and the number of learned feature maps doubles. More precisely, the convolutional stages respectively follow the progression $[\omega, 2\omega, 4\omega, 8\omega]$, where ω is the base width of the network, i.e. the number of feature maps learned at the first layer.

In our experiments, following Nakkiran et al. [35], we vary the base width in the range $\omega = 1, \ldots, 64$. By controlling the network size through the network width, we produce a range of models presenting model-wise double descent in the test error, which captures the essence of the *benign overfitting* phenomenon[3] observed for large interpolating networks, while also presenting *malign overfitting* for models near the interpolation threshold. Furthermore, controlling model size through base width allows us to keep the network depth fixed, and focus our study on effective complexity of fixed-depth networks, for two network architecture families (ConvNets and ResNets).

To tune hyperparameters, we take a random validation split of size 1000 from each CIFAR training set. We train all networks with SGD with momentum 0.9, batch size 128, and fixed learning rate, set at $\eta = 5\text{e}-3$ for the ConvNets and $\eta = 1\text{e}-4$ for the ResNets. We train the

Table 1: Number of model parameters p for representative widths ω on CIFAR-10. Models near the interpolation threshold are marked in bold.

ω	ConvNet	ResNet18
1	510	2,902
2	1,766	11,242
4	6,546	**44,266**
8	25,178	175,690
16	**98,730**	700,042
32	390,986	2,794,762
64	1,556,106	11,168,266

ConvNets for 500 epochs, and the ResNets for 4000 epochs. To stabilize prolonged training, we use learning rate warmup over the first 5 epochs of training, starting from a learning rate $\eta_0 = 10^{-1} \times \eta$.

Transformers on Machine Translation tasks We also train multi-head attention-based Transformers [44] for neural machine translation tasks. We vary model size by controlling the embedding dimension d_e, as well as the width h of all fully connected layers, which we set to $h = 4d_e$ following the architecture described in Vaswani et al. [44]. We train the transformer networks on the WMT'14 En-Fr task [30], as well as ISWLT'14 De-En [7]. The training set of WMT'14 is reduced by randomly sampling 200k sentences, fixed for all models. The networks are trained for 80k gradient steps, to optimize per-token perplexity, with 10% label smoothing, and no dropout, gradient clipping or weight decay.

Hardware specifications Our codebase is implemented in Pytorch version 1.11, running on a local cluster equipped with NVIDIA A100 GPUs with 40GB onboard memory. Our experiments involve training 64 ConvNets and ResNets (each corresponding to a base width ω) for up to 4000 epochs, producing 72 model checkpoints per network. We use 3 random seeds for the ConvNets and 5 for the ResNets, controlling network initialization and the shuffling and sampling of mini-batches from the training set. We use a dedicated random seed for generating the validation split used for hyperparameter tuning, fixed for all networks, as well as a fixed seed for corrupting the CIFAR training labels. The empirical Lipschitz constant is estimated and averaged on every training point for each of the reported configurations.

Number of model parameters Our main empirical finding is that, while network size increases – causing uniform upper bounds like Ma and Ying [29] (Theorem 3) to monotonically increase – the empirical Lipschitz constant of the models decreases past the interpolation threshold. To better frame our observations, we report in Table 1 the number of parameters for a few representative models in our experiments.

C Operator Norm Estimation

For linear operators $\mathbf{A} : (\mathbb{R}^d, \|\cdot\|_p) \to (\mathbb{R}^K, \|\cdot\|_q)$, the operator norm is defined as

$$\|A\|_{\text{op}} := \sup_{\mathbf{x}:\|\mathbf{x}\|_p \neq 0} \frac{\|A\mathbf{x}\|_q}{\|\mathbf{x}\|_p}, \tag{8}$$

where the norms $\|\cdot\|_p$ and $\|\cdot\|_q$ are respectively taken in input and logit space. Crucially, if $p = q = 2$, then the operator norm can be estimated by computing the largest singular value of

A. For any data point $\bar{\mathbf{x}} \in \mathbb{R}^d$, evaluating the Jacobian at $\bar{\mathbf{x}}$ yields $\nabla_{\mathbf{x}}\mathbf{f}(\mathbf{x},\boldsymbol{\theta})|_{\mathbf{x}=\bar{\mathbf{x}}} = \boldsymbol{\theta}_{\varepsilon}$, i.e. the linear function computed by $\mathbf{f}$ on the activation region ε of $\bar{\mathbf{x}}$. Hence, at each point, $\|\boldsymbol{\theta}_{\varepsilon}\|_{\text{op}}$, provides an estimate of worst-case sensitivity of the corresponding linear "piece" of $\mathbf{f}$. We note that, while the supremum $\|\mathbf{A}\|_{\text{op}}$ may not be attained within the activation region of $\bar{\mathbf{x}}$, the operator norm upper bounds worst-case sensitivity within the region. Furthermore, activation regions neighbouring training data tend to compute approximately the same linear function [13, 41].

Computing the operator norm Computing the operator norm of $\boldsymbol{\theta}_{\varepsilon} \in \mathbb{R}^{K\times d}$ entails two steps. First, computing the gradient $\nabla_{\mathbf{x}}\mathbf{f}|_{\mathbf{x}=\bar{\mathbf{x}}} = \boldsymbol{\theta}_{\varepsilon}$ (via automatic differentiation), and then estimating its largest singular value. To perform the latter, we use a standard power method. Starting at iteration $t=0$ with randomly initialized vectors $\tilde{\mathbf{u}}_0 \in \mathbb{R}^K$, $\tilde{\mathbf{v}}_0 \in \mathbb{R}^d$, and corresponding normalized vectors $\mathbf{u}_0 = \frac{\tilde{\mathbf{u}}_0}{\|\tilde{\mathbf{u}}_0\|_q}$, $\mathbf{v}_0 = \frac{\tilde{\mathbf{v}}_0}{\|\tilde{\mathbf{v}}_0\|_p}$, at step t we compute

$$\begin{aligned} \tilde{\mathbf{u}}_t &\leftarrow \nabla_{\mathbf{x}}\mathbf{f}\,\mathbf{v}_{t-1} \\ \tilde{\mathbf{v}}_t &\leftarrow \mathbf{u}_t^T\nabla_{\mathbf{x}}\mathbf{f} \\ \sigma_t &\leftarrow \mathbf{u}_t^T\nabla_{\mathbf{x}}\mathbf{f}\,\mathbf{v}_t \end{aligned} \tag{9}$$

with σ_t storing the largest singular value at convergence, defined based on a relative tolerance $1\text{e}-6$ on the size of the increments of σ_t.

In our experiments, we estimate the Lipschitz constant of the network by its empirical constant, $\mathbb{E}_{\mathcal{D}}\|\nabla_{\mathbf{x}}\mathbf{f}|_{\mathbf{x}=\mathbf{x}_n}\|_{\text{op}}$, for all training points $\mathbf{x}_n \in \mathcal{D}$. We extend the empirical Lipschitz constant estimation to validation data in Figure 4.

C.1 Hessian Eigenvalue Estimation

The power method detailed in section C can be used to estimate the largest eigenvalue of the parameter-space loss Hessian (Figure 6), as well as the first principal component of the gradient noise covariance (Figure 7). Importantly, for large networks, direct computation of any of the two matrices is infeasible due to the large number of parameters. Instead, we use efficient Jacobian-vector products for estimating the noise covariance (which entails accumulating the true gradient $\mathbb{E}_{\boldsymbol{\xi}}\nabla_{\boldsymbol{\theta}}\mathcal{L}$ at each iteration of the algorithm. For the Hessian matrix, Jacobian-vector products can be turned into Hessian-vector products using Pearlmutter's trick [39].

C.2 Hessian Trace Estimation

To estimate the Hessian trace in Figure 3, we use Hutchinson's algorithm [23], which provides an unbiased estimator of the trace. At each iteration t, the algorithm generates a set of V random test vectors, $\mathbf{v}_n \in \mathbb{R}^p$ with zero mean $\mathbb{E}\mathbf{v}_n = \frac{1}{p}\sum_{i=1}^{p} v_n^i = 0$ and variance $\mathbb{E}[\mathbf{v}_n\mathbf{v}_n^T] = I_p$, by sampling each $\mathbf{v}_n$ from the Rademacher distribution. At iteration t, the algorithm computes $\text{tr}_t = \frac{1}{V}\sum_{n=1}^{V}\mathbf{v}_n^T H\mathbf{v}_n$, where H is the expected loss Hessian. Notably, the trace is obtained by computing $\frac{1}{V}\sum_{i=1}^{p}\mathbf{v}_n^T H\mathbf{v}_n = \text{tr}\,(\mathbf{v}_n^T H\mathbf{v}_n)$, where the Hessian is never instantiated and is implicitly computed via Hessian-vector products [39]. In our work, we estimate the trace using $V=100$ test vectors.

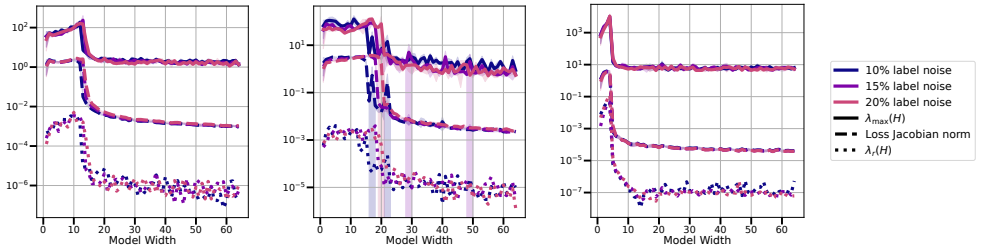

Figure 6: **Maximum and minimum curvature** for the loss in parameter space, and **input-space loss Jacobian norm**. From left to right: ConvNets trained on CIFAR-10 (left), CIFAR-100 (middle) and ResNets trained on CIFAR-10 (right). In all settings, minimum and maximum parameter-space curvature strongly correlate with double descent, peaking at the interpolation threshold, and highlighting a nonlinear dependence on model size.

D Additional Experiments

D.1 Upper Bounding the Lipschitz Constant

We complement our analysis of the empirical Lipschitz lower bound of Equation 1 by studying an upper bound on the true Lipschitz constant $\mathrm{Lip}(\mathbf{f})$, described by Equation 3. Figure 2 presents the upper bound for ConvNets trained on CIFAR-10, CIFAR-100, and ResNets trained on CIFAR-10. Similarly to the empirical Lipschitz lower bound, the upper bound closely follows double descent for the test error, peaking near the interpolation threshold. We note that, since the upper bound is independent of the binary activation pattern of ReLU, it captures global worst-case sensitivity of the network on the whole domain Ω of $\mathbf{f}$, suggesting that the non-monotonic dependency of Lipschitz continuity on model size holds also beyond the training set $\mathcal{D}$. This observation is substantiated by experimentally extending the lower Lipschitz bound in Equation 1 to validation as well as random data in Figure 4, as well as by observing that distance from initialization of trained weights also undergoes double descent (Figure 5). Together, with Theorem 2, these observations suggest that the main factor controlling double descent when the number of model parameters varies is the loss landscape curvature, and which in turn controls input-space sensitivity on the training set through the empirical Lipschitz lower bound. We explore parameter-space curvature in more detail in the next section.

D.2 Parameter Space Curvature

Theorem 2 provides a bound on input-space sensitivity via mean curvature of the loss in parameter space, connecting parameter-space dynamics to input-space sensitivity under double descent. In Figure 6, we explore parameter-space curvature in more detail, by plotting the largest and smallest non-zero Hessian eigenvalues, together with the input-space loss Jacobian norm studied in Corollary 1. We observe that maximum and minimum *parameter space* curvature mirror *input space* sensitivity, as measured by the loss Jacobian norm, peaking near the interpolation threshold, and then decreasing. Our observations support the hypothesis that overparameterization non-monotonically controls flatness of the parameter space, which

in turn controls sensitivity of the model function.

D.3 Mean Curvature, Stochastic Noise and Linear Stability

In this section, we study Theorem 2 in relation to the training dynamics in proximity of a critical point $\boldsymbol{\theta}^*$. Finally, we discuss the influence of training hyperparameters on curvature.

First, we draw a connection between the mean loss Hessian H and gradient noise covariance C, as defined in Corollary 2. Then, we study reachability of the critical point $\boldsymbol{\theta}^*$ by SGD, in relation to training hyperparameters. In turn, this allows us to draw a connection between hyperparameters, their influence on mean curvature, and input-space sensitivity.

At iteration t, the update rule of SGD with batch size B, and learning rate η, is given by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\eta}{B} \sum_{b=1}^{B} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t, \mathbf{x}_{\xi_b}, y_{\xi_b}) \tag{10}$$

with random variables $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_B)$ representing sampling of mini-batches. At step t, the stochastic noise $\boldsymbol{\varepsilon}_t$ of SGD is given by

$$\boldsymbol{\varepsilon}_t = \frac{1}{B} \sum_{b=1}^{B} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t, \mathbf{x}_{\xi_b}, y_{\xi_b}) - \mathbb{E}_{\boldsymbol{\xi}} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t) \tag{11}$$

dependent both on the current parameter $\boldsymbol{\theta}_t$ and $\boldsymbol{\xi}_t$ [33, 50]. Importantly, the noise covariance $C = \mathbb{E}_{\boldsymbol{\xi}}[\boldsymbol{\varepsilon}_t \boldsymbol{\varepsilon}_t^T]$ accounts for fluctuations of the training dynamics around $\boldsymbol{\theta}^*$.

For loss functions without Tikhonov regularization terms such as weight decay, the noise covariance matrix has been shown by several works to be equivalent to the mean Hessian [33, 50]. Hence, the bound in Theorem 2 can be expressed in terms of fluctuations of the parameter gradients around $\boldsymbol{\theta}^*$, providing Corollary 2, restated below. A proof of the statement is given in section F.

Corollary 2. *Let $\boldsymbol{\theta}^*$ be a critical point for the loss $\mathcal{L}(\boldsymbol{\theta}, \mathbf{x}, y)$ on $\mathcal{D}$. Let $\mathbf{f}_{\boldsymbol{\theta}}$ denote a neural network with at least one hidden layer, with $\|\boldsymbol{\theta}^1\| > 0$. Then,*

$$\frac{x_{\min}^2}{\|\boldsymbol{\theta}^1\|_2^2} \mathbb{E}_{\mathcal{D}} \|\nabla_{\mathbf{x}} \mathcal{L}\|_2^2 \leq \operatorname{tr}(S) + o(\mathcal{L}(\boldsymbol{\theta})) \tag{7}$$

with $S = C + \frac{1}{B} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})^T$ denoting the gradient uncentered covariance.

Figure 7 shows the largest principal component $\lambda_{\max}(C)$, as model size increases. Similarly to the mean curvature, stochastic noise strongly correlates with the empirical Lipschitz constant, decreasing considerably in the interpolation regime, and showing that overparameterization non-monotonically affects the dynamics of training.

After having established a clearer connection between training dynamics in proximity of $\boldsymbol{\theta}^*$ and our main bound, we discuss the role of training hyperparameters in affecting mean curvature.

Linear Stability of SGD In proximity of a critical point $\boldsymbol{\theta}^*$, it is possible to derive stability conditions under which the point is attainable by SGD [47]. Essentially, under the quadratic approximation of Equation 5, the dynamics of SGD are said to be linearly stable in a neighbourhood of $\boldsymbol{\theta}^*$ if $\exists\ \gamma > 0$ for which $\mathbb{E}_{\mathcal{D}} \|\boldsymbol{\theta}_t\|^2 \leq \gamma\, \mathbb{E}_{\mathcal{D}} \|\boldsymbol{\theta}_0\|^2$, for all t [22]. Wu and Ma [47] provide linear stability conditions for SGD in the following proposition.

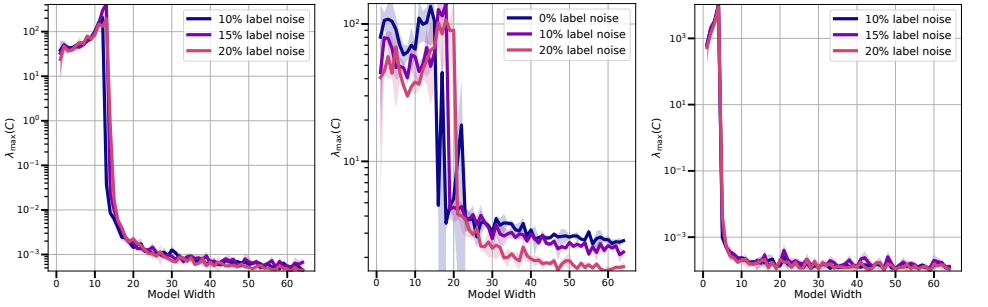

Figure 7: **Dominant noise-covariance eigenvalue.** (Top) From left to right: ConvNets trained on CIFAR-10 (left), CIFAR-100 (middle) and ResNets trained on CIFAR-10 (right). In all settings, the magnitude of stochastic noise strongly correlates with double descent, peaking at the interpolation threshold, and highlighting a nonlinear dependence on network width.

Proposition 3. *(Wu and Ma [47], Theorem 1.) A critical point $\boldsymbol{\theta}^*$ is linearly stable for SGD with learning rate η and batch size B if*

$$\lambda_{\max}\left((I_p - \eta H)^2 + \frac{\eta^2(N-B)}{B(N-1)}\Sigma\right) \leq 1$$

with $N = |\mathcal{D}|$ and $\Sigma = \mathbb{E}_{\mathcal{D}}(H^2) - (\mathbb{E}_{\mathcal{D}}H)^2$.

Additionally, Wu and Ma [47], provide a necessary condition for Proposition 3 to hold, by requiring $\lambda_{\max}(I_p - \eta H) \leq 1$ and $\lambda_{\max}(\frac{\eta^2(N-B)}{B(N-1)}\Sigma) \leq 1$ to hold separately, providing the conditions

$$\begin{cases} 0 \leq & \lambda_{\max}(H) \leq \frac{2}{\eta} \\ 0 \leq & \lambda_{\max}(\Sigma) \leq \frac{1}{\eta}\sqrt{\frac{B(N-1)}{n-B}} \end{cases} \tag{12}$$

The term $\lambda_{\max}(\Sigma)$, called non-uniformity, measures the mean squared deviation of curvature under sampling of mini-batches from $\mathcal{D}$.

Thus, the choice of η and B affects reachability of critical points $\boldsymbol{\theta}^*$ under the dynamics of SGD. Particularly, the conditions in Equation 12 imply that large learning rates η and small batch sizes B will select critical points respectively with low curvature $\lambda_{\max}(H)$ and low non-uniformity $\lambda_{\max}(\Sigma)$. Hence, η and B control parameter space curvature around critical points attainable by the training dynamics and, via Theorem 2, input-sensitivity.

In the next sections, we extend the empirical findings of section 3.

D.4 Beyond Piece-wise Linear Networks

In this section, we extend our main finding beyond vision architectures and focus on natural language processing tasks. Specifically, we consider transformer architectures and train 8-layer multi-head attention transformers [44] on machine translation tasks, controlling the embedding dimension, as well as the width of hidden fully connected layers $\omega = 4h$. We report the test error in Figure 8 (left). We compute Equation 1 on $\nabla_{\mathbf{x}}\mathcal{L}$, where $\mathcal{L}$ is the per-token perplexity. We note that Equation 1 can still be applied to the Jacobian $\nabla_{\mathbf{x}}\mathcal{L}$ – which linearly approximates $\mathcal{L}$ at each point $\mathbf{x}$ – and the expected operator norm should be intended

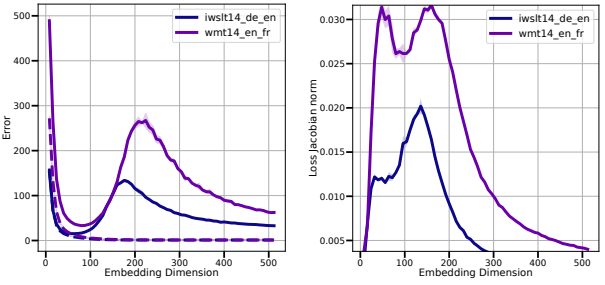

Figure 8: **Input-space smoothness of Transformers**, as the embedding dimension and model width vary. Train error (dashed) and double descent for the test error for Transformers trained machine translation tasks (left) and input-space loss Jacobian norm (right).

as the Sobolev seminorm $\|\mathcal{L}\|_{\mathcal{D},1,2}$ of $\mathcal{L}$ on $\mathcal{D}$. Figure 8 (right panel) extends our main finding, showing that $\nabla_{\mathbf{x}}\mathcal{L}$ depends non-monotonically on model size, peaking near the interpolation threshold, and extending our main result beyond vision architectures.

D.5 Empirical Lipschitz Throughout Training

In this section, we complement the results shown for selected model widths in Figure 4, by plotting the development of the empirical Lipschitz constant throughout training for all model sizes, and discuss its relationship to the test error. Extending our finding to additional model widths, Figure 9 shows that small models maintain a small empirical Lipschitz constant throughout training, while models near the interpolation threshold accumulate a large empirical Lipschitz constant after prolonged training. Finally, large models maintain a relatively low empirical Lipschitz constant, plateauing earlier as model size increases past the interpolating threshold.

At the same time, with reference to the line plots in Figure 4, for all models the initial increase in empirical Lipschitz constant – occurring during "early" training (up until epoch 100 for ConvNets and 400 for ResNets) – is matched by a rapid decrease in test error. During mid-training (epoch e $100 < e < 200$ for ConvNets, and $400 < e < 500$ for ResNets) the rate of increase of the Lipschitz constant changes according to model size. Small models plateau in their empirical Lipschitz constant, train and test error, and remain stable thereafter. Models near the interpolation threshold start slowly increasing the empirical Lipschitz constant as they slowly interpolate the training set, with corresponding increase in test error, showcasing the "malign overfitting" phenomenon [3], Strikingly, large models quickly interpolate the training set, causing relative increase in the empirical Lipschitz constant, inversely correlating with model size. Throughout this phase of "accelerated interpolation" the test error undergoes epoch-wise double descent [45]. Crucially, while for all models the empirical Lipschitz constant is monotonically increasing in epochs, the *rate* at which the empirical Lipschitz constant grows correlates with *epoch-wise* double descent for the test error. This observation suggests that tracking second order information of $\mathbf{f}_{\boldsymbol{\theta}}$ in input space may reveal important properties of interpolation. Indeed, input-space Hessian based measures [27, 32] have been observed to correlate with model performance for fixed-sized models. Our observations suggest that input-space curvature may bear significance for understanding epoch-wise double descent. We leave this exciting direction to future work.

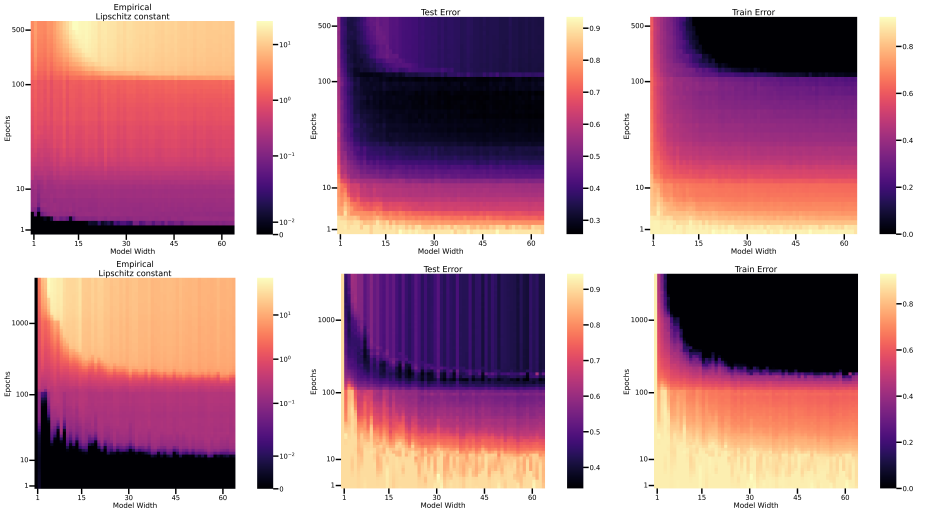

Figure 9: (Top left) **Empirical Lipschitz constant** (color) **as a function of training epochs** ($y-axis$) and model size (x-axis). (Top middle) **Test error** for ConvNets on CIFAR-10 with 20% noisy training labels. (Top right) **Test error** for ConvNets on CIFAR-10 with 20% noisy training labels. (Bottom) Analogous plots for ResNet18s trained on the same dataset.

D.6 Validation of our Bound in the Interpolating Regime

Figure 10 summarizes our main findings, showing a strong correlation between the empirical Lipschitz constant and maximum parameter-space curvature of the loss landscape, mean parameter-space curvature, as well as the first principal component of gradient noise, with networks with large empirical Lipschitz constant incurring in high test error.

E Generating Random Validation Data

To generate random validation data for the experiments reported in Figure 4, we define several distributions over RGB pixels, and sample each pixel independently. We consider the following distributions:

- $\mathbf{x}_n \sim \mathcal{U}\big([\boldsymbol{\mu}_{\text{CIFAR}} - \boldsymbol{\sigma}_{\text{CIFAR}}, \boldsymbol{\mu}_{\text{CIFAR}} + \boldsymbol{\sigma}_{\text{CIFAR}}]\big)$ pixel-wise
- $\mathbf{x}_n \sim \mathcal{N}\big([\boldsymbol{\mu}_{\text{CIFAR}}, \mathcal{I}_3\boldsymbol{\sigma}_{\text{CIFAR}}]\big)$ pixel-wise
- $\mathbf{x}_n \sim \mathcal{U}\big(S_{d-1}\big)$ (pixel-wise) hypersphere
- $\mathbf{x}_n + \boldsymbol{\varepsilon}_n$, with $\boldsymbol{\varepsilon}$ strong random jitter

where $\boldsymbol{\mu}_{\text{CIFAR}}$ and $\boldsymbol{\sigma}_{\text{CIFAR}}$ respectively denote the per-channel mean and standard deviation computed on the CIFAR-10 training set. For each distribution, we generate a validation set of 50k i.i.d. samples, and probe networks trained on the standard CIFAR-10 with 20% corrupted labels.

For reference, we also plot the empirical Lipschitz constant estimated on the CIFAR-10 train and test split. For both out-of-sample and in-sample validation datasets, it can be

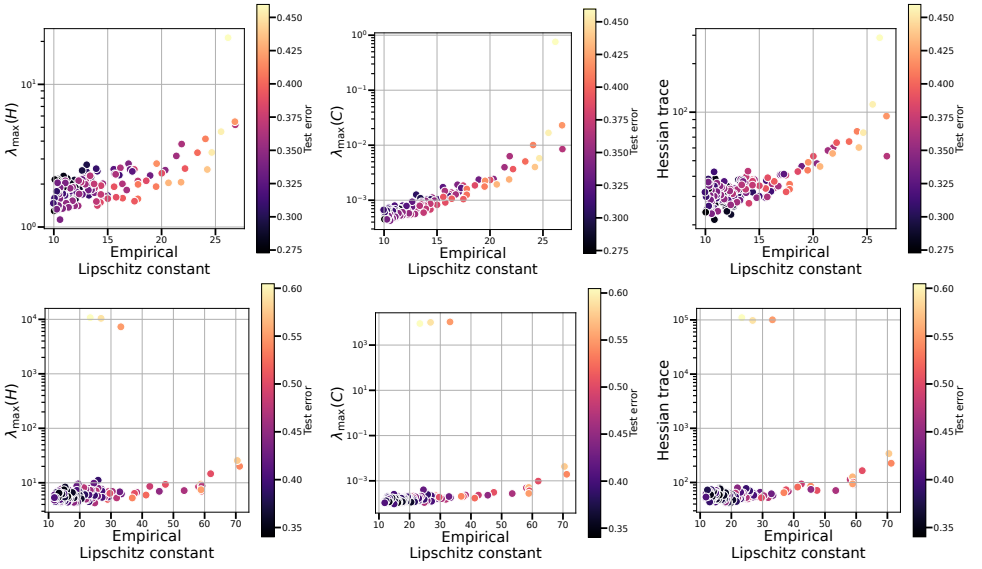

Figure 10: **Correlation between empirical Lipschitz constant and parameter-space curvature** in the interpolating regime. From left to right: maximum curvature (left), dominant noise-covariance eigenvalue (middle) and mean curvature (right), respectively for ConvNets trained on CIFAR-10 (top), and ResNets trained on CIFAR-10 (bottom). In all settings, mean and maximum parameter-space curvature strongly correlate with the empirical Lipschitz constant in the interpolating regime. Furthemore, models with higher empirical Lipschitz present higher mean and maximum curvatures, and incur in higher test error. All values are reported in log-y scale to better separate models.

observed how the empirical Lipschitz constant remains bounded, and closely follows the double descent trend for the test error (c.f.r. Figure 1). Remarkably, the empirical Lipschitz constant on random validation data closely matches the one estimated on the training set, supporting the hypothesis of globally bounded function complexity.

F Proofs

In this section, we provide proofs for the formal statements presented in section 2. We begin by deriving results on boundedness of model function input-space gradients via parameter-space gradients. Then, we prove results of section 2.3.

F.1 Duality of Linear Layers

We begin by providing a general form of Theorem 1.

Theorem 1. *Let* $\mathbf{f}$ *denote a neural network with a least one hidden layer, with* $\|\boldsymbol{\theta}^1\| > 0$ *and arbitrary weights* $\boldsymbol{\theta}^2, \dots, \boldsymbol{\theta}^L$. *Let* $x_{\min} := \min_{\mathbf{x}_n \in \mathcal{D}} \|\mathbf{x}_n\|_2$. *Then, parameter-space gradients*

bound input-space gradients of $\mathbf{f}$ *from above:*

$$\frac{x_{\min}^2}{\|\boldsymbol{\theta}^1\|_2^2}\mathbb{E}_{\mathcal{D}}\|\nabla_{\mathbf{x}}\mathbf{f}\|_2^2 \leq \mathbb{E}_{\mathcal{D}}\|\nabla_{\boldsymbol{\theta}}\mathbf{f}\|_2^2 . \tag{2}$$

Proof. We recall that, by duality of inputs and weights in linear transformations, the partial derivatives $\frac{\partial \mathbf{f}}{\partial \mathbf{x}^{\ell-1}}$ and $\frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}^\ell}$ w.r.t. any layer of the form $\mathbf{x}^\ell = \phi(\boldsymbol{\theta}^\ell \mathbf{x}^{\ell-1})$ are tied by the upstream gradient $\frac{\partial \mathbf{f}}{\partial(\boldsymbol{\theta}^\ell \mathbf{x}^{\ell-1})}$. Indeed, by the chain rule

$$\begin{cases} \frac{\partial \mathbf{f}}{\partial \mathbf{x}^{\ell-1}} & = \boldsymbol{\theta}^{\ell T} \frac{\partial \mathbf{f}}{\partial(\boldsymbol{\theta}^\ell \mathbf{x}^{\ell-1})} \\ \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}^\ell} & = \frac{\partial \mathbf{f}}{\partial(\boldsymbol{\theta}^\ell \mathbf{x}^{\ell-1})} \mathbf{x}^{\ell-1 T} \end{cases} \tag{13}$$

Let $\mathbf{f} : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^K$ be an arbitrary function composing linear layers with (optional) nonlinearities $\phi : \mathbb{R} \to \mathbb{R}$, that are differentiable a.e. Furthermore, let $\mathbb{R}^{d_\ell}$ denote the codomain of layer ℓ, i.e. $\mathbf{x}^\ell \in \mathbb{R}^{d_\ell}$.

Combining the two conditions in Equation 13 gives

$$\begin{aligned} \frac{\partial \mathbf{f}}{\partial \mathbf{x}^{\ell-1}} \mathbf{x}^{\ell-1 T} &= \boldsymbol{\theta}^{\ell T} \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}^\ell} \\ \left\| \frac{\partial \mathbf{f}}{\partial \mathbf{x}^{\ell-1}} \mathbf{x}^{\ell-1 T} \right\| &= \left\| \boldsymbol{\theta}^{\ell T} \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}^\ell} \right\| \\ \frac{1}{\|\mathbf{x}\|} \left\| \frac{\partial \mathbf{f}}{\partial \mathbf{x}^{\ell-1}} \mathbf{x}^{\ell-1 T} \right\| &= \frac{1}{\|\mathbf{x}\|} \left\| \boldsymbol{\theta}^{\ell T} \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}^\ell} \right\| \qquad \text{for} \quad \mathbf{x} \in \mathbb{R}^{d_\ell} \setminus \{\mathbf{0}\} \\ \sup_{\mathbf{x}:\|\mathbf{x}\|\neq 0} \frac{1}{\|\mathbf{x}\|} \left\| \frac{\partial \mathbf{f}}{\partial \mathbf{x}^{\ell-1}} \mathbf{x}^{\ell-1 T} \right\| &= \sup_{\mathbf{x}:\|\mathbf{x}\|\neq 0} \frac{1}{\|\mathbf{x}\|} \left\| \boldsymbol{\theta}^{\ell T} \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}^\ell} \right\| \\ \left\| \frac{\partial \mathbf{f}}{\partial \mathbf{x}^{\ell-1}} \right\|_2 &= \sup_{\mathbf{x}:\|\mathbf{x}\|\neq 0} \frac{1}{\|\mathbf{x}\|} \left\| \boldsymbol{\theta}^{\ell T} \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}^\ell} \right\| \end{aligned} \tag{14}$$

Restricting the domain of $\mathbf{f}$ to a set of data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^d$, with corresponding activations $\{\mathbf{x}_1^\ell, \ldots, \mathbf{x}_n^\ell\} \subset \mathbb{R}^{d_\ell}$ yields:

$$\left\| \frac{\partial \mathbf{f}}{\partial \mathbf{x}^{\ell-1}} \right\|_2 = \frac{1}{\min_n \|\mathbf{x}_n^\ell\|} \left\| \boldsymbol{\theta}^{\ell T} \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}^\ell} \right\| \leq \frac{\|\boldsymbol{\theta}^{\ell T}\|}{\min_n \|\mathbf{x}_n^\ell\|} \left\| \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}^\ell} \right\| \tag{15}$$

Particularly, given the training set $\mathcal{D}$, applying Equation 15 to the gradients $\nabla_{\mathbf{x}}\mathbf{f}$ and $\nabla_{\boldsymbol{\theta}}\mathbf{f}$, and taking the expectation over $\mathcal{D}$ on both sides concludes the proof.

□

We note that, while a similar bound was observed in Ma and Ying [29] for the first layer gradients, the authors propose to bound $\nabla_{\boldsymbol{\theta}}\mathbf{f}$ via a uniform bound that linearly depends on model size, and thus cannot capture double descent. In this work, we improve upon their bounds,generalizing the result of Ma and Ying [29] to any layer beyond the first, and by explicitly studying Equation 4 in connection to parameter-space dynamics and double descent.

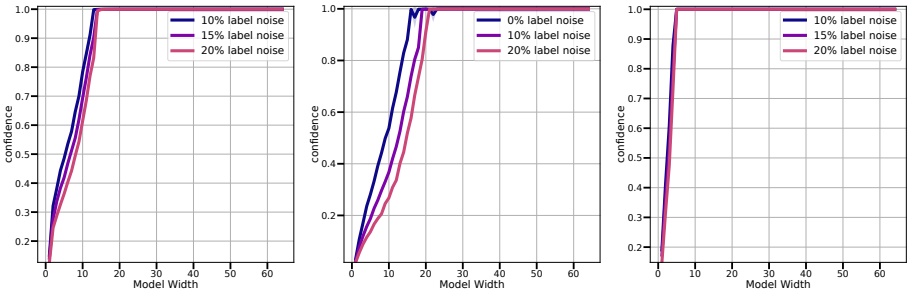

Figure 11: **Prediction confidence** as a function of model size, for ConvNets trained on CIFAR-10 (left), CIFAR-100 (middle) and ResNet18s trained on CIFAR-10. For all experimental settings, the model confidence monotonically depends on model size. By Equation 16, this confirms that the non-monotonic trends reported in this work are caused by the model function **f**.

F.2 Extension to loss functions

For losses $\mathcal{L} : \mathbb{R}^p \times \mathbb{R}^d \times \mathcal{Y} \to \mathbb{R}^+$ of the exponential family [10] like mean squared error and cross entropy, the following corollary holds.

Corollary 1. *Consider the composition of a loss function $\mathcal{L}$ with a neural network* **f** *with a least one hidden layer, with $\|\boldsymbol{\theta}^1\| > 0$ and arbitrary weights $\boldsymbol{\theta}^2, \ldots, \boldsymbol{\theta}^L$. Then,*

$$\frac{x_{\min}^2}{\|\boldsymbol{\theta}^1\|_2^2} \mathbb{E}_{\mathcal{D}} \|\nabla_{\mathbf{x}} \mathcal{L}\|_2^2 \leq \mathbb{E}_{\mathcal{D}} \|\nabla_{\boldsymbol{\theta}} \mathcal{L}\|_2^2 . \tag{4}$$

Proof. For each sample $(\mathbf{x}_n, y_n)$, the gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{f}}$ takes the form $\mathbf{p}_n - \mathbf{e}_{y_n}$. For crossentropy, $\mathbf{p}_n$ denotes the softmax normalized logits, and $\mathbf{e}_{y_n}$ the one-hot encoded label y_n. For mean squared error, $\mathbf{p}_n = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_n)$.

When composing the loss with a model **f**, we have

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} (\mathcal{L} \circ \mathbf{f}) &= (\mathbf{p}_n - \mathbf{e}_{y_n}) \nabla_{\boldsymbol{\theta}} \mathbf{f} \\ \nabla_{\mathbf{x}} (\mathcal{L} \circ \mathbf{f}) &= (\mathbf{p}_n - \mathbf{e}_{y_n}) \nabla_{\mathbf{x}} \mathbf{f} \end{aligned} \tag{16}$$

Applying Theorem 1 trivially concludes the proof. □

Importantly, the term $\|\mathbf{p}_n - \mathbf{e}_{y_n}\|$ is inversely proportional to the model confidence [5] $\sigma = 1 - \frac{1}{N} \sum_{n=1}^{N} \|\mathbf{p}_n - \mathbf{e}_{y_n}\|$, which generally saturates for large models, typically yielding high confidence predictions at convergence. In Figure 11 we empirically study how the quantity is affected by model size, to understand its impact on the bounds presented throughout section 2. Model confidence is observed to monotonically depend on model size. This highlights the fact that the double descent trends observed throughout this paper are to be attributed to the model function, as shown throughout our experiments for the empirical Lipschitz constant.

Next, we provide proofs for section 2.3.

F.3 Connection to Parameter-Space Curvature

In this section, we prove formal statements connecting Theorem 1 to the dynamics of SGD in proximity of a critical point $\boldsymbol{\theta}^*$. For our proofs, we use the mean square error $\mathbb{E}_{\mathcal{D}}\mathcal{L} = \frac{1}{2N}\sum_{n=1}^{N}(f_{\boldsymbol{\theta}}(\mathbf{x}_n)-y_n)^2$, and adopt a recent model of stochastic noise proposed by Liu et al. [28]. The crux of the proof of Theorem 2 is bounding $\mathbb{E}_{\mathcal{D}}\|\nabla_{\boldsymbol{\theta}}\mathcal{L}\|$ with $\mathrm{tr}\,(H)$, which we can later connect to the noise uncentered covariance S.

Theorem 2. *Let $\boldsymbol{\theta}^*$ be a critical point for the loss $\mathcal{L}(\boldsymbol{\theta},\mathbf{x},y)$ on $\mathcal{D}$. Let $\mathbf{f}_{\boldsymbol{\theta}}$ denote a neural network with at least one hidden layer, with $\|\boldsymbol{\theta}^1\| > 0$. Then,*

$$\frac{x_{\min}^2}{\|\boldsymbol{\theta}^1\|_2^2}\mathbb{E}_{\mathcal{D}}\|\nabla_{\mathbf{x}}\mathcal{L}\|_2^2 \leq 2\mathcal{L}_{\max}(\boldsymbol{\theta})\,\Delta(\mathcal{L}(\boldsymbol{\theta})) + o(\mathcal{L}(\boldsymbol{\theta})) \tag{6}$$

with $\Delta(\mathcal{L}(\boldsymbol{\theta})) := \mathrm{tr}\,(H)$ denoting the Laplace operator, $H := \mathbb{E}_{\mathcal{D}}[\frac{\partial^2\mathcal{L}}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^T}]$ denoting the expected parameter-space Hessian of $\mathcal{L}$, and $\mathcal{L}_{\max}(\boldsymbol{\theta}) := \max_{(\mathbf{x}_n,y_n)\in\mathcal{D}}\mathcal{L}(\boldsymbol{\theta},\mathbf{x}_n,y_n)$.

Proof. The proof is broken down in two parts. First, we write out explicitly the expected Hessian H of $\mathcal{L}$.

$$H = \frac{1}{N}\sum_{n=1}^{N}\frac{\partial^2}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^T}\mathcal{L}_n = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}_n''\nabla_{\boldsymbol{\theta}}\mathbf{f}_n\nabla_{\boldsymbol{\theta}}\mathbf{f}_n^T + \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}_n'\frac{\partial^2}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^T}\mathbf{f}_n \tag{17}$$

with $\mathbf{f}_n := \mathbf{f}(\mathbf{x}_n,\boldsymbol{\theta})$, for $n = 1,\dots,N$.

By noting that $\mathcal{L}_n'' = 1, \forall n$, and that $\mathcal{L}_n' \propto \mathcal{L}_n \to 0$ as $\|\boldsymbol{\theta}-\boldsymbol{\theta}^*\|^2 \to 0$ for interpolating models, the expected loss Hessian amounts to the cross term

$$H = \frac{1}{N}\sum_{n=1}^{N}\nabla_{\boldsymbol{\theta}}\mathbf{f}_n\nabla_{\boldsymbol{\theta}}\mathbf{f}_n^T + \mathcal{O}(\mathcal{L}(\boldsymbol{\theta})) \tag{18}$$

Next, we connect $\|\nabla_{\boldsymbol{\theta}}\mathcal{L}\|_2^2$ to H. We note that $\frac{1}{N}\sum_{n=1}^{N}\nabla_{\boldsymbol{\theta}}\mathcal{L} = \mathcal{L}_n'\nabla_{\boldsymbol{\theta}}\mathbf{f}_n$. Then, by definition of norm:

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}}\|\nabla_{\boldsymbol{\theta}}\mathcal{L}\|_2^2 &= \mathbb{E}_{\mathcal{D}}\nabla_{\boldsymbol{\theta}}\mathcal{L}^T\nabla_{\boldsymbol{\theta}}\mathcal{L} \\
&= \mathbb{E}_{\mathcal{D}}\,\mathrm{tr}\,(\nabla_{\boldsymbol{\theta}}\mathcal{L}\nabla_{\boldsymbol{\theta}}\mathcal{L}^T) \\
&= \mathrm{tr}\,\Big(\frac{1}{N}\sum_{n=1}^{N}[\mathcal{L}_n'^{\,2}\,\nabla_{\boldsymbol{\theta}}\mathbf{f}_n\nabla_{\boldsymbol{\theta}}\mathbf{f}_n^T]\Big) \\
&\leq 2\Big(\max_{1\leq n\leq N}\mathcal{L}_n'^{\,2}\Big)\Big(\mathrm{tr}\,\Big(\frac{1}{N}\sum_{n=1}^{N}\nabla_{\boldsymbol{\theta}}\mathbf{f}_n\nabla_{\boldsymbol{\theta}}\mathbf{f}_n^T\Big)\Big) \\
&= 2\mathcal{L}_{\max}(\boldsymbol{\theta})\,\mathrm{tr}\,(H) \\
&= 2\mathcal{L}_{\max}(\boldsymbol{\theta})\,\Delta\mathcal{L}(\boldsymbol{\theta})
\end{aligned} \tag{19}$$

□

Having built a connection between $\nabla_{\boldsymbol{\theta}}\mathcal{L}$ and H, we can prove Corollary 2.

Corollary 2. *Let $\boldsymbol{\theta}^*$ be a critical point for the loss $\mathcal{L}(\boldsymbol{\theta}, \mathbf{x}, y)$ on $\mathcal{D}$. Let $\mathbf{f}_{\boldsymbol{\theta}}$ denote a neural network with at least one hidden layer, with $\|\boldsymbol{\theta}^1\| > 0$. Then,*

$$\frac{x_{\min}^2}{\|\boldsymbol{\theta}^1\|_2^2} \mathbb{E}_{\mathcal{D}} \|\nabla_{\mathbf{x}} \mathcal{L}\|_2^2 \leq \operatorname{tr}(S) + o(\mathcal{L}(\boldsymbol{\theta})) \tag{7}$$

with $S = C + \frac{1}{B} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})^T$ denoting the gradient uncentered covariance.

Proof.

$$\begin{aligned} \mathbb{E}_{\mathcal{D}} \|\nabla_{\boldsymbol{\theta}} \mathcal{L}\|_2^2 &= \mathbb{E}_{\mathcal{D}} \nabla_{\boldsymbol{\theta}} \mathcal{L}^T \nabla_{\boldsymbol{\theta}} \mathcal{L} \\ &= \mathbb{E}_{\mathcal{D}} \operatorname{tr}(\nabla_{\boldsymbol{\theta}} \mathcal{L} \nabla_{\boldsymbol{\theta}} \mathcal{L}^T) \\ &= \operatorname{tr}\Big(\frac{1}{N} \sum_{n=1}^{N} [\mathcal{L}_n'^2 \, \nabla_{\boldsymbol{\theta}} \mathbf{f}_n \nabla_{\boldsymbol{\theta}} \mathbf{f}_n^T]\Big) \\ &= \operatorname{tr}(S) \end{aligned} \tag{20}$$

□