Data Dependent Generalization Bounds for Neural Networks with ReLU

Anonymous authors
Paper under double-blind review

Abstract

We try to establish that one of the correct data dependent quantities to look at while trying to prove generalization bounds, even for overparameterized neural networks, are the gradients encountered by stochastic gradient descent while training the model. If these are small, then the model generalizes. To make this conclusion rigorous, we weaken the notion of uniform stability of a learning algorithm in a probabilistic way by positing the notion of almost sure (a.s.) support stability and showing that algorithms that have this form of stability have generalization error tending to 0 as the training set size increases. Further, we show that for Stochastic Gradient Descent to be a.s. support stable we only need the loss function to be a.s. locally Lipschitz and locally Smooth at the training points, thereby showing low generalization error with weaker conditions than have been used in the literature. We then show that Neural Networks with ReLU activation and a doubly differentiable loss function possess these properties. Our notion of stability is the first data dependent notion to be able to show good generalization bounds for non-convex functions with learning rates strictly slower than $1/t$ at the t-th step. Finally, we present experimental evidence to validate our theoretical results.

1 Introduction

The low generalization error of Deep neural networks is now a well known empirical result (c.f. Jin et al. (2020)), but theoretical explanations of this behaviour are still unsatisfactory. Chatterjee & Zielinski (2022) articulated the main question as follows: *why (or when) do neural networks generalize well when they have sufficient capacity to memorize their training set?* Although a number of formalisms have been used in an attempt to derive theoretical bounds on the generalization error, e.g., VC dimension (Vapnik, 1998), Rademacher complexity (Bartlett & Mendelson, 2003) and uniform stability (Bousquet & Elisseeff, 2002) but, as Zhang et al. (2017) showed, all of these fail to resolve the conundrum thrown up by overly parameterized deep neural networks. One clear failing identified in Zhang et al. (2017) was that many of these notions were data independent. A simple counterexample provided by Zhang et al. (2017) clearly established that a data independent notion was bound to fail to distinguish between data distributions on which deep NNs will generalize well and those on which they will not. Subsequent research on generalization has tried to tackle the question that Chatterjee & Zielinski (2022) formulated as follows: *For a neural network, is there a property of the dataset that controls the generalization error (assuming the size of the training set, architecture, learning rate, etc are held fixed)?* We give an affirmative answer to this question in one direction: We identify data dependent quantities, namely the *Training Lipschitz constant* (L_S) *and the Test Lipschitz constant* (L_g), that control the generalization error bound of an NN trained using SGD. In simple terms, we show that if the data distribution is such that the gradients computed by SGD during training *and* the gradients on the test points are bounded, the model will generalize well. Our techniques are able to rigorously handle nonlinearities like RELU and work for non-convex loss functions. We also allow for a learning rate that is asymptotically strictly slower than $\theta(1/t)$ at the t-th step of SGD.

Our work is within the theoretical paradigm of stability. We asked the question, *Is there an appropriate version of stability that is flexible enough to incorporate dataset properties and can also adapt to most neural networks?* In a partial answer to this question, we introduce a notion called *almost sure (a.s.) support*

Paper	**Number of Epochs**	**Step Size**	**Neural Network Type**	**Key Assumptions**
Hardt et al. (2016)	$O(m^c), c > 1/2$	$O(1/t)$	No restrictions	No data dependence.
Kuzborskij & Lampert (2018)	1 epoch	$O(1/t)$	No restrictions	Bounded Hessian
Lei & Ying (2020)	$O(1)$	$O(1/t)$	No restrictions	Strongly convex objective but non convex loss function
Charles & Papailiopoulos (2018)	$O(m)$	$O(1)$	1-layered networks with leaky ReLU or linear	PL and QG growth conditions
Lei et al. (2022)	$O(m)$	$O(1)$	1-layered networks with smooth activation functions	Smooth loss function, Bound in expectation, lower bound on number of parameters $n > m^{\frac{3}{(\alpha+1)}}, \alpha > 0$
Our Paper	$O(\ln m)$	$O\left(1/t^{1-\frac{c}{\rho(\tau,m)}}\right)$, $c \in (0,1)$	No restrictions	Bounded Spectral Complexity

Table 1: Recent related works addressing the question of generalization error and stability of neural networks in comparison to the results in this paper.

stability which is a data dependent probabilistic weakening of uniform stability. Following the suggestions made by Zhang et al. (2017), data dependent notions of stability were defined in Kuzborskij & Lampert (2018) and Lei & Ying (2020) as well. However, a.s. support stability is a more useful notion on three counts: it can handle SGD learning rates that are strictly slower than $\theta(1/t)$, its initial learning rate is much higher, and, while these past works bound generalization error in expectation, a.s. support stability can be used to show high probability bounds on generalization error. But, over and above these technical benefits, our main contribution here is the identification of the data dependent Lipschitz constants as a key indicator of generalization. A brief description of recent related works are summarized in table 1.

Technically, our approach has several ingredients. We begin by widening the scope of the generalization results of Feldman & Vondrak (2018; 2019a) by showing that they hold for algorithms that have a.s. support stability. To do so, we prove a mild generalization of McDiarmid's Inequality that could be of independent interest. Secondly, we show that when SGD is used to minimize a non-convex function, its a.s. support stability can be bounded in terms of the gradients encountered during the training and testing. This naturally leads to the idea of the two data dependent constants mentioned above, the Training and Test Lipschitz constants. Thirdly we show that NN with ReLU activation will be *locally* Lipschitz and smoothness (w.r.t. to the parameter set) with probability 1 as long as the unknown distribution places probability 0 on sets of Lebesgue measure 0, a constraint that holds for most natural data distributions. This directly implies the existence of the constants that control the rate of convergence to 0 of the generalization error. The Training Lipschitz constant (L_S) depends on the training set picked, and the Test Lipschitz constant (L_g) is a function of data distribution. We show bounds on these constants based on the spectral property of NN, arguing that although these constants are always small for small sized networks, they could also be small for large networks depending on the parameter (weights) values. Our theory can also help explain why the Neural Tangent Kernel generalizes well because of a good bound on Lipschitz constants, although we do not go into detail in this work. We also consider the randomly labelled example suggested by Zhang et al. (2018) and show that *the assumption of bounded Lipschitz constants w.r.t training set size breaks.* We show that the Lipschitz constants are high and keep increasing with the training set size of this bad distribution. Thereby our theory does not guarantee any generalization in these cases, which is as it should be.

We note that although we can say that when the data dependent Training and Test Lipschitz constants are small, our results guarantee good generalization performance, we do not establish that this condition is necessary.

In particular our contributions are:

- In Section 3 we define a new notion of stability called *a.s. support stability* and show in Theorem 3.2 that algorithms with a.s. support stability $o(1/\log^2 m)$ have generalization error tending to 0 as $m \to \infty$ where m is the size of the training set.

- In Section 4 we show that if we run stochastic gradient descent on a parameterized optimization function that is only *locally* Lipschitz and *locally* smooth in the parameter space and has data dependent Training and Test Lipschitz constants that are bounded with probability 1, then the replace-one error is bounded even if the training is conducted for $c \log m$ epochs. This implies (Corollary 4.5) that any learning algorithm trained this way has a generalization error that goes to 0 as $m \to \infty$. If SGD is trained for τ epochs, the slowest learning rate for which our result holds is $\alpha_0/t^{1-\rho(\tau,m)}$ at step t where $\rho(\tau, m)$ is $O\left(\ln\ln m/\left(\ln\tau + \ln m\right)\right)$ for an appropriately selected value of α_0. For all reasonable values of m and τ, this marks a significant slowing down of the training rate from $\theta(1/t)$.

- In Section 5.1 we show that the output of an NN with ReLU activations when used with a doubly differentiable loss function is locally Lipschitz and locally smooth in the parameter space for all inputs except those from a set of Lebesgue measure 0 (Theorem 5.2). We also show a spectral norm based bound for Training and Test Lipschitz constants (L_g and L_S). We then experimentally verify our theory showing that the bounded Lipschitz condition holds and we also plot the generalization error.

- Then in Section 5.2 we experimentally analyze the Test Lipschitz constant (L_g) for random labelling setting suggested by Zhang et al. (2018) and conclude the Test Lipschitz constant is actually not bounded and increase with the training set size. We relate this to the high variance of the loss function in random labelling case and hence provide an explanation of which this example cannot be proved, incorrectly, to generalize using our methods.

2 Related Work

Although NNs are known to generalize well in practice, many different theoretical approaches have been tried without satisfactorily explaining this phenomenon, c.f., Jin et al. (2020); Chatterjee & Zielinski (2022). We refer the reader to the work of Jin et al. (2020) which presents a concise taxonomy of these different theoretical approaches. Several works seek to understand what a good theory of generalization should look like, c.f. Kawaguchi et al. (2017); Chatterjee & Zielinski (2022). Our own work falls within the paradigm that seeks to use notions of algorithmic stability to bound generalization error that began with Vapnik & Chervonenkis (1974) but gathered steam with the publication of the work by Bousquet & Elisseeff (2002).

The applicability of the algorithmic stability paradigm to the study of generalization error in NNs was brought to light by Hardt et al. (2016) who showed that functions optimized via Stochastic Gradient Descent have the property of uniform stability defined by Bousquet & Elisseeff (2002), implying that NNs should also have this property. Subsequently, there was renewed interest in uniform stability and a sequence of papers emerged using improved probabilistic tools to give better generalization bounds for uniformly stable algorithms, e.g., Feldman & Vondrak (2018; 2019a) and Bousquet et al. (2020). Some other works, e.g. Klochkov & Zhivotovskiy (2021), took this line forward by focussing on the relationship of uniform stability with the excess risk. However, the work of Zhang et al. (2017) complicated the picture by pointing out examples where the theory suggests the opposite of what happens in practice. This led to two different strands of research. In one thread an attempt was made to either discover those cases where uniform stability, (e.g. Charles & Papailiopoulos (2018)), or to show lower bounds on stability that ensure that uniform stability does not exist, (e.g. Zhang et al. (2022)). The other strand of research, our work falls in this category, focuses on weakening the notion of uniform stability, specifically by making it data dependent, thereby following the suggestion made by Zhang et al. (2017). Kuzborskij & Lampert (2018) defined "on-average stability" which is weaker than our definition of a.s. support stability. Consequently, their definition leads to a weaker in-expectation bound on the generalization error where the expectation is over the training set as well as the random choices of the algorithm. Our Theorem 3.2, on the other hand, provides a sharp concentration bound on the choice of the training set. Lei & Ying (2020) define an "on-average model stability" that requires the average replace-one error over all the training points to be bounded in expectation. While their smoothness requirements are less stringent, the problem is that their generalization results are all relative to the optimal choice of the weight vector, which implies a high generalization error in case of early stopping.

3 a.s. Support Stability and Generalization

In this section, we present a weakening of the notion of uniform stability defined by Bousquet & Elisseeff (2002) and show that exponential concentration bounds on the generalization error can be proved for learning algorithms that have this weaker form of stability.

3.1 Terminology

Let $\mathcal{X}$ and $\mathcal{Y}$ be the input and output spaces respectively. We assume we have a training set $S \in \mathcal{Z}^m$ of size m where each point is chosen independently at random from an unknown distribution D over $\mathcal{Z} \subset \mathcal{X} \times \mathcal{Y}$. For $z = (x, y) \in \mathcal{Z}$ we will use the notation x_z to denote x and y_z to denote y. Let $\mathcal{R}$ be the set of all finite strings on some finite alphabet, and let us call the elements of $\mathcal{R}$ *decision strings* and let us assume that there is some probability distribution D_r according to which we will select r randomly from $\mathcal{R}$. Further, let $\mathcal{F}$ be the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$. In machine learning settings we typically compute a map from $\mathcal{Z}^m \times \mathcal{R}$ to $\mathcal{F}$. We will denote the function computed by this map as $A_{S,r}$. Since the choice of S and r are both random, $A_{S,r}$ is effectively a random function and can also be thought of as a randomized algorithm.

Given a bound $M > 0$, we assume that we are given a bounded *loss function* $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, M]$. We define the *risk* of A_S as

$$R(A, S, r) = \mathrm{E}_{z \sim D} \left[\ell(A_{S,r}(x_z), y_z)\right],$$

where the expectation is over the random choice of point z according to data distribution D. Note that the risk is a random variable since both S and r are randomly chosen. The *empirical risk* of A_S is defined as

$$R_e(A, S, r) = \frac{1}{|S|} \sum_{z \in S} \ell(A_{S,r}(x_z), y_z).$$

We are interested in bounding the *generalization error*

$$|R(A, S, r) - R_e(A, S, r)| .$$

3.2 A Weakening of Uniform Stability

Given $S = \{Z_1, \ldots, Z_m\}$ chosen randomly according to D^m, we choose $\{Z_{1+m}, \ldots, Z_{2m}\}$ also from D^m independently from S and, for each $i \in [m]$, we define

$$S^i = \{Z_1, \ldots, Z_{i-1}, Z_{i+m}, Z_{i+1}, \ldots, Z_m\}.$$

Definition 3.1 (Almost (Sure) Support Stability)**.** We say an algorithm $A_{S,r}$ has *η-almost support stability β with respect to the loss function $\ell(\cdot, \cdot)$* if for $Z_1, \ldots, Z_{2m}$ chosen i.i.d. according to an unknown distribution D defined over $\mathcal{Z}$,

$$\forall i \in [m] : \forall z \in \operatorname{supp}(D) : \mathrm{E}_r \left[|\ell(A_{S,r}(x_z), y_z) - \ell(A_{S^i,r}(x_z), y_z)|\right] \leq \beta$$

with probability $1 - \eta$. If $\eta = 0$ then we say the algorithm has *almost sure (a.s.) support stability β.*

We note that this notion weakens the notion of uniform stability introduced by Bousquet & Elisseeff (2002) by requiring the bound on the difference in losses to hold with a certain probability. This probability is defined over the random choices of $Z_1, \ldots, Z_{2m}$. Besides the condition on the loss is required to hold only for those data points that lie in the support of D. These conditions make almost support stability a *data dependent quantity* on the lines of the suggestion made by Zhang et al. (2017). We also observe that almost support stability is comparable to but stronger than the hypothesis stability of Kearns & Ron (1999) as formulated by Bousquet & Elisseeff (2002).

3.3 Exponential Convergence of Generalization Error

A.s. support stability can be used in place of uniform stability in conjunction with the techniques of Feldman & Vondrak (2019a) to give guarantees on generalization error. In particular, we can derive the following theorem.

Theorem 3.2. *Let $A_{S,r}$ be an algorithm that is symmetric in distribution and has a.s. stability β with respect to the loss function $\ell(\cdot,\cdot)$ such that $0 \le \ell(A_{S,r}(x_z), y_z) \le 1$ for all $S \in \mathcal{Z}^m$, for all $r \in \mathcal{R}$ and for all $z = (x_z, y_z) \in \mathcal{Z}$. Then, there is a constant $c > 0$ s.t. for any $m \ge 1$ and $\delta \in (0,1)$, with probability $1-\delta$,*

$$E_r\left[R(S,r) - R_e(S,r)\right] \le c\left(\beta \log(m) \log\left(\frac{m}{\delta}\right) + \sqrt{\frac{\log(1/\delta)}{m}}\right).$$

Proof outline. We give a high-level outline here. Feldman & Vondrak (2019b) used two steps to get a better generalization guarantee. The first step is range reduction, where the range of the loss function is reduced. For this, they define a new clipping function in Lemma 3.1 Feldman & Vondrak (2019a) which preserves uniform stability and hence it will also preserve a.s. support stability. They also use uniform stability in Lemma 3.2 Feldman & Vondrak (2019a) where they show the shifted and clipped function will still be stable which is done by applying McDiarmid's inequality to β sensitive functions. Here use a modification of McDiarmid's Inequality (Lemma A.2 given in Appendix A) to get bounds for a.s. support stability. The second step is dataset size reduction (as described in Section 3.3 Feldman & Vondrak (2019a)) which will remain the same for a.s. support stability as this only involves stating the result for a smaller dataset and the probability, and then taking a union bound. Therefore both steps of the argument given in Feldman & Vondrak (2019a) go through for a.s. support stability.

4 Proof of a. s. Support Stability of Stochastic Gradient Descent

A large family of machine learning algorithms follow a paradigm in which the learned function is parameterized by a vector $\boldsymbol{w} \in \mathbb{R}^n$ for some $n \ge 1$, i.e., we have some fixed function $g : \mathbb{R}^n \times \mathcal{X} \to \mathcal{Y}$. The training set is used to learn a suitable parameter vector $\boldsymbol{w} \in \mathbb{R}^n$ such that the value $g(\boldsymbol{w}, x_z)$ is a good estimate of y_z for all $z \in \mathcal{Z}$. This value of $\boldsymbol{w}$ is learned by running Stochastic Gradient Descent (SGD) using a training set drawn from the unknown distribution. We will say that the size of the training set is m and the algorithm proceeds in *epochs* of m steps each. The parameter vector at step t being denoted $\boldsymbol{w}_t$ for $0 \le t \le \tau \cdot m$, where τ is the total number of epochs during training. To frame the learned function output by this algorithm in the terms defined in Section 3.1, the random decision string r consists of the pair $(\boldsymbol{w}_0, (\pi_0, \ldots, \pi_{\tau-1}))$, i.e., the random initial parameter vector $\boldsymbol{w}_0$ from which SGD begin and the set of τ random permutations used in the τ epochs.

4.1 Some Properties of Parameterized Functions

In Hardt et al. (2016), proving that the learning algorithm derived by SGD is stable requires smoothness and Lipschitz properties of f, but *only* for partial derivatives taken on $\mathbb{R}^n$, i.e., on the parameter space. The requirement there is that *every function* in the family of functions $\{f(\cdot, z) : z \in \mathcal{Z}\}$ is smooth and has a bounded Lipschitz constant. Our key insight is that this requirement is stronger than required. All we need is that *the functions induced by the data points that we pick to train SGD have these properties.* We now present some definitions that encapsulate this idea.

Definition 4.1 (Almost Locally Parameter-Lipschitz functions)**.** Given a set Ω defined over $\mathcal{Z}$, a parameterized function $f : \mathbb{R}^n \times \mathcal{Z} \to \mathbb{R}$ is said to be *β-almost L_l locally Lipschitz w.r.t Ω* if for any $\boldsymbol{w} \in \mathbb{R}^n$ there exists constants $L_l > 0$ and $\epsilon > 0$ such that, with probability β (over the choice of z), for all $\boldsymbol{w'} \in \mathbb{R}^n$, $\|\boldsymbol{w'} - \boldsymbol{w}\| < \epsilon$ implies

$$|f(\boldsymbol{w'}, z) - f(\boldsymbol{w}, z)| \leq L_l \|\boldsymbol{w'} - \boldsymbol{w}\|.$$

Note that L_l can depend on z. If $\beta = 1$ then we say that f is *almost surely locally L_l-parameter Lipschitz* (*a.s. L_l-LPL* for short).

Definition 4.2 (Almost Locally Parameter-Smooth functions)**.** Given a set Ω defined over $\mathcal{Z}$, a parameterized function $f : \mathbb{R}^n \times \mathcal{Z} \to \mathbb{R}$ is said to be *γ-almost K_l-Parameter Smooth w.r.t Ω* for some $\gamma \in (0, 1]$ if for any $\boldsymbol{w} \in \mathbb{R}^n$ there exist constants $K_l > 0$ and $\epsilon > 0$ such that, with probability γ (over the choice of z), for all $\boldsymbol{w'} \in \mathbb{R}^n$, $\|\boldsymbol{w'} - \boldsymbol{w}\| < \epsilon$ implies

$$\|\nabla f(\boldsymbol{w'}, z) - \nabla f(\boldsymbol{w}, z)\| \leq K_l \|\boldsymbol{w'} - \boldsymbol{w}\|.$$

Note that K_l can depend on z. If $\gamma = 1$ then we say that f is *almost surely locally K_l-parameter smooth* (*a.s. K_l-LPS* for short).

If the function (or its gradient) is locally bounded, and, if we only look at this function at a finite number of points ($\boldsymbol{w}$), we get a “global” property within this finite set of points:

Lemma 4.3. *Given $f : \mathbb{R}^n \times \mathcal{Z} \to \mathbb{R}$ we have that (1) if f is bounded and is locally Lipschitz at a finite set of points $A \subset \mathbb{R}^n$ and for a set $\Omega \subseteq \mathcal{Z}$, then there is an $L > 0$ such that for every pair $\boldsymbol{w}, \boldsymbol{w'} \in A$ and $z \in \Omega$*

$$|f(\boldsymbol{w}, z) - f(\boldsymbol{w'}, z)| \leq L \|\boldsymbol{w} - \boldsymbol{w'}\|, and$$

2) if ∇f is bounded and is locally smooth at a finite set of points $A \subset \mathbb{R}^n$ and for a set $\Omega \subseteq \mathcal{Z}$, then there is a $K > 0$ such that for every pair $\boldsymbol{w}, \boldsymbol{w'} \in A$ and $z \in \Omega$

$$\|\nabla f(\boldsymbol{w}, z) - \nabla f(\boldsymbol{w'}, z)\| \leq K \|\boldsymbol{w} - \boldsymbol{w'}\|.$$

The proof is in Appendix B.

Discussion: Local properties imply “Global” properties. SGD trained on a finite training set will encounter a finite number of parameter vectors in its execution, and hence Lemma 4.3 can be used to say that the local properties of a bounded Lipschitz constant and bounded smoothness can be extended to the entire set of parameters encountered by SGD. Specifically, we use the lemma in two ways.

- Setting $\Omega = S$, and taking A to be the set of weights encountered during training over all possible permutations of S, we get the *Training Lipschitz constant L_S* and the *Training Smoothness constant K_S*.
- Setting $\Omega = \operatorname{supp}(D)$ and taking A to be the set of final parameter vectors produced by SGD for each of the possible permutations, we get the *Test Lipschitz constant L_g* which is applicable for each test point.

Note that although we use the term “constant” in their names, all these quantities are random variables that depend on the random choice of the initial weight $\boldsymbol{w}_0$.

4.2 a.s. Support Stability of SGD

We now work towards the a.s. support stability of SGD. First, we state a theorem that bounds the replace-one error of SGD up to a certain number of epochs.

Theorem 4.4. *We are given a labelled data set $\mathcal{Z}$ and probability distribution D defined over it. Suppose we have a parameterized loss function $f : \mathbb{R}^n \times \mathcal{Z} \to \mathbb{R}$ that is a.s. L_l-LPL and a.s. K_l-LPS w.r.t D. Suppose further for some integer $\tau > 0$, for each $i \in [m]$ we run Stochastic Gradient Descent on f for τ epochs* *using a training set S of size m chosen i.i.d. according to D with random choices r and, parallelly, with the same set of random choices r, on a set S^i wherein the i-th data point z_i of S has been replaced by another data point z_i' chosen from D independent of all other random choices and with a decreasing learning rate of $\alpha_t \leq \alpha_0 / t^{(1-\rho(\tau,m))}, \rho(\tau,m) = \frac{\ln \ln m}{\ln \tau + \ln m}$. Then, if for all $t > 0$, we denote by $\boldsymbol{w}_t$ and $\boldsymbol{w}_t'$ the parameter vectors obtained while training with S and S^i respectively, we have constants L_S, K_S and L_g as the Training Lipschitz constant, Training Smoothness constant and Test Lipschitz constant such that with probability 1 over z*

$$E_r \left[f(\boldsymbol{w}_{\tau m}, z) - f(\boldsymbol{w}'_{\tau m}, z) \right] \leq E_r \left[\frac{F(\tau) \cdot U(\alpha_0, K_S, \rho(\tau,m))}{m^{\left(1 - \frac{\alpha_0 K_S}{\rho(\tau,m)}\right)}} \right], \tag{1}$$

where $U(\alpha_0, K_S, \rho(\tau,m)) \leq 1 + \frac{1}{K_S \alpha_0}$, and as $\alpha_0 \to 0, U(\alpha_0, K_S, \rho(\tau,m)) \to 1$, and also $F(\tau) := 2^\tau \alpha_0 \cdot L_S L_g$.

Note that here Expectation will be over random variables L_g, L_S and K_S which are a function of random initialization of initial weights ($\boldsymbol{w}_0$).

Proof outline. The proof follows the lines of the argument presented by Hardt et al. (2016) with the difference that we allow for a probabilistic relaxation of the smoothness conditions in line with our definition of a.s. stability. Also, note that we have to account for an expectation over the random string r and that we have been able to extend the argument to multiple epochs which was not possible in Kuzborskij & Lampert (2018). The complete proof of Theorem 4.4 is in Appendix B.

Data dependence with Training Lipschitz constant and Test Lipschitz constant. A key feature of the bound presented in equation 1 is that the dependence on the data is expressed through the data dependent Training Lipschitz constant L_S and Test Lipschitz constant L_g. Training Lipschitz constant depends on the gradients at training points and the replacement point z_i' which is also picked from the data distribution and Training Lipschitz constant depends on the gradients of the trained network calculated at points from distribution. In general, we expect that if the unknown distribution D has a low variance then the Lipschitz constants will be small. Further, a line of research in the optimization literature has shown that the gradients associated with SGD decay as training proceeds, even for non-convex loss functions, c.f. Section 4 of Bottou et al. (2018). Therefore we can conclude that the a.s. stability bound of equation 1 is closely connected to the data distribution and is likely to be useful for cases where SGD returns a meaningful solution and vacuous for bad cases like the one presented by Zhang et al. (2017).

Corollary 4.5. *For a learning algorithm that is symmetric in distribution and trained as described in the statement of Theorem 4.4 under the condition that K_S is a constant $\forall r$ w.r.t. m and $E_r[L_g L_S]$ is also constant w.r.t m then there is a constant $c \in (0,1)$ that depends on α_0 and K_S such that for training at most till $c \log m$ epochs the expectation of the generalization error of the algorithm taken over the random choices of the algorithm decreases as $O\left(\frac{\log(m)^2}{m^\epsilon} + \sqrt{\left(\frac{\log(m)}{m}\right)}\right)$ with probability at least $1 - 1/m$ over the choice of the training set if $\alpha_0 < \rho(\tau,m)/(K_S)$, where $\epsilon > c$ and $\in (0,1)$ is some constant.*

Proof. It is easy to check from Theorem 4.4 that with the conditions given in the statement of Corollary 4.5 the learning algorithm has a.s. support stability β where β is $o(1/m)$ if $\alpha_0 < \rho(\tau,m)/K_S$.

We can therefore apply Theorem 3.2 with $\delta = 1/m$ to get the result. □

We would like to highlight the importance of $\rho(\tau,m)$, notice that $\rho(\tau,m) = O\left(\frac{\ln \ln m}{\ln \tau + \ln m}\right)$ so it is decreasing in m but very slowly, so because of this even for datasets with say 10^6 points, it turns out to be $\rho(\tau,m) = 0.19$, so this helps us achieve a descent value of initial learning rate $\alpha_0 = 0.19$ and also a much slower decay of learning rate $\alpha_t = \alpha_0 / t^{0.81}$. If we directly compare this corollary with Theorem 4 of Kuzborskij & Lampert (2018), we note that their analysis requires an $\alpha_t = \alpha_0 / t$ and $\alpha_0 = O\left(1/\ln(m)^2\right)$ which is a very steep decay in learning rate and a very small value of alpha, just to compare for $m = 10^6$, for Kuzborskij & Lampert (2018) $\alpha_0 = 0.005$. Also, their analysis bounds generalization error only up to the end of a single epoch whereas we can bound the error well beyond that. Kuzborskij & Lampert (2018) also require the Hessian to have a bounded Lipschitz constant, i.e., the third derivative of the loss function has to be bounded. We do not need any such constraint.

5 Neural Networks with ReLU Activation

The main result of this section presents the conditions required for low generalization error for Neural Networks with ReLU activation:

Theorem 5.1. *For a neural network with ReLU activation and* 1 *output neuron, trained on set* $S \in D^m$ *using SGD for* τ *epochs, where* D *is over* $\mathbb{R}^d \times \mathcal{Y}$*, such that* $\mathcal{Y}$ *is countable and for each* $y \in \mathcal{Y}$ *we get a countable set* $\{x \in \mathbb{R}^d : Pr_D \{\mathsf{lab}(x) = y\} > 0\}$*, where* $\mathsf{lab}(x)$ *is label of* x*. For a doubly differentiable loss function with bounded first and second order derivatives and learning rate* $\alpha_t = \alpha_0 / t^{(1-\rho(\tau,m))}$*, where* $\rho(\tau, m) = \frac{\ln \ln m}{\ln \tau + \ln m}$ *if the data points of* S *and the spectral norms of weight matrices explored by SGD are bounded, then* K_S *is constant w.r.t* m *for all* r *and* $E_r [L_g \cdot L_S]$ *is also constant w.r.t.* m*, with* $c > 0$ *such that*

$$E_r [|R(S,r) - R_e(S,r)|] \leq c \left(E_r [F(\tau)] \cdot U(\alpha_0, K_S, \rho(\tau, m)) \frac{\log(m)^2}{m^{\left(1 - \frac{\alpha_0 K_S}{\rho(\tau,m)}\right)}} + \sqrt{\frac{\log(m)}{m}} \right),$$

with probability at least $1 - 1/m$*, where* $U(\alpha_0, K_S, \rho(\tau, m)) \leq 1 + \frac{1}{\alpha_0 K_S}$*,* $\alpha_0 \to 0$ *implies* $U(\alpha_0, K_S, \rho(\tau, m)) \to 1$ *and* $F(\tau) := 2^\tau \alpha_0 \cdot L_g L_S$*.*

Note that for some $c_1 \log(m)$ epochs and with an initial learning rate of $\alpha_0 < \rho(\tau, m)/K_S$ for some $c_1 \in (0, 1)$ the RHS decreases as m increases. It is important to note that L_g is the constant that depends on the actual distribution D and is calculated for a trained neural network (i.e. at $\boldsymbol{w}_{\tau m}$).This aligns with the notion that if the network has reached a "good enough" minima then the gradient values should be less and hence this will show better generalization. Also, L_S and K_S are constants that depend on the training set S. This dependence on S is also crucial because if the model can't fit the training set properly then these values will be high and reflect a bad generalization. Next in Section 5.1, we first establish that the theory of a.s. support stability applies to NNs under conditions specified, then we prove the above theorem along with empirical validation.

5.1 Support Stability of Neural Networks with ReLU Activation

The key to showing the support stability of NNs with ReLU is to establish that they are locally parameter-Lipschitz and locally parameter-smooth. First, we show the existence of these constants and then we will show an upper bound under some reasonable assumptions.

Theorem 5.2. *For every* $\boldsymbol{w} \in \mathbb{R}^n$*, a doubly differentiable loss function,* $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$*, applied to the output of a NN with ReLU activation is locally parameter-Lipschitz and locally parameter-smooth for all* $x \in \mathbb{R}^d$ *except for a set of measure 0.*

Proof outline. The proof of this theorem is based on the argument that for a given $\boldsymbol{w}$ a point of discontinuity exists at a given neuron if the input x lies in the set of solutions to a family of equations, i.e., in a lower dimensional subspace of $\mathbb{R}^d$. This proof is an adaption of an idea of Milne (2019) and can be found in Appendix C.

Theorem 5.2 begs the question: How large are these Lipschitz and smoothness constants? We provide some general bounds that can be improved for specific architectures:

Proposition 5.3. *Suppose we have a fully connected NN of depth* $H + 1$*, with ReLU activation at the inner nodes. Then, if the spectral norms of weight matrices are bounded for every layer i.e.,* $\|W^i\|_\sigma$ *is bounded* $\forall i \in [H]$*, and the size of each layer be* $\{l_0, \ldots, l_H\}$ *and the distribution of dataset is normalized with* $\|x\|_2 \leq 1$ *then,*

$$L_g \leq \left(\prod_{k=1}^{H} \|W^k\|_\sigma \right) \times \mathcal{A}(M, W)^{1/2}$$

$$K_S \leq \left(\prod_{k=1}^{H} \|W^k\|_\sigma \right) \times \mathcal{A}(M, W)$$

where

$$\mathcal{A}(M, W) = \sum_{l=1}^{H} \frac{\|M^l\|_{2,2}^2}{\|W^{l-1}\|_\sigma^2 \cdot \|W^l\|_\sigma^2 \cdot \|W^{l+1}\|_\sigma^2}$$

where $(i, j)^{th}$ *element of matrix* $M^l[i, j] = \|M'(l, i, j)\|_\sigma$*, and where* $M'(l, i, j)$ *is a matrix such that* $(p, q)^{th}$ *element is* $M'(l, i, j)[p, q] = w_{j,p}^{(l+1)} w_{q,i}^{(l-1)}$*. Note that this equation holds for both Training Lipschitz constant* L_S *and Test Lipschitz constant* L_g*.*

The proof of the proposition is in Appendix C. The bound on Lipschitz constants should be compared to the bounds given in the context of Rademacher complexity by Bartlett et al. (2017) and Golowich et al. (2018). Our bound is related to the spectral complexity and can potentially be independent of the size of the network. We are now ready to prove our main theorem.

Proof of Theorem 5.1. Theorem 5.2 tells us that a NN with ReLU activations is locally parameter-Lipschitz and locally parameter-smooth. From Proposition 5.3 we see that the boundedness of the first and second derivatives of the loss function and the boundedness of the spectral norm of weight matrices and data points ensures that the Lipschitz constants and smoothness constants associated with the NN's training are bounded w.r.t. m. With all these in place, we can apply Theorem 4.4 to get the a.s. support stability followed by Theorem 3.2 to get the desired result.

$\square$

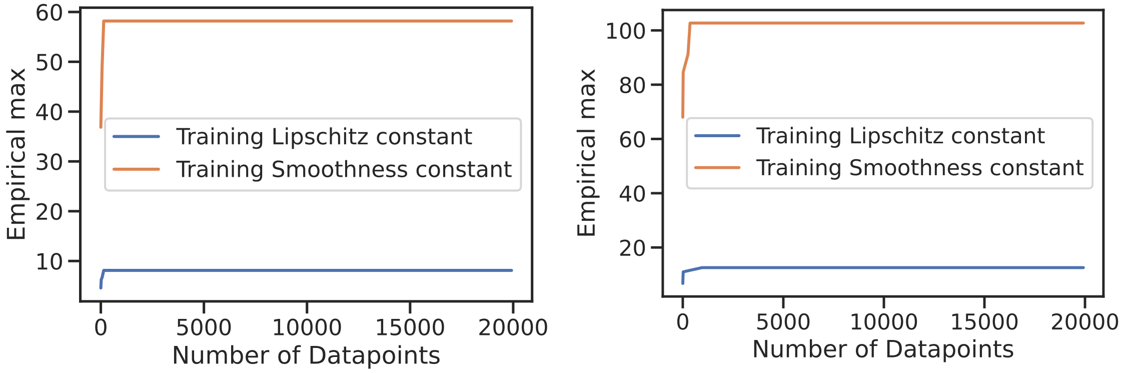
5.1.1 Experimental Validation of Results

Here we will experimentally show that the Training Lipschitz constant (L_S), Test Lipschitz constant (L_g) and Training Smoothness constant (K_S) that we reasoned with are indeed bounded, and that the theoretical upper bound that we derived for the generalization error of a neural network holds in practice. For simplicity in this experiment, we assume Training Lipschitz constant to be a good proxy for Test Lipschitz constant ($L_g = L_S$).

Setup. For our experiments we use *MNIST* and *FashionMNIST* datasets. In both datasets, we randomly selected 20,000 training and 1,000 test points. All experiments were conducted using a fully connected feed forward neural network with a single hidden layer and ReLU activation. We train the model using SGD (batch size = 1), with cross-entropy loss, starting with randomly initialized weights. As suggested in our analysis we use a decreasing learning rate $\alpha_t = \frac{\alpha_0}{t}$. In each epoch, we consider a random permutation of the training set. Training Lipschitz constant and Training Smoothness constant are computed by calculating the norm of gradients and Hessian across the training steps and taking their max.

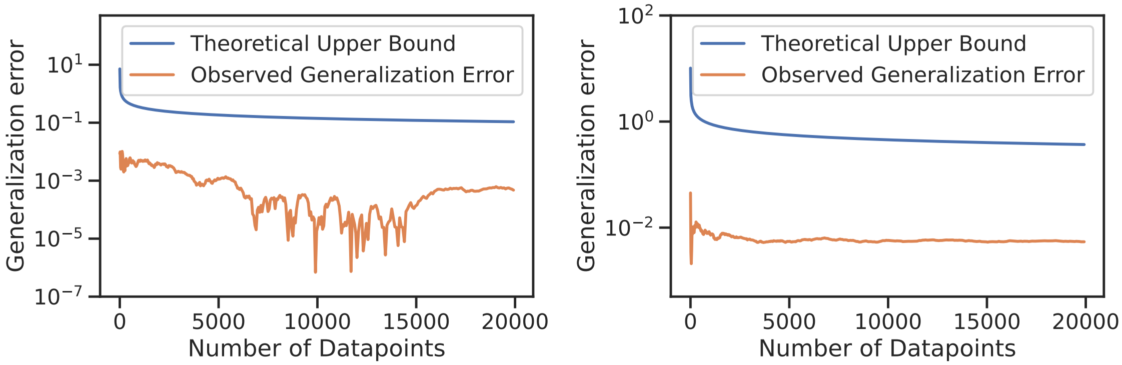
Experiment 1. Our first experiment is aimed towards establishing that the Training Lipschitz constant (L_S) and Training Smoothness constant (K_S) values estimated using local values at each step are bounded. Figure 1 summarizes the results of these experiments over MNIST and FashionMNIST datasets ($\alpha_0 = 0.001$). The plots contain the maximum of the local parameter Lipschitz and smoothness values obtained after running each experiment 10 times with random weight initialization. These results support our Theorem 5.2 since the upper bound values quickly stabilize and do not grow with the size of the training set in both datasets. Similarly, the bounded smoothness constant supports our constraint on the learning rate, $\alpha_0 \leq \frac{\rho(\tau,m)}{K_S}, \rho(\tau, m) = \frac{\ln \ln m}{\ln \tau + \ln m}$. We find L_S to be 8.1174 (MNIST) & 12.5737 (FashionMNIST), and K_S to be 58.185 (MNIST) and 102.7096 (FashionMNIST).

Experiment 2. We now turn our attention to the experiment to support our main result, i.e., the empirical generalization error estimated using validation set is upper bounded by our theoretical upper bound. We first split each dataset in a 20:1 ratio into training and validation sets, and train the model at varying sizes of



(a) MNIST (b) FashionMNIST

Figure 1: Experiment 1, Maximum of the Lipschitz and smoothness constants at every p $(= 20)$ interval of updates of SGD (we plot both the running average and the highest value found so far). Notice that these constants have a clear upper bound throughout the training process.



(a) MNIST (b) FashionMNIST

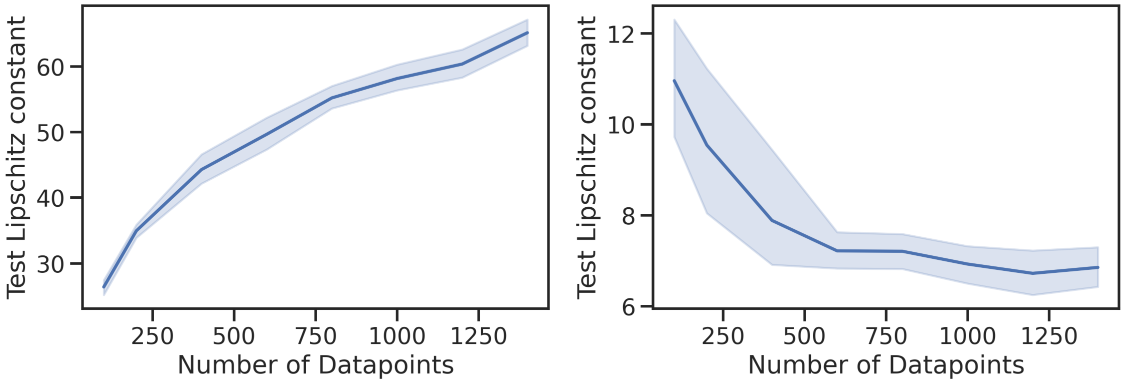
Figure 2: Experiment 2, Comparison of empirical generalization error (red) vs. theoretical upper bound (blue) with varying training set size for different datasets.

training sets. We empirically compute the generalization error at each training set size using the validation set. Figure 2 compares this empirical generalization error (in red) with the theoretical upper bound (in blue). From these results, we can see that our bound decreases along with the generalization error thus empirically validating our reasoning. Clearly, the bound is not as tight as we would like it to be. Our conjecture is that this arises from the fact that we use a single value of L_S as an upper bound for the gradients encountered during the course of the training. Our framework is flexible in the sense that we can use it with a more fine grained analysis of the gradients averaged over time. This is likely to achieve a better bound. We leave this direction for future work.

nn

5.2 Random Labelling Case

In making their case against the applicability of uniform stability as a tool for theoretically establishing the good generalization properties of Neural Networks, Zhang et al. (2017) presented the following classification problem: Given points picked from Euclidean space using some well-behaved distribution, say a Gaussian, each point was assumed to have a class label picked uniformly at random from a finite set of labels independent of all other points. Clearly, any classification algorithm trained on a finite training set will have $\omega(1)$ generalization error for this problem. We now demonstrate that our results *do not* imply good generalization



(a) Increasing L_g for random label MNIST (b) Decreasing L_g for standard (non random) MNIST

Figure 3: Experiment 3, Test Lipschitz constant plot as training set size increases

for this problem. Specifically, we show empirically that the assumption of Test Lipschitz constant (L_g) being independent of m breaks in this case and this "constant" actually increases with m.

Setup. We pick images from the 0 and 1 label class of MNIST dataset. For random labelling case, we assign random labels to all the points. We then randomly sample a test set $\mathcal{T}$ ($|\mathcal{T}| = 50$). We take a single hidden layer (128 neurons) fully connected neural network having ReLU activation in the hidden layer. We take the loss function as $l(\hat{y}, y) = 1 - \text{Softmax}(c \cdot \hat{y}, y)$ where $c = 6$. We use a constant learning rate of 0.003, batch size of size 8.

Experiment 3. The experiment proceeds by selecting initial random weights for a model say $\boldsymbol{w}_0$ (we do this 10 times). Then for every initialization we pick training set S from our modified dataset (we do this for 5 times). Now for every training set, we train the model either till accuracy is $\geq 98\%$ or till 500 epochs whichever is reached first. Now we calculate the loss i.e. $f(r, S, z)$ and the gradient $\nabla_{\boldsymbol{w}} f(r, S, z)$ for all $z \in \mathcal{T}$. For the Test Lipschitz constant we do $L_g \simeq \max_{z \in \mathcal{T}} \{\|\nabla_{\boldsymbol{w}} f(r, S, z)\|\}$. In figure 3a we can clearly see that the Test Lipschitz constant L_g scales as the size of the training set (m) increases. On the contrary for the standard (non-random) dataset the Test Lipschitz constant L_g shows a decreasing trend with m see figure 3b. Therefore we can expect that in the random labelling case, the upper bound in Theorem 5.1 becomes so large as to become vacuous.

Discussion. We note that the random labelling example has the property that the variance over the choice of training sets of the loss of any algorithm, $Var_S[f(r, S, z)]$, is bound to be high. One possible direction for theoretically showing that this implies that the Lipschitz constants are likely to be high is by using Poincare-type inequalities that show that the norm of the gradients of a function of a random vector is lower bounded by the variance of the function. We do not pursue this direction further here, but we point out that it may help develop a general theory for the limitations of what can be learned using parametrized methods trained using gradient descent methods.

5.3 Discussion on the Applicability of Our Results

- *The NTK setting.* In the Neural Tangent Kernel (NTK) setting, kernel regression is carried out via the so-called NTK. The NTK itself can be shown to be essentially independent of the parameters and hence its norm can be controlled. At the same time the NTK is shown to be a good approximation of a kernel that is computed from the gradients of the parameters. Hence the empirically observed good generalization properties of NTK can be derived using our theory by performing a more careful analysis than what Proposition 5.3 implies. Since this analysis would take us far outside the scope of the current paper, we do not include it here, leaving it for future work.

- *Removing the fully connectedness constraint.* Although Theorem 5.1 is stated for a fully connected network, it can be applied to networks like CNNs which have partially connected convolutional layers with intermediate pooling and normalization layers (e.g., LeNet, AlexNet, etc.). In such cases, the symmetry in distribution condition required for Theorem 3.2 holds as long as the training set is chosen i.i.d. from the unknown distribution.

- *Adding regularization terms to the loss function.* Several popular regularizers, the ℓ_2 regularizer being a prominent example, are doubly differentiable and therefore Theorem 5.1 can be applied when such regularizers are used along with a doubly differentiable loss function.

- *Activation functions apart from ReLU.* We present a comprehensive treatment of ReLU activation but that does not mean that our results are restricted to this kind of activation. Non-linearities like max-pool can also be handled in our framework by proving that, like with ReLU, the points of discontinuity of such a non-linearity also lie in a set of Lebesgue measure 0.

- *The case of multiple outputs* Although we state the Theorem 5.1 for the case of a NN with a single output, it is not difficult to extend the technique to cover the case of multiple outputs.

What about other distributions? The data dependent Training and Test Lipschitz constants turns out to be the deciding factor of generalization error. But our analysis is limited to the bounds we derive for them. There is a requirement for a more fine-grained analysis of Training and Test Lipschitz constants and we believe that optimizing these data dependent Lipschitz constants will be the right direction to proceed. This may be made possible by looking at the network structures, the data distribution and the training set in more detail. We hope that the polynomial characterization of the NN presented in Section C.1.2 will help this process. We conjecture that it may be able to show that for certain distributions the constants actually improve (decrease) as the training proceeds, resulting in a much slower decay of learning rate and this could lead to a proof of a.s. support stability in these cases.

6 Conclusion

We have shown the data dependent quantities which derive the generalization error for NNs. We devised a theoretical framework for using algorithmic stability, introduced the data distribution part and proved generalization bounds for NNs with ReLU nonlinearities. We feel that it is possible to prove stronger results in this framework than the ones we have presented here, and more widely applicable ones. Immediate lines of research that suggest themselves are to apply our methods for CNNs and GNNs and to investigate what other architectures can be approached with our method and does the Lipschitz constants play some significant role because of a different network structure. It would be particularly interesting to see if there is some analog of our polynomial characterisation for GNNs. Although we tackle the overparameterization setting to some extent we feel more in-depth analysis is required to give better insights into when we can expect overparametrized NNs to generalize.

References

Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *JMLR*, 3(null):463–482, march 2003. ISSN 1532-4435.

Peter L. Bartlett, Dylan J. Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Proc. Advances in Neural Information Processing Systems 30*, pp. 6240–6249, 2017.

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Rev.*, 60(2):221–487, 2018.

Olivier Bousquet and André Elisseeff. Stability and Generalization. *J. Mach. Learn. Res.*, 2(Mar):499–526, 2002.

Olivier Bousquet, Yegor Klochkov, and Nikita Zhivotovskiy. Sharper bounds for uniformly stable algorithms. *Proc. Machine Learning Research*, 125:1–17, 2020.

Zachary Charles and Dimitris Papailiopoulos. Stability and generalization of learning algorithms that converge to global optima. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 745–754. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/charles18a.html`.

Satrajit Chatterjee and Piotr Zielinski. On the generalization mystery in Deep Learning. arXiv:2203.10036, 2022.

Vitaly Feldman and Jan Vondrak. Generalization bounds for uniformly stable algorithms. In *Advances in Neural Information Processing Systems*, 2018.

Vitaly Feldman and Jan Vondrak. High probability generalization bounds for uniformly stable algorithms with nearly optimal rate. In Alina Beygelzimer and Daniel Hsu (eds.), *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pp. 1270–1279. PMLR, 25–28 Jun 2019a. URL `https://proceedings.mlr.press/v99/feldman19a.html`.

Vitaly Feldman and Jan Vondrak. High probability generalization bounds for uniformly stable algorithms with nearly optimal rate. In Alina Beygelzimer and Daniel Hsu (eds.), *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pp. 1270–1279. PMLR, 25–28 Jun 2019b. URL `https://proceedings.mlr.press/v99/feldman19a.html`. Supplementary section.

N. Golowich, A. Rakhlin, and O. Shamir. Size-independent sample complexity of neural networks (extended abstract). *Proc. Machine Learning Research*, 75:1–3, 2018.

Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1225–1234, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `https://proceedings.mlr.press/v48/hardt16.html`.

Pengzhan Jin, Lu Lu, Yifa Tang, and George Em Karniadakis. Quantifying the generalization error in deep learning in terms of data distribution and neural network smoothness. *Neural Networks*, 130:85–99, 2020. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2020.06.024. URL `https://www.sciencedirect.com/science/article/pii/S0893608020302392`.

Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *CoRR*, abs/1710.05468, 2017.

M. Kearns and D. Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross- validation. *Neural Computation*, 11(6):1427–1453, 1999.

Yegor Klochkov and Nikita Zhivotovskiy. Stability and deviation optimal risk bounds with convergence rate o(1/n). In *Advances in Neural Information Processing Systems*, 2021.

Ilja Kuzborskij and Christoph Lampert. Data-dependent stability of stochastic gradient descent. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2815–2824. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/kuzborskij18a.html`.

Yunwen Lei and Yiming Ying. Fine-grained analysis of stability and generalization for stochastic gradient descent. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5809–5819. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/lei20c.html`.

Yunwen Lei, Rong Jin, and Yiming Ying. Stability and generalization analysis of gradient methods for shallow neural networks. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=BWEGx_GFCbL`.

Tristan Milne. Piecewise strong convexity of neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/b33128cb0089003ddfb5199e1b679652-Paper.pdf`.

Vladimir Vapnik. *Statistical learning theory.* Wiley, 1998. ISBN 978-0-471-03003-4.

Vladimir Vapnik and Alexey Chervonenkis. *Theory of Pattern Recognition.* Nauka, Moscow, 1974.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=Sy8gdB9xx`.

Chiyuan Zhang, Qianli Liao, Alexander Rakhlin, Brando Miranda, Noah Golowich, and Tomaso A. Poggio. Theory of deep learning iib: Optimization properties of sgd. *ArXiv*, abs/1801.02254, 2018.

Yikai Zhang, Wenjia Zhang, Sammy Bald, Vamsi Pritham Pingali, Chao Chen, and Mayank Goswami. Stability of SGD: Tightness analysis and improved bounds. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022. URL `https://openreview.net/forum?id=Sl-zmO8j5lq`.

A Modification of McDiarmid's Theorem

We first define a probabilistic weakening of Lipschitz functions.

Definition A.1. Given $2m$ i.i.d. random variables $X_1, \ldots, X_{2m}$ drawn from some domain $\mathcal{Z}$ according to some probability distribution D, for some $\beta > 0$ and $\eta \in [0,1]$, a function $f : \mathcal{Z}^m \to \mathbb{R}$ is called *η-almost β-Lipschitz w.r.t. D* if

$$\forall i \in \{1, \cdots, m\} : \left| f\left(X_1, \ldots, X_m\right) - f\left(X_1, \ldots, X_{i-1}, X_i', X_{i+1}, \ldots, X_m\right)\right| \leq \beta,$$

with probability at least $1-\eta$. In case $\eta = 0$ we say that f is *almost surely β-Lipschitz w.r.t D*. When D is understood we will omit it, and for the case $\eta = 1$ we will simply write that f is almost surely (or just a.s.) β-Lipschitz.

We now state a modified version of McDiarmid's theorem that holds for η-almost β-Lipschitz functions.

Lemma A.2. *Let $X_1, \ldots, X_m$ be i.i.d. random variables. If f is η-almost β-Lipschitz and takes values between 0 and M, then,*

$$Pr\{f(X_1, \ldots, X_m) - E[f(X_1, \ldots, X_m)] \geq \epsilon\} \leq \exp\left[\frac{-2\epsilon^2}{m\left(\beta + M\eta\right)^2}\right] + \eta.$$

For simplicity, we adopt the notation $A_{i,j} = A_i, \cdots, A_j$ (i.e. with subscript (i,j) we represent $j-i+1$ variables) and A_i represents only one variable in the rest of the section. Before we prove Theorem A.2, we state and prove the following lemma:

Lemma A.3. *Let $f : \mathcal{Z}^m \to \mathbb{R}$ be a function such that $0 \leq f(\cdot) \leq M$. Define $V_i = E_{i+1,m}\left[f(X_{1,m})\right] - E_{i,m}\left[f(X_{1,m})\right]$, where $E_{i,j}\left[\cdot\right]$ denotes $E_{X_i, \cdots, X_j}\left[\cdot\right]$. Then, if f is η-almost β-Lipschitz, then with probability $1-\eta$, $\forall i, V_i \leq (1-\eta)\beta + \eta M$.*

Proof.

$$\begin{aligned}
V_i &= \mathrm{E}_{i+1,m}\left[f(X_{1,m})\right] - \mathrm{E}_{i,m}\left[f(X_{1,m})\right] \\
&= \int_{y_{i+1} \cdots y_m} \cdots \int f(X_{1,i}, y_{i+1,m}) d\mu(y_{i+1,m}) - \int_{y_i \cdots y_m} \cdots \int f(X_{1,i-1}, y_{i,m}) d\mu(y_{i,m}) \\
&= \int_{y_{i+1} \cdots y_m} \cdots \int \left[f(X_{1,i}, y_{i+1,m}) d\mu(y_{i+1,m}) - \int_{y_i} f(X_{1,i-1}, y_{i,m}) d\mu(y_{i,m}) d\mu(y_i) \right] d\mu(y_{i+1,m})
\end{aligned}$$

where μ is the measure associated with the random variables. Thus, V_i, a function of $X_{1,i}$, is an expectation over the random variables $y_{i+1,m}$. Thus, we can write $V_i = \mathrm{E}_{y_{i+1,m}}\left[\Phi(X_{1,i}, y_{i,m})|X_{1,i}\right]$.

Now, let's define two sets as follows:

$$S_i = \{x \in \mathcal{Z}^i : \exists y \in \mathcal{Z}^{m-i} : \Phi(x, y) \leq \beta\},$$

$$S'_x = \{y \in \mathcal{Z}^{m-i} : \Phi(x, y) \leq \beta\}.$$

Then, since f is η-almost β-Lipschitz, and renaming y_i as y_{i+m} in Φ, since they are *i.i.d.*, for any $x \in S_i$, we have

$$\mathrm{E}\left[\Phi(x, y)|x\right] \leq \sum_{y \in S'_x} \Pr\left\{y\right\}\beta + \left(1 - \sum_{y \in S'_x} \Pr\left\{y\right\}\right) M, \tag{2}$$

and for any $x \notin S_i$, we have

$$\mathrm{E}\left[\Phi(x, y)|x\right] \leq M.$$

We further note that $\sum_{x \in S_i} \Pr\left\{x\right\} \geq 1 - \eta$, and $\sum_{y \in S'_x} \Pr\left\{y\right\} \geq 1 - \eta$. Therefore, applying these bounds to Equation 2, we get, with probability at least $1 - \eta$,

$$V_i \leq \beta + \eta M.$$

☐

Proof of Theorem A.2. We will use Azuma's inequality to prove the result. First we show that the sequence $V_1, \cdots, V_m$ is a martingale difference sequence with respect to $X_{1,m}$. To see this, we first notice that V_i is a function of $X_{1,i}$. Also, $\mathrm{E}\left[|V_i|\right]$ is finite since f is bounded. Finally,

$$\begin{aligned}\mathrm{E}\left[V_{i+1}|X_{1,i}\right] &= \mathrm{E}_{i+1,m}\left[\mathrm{E}_{i+1,m}\left[f(X_{1,m})\right] - \mathrm{E}_{i,m}\left[f(X_{1,m})\right]|X_{1,i}\right] \\ &= 0\end{aligned}$$

Let $\boldsymbol{E}$ be the event that $|f(X_{1,m}) - \mathrm{E}\left[f(X_{1,m})\right]| \geq \epsilon$, and $\boldsymbol{F}$ the event that V_i is bounded by $\beta + \eta M$. We have,

$$\Pr\left\{\boldsymbol{E}\right\} \leq \Pr\left\{\boldsymbol{E}|\boldsymbol{F}\right\}\Pr\left\{\boldsymbol{F}\right\} + \Pr\left\{\overline{\boldsymbol{F}}\right\}. \tag{3}$$

Using Lemma A.3, we have that with probability $1 - \eta$, for all i, V_i is bounded by $\beta + \eta M$.

Applying Azuma's inequality to $\Pr\left\{\boldsymbol{E}|\boldsymbol{F}\right\}$, we have the result. ☐

B a.s. Support Stability of SGD Proved

Proof of Lemma 4.3. If f is locally Lipschitz at $\boldsymbol{w} \in A$, there is an $\varepsilon_{\boldsymbol{w}} > 0$ and an $L_{\boldsymbol{w}} > 0$ such that for all $\boldsymbol{w}' \in \mathbb{R}^n$ with $\|\boldsymbol{w} - \boldsymbol{w}'\| \leq \varepsilon_{\boldsymbol{w}}$, $|f(\boldsymbol{w}) - f(\boldsymbol{w}')| \leq L_{\boldsymbol{w}}\|\boldsymbol{w} - \boldsymbol{w}'\|$. So, let us turn our attention to those $\boldsymbol{w}' \in A$ that lie outside the ball of radius $\varepsilon_{\boldsymbol{w}}$ around $\boldsymbol{w}$. Note that for such a $\boldsymbol{w}'$, if $B > 0$ is the bound on f, we have that

$$\frac{|f(\boldsymbol{w}) - f(\boldsymbol{w}')|}{\|\boldsymbol{w} - \boldsymbol{w}'\|} \leq \frac{2B}{\varepsilon_{\boldsymbol{w}}}.$$

Therefore the "global" Lipschitz constant for f within A is $\max\{L_{\boldsymbol{w}}, 2B/\varepsilon_{\boldsymbol{w}} : \boldsymbol{w} \in A\}$ which is bounded since A is finite. This proves the first part of the lemma. The second part follows similarly. ☐

Proof of Theorem 4.4. For some $i \in [m]$ we couple the trajectory of SGD on S and S^i where $z_i \in S$ has been replaced with z'_i. Our random string r, in this case, is a random choice of an initial parameter vector, $\boldsymbol{w}_0$, and a random set of τ i.i.d permutations $\pi_0, \ldots, \pi_{\tau-1}$ of $[m]$ chosen uniformly at random. We use these random choices for training both the algorithms with S and S^i. For $0 \leq j \leq \tau - 1$, we denote $\pi_j^{-1}(i)$ by I_j, i.e., I_j is the (random) position where the ith training point is encountered in the jth training epoch. The key quantity we will track through the coupled training process will be

$$\delta_t = \|\boldsymbol{w}_t - \boldsymbol{w}'_t\|,$$

for $1 \leq t \leq \tau m$. If we can show that $\mathrm{E}_r\left[L_g \delta_{\tau m}\right]$ is bounded by some quantity B almost surely, we can invoke the fact that f is a.s. L_l-LPL to say that $\|\mathrm{E}_r\left[f(\boldsymbol{w}_t, z) - f(\boldsymbol{w}'_t, z)\right]\| \leq \mathrm{E}_r\left[L_g \delta_{\tau m}\right] \leq B$ for all $z \in \operatorname{supp}(D)$, where L_g is the Test Lipschitz constant.

We argue differently for the first epoch and differently for later epochs. For the first epoch, we note that for $t \leq I_0$, $\delta_t = 0$ since SGD performs identical moves in both cases. At $t = I_0 + 1$

$$\delta_{I_0+1} = \|\boldsymbol{w}_{I_0} - \alpha_{I_0}\nabla f(\boldsymbol{w}_{I_0}, z_i) - (\boldsymbol{w}'_{I_0} - \alpha_{I_0}\nabla f(\boldsymbol{w}'_{I_0}, z'_i))\| = \alpha_{I_0}\|\nabla f(\boldsymbol{w}_{I_0}, z_i) - \nabla f(\boldsymbol{w}'_{I_0}, z'_i)\|,$$

where the second equality follows from the fact that $\boldsymbol{w}_{I_0} = \boldsymbol{w}'_{I_0}$ by the definition of I_0. Using Lemma 4.3 we can say that $\delta_{I_0+1} \leq 2\alpha_{I_0} L_S$ almost surely. Notice here we used data dependent Training Lipschitz constant L_S which is only defined for points in set S, unlike Test Lipschitz constant. Now,

$$\delta_{I_0+2} \leq \|\boldsymbol{w}_{I_0+1} - \boldsymbol{w}'_{I_0+1}\| + \alpha_{I_0+1}\|\nabla f(\boldsymbol{w}_{I_0+1}, z_i) - \nabla f(\boldsymbol{w}'_{I_0+1}, z'_i))\|.$$

Here although the parameter vectors $\boldsymbol{w}_{I_0+1}$ and $\boldsymbol{w}'_{I_0+1}$ are not the same, $z_{\pi_0(I_0+1)}$ and $z'_{\pi_0(I_0+1)}$ are the same by the definition of I_0 (assuming that $I_0 \neq m$). Therefore we get that

$$\delta_{I_0+2} \leq \delta_{I_0+1} + \alpha_{I_0+1} K_g \delta_{I_0+1}$$

with probability 1 since from Lemma 4.3 we have that f has a "global" smoothness property for the entire set of at most $2\tau m$ parameter vectors that will be encountered during the coupled training of S and S^i. Noting that a similar recursion can be applied all the way to the end of the first epoch, i.e. till $t = m$ we get

$$\delta_m \leq 2\alpha_{I_0} L_S \prod_{t=I_0+1}^{m} (1 + \alpha_t K_g) \leq 2\alpha_{I_0} L_S \exp\left\{\sum_{t=I_0+1}^{m} \alpha_t K_g\right\}, \tag{4}$$

with probability 1. Moving on to the next epoch we note that we can make the argument above till the next point where the two training sequences differ, i.e., till the $m + I_1 + 1$st step. At this point we have,

$$\delta_{m+I_1+1} \leq \delta_{m+I_1} + \alpha_{m+I_1}\|\nabla f(\boldsymbol{w}_{m+I_1}, z_i) - \nabla f(\boldsymbol{w}'_{m+I_1}, z'_i))\|.$$

Since neither the parameter vector nor the training points are the same in the second term, we have no option but to use the almost data dependent Lipschitz constant to say that

$$\delta_{m+I_1+1} \leq \delta_{m+I_1} + \alpha_{m+I_1} 2L_S.$$

Since $\alpha_{m+I_1} < \alpha_{I_0}$, observing that our current bound for δ_{m+I_1} is larger than $\alpha_{m+I_1} 2L_S$. Therefore

$$\delta_{m+I_1+1} \leq 2\delta_{m+I_1}.$$

So, we see that in the second and subsequent epochs, for time step $jm + I_j + 1$, $1 \leq j < \tau$ we have the bound

$$\delta_{jm+I_j+1} \leq 2\delta_{jm+I_j},$$

and for all $t > m + I_1, t \neq I_1, \ldots, I_{\tau-1}$ we have, as before, by the smoothness property that

$$\delta_{t+1} \leq \delta_t(1 + \alpha_{t+1} K_S).$$

Therefore, we have that

$$\delta_{\tau m} \leq 2\alpha_{I_0} L_S (2)^{\tau-1} \exp\left\{\sum_{t=I_0+1}^{\tau m} \alpha_t K_S\right\} \leq \alpha_0 L_S 2^{\tau} \frac{1}{I_0^{1-\rho(\tau,m)}} \exp\left\{\frac{\alpha_0 K_S \left((\tau m)^{\rho(\tau,m)} - I_0^{\rho(\tau,m)}\right)}{\rho(\tau,m)}\right\}. \tag{5}$$

where, in the first inequality for ease of calculation we have retained the terms of the form $(1 + \alpha_{I_j} K_S)$, $2 \leq j < \tau$ in the product on the right although we can ignore them. In the second inequality, we have substituted $\alpha_t = \alpha_0/t^{(1-\rho(\tau,m))}$ and bound the summation using integration.

Finally, in order to compute $\mathrm{E}_r [L_g \delta_T]$ remember there were two source of randomness first is random initialization $\boldsymbol{w}_0$ or lets call it r_{init} and random permutation π lets call it r_p. Now because r_{init} and r_p are independent we can write $\mathrm{E}_r [L_g \delta_{\tau m}] = \mathrm{E}_{r_{init}} \left[\mathrm{E}_{r_p} [L_g \delta_{\tau m} | r_{init}]\right]$. Now in order to compute $\mathrm{E}_{r_p} [L_g \delta_{\tau m} | r_{init}]$ note that L_g, L_S and K_S are constant.

Note that, since π_0 is uniformly drawn from the set of permutations of $[m]$, I_0 is uniformly distributed on $[m]$. Summing up the last term of (5) over $I_0 \in [m]$ and dividing further by m we get

$$\mathrm{E}_{r_p} [L_g \delta_{\tau m} | r_{init}] \leq 2^\tau \alpha_0 L_g L_S \times \frac{1}{m} \sum_{I_0=1}^{m} \frac{1}{I_0^{1-\rho(\tau,m)}} \exp\left\{ \frac{\alpha_0 K_S \left((\tau m)^{\rho(\tau,m)} - I_0^{\rho(\tau,m)}\right)}{\rho(\tau,m)} \right\}$$

Using integration we bound the summation part and also using $\exp(-\alpha_0 K_S/\rho(\tau,m)) \leq 1$ we get the upper bound for the summation part as

$$\leq U\left(\alpha_0, K_S, \rho(\tau,m)\right) \cdot \exp\left(\frac{\alpha_0 K_S}{\rho(\tau,m)} (\tau m)^{\rho(\tau,m)} \right)$$

where

$$U\left(\alpha_0, K_S, \rho(\tau,m)\right) := 1 + \frac{1 - \exp(-\alpha_0 K_S m^{\rho(\tau,m)}/\rho(\tau,m))}{\alpha_0 K_S},$$

we get

$$\mathrm{E}_{r_p} [L_g \delta_T | r_{init}] \leq 2^\tau \alpha_0 L_g L_S \cdot U\left(\alpha_0, K_S, \rho(\tau,m)\right) \cdot \frac{\exp\left\{ \frac{\alpha_0 K_S}{\rho(\tau,m)} (\tau m)^{\rho(\tau,m)} \right\}}{m} \tag{6}$$

taking $\rho(\tau,m) = \frac{\ln \ln m}{\ln \tau + \ln m}$ and expectation over r_{init} we get the desired result. ☐

C Neural Networks: Characterization and Proofs

In order to prove Theorem 5.2 we first need to describe a characterization of Neural Networks that allows us to get a better insight into their smoothness properties. We present the characterization in Section C.1 and the proof in Section C.2.

C.1 A Polynomial-based Characterization Neural Networks

C.1.1 Neural Network Terminology

Neural networks provide a family of parameterized functions of the form we have discussed in Section 4. The parameter vector $\boldsymbol{w} \in \mathbb{R}^n$ is applied over a network structure with layers. In this case, we specify $\mathcal{Z}$ to be $\mathbb{R}^d \times \mathbb{R}$, i.e., the data points are from $\mathbb{R}^d$ and the label is from $\mathbb{R}$, i.e., the NN has a single output. We will denote the depth of the network by H. The layers will be numbered 0 to H with layer 0 being the *input layer.* The number of neurons in layer i will be k_i. For this discussion, we assume a fully connected network. We will denote by $w^i_{j,k}$ the weight of the edge from the j neuron of the ith layer to the kth neuron of the $i+1$st layer. For the NN with parameters $\boldsymbol{w}$ at a point $x \in \mathbb{R}^d$ we will denote the input into the jth neuron of the ith layer by $\mathsf{in}_{i,j}(\boldsymbol{w}, x)$ and its output by $\mathsf{out}_{i,j}(\boldsymbol{w}, x)$. Further, we will assume that all neurons in all layers of the network except the input layer and the output layer have ReLU activation applied to them. In case the output of a node is 0 due to ReLU activation we will say the ReLU gate is *closed* otherwise we will say it is *open.* The label output by the network will be $\mathsf{out}_{H,1} = \mathsf{out}(\boldsymbol{w}, x)$. For each exposition, we will assume that $\mathsf{out}(\boldsymbol{w}, x) = 1$ if $\mathsf{in}(\boldsymbol{w}, x) > 0$ and 0 otherwise, i.e., there are only two labels in $\mathcal{Y}$.

C.1.2 Multivariate Polynomials Associated with a Neural Network

Given a set of indeterminates $x = x_1, \ldots, x_l$, let $\mathcal{P}(x)$ be the set of multivariate polynomials on $x_1, \ldots, x_l$ with real coefficients. For any polynomial $p(x)$, $i_1, \ldots, i_q \in [l]$ and any $\alpha_1, \ldots, \alpha_q \in \mathbb{R}$ for some $q \leq l$, we will denote by $p(x)\left\{_{x_{i_j}=\alpha_j : j \in [q]}\right\}$ the polynomial in $\mathcal{P}(x \setminus \{x_{i_1}, \ldots, x_{i_q}\})$ that is obtained by setting all occurrences of x_{i_j} to α_j in $p(x)$. In particular, $p(x)\left\{_{x_i=0}\right\}$ is the polynomial $p(x)$ with all monomials containing x_i removed, and $p(x)\left\{_{x_i=1}\right\}$ retains all the monomials of $p(x)$ but those monomials that contain x_i appear without the term x_i.

Returning to NNs, let us consider two sets of indeterminates: $x = \{x_i : i \in [d]\}$ and $\boldsymbol{w} = \{w^{(i)}_{j,k} : 0 \leq i < H, 1 \leq j \leq k_i, 1 \leq k \leq k_{i+1}\}$ and $k_0 = d$. For a fully connected NN defined in Sec. C.1.1 with all ReLU gates open we will say that it has the following polynomial associated with it:

$$\phi(\boldsymbol{w}, x) = \sum_{j_0=1}^{k_0} \sum_{j_1=1}^{k_1} \cdots \sum_{j_{H-1}=1}^{k_{H-1}} x_{j_0} w^{(0)}_{j_0,j_1} w^{(1)}_{j_1,j_2} \cdots w^{(H-1)}_{j_{H-1},1}.$$

Note that the output layer has only one neuron. We will refer to this as the *base polynomial* of the NN. The base polynomial associated with the jth neuron in layer i can be derived from the base polynomial of the network as follows

$$\phi_{i,j}(\boldsymbol{w}, x) = \frac{\phi(\boldsymbol{w}, x)\left\{_{w^{(i)}_{l_1,l_2}=0, w^{(l_3)}_{l_4,l_5}=1 : l_1 \in [k_i] \setminus \{j\}, l_2 \in [k_{i+1}], l_3 > i, l_4 \in [k_{l_3}], l_5 \in [k_{l_3+1}]}\right\}}{\prod_{p=i+1}^{H} k_i}.$$

Also we could describe a Network whose say i^{th} layer j^{th} neuron's gate is closed by $\phi(\boldsymbol{w}, x)\{w^i_{l_1,j} = 0, \forall l_1 \in k_{i-1}\}$. We will write $G_{\boldsymbol{w},x}$ as the set of weights needed to be equated to zero for all closed ReLU gates. It's clearly visible that due to ReLU activations varying at different points, there is no single polynomial that captures the output of the NN everywhere in $\mathbb{R}^n \times \mathbb{R}^d$. However, the following observation shows a way of defining polynomials that describe the output over certain subsets of space.

Observation C.1. *Given* $\boldsymbol{w} \in \mathbb{R}^n$ *and* $x \neq (0, \ldots, 0) \in \mathbb{R}^d, i \in [H], j \in [k_i]$ *and* $\phi_{i,j}(\boldsymbol{w}, x)\{G_{\boldsymbol{w},x} = 0\}$ *be the polynomial representing output and* $G_{\boldsymbol{w},x}$ *be the set of weights for closed ReLU gates as discussed above. For the case where* $\mathsf{in}_{l_1,l_2}(\boldsymbol{w}, x) \neq 0$ *for all* $1 \leq l_1 \leq i$ *and all* $1 \leq l_2 \leq k_{l_1}$*, there is an* $\epsilon > 0$ *depending on* $\boldsymbol{w}, x$ *such that, for all* $\boldsymbol{w}'$ *with* $\|\boldsymbol{w} - \boldsymbol{w}'\| < \epsilon$*,*

$$\phi_{i,j}(\boldsymbol{w}', x)\{G_{\boldsymbol{w}',x} = 0\} = \phi_{i,j}(\boldsymbol{w}', x)\{G_{\boldsymbol{w},x} = 0\}$$

i.e. the polynomial remains same for $\boldsymbol{w}'$ *and* $\boldsymbol{w}$*.*

Proof. Since $\mathsf{in}_{i,j}(\boldsymbol{w}, x)$ is strictly separated from 0 and there are only a finite number of neurons in the network there must be an ϵ small enough for which all open ReLU gates remain open and all closed gates remain closed. And because of this, we can use the same polynomial with new weights as no ReLU gate switches their state. ☐

C.2 Proof of Theorem 5.2

Proof of Theorem 5.2. The idea behind this proof is due to Milne (2019) who used it for a different purpose. From Observation C.1 it follows that if we have $x \neq (0, \ldots, 0) \in \mathbb{R}^d$ such that $\mathsf{in}_{i,j}(\boldsymbol{w}, x) \neq 0$ for all $1 \leq i \leq H$ and all $1 \leq j \leq k_i$, then $\mathsf{out}(\boldsymbol{w}, x)$ is, in fact, just the polynomial $\phi(\boldsymbol{w}', x)\{G_{\boldsymbol{w},x} = 0\}$ within a small neighbourhood of $\boldsymbol{w}$. Therefore it is doubly differentiable. Since the loss function is also differentiable, we are done for all such values of x.

So now let us consider the set of points x for which i is the smallest layer index such that $\mathsf{in}_{i,j}(\boldsymbol{w}, x) = 0$. In case there are two such indices, we break ties using the neuron index j. By Observation C.1, in a neighbourhood of $\boldsymbol{w}$, $\mathsf{in}_{i,j}(\boldsymbol{w}, x)$ is a polynomial in $\boldsymbol{w}$ and x for each x.

Now, we consider two cases. In the first case, $\mathsf{out}_{i-1,j'}(\boldsymbol{w}, x) = 0$ for all $j' \in [k_{i-1}]$, i.e., all the ReLU gates from the previous layers are closed because $\mathsf{in}_{i-1,j'}(\boldsymbol{w}, x) < 0$ for all $j' \in [k_{i-1}]$. In this case $\mathsf{out}(\boldsymbol{w}', x) = 0$

everywhere in the neighbourhood guaranteed by Observation C.1 and therefore $\ell(\mathsf{out}(\boldsymbol{w}', x), \mathsf{lab}(x))$ is doubly differentiable in the parameter space at $\boldsymbol{w}$ for all such x, where we assume that each data point has a label $\mathsf{lab}(x) \in \{0, 1\}$ associated with it. We note that this argument is easily portable to the case of a more general label set $\mathcal{Y}$ with the property described in the statement of Theorem 5.1 since $\mathsf{in}_{H,1}$ will be 0 everywhere in a small neighbourhood.

In the second case we have some $j' \in [k_{i-1}]$ such that $\mathsf{out}_{i-1,j'}(\boldsymbol{w}, x) > 0$. Let $C_{i,j} \subseteq \mathbb{R}^d$ be those x for which this case holds. $C_{i,j}$ contains the solutions to $\mathsf{in}_{i,j}(\boldsymbol{w}, x) = 0$. Since we are working with a specific value of $\boldsymbol{w}$, this simply becomes a polynomial in x. In fact, inspecting the definition of base polynomials we note that when $\boldsymbol{w}$ is fixed $\mathsf{in}_{i,j}(\boldsymbol{w}, x)$ is simply a linear combination of $x_1, \ldots, x_{\mathbb{R}}^d$. This implies that $C_{i,j}$ is a hyperplane in $\mathbb{R}^d$. We note that this argument can also be made of the output node under the condition on the label set given in the statement of Theorem 5.1 because for $\mathsf{in}_{H,1}(\boldsymbol{w}, x)$ to give a value that lies on the boundary between two sets with different labels for a given $\boldsymbol{w}$, x must be drawn from a set of Lebesgue measure 0.

Since the network size is finite the set of all possible values of x for which case 2 occurs, i.e., $\bigcup_{i\in[H],j\in[k_i]} C_{i,j}$ is a finite union of hyperplanes in $\mathbb{R}^d$ and therefore a set of Lebesgue measure 0. □

Proof of Proposition 5.3. Let us consider the partial derivative w.r.t $w_{i,j}^{(l)}$. For this let $I_{i,j}^{(l)}, A_j^{(l+1)}$ and $B_i^{(l-1)}$ be 3 matrices of size $W^{(l)}, W^{(l+1)}$ and $W^{(l-1)}$ respectively such that $I_{i,j}^{(l)}[i, i] = 1$ and reset all entries are 0, $A_j^{(l+1)}[k, j] = W^{(l+1)}[k, j], \forall k$ and rest all entries are 0 and $B_i^{(l-1)}[i, k] = W^{(l-1)}[i, k], \forall k$ and rest all entries are one. Using these 3 matrices and the weight matrices we can compute the gradient as

$$\frac{\partial \phi(\boldsymbol{w}, x)}{\partial w_{i,j}^{(l)}} = W^{(H)} \cdots W^{(l+2)} \cdot A_j^{(l+1)} \cdot I_{i,j}^{(l)} \cdot B_i^{(l-1)} \cdot W^{l-2} \cdots W^1 \cdot \|x\| \tag{7}$$

Let M'(l,i,j) be a matrix such that

$$M'_{l,i,j} = A_j^{(l+1)} \cdot I_{i,j}^{(l)} \cdot B_i^{(l-1)}$$

Although we have scalar values taking spectral norm on both sides of eq 7 we get

$$\left| \frac{\partial \phi(\boldsymbol{w}, x)}{\partial w_{i,j}^{(l)}} \right| = \prod_{k=1}^{H} \|W^{(k)}\|_\sigma \frac{\|M'_{l,i,j}\|_\sigma}{\|W^{(l+1)}\|_\sigma \cdot \|W^{(l)}\|_\sigma \cdot \|W^{(l-1)}\|_\sigma} \|x\|$$

Now lets define another matrix M_l such that $(p, q)^{th}$ element of matrix $M_l[p, q] = \|M'_{l,i,j}\|_\sigma$. Now the expression for $2, 2$ norm (Frobenius norm) of the gradient vector directly gives us the required expression for Lipschitz constants.

We can give a similar argument for bounding K_S, for some $\boldsymbol{w}_{i_1,j_1}^{(l_1)}$ and $\boldsymbol{w}_{i_2,j_2}^{(l_2)}$ we have

$$\begin{aligned}\left| \frac{\partial^2 \phi(\boldsymbol{w}, x)}{\partial w_{i_2,j_2}^{(l_2)} \partial w_{i_1,j_1}^{(l_1)}} \right| \le \prod_{k=1}^{H} \|W^{(k)}\|_\sigma &\left(\frac{\|M'_{l_1,i_1,j_1}\|_\sigma}{\|W^{(l_1+1)}\|_\sigma \cdot \|W^{(l_1)}\|_\sigma \cdot \|W^{(l_1-1)}\|_\sigma} \right) \\ &\cdot \left(\frac{\|M'_{l_2,i_2,j_2}\|_\sigma}{\|W^{(l_2+1)}\|_\sigma \cdot \|W^{(l_2)}\|_\sigma \cdot \|W^{(l_2-1)}\|_\sigma} \right) \|x\|\end{aligned}$$

Note that the above equation is exactly if $l_1 + 2 < l_2$ or $l_1 - 2 > l_2$ and for the rest of the case we can use this as the upper bound this is because for a matrix M spectral norm $\|M\|_\sigma$ is upper bound for when we set all except one row or column of matrix to zero and calculate the spectral norm. Now if we take the $2, 2$ norm (Frobenius norm) of the Hessian matrix we get the desired result. □