What training reveals about neural network complexity

Anonymous Author(s)
Affiliation
Address
email

Abstract

This work explores the hypothesis that the complexity of the function a deep neural network (NN) is learning can be deduced by how fast its weights change during training. Our analysis provides evidence for this supposition by relating the network's distribution of Lipschitz constants (i.e., the norm of the gradient at different regions of the input space) during different training intervals with the behavior of the stochastic training procedure. We first observe that the average Lipschitz constant *close to the training data* affects various aspects of the parameter trajectory, with more complex networks having a longer trajectory, bigger variance, and often veering further from their initialization. We then show that NNs whose biases are trained more steadily have bounded complexity even in regions of the input space that are *far from any training point*. Finally, we find that steady training with Dropout implies a training- and data-dependent generalization bound that grows poly-logarithmically with the number of parameters. Overall, our results support the hypothesis that good training behavior can be a useful bias towards good generalization.

1 Introduction

Though neural networks (NNs) trained on relatively small datasets can generalize well, often significant trial and error is needed to select an architecture that does not overfit. Could it be possible that, besides relying on validation error, NN designers favor architectures that can be easily trained and this biases them towards models with better generalization?

In the heart of our hypothesis lies the conjecture that *the behavior of the training procedure can be used as an indicator of the complexity of the function a NN is learning*. Some empirical evidence for this supposition already exists. Zhang et al. [1] found that on CIFAR10 image classification the training was becoming more tedious the more the data were corrupted, e.g., the Inception [2] architecture is $3.5\times$ slower to train when used to predict random labels than real images. It has also been observed that the loss is more sensitive w.r.t. specific training points when the network is memorizing data and that training slows down faster as the NN size decreases when the data contain noise [3]. From the theory side, it is known that the training of shallow networks converges faster for more separable classes [4] and slower when fitting random labels [5]. In addition, the stability [6] of stochastic gradient descent (SGD) implies that (under strong assumptions) NNs that can be trained with a small number of iterations provably generalize [7, 8]. Intuitively, since each gradient update conveys limited information, a NN that sees each training point few times (typically one or two) will not learn enough about the training set to overfit. Despite the elegance of this claim, the provided explanation does not necessarily account for what is observed in practice, where NNs trained for thousands of epochs can generalize even without rapidly decaying learning rates.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

1.1 Quantifying NN complexity

This work takes a further step towards theoretically characterizing the relationship between the SGD trajectory and the complexity of the learned function. We study neural networks with ReLU activations, i.e., parametric piece-wise linear functions. Though many works measure the complexity of these networks via their maximum number of linear regions [9–12], it is suspected that the average NN behavior is far from the extremal constructions usually employed theoretically [13].

We instead focus on the Lipschitz continuity of a NN at different regions of its input. For networks equipped with ReLU activations, the Lipschitz constant in a region is simply the norm of the gradient at any point within it. The distribution of Lipschitz constants presents a natural way to quantify the complexity of NNs. Crucially, NNs that are everywhere Lipschitz continuous can generalize beyond the training data, a phenomenon that has been demonstrated both theoretically [14–16] and empirically [17]. The generalization bounds in question grow with the Lipschitz constant and the intrinsic dimensionality of the data manifold, but not necessarily with the number of parameters[1], which renders them ideal for the study of overparameterized networks.

1.2 Main findings: connecting training behavior and neural network complexity

Our analysis links SGD behavior with NN complexity close and far from the training data.

NN complexity close to the training data. Section 4 commences with a simple observation: SGD updates the 1st layer bias more quickly if the learned function has a large Lipschitz constant near a sampled data point. This implies that the length of the optimization trajectory grows linearly with the average Lipschitz constant of the NN on its linear regions that contain training data (Theorem 1). Based on this insight, we deduce that (a) near convergence, the parameters of more complex NNs vary more across successive SGD iterations (Corollary 2), and (b) the distance of the trained network to initialization is small if the learned NN has a low complexity (near training data) throughout its training, with the first few high-error epochs playing a dominant role (Corollary 3).

NN complexity far from the training data. Section 5 focuses on the relationship between training and the Lipschitz constant in empty regions of the input space, i.e., linear regions of the NN that do not contain training points. We first show that the Lipschitz constants in empty regions are linked with those of regions containing training data (Theorem 2). Our analysis implies that NNs whose parameters are updated more slowly during training have bounded complexity in a larger portion of the input space. We also demonstrate how training NNs with Dropout enables us to grasp more information about the properties of the learned function and, as such, to yield tighter estimates for the global Lipschitz constant. Our findings yield a training-dependent generalization bound that features a *poly-logarithmic* dependence on the number of parameters when the data have low intrinsic dimensionality (Theorem 3). On the contrary, in typical NN generalization bounds the number of samples needs to grow *exponentially* with depth [20–26].

All proofs can be found in Appendix A.

2 Related works

The Lipschitz constant of NNs. Since exactly computing the Lipschitz constant is NP-hard [27], its efficient estimation is an active topic of research [27–30, 19, 31]. Our work stands out from these works both in motivation (i.e., we connect training behavior with NN complexity) and in the techniques developed (we are not employing any complex algorithmic machinery to estimate the Lipschitz constant of a trained model, but we bound it as the NN is being trained based on how weights change). Empirically, Lipschitz regularization has been used to bias training towards simple and adversarially robust networks [32–38]. Theoretically, the Lipschitz constant is featured prominently in the generalization analysis of NNs (e.g., [22–24]), but most analyses depend on sensitivity w.r.t. parameter perturbation, which is related but not identical to the Lipschitz constant.

Dropout and generalization. The Dropout mechanism and its variants are standard tools of the NN toolkit [39–41] that regularize training [42, 43] and help prevent memorization [3]. The effect of Dropout on generalization have been theoretically studied primarily for shallow networks [44, 45, 43]

[1] While the Lipschitz constant is typically upper bounded by the product of spectral norms of the layer weight matrices (thus yielding an exponential dependency on the depth), the product-of-norms bound is known to be far from the real Lipschitz constant [18, 19]

as well as for general classifiers [46]. The generalization bounds that apply to deep networks are norm-based and generally grow exponentially with depth [47, 48] or are shown to scale the Rademacher complexity by the Dropout probability (for Dropout used in the last layer) [49]. We instead base our analysis on arguments from [15, 16] and exploit the properties of ReLU networks to derive a bound that features a significantly milder dependency on the NN depth.

Flat and sharp minima. Flat minima correspond to large connected regions with low empirical error in weight space and have been argued to correspond to networks of low complexity and good generalization [50]. It has also been shown that SGD converges more frequently to flat minima [51–55]. Different from the current work that focuses on the sensitivity w.r.t. changes in the data, flatness corresponds to a statement about local Lipschitz continuity w.r.t. weight changes. In addition, whereas flat minima are regions of the space where the loss is low, our main results account for more complex loss behaviors (by means of an appropriate normalization). Note also that some works argue that the flat/sharp dichotomy may not capture all necessary properties [56–58] as flat minima can be made sharp by a suitable reparameterization [57], and flat and sharp minima may be connected [57].

Training dynamics of NNs. Many authors have studied the training dynamics of NNs [59–65], arguing that, with correct initialization and significant overparameterization, SGD converges to a good solution that generalizes. Our work complements these studies by focusing on how the SGD trajectory can be used to infer NN complexity. Arora et al. [5] connect the trajectory length and generalization performance via the Neural Tangent Kernel (NTK). Most analyses based on the NTK ("lazy" regime) or mean field approximation ("adaptive" regime) focus on 2- or 3-layer networks. In contrast to these works, we make no assumptions on initialization or network size.

3 Preliminaries and background

Suppose we are given a training dataset (X, Y) consisting of N training points $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ and the associated labels $Y = (y_1, \ldots, y_N)$, with $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^n$ and $y_i \in \mathcal{Y} \subseteq \mathbb{R}$.

We focus on NNs defined as the composition of d layers $f = f_d \circ \cdots \circ f_1$, with

$$f_l(\boldsymbol{x}, \boldsymbol{w}) = \rho_l(\boldsymbol{W}_l\, \boldsymbol{x} + \boldsymbol{b}_l) \quad \text{for} \quad l = 1, \ldots, d.$$

Above, $\boldsymbol{W}_l \in \mathbb{R}^{n_l \times n_{l-1}}$ and $\boldsymbol{b}_l \in \mathbb{R}^{n_l}$ with $n_0 = n$ and $n_d = 1$, and $\boldsymbol{w} = (\boldsymbol{W}_1, \boldsymbol{b}_1, \ldots, \boldsymbol{W}_d, \boldsymbol{b}_d)$ are the network's parameters. For all layers but the last, ρ_l will be the ReLU activation function, whereas ρ_d may either be the identity $\rho_d(x) = x$ (regression) or the sigmoid function $\rho_d(x) = 1/(1 + e^{-x})$ (classification).

We optimize $\boldsymbol{w}$ to minimize a differentiable loss function ℓ using stochastic gradient descent (SGD). The optimization proceeds in iterations t and each parameter is updated as follows:

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \alpha_t \frac{\partial \ell(f(\boldsymbol{x}^{(t)}, \boldsymbol{w}^{(t)}), y^{(t)})}{\partial \boldsymbol{w}^{(t)}},$$

where $\boldsymbol{x}^{(t)} \in X$ is a point sampled with replacement from the training set at iteration t, $y^{(t)}$ is its label, and α_t is the learning rate. It will also be convenient to refer to $f(\cdot, \boldsymbol{w}^{(t)})$ as $f^{(t)}$.

3.1 Linear regions

A well-known property of NNs with ReLU activations is that they partition the input space into regions (convex polyhedra) $\mathcal{R} \subseteq \mathbb{R}^n$ within which f is linear. This viewpoint will be central to our analysis.

There is a simple way to deduce this property from first principles. When ρ_d is the identity, each f can be equivalently expressed as

$$f(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{S}_d(\boldsymbol{x})(\boldsymbol{W}_d(\cdots\, \boldsymbol{S}_2(\boldsymbol{x})(\boldsymbol{W}_2 \boldsymbol{S}_1(\boldsymbol{x})(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2)\, \cdots) + \boldsymbol{b}_d),$$

where we have defined the input-dependent binary diagonal matrices

$$\boldsymbol{S}_l(\boldsymbol{x}) := \operatorname{diag}(\mathbf{1}[f_l \circ \cdots \circ f_1(\boldsymbol{x}, \boldsymbol{w}) > 0]) \quad \text{and} \quad \boldsymbol{S}_d(\boldsymbol{x}) = 1,$$

with $\mathbf{1}[\boldsymbol{x} > 0]$ being the indicator function applied element-wise. The key observation is that, when the neuron activations $\boldsymbol{S}_l(\boldsymbol{x})$ are fixed for every layer, the above function becomes linear. Thus, each linear region $\mathcal{R}$ of f contains those points that yield same neuron activation pattern.

Since the activation pattern of any region is uniquely defined by a single point in that region, we write $\mathcal{R}_{\boldsymbol{x}}$ to refer to the region that encloses $\boldsymbol{x}$.

3.2 Local and global Lipschitz constants

A function f is Lipschitz continuous with respect to a norm $\|\cdot\|_2$ if there exists a constant λ such that for all $\boldsymbol{x}, \boldsymbol{x}'$ we have $\|f(\boldsymbol{x}) - f(\boldsymbol{x}')\|_2 \leq \lambda \|\boldsymbol{x} - \boldsymbol{x}'\|_2$. The minimum λ satisfying this condition is called the Lipschitz constant of f and is denoted by λ_f.

The Lipschitz constant is intimately connected with the gradient. This can be easily seen for differentiable functions $f : \mathcal{X} \to \mathbb{R}$, in which case $\lambda_f = \sup_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{\nabla}^\top f(\boldsymbol{x})\|_2$, where $\mathcal{X}$ is a convex set and $\boldsymbol{\nabla} f(\boldsymbol{x})$ is the gradient of f at $\boldsymbol{x}$ [66, 27, 30]. Although NNs with ReLU activations are not differentiable everywhere, their Lipschitz constant can be determined in terms of their gradient within their regions. Specifically, the local Lipschitz constant within a linear region $\mathcal{R}_{\boldsymbol{x}}$ of f is

$$\lambda_f(\mathcal{R}_{\boldsymbol{x}}) = \|\boldsymbol{\nabla}^\top f(\boldsymbol{x}, \boldsymbol{w})\|_2$$

The Lipschitz constant of f is then simply the largest gradient within any linear region $\lambda_f = \sup_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{\nabla}^\top f(\boldsymbol{x}, \boldsymbol{w})\|_2$. The latter is typically upper bounded by $\lambda_f^{\text{prod}} = \prod_l \|\boldsymbol{W}_l\|_2$ which is known to be a loose bound [18, 19]. For a more formal treatment that also accounts for different types of activation functions and vector-valued outputs, the reader may refer to [19].

4 Relating training behavior to NN complexity close to the training data

Our analysis commences in Section 4.1 by deriving a general result that bounds the (appropriately normalized) length of the SGD trajectory over any training interval with the average Lipschitz constant of the NN close to training data. Our results on the distance to initialization and weight variance will be implied as corollaries in Section 4.2.

4.1 Bounding the length of the SGD trajectory

Theorem 1 formalizes a simple observation: the gradient of a neural network w.r.t. to its input is intimately linked to that w.r.t. to the bias of the 1st layer. This implies that, by observing how fast the bias of the network is updated, we can deduce what is the average Lipschitz constant of the learned function on the linear regions of the input space encountered during training.

Theorem 1 (Trajectory length)**.** *Let $f^{(t)}$ be a d-layer NN being trained by SGD. Further, define*

$$\lambda^{avg}_{f^{(t)}}(X) := \operatorname*{avg}_{\boldsymbol{x}\in X} \lambda_{f^{(t)}}(\mathcal{R}_{\boldsymbol{x}}), \quad \lambda^{max}_{f^{(t)}}(X) := \max_{\boldsymbol{x}\in X} \lambda_{f^{(t)}}(\mathcal{R}_{\boldsymbol{x}}), \quad \epsilon_{f^{(t)}}(\boldsymbol{x}, y) := \left| \frac{\partial \ell(\hat{y}, y)}{\partial \hat{y}} \right|_{\hat{y}=f^{(t)}(\boldsymbol{x})}.$$

For any $\delta > 0$ and set T of iteration indices within which the gradient is not zero, we have with high probability (over the SGD sampling)

$$\sum_{t\in T} \left(\frac{\lambda^{avg}_{f^{(t)}}(X)}{\sigma_1(\boldsymbol{W}_1^{(t)})} - \delta \right) \leq \sum_{t\in T} \frac{\|\boldsymbol{b}_1^{(t+1)} - \boldsymbol{b}_1^{(t)}\|_2}{\alpha_t\, \epsilon_{f^{(t)}}(\boldsymbol{x}^{(t)}, y^{(t)})} \leq \sum_{t\in T} \left(\frac{\lambda^{avg}_{f^{(t)}}(X)}{\sigma_n(\boldsymbol{W}_1^{(t)})} + \delta \right),$$

for $|T| = \Omega\left(\operatorname*{avg}_{t\in T} \left(\frac{\lambda^{max}_{f^{(t)}}(X)}{\sigma_n(\boldsymbol{W}_1^{(t)})\,\delta} \right)^2 \right)$, where $\sigma_1(\boldsymbol{W}_1^{(t)}) \geq \cdots \geq \sigma_n(\boldsymbol{W}_1^{(t)})$ are singular values of $\boldsymbol{W}_1^{(t)}$.

Theorem 1 thus shows that lower complexity learners will have a shorter (normalized) bias trajectory. If $\{\epsilon_{f^{(t)}}(\boldsymbol{x}^{(t)}, y^{(t)})\}_{t\in T}$ remain approximately constant, the trajectory length will grow linearly with the average Lipschitz constant of the learner close to the training data.

Why we focus on the 1st layer bias. $\boldsymbol{b}_1$ can be interpreted as the input of the second layer when the 1st layer's output is zeroed out at the end of the forward pass. Then, via the chain rule, the gradient of $\ell(f(\boldsymbol{x}), y)$ with respect to $\boldsymbol{b}_1$ is directly related with the gradient of $f_d \circ \cdots \circ f_2(\boldsymbol{x})$ with respect to $\boldsymbol{x}$ and thus with the Lipschitz constant of the same sub-network. This also explains why the singular values of $\boldsymbol{W}_1$ appear in the bound: since the gradient of $\boldsymbol{b}_1$ does not yield information about $\boldsymbol{W}_1$, we account for it separately. Alternatively, as explained in Appendix C.2, the 1st layer Lipschitz constant can be controlled in terms of the change in parameters. We also note that an identical argument can be utilized to connect the gradient of $\boldsymbol{b}_{l-1}$ with the Lipschitz constant of $f_d \circ \cdots \circ f_l(\boldsymbol{x})$.

Understanding the normalization. The normalization by $\alpha_t\, \epsilon_{f^{(t)}}(\boldsymbol{x}^{(t)}, y^{(t)})$ renders the bound independent of the learning rate α_t as well as of how well the network fits the training data. For

instance, for a mean-squared error (MSE) and a binary cross-entropy (BCE) loss,

$$\ell_{\text{MSE}}(\hat{y}, y) = \frac{(\hat{y} - y)^2}{2} \quad \text{and} \quad \ell_{\text{BCE}}(\hat{y}, y) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}),$$

with $y, \hat{y} \in \mathbb{R}$ and $y, \hat{y} \in [0, 1]$, respectively, we have

$$\epsilon_f(\boldsymbol{x}, y) = |f(\boldsymbol{x}) - y| \quad \text{and} \quad \epsilon_f(\boldsymbol{x}, y) = \frac{1}{|1 - y - f(\boldsymbol{x})|}.$$

In both cases, $\epsilon_f(\boldsymbol{x}, y)$ measures the distance between the true label and the network's output.

Lower bound on the interval length. The requirement that $|T|$ is not too small ensures that the stochasticity of SGD is averaged-out: over a sufficiently large window, the trajectory length becomes independent of which point was sampled in each iteration.

Applicability to other architectures. Beyond fully-connected layers, Theorem 1 directly applies to layers that involve weight sharing and/or sparsity constraints, such as convolutional and locally-connected layers, as long as $\boldsymbol{b}_1$ remains non-shared. In addition, the result also holds unaltered for networks that utilize skip connections or max/average pooling after the 1st layer (since, once more, $\boldsymbol{b}_1$ can be seen as the input of $f_d \circ \cdots \circ f_2(\boldsymbol{x})$ if the real input is zero-ed out at the end of the forward pass), as well as for NNs with general element-wise activation functions (see Appendix C.1).

4.2 Corollaries: steady learners, variance of bias, and distance to initialization

Suppose that after some iteration our NN has fit the training data relatively well. We will say that the NN is a "steady learner" if its 1st layer bias is updated slowly:

Definition 1 (Steady learner)**.** *NN $f^{(t)}$ is (τ, φ)-steady if $\|\boldsymbol{b}_1^{(t+1)} - \boldsymbol{b}_1^{(t)}\|_2 \leq \varphi \cdot \alpha_t \cdot \epsilon_{f^{(t)}}(\boldsymbol{x}^{(t)}, y^{(t)})$ for all $t \geq \tau$.*

The following is an immediate corollary (the proof follows from Lemma 1 and Definition 1):

Corollary 1. *A NN $f^{(t)}$ that is (τ, φ)-steady has bounded Lipschitz constant close to any training point $\boldsymbol{x} \in X$: $\lambda_{f^{(t)}}(\mathcal{R}_{\boldsymbol{x}}) \leq \beta\, \varphi$, where $\sigma_1(\boldsymbol{W}_1^{(t)}) \leq \beta$ for every $t \geq \tau$.*

Crucially, the bound of Corollary 1 can be exponentially tighter than the product-of-norms bound λ_f^{prod}: whereas $\beta\varphi$ does not generally depend on depth, $\lambda_f^{\text{prod}}(\mathcal{R}_{\boldsymbol{x}}) = w^d$ when $\|\boldsymbol{W}_l\|_2 = w$.

We will also use Theorem 1 to characterize two other aspects of the training behavior: the parameter variance as well as the distance to initialization. The first corollary shows that the weights of high complexity NNs cannot concentrate close to some local minimum.

Corollary 2 (Variance of bias)**.** *Let $f^{(t)}$ be a d-layer NN with ReLU activations being trained by SGD. Fix $\delta > 0$, let T be a set of iteration indices, write*

$$\epsilon_{f^{(t)}}^{avg}(X) := \operatorname*{avg}_{i=1}^{N} \epsilon_{f^{(t)}}(\boldsymbol{x}_i, y_i) \quad \textit{and} \quad \epsilon_{f^{(t)}}^{var}(X) := \operatorname*{avg}_{i=1}^{N} \Big(\epsilon_{f^{(t)}}(\boldsymbol{x}_i, y_i) - \epsilon_{f^{(t)}}^{avg}(X)\Big)^2,$$

for the mean and variance of the absolute loss derivatives and suppose that $\epsilon_{min} \leq \epsilon_{f^{(t)}}(\boldsymbol{x}^{(t)}, y^{(t)}) \leq \epsilon_{max}$ and $\|\boldsymbol{b}_1^{(t)}\|_2 \leq \beta_1$ for all $t \in T$. With high probability, the bias of the 1st layer will exhibit variance:

$$\operatorname*{avg}_{t \in T} \|\boldsymbol{b}_1^{(t)} - \operatorname*{avg}_{t \in T} \boldsymbol{b}_1^{(t)}\|_2^2 = \Omega\left(\frac{1}{\delta/3 + c} \cdot \left(\operatorname*{avg}_{t \in T} \frac{\alpha_t\, \lambda_{f^{(t)}}^{avg}(X)}{\sigma_1(\boldsymbol{W}_1^{(t)})} - \delta\right)^2\right),$$

where $c = \operatorname*{avg}_{t \in T} \frac{\epsilon_{f^{(t)}}^{var}(X)}{\epsilon_{f^{(t)}}^{avg}(X)^4}$ *and* $|T| = \Omega\left(\left(\frac{\max\{\epsilon_{min}^{-2} - \epsilon_{max}^{-2}, \operatorname*{avg}_{t \in T} \alpha_t \lambda_{f^{(t)}}^{max}(X)/\sigma_1(\boldsymbol{W}_1^{(t)})\}}{\delta}\right)^2 + \beta_1^2\right)$.

Thus, a larger complexity NN will need to fit the training data more closely to achieve the same variance as that of a lower complexity NN.

We can also deduce that the bias will remain closer to initialization for NNs that have a smaller Lipschitz constant:

Corollary 3 (Distance to initialization)**.** *Let $f^{(t)}$ be a d-layer NN being trained by SGD with an MSE loss. Fix $\delta > 0$, some iteration τ, and suppose that $\epsilon_{f^{(t)}}(\boldsymbol{x}^{(t)}, y^{(t)}) \leq \epsilon^{max}_{f^{(t)}}(X)$ for every $t \leq \tau$. With high probability, we have*

$$\|\boldsymbol{b}_1^{(\tau)} - \boldsymbol{b}_1^{(0)}\|_2 \leq \sum_{t=0}^{\tau-1} \left(\frac{\alpha_t \, \epsilon^{max}_{f^{(t)}}(X) \, \lambda^{avg}_{f^{(t)}}(X)}{\sigma_n(\boldsymbol{W}_1^{(t)})} + \delta \right),$$

whenever $\tau = \Omega\left(\mathrm{avg}_{t \in T} \left(\alpha_t \, \epsilon^{max}_{f^{(t)}}(X) \, \lambda^{max}_{f^{(t)}}(X) / \sigma_n(\boldsymbol{W}_1^{(t)}) \, \delta \right)^2 \right)$.

It is important to stress that the latter result is only meaningful in a regression setting. When the BCE loss is utilized, $\epsilon^{\max}_{f^{(t)}}(X) = \max_i 1/(1 - y_i - f^{(t)}(\boldsymbol{x}_i))$ can become unbounded when the classifier is confidently wrong and thus the bound does not have predictive power. This is not an issue for Theorem 1 since the growth of $\epsilon_{f^{(t)}}(\boldsymbol{x}^{(t)}, y^{(t)})$ is evenly matched by that of the numerator $\|\boldsymbol{b}_1^{(t+1)} - \boldsymbol{b}_1^{(t)}\|_2$. However, to decouple the bias update from its normalization, Corollary 3 needs to assume a uniform upper bound on every $\epsilon_{f^{(t)}}(\boldsymbol{x}^{(t)}, y^{(t)})$ and thus becomes vacuous when the latter follows a heavy-tailed distribution. On the contrary, with an MSE loss in place $\epsilon^{\max}_{f^{(t)}}(X) = \max_i |f^{(t)}(\boldsymbol{x}_i) - y_i|$ grows only linearly with the error, rendering the bound more meaningful.

When α_t and $\epsilon^{\max}_{f^{(t)}}(X)$ decay sufficiently fast, the bound depends on $\sigma_n(\boldsymbol{W}_1^{(t)})$ and the (normalized) average Lipschitz constant at and close to initialization. Therefore, the corollary asserts that SGD with an MSE loss can find near solutions if two things happen: the NN fits the data from relatively early on in the training while implementing a low-complexity function (close to the training data).

5 NN complexity far from the training data

Our exploration on the relationship between training and the complexity of the learned function, thus far, focused only on regions of the input space that contain at least one training point. It is natural to ask how the function behaves in empty regions. After all, to make generalization statements we need to ensure that the learned function is globally Lipschitz.

Next, we provide conditions such that a NN that undergoes steady bias updates, as per Definition 1, during training also has low complexity in linear regions that do not contain any training points. Our analysis starts in Section 5.1 by relating the Lipschitz constant of regions in and outside the training data. We then show in Section 5.2 how learners that remain steady while trained with Dropout have a bounded generalization error.

Analysis setup. A central quantity in our analysis is the activation $\boldsymbol{s}_t(\boldsymbol{x})$ associated with each $\boldsymbol{x}$:

$$\boldsymbol{s}_t(\boldsymbol{x}) := \bigotimes_{l=d-1}^{1} \mathrm{diag}\left(\boldsymbol{S}_l^{(t)}(\boldsymbol{x})\right) = \bigotimes_{l=d-1}^{1} \mathbf{1}[f_l \circ \cdots \circ f_1(\boldsymbol{x}, \boldsymbol{w}) > 0] \in \{0,1\}^{n_{d-1} \cdots n_1}$$

Thus, $\boldsymbol{s}_t(\boldsymbol{x})$ is the Kronecker product of all activations when the network's input is $\boldsymbol{x}$.

We will also assume that the learned function $f^{(t)}$ eventually becomes consistent on the training data:

Assumption 1. *There exists $\tau, \gamma > 0$ such that $\boldsymbol{s}_t(\boldsymbol{x}) = \boldsymbol{s}_{t'}(\boldsymbol{x})$ and $\lambda_{f^{(t)}}(\mathcal{R}_{\boldsymbol{x}}) \leq (1 + \gamma) \, \lambda_{f^{(t')}}(\mathcal{R}_{\boldsymbol{x}})$ for all $\boldsymbol{x} \in X$ and $t, t' \geq \tau$.*

Assumption 1 is weaker than assuming convergence: it allows for the parameters of the NN to keep changing as long as the slope and activation pattern on each training point remains similar. Naturally, it is always possible to satisfy it by decaying the learning rate appropriately.

5.1 The Lipschitz constant of empty regions

As we show next, the Lipschitz constant of our neural network can also be controlled in regions whose activation can be written as a combination of activations of training points.

Theorem 2. *Let T be any interval of SGD iterations that satisfies Assumption 1, and suppose that $\sigma_1(\boldsymbol{W}_1^{(t)}) \leq \beta$ for all $t \in T$. Furthermore, denote by*

$$\boldsymbol{S}_T := \left[\boldsymbol{s}_t(\boldsymbol{x}^{(t)})\right]_{t \in T}, \quad \boldsymbol{\varphi}_T := \left[\frac{\|\boldsymbol{b}_1^{(t+1)} - \boldsymbol{b}_1^{(t)}\|_2}{\alpha_t \, \epsilon_{f^{(t)}}(\boldsymbol{x}^{(t)}, y^{(t)})} \right]_{t \in T}, \quad \mu_T := \min_{t \in T} \{ f^{(t)}(\boldsymbol{x}^{(t)}), 1 - f^{(t)}(\boldsymbol{x}^{(t)}) \}$$

the binary matrix whose columns are the neural activations of all points sampled within T, the vector containing the normalized bias updates, and the distance to integrality if a sigmoid is used in the last layer. Select a point $\boldsymbol{x} \in \mathbb{R}^n$ that is not in the training set. For all $t \in T$, the Lipschitz constant of $f^{(t)}$ in $\mathcal{R}_{\boldsymbol{x}}$ is bounded by the following Basis Pursuit problem:

$$\lambda_{f^{(t)}}(\mathcal{R}_{\boldsymbol{x}}) \leq (1+\gamma)\,\beta\,\xi \min_{\boldsymbol{k}} \|\boldsymbol{k} \odot \boldsymbol{\varphi}_T\|_1 \quad \textit{subject to} \quad \boldsymbol{s}_t(\boldsymbol{x}) = \boldsymbol{S}_T\,\boldsymbol{k},$$

where $\odot$ is the Hadamard product, $\xi = \frac{0.25}{\mu_T(1-\mu_T)}$ if a sigmoid is used and $\xi = 1$, otherwise.

To grasp an intuition of the bound, suppose that we are in a regression setting ($\xi = 1$) and that the interval T is large enough so that we have seen all training points. Theorem 2 then implies:

$$\exists\, \boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k} \in X,\ \boldsymbol{s}_t(\boldsymbol{x}) = \boldsymbol{s}_t(\boldsymbol{x}_{i_1}) + \cdots + \boldsymbol{s}_t(\boldsymbol{x}_{i_k}) \implies \lambda_f^{(t)}(\mathcal{R}_{\boldsymbol{x}}) \leq k\,\beta(1+\gamma)\,\|\boldsymbol{\varphi}_T\|_\infty.$$

By itself, Theorem 2 does not suffice to ensure that the function is globally Lipschitz (and thus to derive generalization guarantees). A sufficient condition for the theorem to yield a global bound is that $\boldsymbol{S}_T$ is full rank, but this can occur only when $N \geq \prod_{l=1}^{d-1} n_l$. Section 5.2 will provide a much milder condition for networks trained with Dropout.

5.2 Learners that remain steady with Dropout generalize

Dropout entails deactivating each active neuron independently with probability p. We here consider the variant that randomly deactivates each neuron independently at the end of the forward-pass with probability $1/2$. We focus on binary NN classifiers

$$g^{(t)}(\boldsymbol{x}) := \mathbf{1}\left[f^{(t)}(\boldsymbol{x}) > 0.5\right]$$

trained with a BCE loss, and with the NN's last layer using a sigmoid activation. The empirical and expected classification error is, respectively, given by

$$\mathrm{er}_t^{\mathrm{emp}} = \operatorname*{avg}_{i=1}^{N} \mathbf{1}\left[g^{(t)}(\boldsymbol{x}_i) \neq y_i\right] \quad \text{and} \quad \mathrm{er}_t^{\mathrm{exp}} = \mathrm{E}_{(\boldsymbol{x},y)}\left[\mathbf{1}\left[g^{(t)}(\boldsymbol{x}) \neq y\right]\right].$$

Theorem 3 controls the generalization error in terms of the number $\mathcal{N}(\mathcal{X}; \ell_2, r)$ of ℓ_2 balls of radius $r(X)$ needed to cover the data manifold $\mathcal{X}$. The radius is shown to be larger for more steadily trained classifiers (through $1/\varphi$) and to depend logarithmically on the number of neurons.

Theorem 3. *Let $f^{(t)}$ be (τ, φ)-steady while being trained with a BCE loss and $1/2$-Dropout. Let Assumption 1 hold, choose any $t > \tau$, let $\boldsymbol{s}_t(\boldsymbol{x}) \leq \sum_{i=1}^N \boldsymbol{s}_t(\boldsymbol{x}_i)$ for every $\boldsymbol{x} \in \mathcal{X}$, and define*

$$r(X) := \frac{\min_{i=1}^N |1 - 2f^{(t)}(\boldsymbol{x}_i)|}{c\,\log\left(\sum_{l=1}^{d-1} n_l\right)\varphi},$$

where $c = (1+\gamma)\,\beta\,(1+o(1))/(\mu\,(1-\mu)\,p_{min})$, $p_{min} = \min_{l<d} \min_{i \leq n_l} [\mathrm{avg}_{\boldsymbol{x}\in X}\, \mathit{diag}\left(\boldsymbol{S}_l^{(t)}(\boldsymbol{x})\right)]_i$ is the minimum frequency that any neuron is active before Dropout is applied, $\sigma_1(\boldsymbol{W}_1^{(t)}) \leq \beta$ and $\mu \leq f^{(t)}(\boldsymbol{x}^{(t)}) \leq 1-\mu$ for every $t \geq \tau$. Then, for any $\delta > 0$, with probability at least $1 - \delta - o(1)$, the generalization error is at most

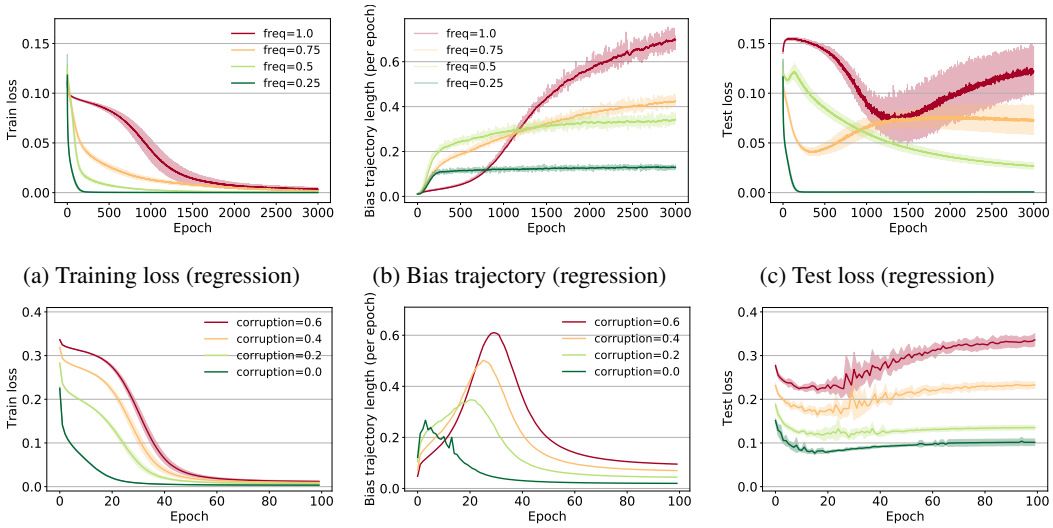
$$|er_t^{emp} - er_t^{exp}| \leq \sqrt{\frac{4\log(2)\,\mathcal{N}(\mathcal{X}; \ell_2, r(X)) + 2\log(1/\delta)}{N}},$$

where $\mathcal{N}(\mathcal{X}; \ell_2, r)$ is the minimal number of ℓ_2-balls of radius r needed to cover $\mathcal{X}$.

Intuitively, with Dropout enabled, training reflects the global behavior of the learned function. The proof entails first approximating the global Lipschitz constant as follows:

$$\lambda_{f^{(t)}}^{\text{steady}} = \frac{\varphi\,c}{4}\,\log\left(\sum_{l=1}^{d-1} n_l\right) \quad \text{with} \quad \lambda_{f^{(t)}} \leq \lambda_{f^{(t)}}^{\text{steady}} \leq \lambda_{f^{(t)}} \cdot \frac{c}{4}\log\left(\sum_{l=1}^{d-1} n_l\right).$$

We then invoke a robustness argument [15, 16] to control the generalization error. Note that the bound above comes in sharp contrast with the product-of-norms bound λ_f^{prod}, which grows exponentially



(a) Training loss (regression) (b) Bias trajectory (regression) (c) Test loss (regression)

(d) Training loss (CIFAR classif.) (e) Bias trajectory (CIFAR classif.) (f) Test loss (CIFAR classif.)

Figure 1: Training behavior of MLP (top) and CNN (bottom) solving a task of increasing complexity (green→red): fitting a function of increasing spatial frequency (top) and classifying CIFAR images with increasing label corruption (bottom). In accordance with Theorem 1, the per epoch bias trajectory (middle subfigures) is longer when the network is asked to fit a more complex training set.

with d and can be arbitrary worse that the actual Lipschitz constant (there exists parameters such that $\lambda_f^{\text{prod}}/\lambda_f = \infty$).

Understanding the assumptions made. The main requirement posed by Theorem 3 is that $\boldsymbol{s}_t(\boldsymbol{x}) \leq \sum_{\boldsymbol{x}\in X} \boldsymbol{s}_t(\boldsymbol{x})$ for every $\boldsymbol{x} \in \mathcal{X}$. In contrast to Theorem 2, the latter can be satisfied even when N is very small, e.g., if there exist some training point for which all neurons are active. On the other hand, the assumption will not hold when some entries of $\boldsymbol{s}_t(\boldsymbol{x})$ are never activated after iteration τ. Unfortunately, little can be said about the global behavior of $f^{(t)}$ if there are neurons that are not periodically active (which would also imply $p_{min} = 0$). We argue however that such neurons can be eliminated without any harm as, by definition, they are not affecting the network's output after τ.

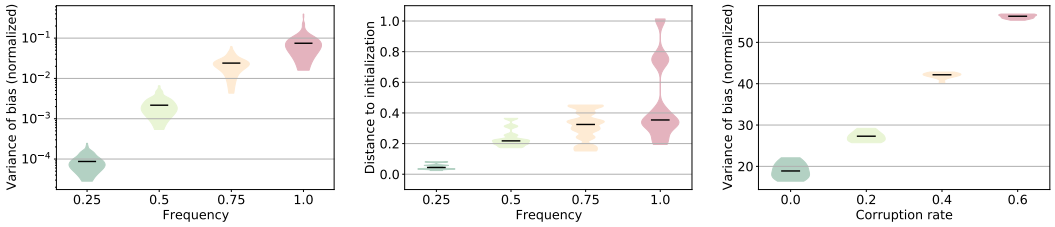
Dependence on the classifier's confidence. According to Theorem 3, the best generalization is attained when the classifier has some certainty about its decisions on the training set (so that $|1 - 2f^{(t)}(\boldsymbol{x}_i)| = \Omega(1)$), while also not being overconfident (so that $\mu\,(1-\mu) = O(1)$).

Dependence on the data distribution and the number of parameters. A key property of the bound is that it depends on the intrinsic dimension of the data rather than the ambient dimension. For instance, if $\mathcal{X}$ is a C_M-regular k-dimensional manifold with $C_M = O(1)$, the covering number is given by $\mathcal{N}(\mathcal{X}; \ell_2, r(X)) = (C_M/r(X))^k$ [67] implying that $N = O(r(X)^{-k})$ training points suffice to ensure generalization. The latter grows poly-logarithmically with the number of neurons $\sum_{l=1}^{d-1} n_l$ (and thus also with the number of parameters) when $\varphi\, c = O(1)$. On the contrary, if λ_f^{prod} was used instead of $\lambda_f^{\text{steady}}$ in our proof, then the radius would be exponentially smaller: e.g., if $\|\boldsymbol{W}_l\|_2 = w$ then $r(X) = O(w^{-d})$ implying a $N = O(w^{dk})$ sample complexity. Other examples of data distributions with covering numbers that grow polynomially with k include rank-k Gaussian mixture models [68], and k-sparse signals under a dictionary [69] (see discussion in [16]).

6 Experiments

We test our theoretical results in the context of two tasks:

Task 1. Regression of a sinusoidal function with increasing frequency. In this toy problem, a multi-layer perceptron (MLP) is tasked with fitting a randomly-sampled 2D sinusoidal function with increasing frequency (0.25,0.5,0.75,1) isometrically embedded in a 10-dimensional space. More details can be found in Appendix B.1. The setup allows us to test our results while precisely



(a) Variance (regression) (b) Distance to init. (regression) (c) Variance (CIFAR classif.)

Figure 2: A closer inspection of how the bias is updated. The variance is computed over the last 10 epochs. As seen, the bias of higher complexity NNs varies more close to convergence (Corollary 2). Further, with an MSE loss, high complexity NNs may veer off further from initialization (Corollary 3).

controlling the complexity of the ground-truth function: fitting a low-frequency function necessitates a smaller Lipschitz constant than a high-frequency one. We trained an MLP with 5 layers consisting entirely of ReLU activations and with the 1st layer weights being identity. We repeated the experiment 10 times, each time training the network with SGD using a learning rate of 0.001 and an MSE loss until it had fit the sinusoidal function at 100 randomly generated training points.

Task 2. CIFAR classification under label corruption. In our second experiment, we trained a convolutional neural network (CNN) to classify 10000 images from the 'dog' and 'airplane' classes of CIFAR10 [70]. We focus on binary classification to remain consistent with the theory. Inspired by [1], we artificially increase the task complexity by randomly corrupting a (0, 0.2, 0.4, 0.6) fraction of the training labels. Thus, a higher corruption implies a larger complexity function. Differently from the 1st task, we used a CNN with 2 convolutional layers featuring ReLU activations in intermediate layers and a sigmoid activation in the last. We repeated the experiment 8 times, each time training the network with SGD using a BCE loss and a learning rate of 0.0025.

As similar studies have reported [1, 3], Figure 1 shows that training slows down as the complexity of the fitted function increases—though, we lack conclusive mathematical evidence that justifies this phenomenon. Figures 1b and 1e depict the per-epoch bias trajectory (i.e., $\sum_{t \in T} \|\boldsymbol{b}_1^{(t+1)} - \boldsymbol{b}_1^{(t)}\|_2 / \alpha_t \cdot \epsilon_{f^{(t)}}(\boldsymbol{x}^{(t)}, y^{(t)})$ within every epoch T). According to Theorem 1, this measure captures the average Lipschitz constant $\lambda_{f^{(t)}}^{\text{avg}}$ of the NN during each epoch and across all training points. In agreement with our theory, the bias trajectory is significantly longer when fitting higher complexity functions (note that the length of the total trajectory is the integral of the depicted curve). Moreover, as shown in Fig 1c, the trajectory length also correlates with the loss of the network on a held-out test set, with longer trajectories consistently corresponding to poorer test performance.

We proceed to examine more closely the behavior of $\boldsymbol{b}_1^{(t)}$ during training. Figures 2a and 2b corroborate the claims of Corollaries 2 and 3, respectively: when fitting a lower complexity function and an MSE loss is utilized, the bias will remain more stable (here we show the variance in the last 10 epochs) and closer to initialization. The same variance trend can be seen in Figure 2c for image classification. Though the distance-to-initialization analysis is not applicable to the classification setting (due to the BCE gradient being unbounded), we include the figure in Appendix B.2 for completeness.

7 Conclusion

This paper showed that the training behavior and the complexity of a NN are interlinked: networks that fit the training set with a small Lipschitz constant will exhibit a shorter trajectory and their parameters will vary less. Though our study is of primarily theoretical interest, our results support for the hypothesis that favoring NNs that exhibit good training behavior can be a useful bias towards models that generalize well.

At the same time, there are many things we do not understand: what is the effect of optimization algorithms and of batching on the connection between complexity and training behavior? Does layer normalization play a role? What can be glimpsed by the trajectory of other parameters (i.e., beyond $\boldsymbol{b}_1$)? We believe that a firm understanding of these questions will be essential in fleshing out the interplay between training, NN complexity, and generalization.

References

[1] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[2] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[3] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pages 233–242. PMLR, 2017.

[4] Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. Sgd learns over-parameterized networks that provably generalize on linearly separable data. In *International Conference on Learning Representations*, 2018.

[5] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

[6] Olivier Bousquet and André Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526, 2002.

[7] Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pages 1225–1234. PMLR, 2016.

[8] Ilja Kuzborskij and Christoph Lampert. Data-dependent stability of stochastic gradient descent. In *International Conference on Machine Learning*, pages 2815–2824. PMLR, 2018.

[9] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pages 2924–2932, 2014.

[10] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2847–2854, 2017.

[11] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*, 2018.

[12] Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. Bounding and counting linear regions of deep neural networks. In *International Conference on Machine Learning*, pages 4558–4566. PMLR, 2018.

[13] Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In *International Conference on Machine Learning*, pages 2596–2604. PMLR, 2019.

[14] Ulrike von Luxburg and Olivier Bousquet. Distance-based classification with lipschitz functions. *J. Mach. Learn. Res.*, 5:669–695, 2004.

[15] Huan Xu and Shie Mannor. Robustness and generalization. *Machine learning*, 86(3):391–423, 2012.

[16] Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017.

[17] 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.

[18] Patrick L Combettes and Jean-Christophe Pesquet. Lipschitz certificates for layered network structures driven by averaged activation operators. *SIAM Journal on Mathematics of Data Science*, 2(2):529–557, 2020.

[19] 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. URL `https://proceedings.neurips.cc/paper/2020/file/5227fa9a19dce7ba113f50a405dcaf09-Paper.pdf`.

[20] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.

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

[22] Peter L Bartlett, Dylan J Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6241–6250, 2017.

[23] Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations*, 2018.

[24] Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. In *Conference On Learning Theory*, pages 297–299. PMLR, 2018.

[25] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning*, pages 254–263. PMLR, 2018.

[26] Konstantinos Pitas, Andreas Loukas, Mike Davies, and Pierre Vandergheynst. Some limitations of norm based generalization bounds in deep neural networks. *CoRR*, abs/1905.09677, 2019. URL `http://arxiv.org/abs/1905.09677`.

[27] Kevin Scaman and Aladin Virmaux. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 3839–3848, 2018.

[28] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. In *NeurIPS*, 2019.

[29] Dongmian Zou, Radu Balan, and Maneesh Singh. On lipschitz bounds of general convolutional neural networks. *IEEE Transactions on Information Theory*, 66(3):1738–1759, 2019.

[30] Fabian Latorre, Paul Thierry Yves Rolland, and Volkan Cevher. Lipschitz constant estimation for neural networks via sparse polynomial optimization. In *8th International Conference on Learning Representations*, number CONF, 2020.

[31] Tong Chen, Jean-Bernard Lasserre, Victor Magron, and Edouard Pauwels. Semialgebraic optimization for lipschitz constants of relu networks. *arXiv e-prints*, pages arXiv–2002, 2020.

[32] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: scalable certification of perturbation invariance for deep neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 6542–6551, 2018.

[33] Guang-He Lee, David Alvarez-Melis, and Tommi S Jaakkola. Towards robust, locally linear deep networks. In *International Conference on Learning Representations*, 2018.

[34] Cem Anil, James Lucas, and Roger Grosse. Sorting out Lipschitz function approximation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 291–301. PMLR, 09–15 Jun 2019.

[35] Patricia Pauli, Anne Koch, Julian Berberich, Paul Kohler, and Frank Allgower. Training robust neural networks using lipschitz bounds. *IEEE Control Systems Letters*, pages 1–1, 2021. doi: 10.1109/LCSYS.2021.3050444.

[36] Zac Cranko, Simon Kornblith, Zhan Shi, and Richard Nock. Lipschitz networks and distributional robustness. *arXiv preprint arXiv:1809.01129*, 2018.

[37] Adam M. Oberman and Jeff Calder. Lipschitz regularized deep neural networks converge and generalize. *CoRR*, abs/1808.09540, 2018. URL `http://arxiv.org/abs/1808.09540`.

[38] 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.

[39] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[40] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[41] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[42] Colin Wei, Sham Kakade, and Tengyu Ma. The implicit and explicit regularization effects of dropout. In *International Conference on Machine Learning*, pages 10181–10192. PMLR, 2020.

[43] Raman Arora, Peter Bartlett, Poorya Mianjy, and Nathan Srebro. Dropout: Explicit forms and capacity control. *arXiv preprint arXiv:2003.03397*, 2020.

[44] Wenlong Mou, Yuchen Zhou, Jun Gao, and Liwei Wang. Dropout training, data-dependent regularization, and generalization bounds. In *International Conference on Machine Learning*, pages 3645–3653. PMLR, 2018.

[45] Poorya Mianjy and Raman Arora. On convergence and generalization of dropout training. *Advances in Neural Information Processing Systems*, 33, 2020.

[46] David McAllester. A pac-bayesian tutorial with a dropout bound. *arXiv preprint arXiv:1307.2118*, 2013.

[47] Wei Gao and Zhi-Hua Zhou. Dropout rademacher complexity of deep neural networks. *Science China Information Sciences*, 59(7):1–12, 2016.

[48] Ke Zhai and Huan Wang. Adaptive dropout with rademacher complexity regularization. In *International Conference on Learning Representations*, 2018.

[49] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL `http://proceedings.mlr.press/v28/wan13.html`.

[50] Sepp Hochreiter and Jargen Schmidhuber. Flat Minima. *Neural Computation*, 9(1):1–42, 01 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.1.1. URL `https://doi.org/10.1162/neco.1997.9.1.1`.

[51] 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.

[52] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *NIPS*, 2017.

[53] 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.

[54] Samuel L Smith and Quoc V Le. A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*, 2018.

[55] Chiyuan Zhang, Qianli Liao, Alexander Rakhlin, Brando Miranda, Noah Golowich, and Tomaso Poggio. Theory of deep learning iib: Optimization properties of sgd. *arXiv preprint arXiv:1801.02254*, 2018.

[56] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR, 2017.

[57] Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

[58] Haowei He, Gao Huang, and Yang Yuan. Asymmetric valleys: Beyond sharp and flat local minima. *arXiv preprint arXiv:1902.00744*, 2019.

[59] Itay Safran and Ohad Shamir. On the quality of the initial basin in overspecified neural networks. In *International Conference on Machine Learning*, pages 774–782. PMLR, 2016.

[60] C Daniel Freeman and Joan Bruna. Topology and geometry of half-rectified network optimization. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[61] Yuanzhi Li and Yang Yuan. Convergence analysis of two-layer neural networks with relu activation. In *NIPS*, 2017.

[62] Quynh Nguyen, Mahesh Chandra Mukkamala, and Matthias Hein. On the loss landscape of a class of deep neural networks with no bad local valleys. In *International Conference on Learning Representations*, 2018.

[63] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2018.

[64] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.

[65] Chao Ma, Qingcan Wang, Lei Wu, et al. Analysis of the gradient descent algorithm for a deep neural network model with skip-connections. *arXiv e-prints*, pages arXiv–1904, 2019.

[66] Remigijus Paulavičius and Julius Žilinskas. Analysis of different norms and corresponding lipschitz constants for global optimization. *Technological and Economic Development of Economy*, 12(4):301–306, 2006.

[67] Nakul Verma. Distance preserving embeddings for general n-dimensional manifolds. In *Conference on Learning Theory*, pages 32–1. JMLR Workshop and Conference Proceedings, 2012.

[68] Shahar Mendelson, Alain Pajor, and Nicole Tomczak-Jaegermann. Uniform uncertainty principle for bernoulli and subgaussian ensembles. *Constructive Approximation*, 28(3):277–289, 2008.

[69] Raja Giryes, Guillermo Sapiro, and Alex M Bronstein. Deep neural networks with random gaussian weights: A universal classification strategy? *IEEE Transactions on Signal Processing*, 64(13):3444–3457, 2016.

[70] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See: (a) Section 4.2 about the in-applicability of Corollary 3 to NNs trained with BCE loss, (b) Sections 5.1 and 5.2 for the theoretical assumptions required to guarantee generalization, and (c) Section 7 for aspects not covered by the theory.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We could not identify any negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] The proofs can be found in Appendix A
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] The experiments that we conducted are relatively small for deep learning standards (1 GPU used for a few dozen hours)
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]