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

Deep neural networks are known to perform well on unseen data (test data), c.f. e.g. Jin et al. (2020)), but theoretical explanations of this behaviour are still unsatisfactory. Under the assumption that the error on the training set (empirical error) is low, studying the gap between the empirical error and the population error is one route to investigating why this performance is good. In this paper we look at the gap between the population error (risk) and empirical error (empirical risk) 1. Following works like Bousquet & Elisseeff (2002) we use the term *generalization error* for this quantity. 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 and this holds for classification case. We also allow for a learning rate that is

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	$O(\log m)$	$O\left(1/t^{1-\frac{c}{\rho(\tau,m)}}\right)$, $c \in (0,1)$	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.

asymptotically strictly slower than $\theta(1/t)$ at the t-th step of SGD. All this holds for any bounded value loss function which is twice differentiable.

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 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, Definition 2) and (Lei & Ying, 2020, Definition 4) 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 using a mild generalization of McDiarmid's Inequality. 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. 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 number of epochs proportional to $\log m$. This implies (Corollary 4.4) 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 $O\left(\log\log m/\left(\log \tau + \log m\right)\right)$ 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, a category in which our work falls, 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, Definition 2) 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, Definition 4) 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.

The role of the norm of gradients for bounding generalization error has been observed from a different angle by Negrea et al. (2020) and also by Haghifam et al. (2020). Their work focuses on giving information-theoretic generalization bounds for Stochastic Gradient Langevin Dynamics (SGLD) and Langevin dynamics algorithm. Although their setting and methods is different from ours, the general flavour is similar.

3 Almost Sure (a.s.) Support Stability and Generalization

In this section, we present a weakening of the notion of uniform stability defined by (Bousquet & Elisseeff, 2002, Definition 6) 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 $\mathcal{R}$. This random string abstracts the random choices of the algorithm. For example, in an NN trained with SGD it encapsulates the random initial parameter vector and the random permutation of the training set as seen by SGD. For an algorithm like Random Forest r would abstract out the random points chosen to divide the space.

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 $A_{S,r}$ 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 $A_{S,r}$ 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*

$$\left|R(A_{S,r}) - R_e(A_{S,r})\right|. \tag{1}$$

When talking about SGD we omit A and just use $R(S, r)$ and $R_e(S, r)$ to represent $R(A_{S,r})$ and $R_e(A_{S,r})$ respectively.

About the loss function $l(\cdot, \cdot)$. When we talk about the loss function we refer to the commonly used loss functions in machine learning, like cross entropy, focal loss, Mean Squared Error (for bounded inputs) etc. Our results are valid for any bounded value loss function which is doubly differentiable. In Machine Learning an implicit assumption is that the algorithm is able to successfully minimize the loss function chosen, i.e., a reasonable loss function is used that can be minimized to a reasonable value over a training set. We also work with this assumption.

3.2 A Weakening of Uniform Stability

Given $S = \{Z_1, \ldots, Z_m\}$ where all points are chosen randomly from D, we construct S^i via replacing the i-th element of S by an independently generated element from D, to quote it formally we choose $\{Z_{1+m}, \ldots, Z_{2m}\}$ points such that all are chosen randomly from D such that they are independent from all points in S, for each $i \in [m]$ we define

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

Where $[m]$ represents integer points from $[1, m]$.

Definition 3.1 (Almost Sure (a.s.) Support Stability)**.** We say an algorithm $A_{S,r}$ has *almost sure (a.s.) 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 over the choice of points $Z_1, \ldots, Z_{2m}$ where $\forall i, Z_i \sim D$ or in other words $\{Z_1, \ldots, Z_{2m}\} \sim D^{2m}$.

We note that this notion weakens the notion of uniform stability introduced by (Bousquet & Elisseeff, 2002, Definition 6) by requiring the bound on the difference in losses to hold D^{2m}- almost everywhere. 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 a.s. support stability a *data-dependent quantity* on the lines of the suggestion made by Zhang et al. (2017). We also observe that a.s. support stability is comparable to but stronger than the hypothesis stability of Kearns & Ron (1999) as formulated by Bousquet & Elisseeff (2002).

While the quantification of z, i.e., $\forall z \sim \operatorname{supp}(D)$ appears to be a very strong condition it is a weakening of uniform stability. In (Bousquet & Elisseeff, 2002, Section 5) it was shown that that uniform stability (which is $\forall z \sim D$) holds for several classical Machine Learning algorithms like soft margin SVM, bounded SVM regression and regularized least square regression. Hence a.s. support stability also holds for these algorithms. As we will see ahead the weakening helps us fulfill key technical requirements when it comes to the study of Neural Networks.

3.3 Exponential Convergence of Generalization Error

Almost Sure (a.s.) Support Stability can be used in place of uniform stability in conjunction with the techniques of (Feldman & Vondrak, 2019a, Theorem 1.1) to give guarantees on generalization error for algorithms that are *symmetric in distribution*. A function $f(x_1, \ldots, x_m)$ is called symmetric if $f(x_1, \ldots, x_m) = f(\sigma(x_1), \ldots, \sigma(x_m))$ for any permutation σ . But if we have a function f which is not symmetric but the probability of choosing any permutation of a given set of elements is equal then we use the term “symmetric in distribution” to refer to such a function along with the distribution by which its inputs are picked. In (Bousquet & Elisseeff, 2002, Section 2.1) the term “symmetric algorithm” was used but it was potentially misleading since what they meant was “symmetric in distribution” in the sense that we have used it. Since SGD randomly permutes the training points it is clearly “symmetric in distribution”.

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 exists a constant $c > 0$ independent of m s.t. for any $m \ge 1$ and $\delta \in (0,1)$, with probability $1 - \delta$,*

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

The constant c is independent of m and, because our analysis is asymptotic in m, this is sufficient for us.

Proof outline. We give a high-level outline here. Our proof extends the proof of Feldman and Vondrak ((Feldman & Vondrak, 2019a, Theorem 1.1)) to accommodate the generalization of McDiarmid's Lemma A.2 from (Combes, 2015, Proposition 2). 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 Almost Sure (a.s.) Support Stability of Stochastic Gradient Descent

As 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 is 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 needed. 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 Lipschitzness of vector-valued function)**.** Given a subset Ω of $\mathcal{Z}$, a parameterized function $f : \mathbb{R}^n \times \mathcal{Z} \to \mathbb{R}^o$ is said to follow *β-almost L Lipschitzness property w.r.t Ω* if for any $\boldsymbol{w} \in \mathbb{R}^n$ and $\forall z \in \Omega$ there exists constants $L > 0$ 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

$$\|f(\boldsymbol{w}', z) - f(\boldsymbol{w}, z)\| \le L\|\boldsymbol{w}' - \boldsymbol{w}\|.$$

If $\beta = 1$ then we say that f follows *almost surely L-Lipschitzness* property w.r.t Ω.

From Lipschitzness to local parameter Lipschitz and local parameter Smoothness. From Definition 4.1 we get two local properties of a function which are of interest to us.

- For a parameterized function $f : \mathbb{R}^n \times \mathcal{Z} \to \mathbb{R}$ we call it *almost surely locally L_l-parameter Lipschitz (a.s. L_l-LPL for short)* if it satisfies the Definition 4.1.
- Also we call this function *almost surely locally K_l-parameter Smooth (a.s. K_l-LPS for short)* if $\nabla_{\boldsymbol{w}} f(\cdot,\cdot)$ satisfies the Definition 4.1.

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 4.2. *Given $f : \mathbb{R}^n \times \mathcal{Z} \to \mathbb{R}$ we have that if f is bounded and follows almost surely L Lipschitzness property 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 $\forall z \in \Omega$*

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

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 4.2 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 applying it to the gradients of function we get 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 $\boldsymbol{w}_0$. We also give a formal definition of set A for Test Lipschitz constant which can be found in Appendix as Definition B.1.

4.2 Almost Sure (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. To make the theorem statement easier to read, we first separate out our assumptions.

S1. We are given a space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and a probability distribution D defined over it. We have a parameterized loss function $f : \mathbb{R}^n \times \mathcal{Z} \to \mathbb{R}$ that is a.s. L_l-LPL w.r.t $\operatorname{supp}(D)$ and a.s. K_l-LPS w.r.t S.

S2. For a training set S of size m for each $i \in [m]$ chosen i.i.d. according to D we run Stochastic Gradient Descent on f for τ epochs with random choice 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.

S3. At step t of SGD let the learning rate $\alpha_t \leq \alpha_0/t^{(1-\rho(\tau,m))}$, $\rho(\tau, m) = \frac{\log\log m}{\log\tau + \log m}$, $\boldsymbol{w}_t$ and $\boldsymbol{w}_t'$ the parameter vectors obtained while training with set S and S' respectively.

Theorem 4.3. *Given Assumptions S1, S2 and S3, 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*

$$E_r\left[f(\boldsymbol{w}_{\tau m}, z) - f(\boldsymbol{w}'_{\tau m}, z)\right] \leq 2^{\tau}\alpha_0 \cdot E_r\left[\frac{L_S L_g \cdot U(\alpha_0, K_S, \rho(\tau, m))}{m^{\left(1-\frac{\alpha_0 K_S}{\rho(\tau,m)}\right)}}\right], \tag{2}$$

where $U(\alpha_0, K_S, \rho(\tau, m)) \leq 1 + \frac{1}{K_S\alpha_0}$, and as $\alpha_0 \to 0$, $U(\alpha_0, K_S, \rho(\tau, m)) \to 1 + \frac{m^{\rho(\tau,m)}}{\rho(\tau,m)}$.

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, Theorem 3.12) with the difference that we allow for a probabilistic relaxation of the smoothness conditions and more relaxed constraint on Lipschitz constants 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, Theorem 4). The complete proof of Theorem 4.3 is in Appendix B.

Data-dependence with Training Lipschitz constant and Test Lipschitz constant. A key feature of the bound presented in equation 2 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. 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 2 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 4.4. *Given assumptions S1, S2 and S3, and under the condition that K_S is a constant, w.r.t. m, for all r, and $E_r\left[L_g L_S\right]$ is also constant w.r.t m, there is a constant $c \in (0, 1)$ that depends on α_0 and K_S such that if the number of epochs τ is at most $c \log m$ epochs, the expectation of the generalization error of the algorithm taken over the random choices of the algorithm decreases as $\tilde{O}\left(m^{-\min(\epsilon,1/2)}\right)$ (Where tilde hides the logarithmic factors) with probability at least $1 - 1/m$ over the choice of the training set if $\frac{\alpha_0 K_S}{\rho(1,m)} + c\log 2 < 1$, where $\epsilon = 1 - c\log 2 - \frac{\alpha_0 K_S}{\rho(1,m)}$.*

Proof. Let us consider two cases. In the first case when $\epsilon > 1/2$ (i.e. we get the usual rate $\tilde{O}(m^{-1/2})$), this happens when $\alpha_0 < \frac{\rho(1,m)}{2K_S}$ and we choose a small enough c. One the other hand for case where $\epsilon < 1/2$ (i.e. rate of $\tilde{O}(m^{-\epsilon})$), which allows for a larger learning rate $\frac{\rho(1,m)}{2K_S} < \alpha_0 < \frac{\rho(1,m)}{K_S}$ (for small enough c). This clearly shoes that larger initial learning rate could be bad for generalization. It is easy to check from Theorem 4.3 that with the conditions given in the statement of Corollary 4.4 the learning algorithm has a.s. support stability β where β is $o(1/m^{\epsilon})$ if $\frac{\alpha_0 K_S}{\rho(1,m)} + c\log m < 1$. 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 $\rho(\tau, m) = O\left(\frac{\log\log m}{\log\tau + \log m}\right)$ 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 $\alpha_0 = O\left(1/\log(m)^2\right)$ 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. For ease of reading, we first state our assumptions:

N1. We have a fully connected Neural Network with ReLU activation and 1 output neuron.

N2. The NN is trained on set $S \sim D^m$ 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.

N3. We have 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 $\rho(\tau, m) = \frac{\log\log m}{\log\tau + \log m}$ and the data points of S and the spectral norms of weight matrices explored by SGD are bounded

Theorem 5.1. *If N1, N2 and N3 hold, 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*

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

with probability at least $1 - 1/m$, where $U(\alpha_0, K_S, \rho(\tau,m)) \le 1 + \frac{1}{\alpha_0 K_S}$, $\alpha_0 \to 0$ implies $U(\alpha_0, K_S, \rho(\tau,m)) \to 1 + \frac{m^{\rho(\tau,m)}}{\rho(\tau,m)}$.

Note that for some $c_1 \log(m)$ epochs and with an initial learning rate of α_0 such that $\frac{\alpha_0 K_S}{\rho(1,m)} + c_1 \log 2 < 1$, 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. These are "global" over the data set in the sense that the expectation is for the entire training process over the (random) choice of initial parameters and the permutation that SGD chooses. For cases where SGD chooses a good set of initial parameters with good probability these are likely to be small.

However, our framework allows for a more nuanced analysis that we have presented in brief in Appendix C. To understand this line of analysis let us first note that as SGD training proceeds, and if it is able to converge to a minima, the Lipschitz constants encountered decrease. Now if we consider the index i of the training set that has been changed as per the definition of a.s. support stability, a random permutation of S will locate this index at a randomly selected position between 1 and m in the training sequence. If we look at a particular trace of SGD, the bounding constant L_S can actually be replaced with the constant encountered at this position. Hence, when the expectation is taken over the random choices, the bounding value is related to the expected value of the Lipschitz constants encountered. *The significance of this is that this expected value could be substantially smaller than the worst case value in many cases, especially when the training converges from a high value of the loss to a low value rapidly.* L_g can be significantly smaller that L_S but in case the training converges early or if the initial choice of weights is close to the final choice of parameters these two could be close. Further details of this analysis and alternate theorem statements can be found in Appendix C.

For K_S we are constrained in the sense that we need this value to be small throughout the training, even at the beginning. Also it is interesting to note that when L_S and $L_g \to 0$ the generalization error becomes zero, but when $K_S \to 0$, $U(\alpha_0, K_S, \rho(\tau,m)) \to 1 + \frac{m^{\rho(\tau,m)}}{\rho(\tau,m)}$ which leads to generalization error behave like $O\left(\frac{\log m^3}{m} + \sqrt{\frac{\log m}{m}}\right)$, which is still a decreasing function of m and does not directly reaches 0.

We note that the role of gradients in bounding generalization error that we identify has the same flavour, in a very different context, as the work of (Negrea et al., 2020, Theorem 3.1) and (Haghifam et al., 2020, Theorem 4.2) where Information-Theoretic generalization bounds for SGLD (Stochastic Gradient Langevin Dynamics) are given in terms of gradient values of the loss function. 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 Almost Sure (a.s.) Support Stability of Neural Networks with ReLU Activation

The key to showing the a.s. 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, Lemma 1) and can be found in Appendix D.

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,*

$$L_g \leq \left(\prod_{k=1}^{H} \|W^k\|_\sigma\right) \times \mathcal{A}(M,W)^{1/2} \tag{3}$$

$$K_S \leq \left(\prod_{k=1}^{H} \|W^k\|_\sigma\right) \times \mathcal{A}(M,W) \tag{4}$$

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 equation 3 holds for both Training Lipschitz constant L_S and Test Lipschitz constant L_g.

The proof of the proposition is in Appendix D. Note that it's possible to give a tighter bound for above theorem by not bounding product of weight matrices (which we do after equation 20 in Appendix) but we keep the above equation because of its clarity. The bound on Lipschitz constants should be compared to the bounds given in the context of Rademacher complexity by (Bartlett et al., 2017, Equation 1.2) 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 4.3 to get the a.s. support stability followed by Theorem 3.2 to get the desired result.

$\square$

Discussion: An example showing the benefits of data-dependent Lipschitz constants. In general data-dependent Lipschitz constants can be much smaller than the Lipschitz constants of the space from which the data might appear. There are probably many scenarios in which this can be demonstrated but we turn to a well-appreciated scenario: a data set which has much smaller dimensionality than the space in which it is embedded. We will now argue that in such scenarios data-dependent Lipschitz constants can be significantly smaller than the

Suppose we have data as $x \in \mathbb{R}^d$ but the actual dimension of the data is $D \ll d$, a situation that is often seen in many cases, for example image data. For simplicity of presentations we assume that each data point has $x_1, \ldots, x_D$ non-zero and the remaining coordinates are 0. The arguments we make can be made even without this assumption by considering the data points with coordinates based on their projection onto a basis of the subspace they are taken from.

Suppose we have a neural network with 1 hidden layer of d_1 neurons and a single output layer. Let $W^1 \in \mathbb{R}^{d_1 \times d}$ and $W^2 \in \mathbb{R}^{1 \times d_1}$ be the weights of 1^{st} and 2^{nd} layer respectively and we use $w_{i,j}^{(l)}$ to represent i, j weight of l^{th} layer. For simplicity, let the output of the neural network $O(\boldsymbol{w}, x) = W^2 W^1 x$. Now, assuming MSE loss we calculate the gradients and show that the effective upper bound of this could be smaller because of the fact that our L_S and L_g are calculated only from S and $supp(D)$. This is under the assumption that the weights are upper bounded by some quantity B_1. We will also assume that all data points have been re-scaled so that their norm is at most 1.

Computing the squared ℓ_2 norm of the gradients of the parameter vector $\boldsymbol{w}$ we get,

$$\|\nabla_{\boldsymbol{w}} f(\boldsymbol{w}, x)\|_2^2 = \|\nabla_{W^1} f(\boldsymbol{w}, x)\|_2^2 + \|\nabla_{W^2} f(\boldsymbol{w}, x)\|_2^2$$

Calculating gradients norm for both layers, assuming for given $\boldsymbol{w}_0$ (i.e., weight at initialization, note that this is a part of r randomness) we have value of weights bounded above by B_1

$$\begin{aligned}\left\|\frac{\partial f(\boldsymbol{w}, x)}{\partial w_j^{(2)}}\right\|_2^2 &= |\langle W_{j,:}^1, x\rangle|^2, \text{Where } W_{j,:}^1 \text{ are the } j^{th} \text{ row of } W^1 \\ &\leq B_1^2 \cdot D^2\end{aligned}$$

The effective dimension of x is D so the above dot product dimension will also be bounded by D as x is 0 in all other dimensions.

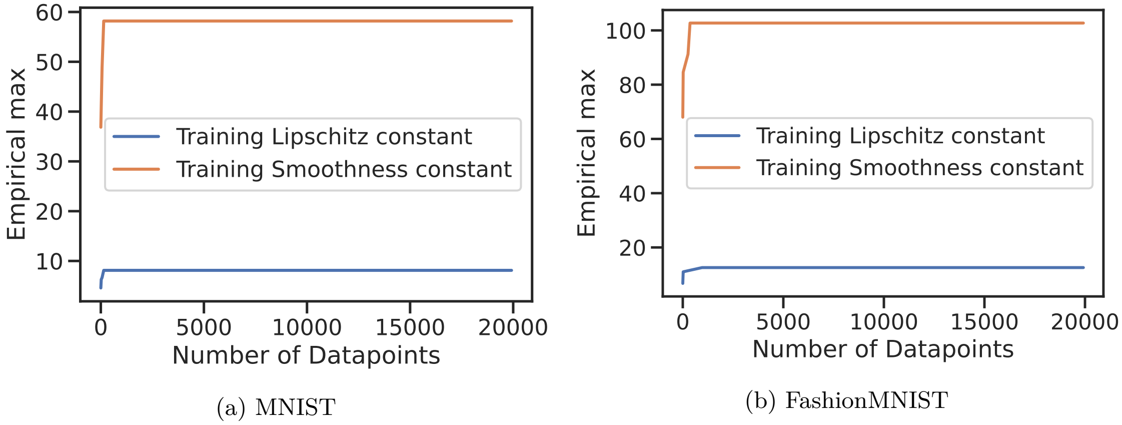
$$\begin{aligned}\left\|\frac{\partial f(\boldsymbol{w}, x)}{\partial w_{i,j}^{(1)}}\right\|_2^2 &= |w_i^{(2)} x_j|^2 \\ &\leq B_1^2\end{aligned}$$

So summing up the squared partial derivatives across all parameters we get,

$$\begin{aligned}\|\nabla_{\boldsymbol{w}} f(\boldsymbol{w}, x)\|_2^2 &\leq d_1 \cdot D \cdot B_1^2 + d_1 \cdot D^2 \cdot B_1^2 \\ &= B_1^2 \cdot d_1 \cdot D(1 + D)\end{aligned}$$

Here we see that we obtain a bound on the norm of the gradients that is related to D which is significantly smaller than d, whereas in general we can expect the norm of the gradients to be of the order of d even under the assumptions of bounded weights and rescaled data points.

Note that we show here the value of L_S and L_g but for generalization we actually need $\mathbb{E}_r[L_S \cdot L_g]$ to be bounded. Using Cauchy-Schwarz inequality we could see that we need bound on just the expectation 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.

square (or the second moment) of each term. This means that when we select the initial weight parameter vector w_0 we need the boundedness constraints on weights only which is a fairly mild constraint.

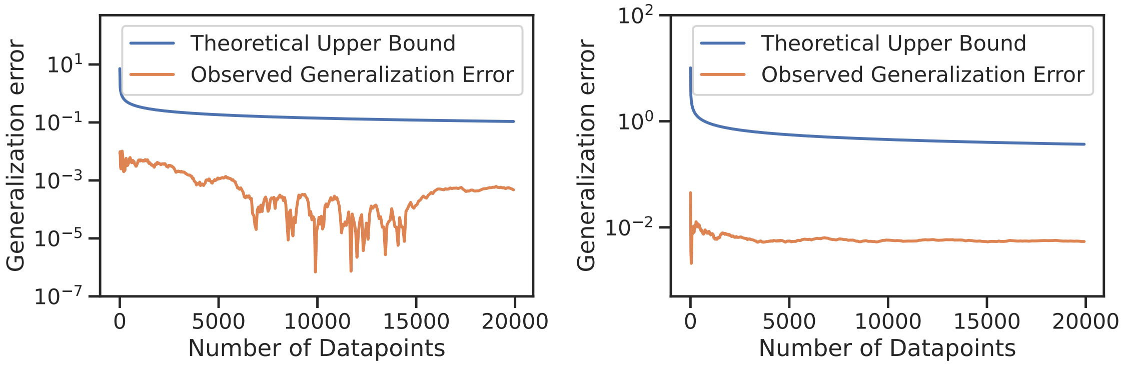
5.1.1 Experimental Validation of Results

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 ($L_g \leq L_S$).

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, $\alpha_0 \leq \frac{\rho(\tau,m)}{K_S}, \rho(\tau, m) = \frac{\log\log m}{\log\tau+\log m}$. 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 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



(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.

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.

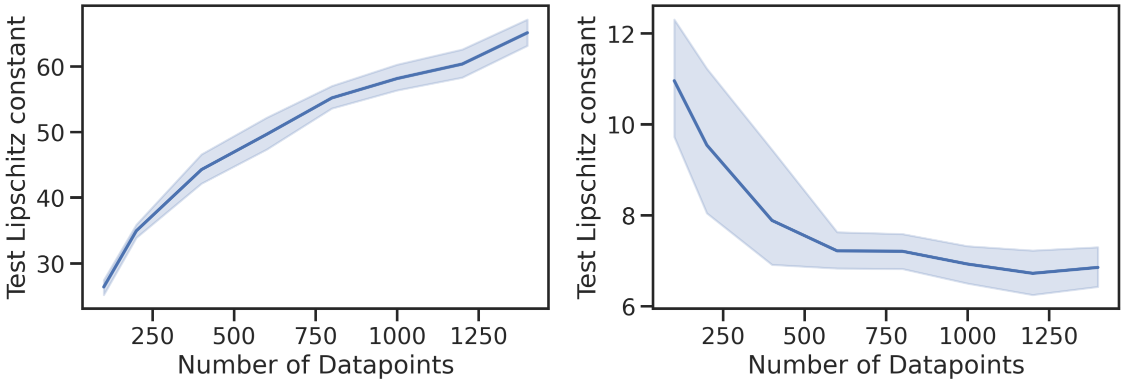
5.2 Random Labelling Case

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 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



(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

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

- *Removing the fully connectedness constraint.* Although Theorem 5.1 is stated for a fully connected network, we conjecture that 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. Our work provides a framework in which the study of the gradients obtained during training such networks can help guide our understanding of their generalization properties.

- *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. Here as well a mild addition for bound on derivative of regularization term in Theorem 5.3 may be able to help us prove results for this setting. However, it requires further investigation to establish such a result.

- *Activation functions apart from ReLU.* We present a comprehensive treatment of ReLU activation but we conjecture that results are not 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. This provides a direction for future research in this area.

- *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. However, this requires a full treatment which we postpone to future work.

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 D.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*, 3:463–482, 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.

Richard Combes. An extension of McDiarmid's inequality, 2015. URL `https://arxiv.org/abs/1511.05240`.

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.

Mahdi Haghifam, Jeffrey Negrea, Ashish Khisti, Daniel M. Roy, and Gintare Karolina Dziugaite. Sharpened generalization bounds based on conditional mutual information and an application to noisy, iterative algorithms, 2020.

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`.

Zinan Lin, Vyas Sekar, and Giulia Fanti. Why spectral normalization stabilizes gans: Analysis and improvements. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 9625–9638. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper_files/paper/2021/file/4ffb0d2ba92f664c2281970110a2e071-Paper.pdf`.

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://papers.nips.cc/paper/2019/file/b33128cb0089003ddfb5199e1b679652-Paper.pdf`.

Jeffrey Negrea, Mahdi Haghifam, Gintare Karolina Dziugaite, Ashish Khisti, and Daniel M. Roy. Information-theoretic generalization bounds for sgld via data-dependent estimates, 2020. URL `https://arxiv.org/abs/1911.02151v3`.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks, 2019.

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.

Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning, 2017. URL `https://arxiv.org/pdf/1705.10941.pdf`.

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-zm08j5lq`.

A Modification of McDiarmid's Theorem

Symbol	Explanation
L_S	Train Lipschitz constant.
L_g	Test Lipschitz constant.
K_S	Train Smoothness constant.
$\mathcal{X}, \mathcal{Y}$	Input and output Space.
D	Distribution over $\mathcal{Z} \subset \mathcal{X}\mathcal{Y}$.
$z = (x_z, y_z)$	Input point and label picked from distribution D defined over $\mathcal{Z}$.
$r \in \mathcal{R}$	random string from a random set to show randomness in an algorithm.
S	Training set of size m.
S^i	Training set S with i^{th} point replaced by another point picked i.i.d from D.
$A_{S,r}$	Training Algorithm.
ℓ	Bounded value Loss function with domain $\mathcal{Y} \times \mathcal{Y} \to [0, M]$.
$R(A_{S,r})$	Risk (Population error).
$R_e(A_{S,r})$	Empirical Risk (Training error).
$\boldsymbol{w}_t$	Weight of the parameterized function trained by SGD at t^{th} step.
α_t	Learning rate at t-th step of SGD.
$\|\cdot\|_\sigma$	Spectral norm of matrix.
τ	Total number of epochs, each epoch is of m step, $\boldsymbol{w}_0$ is the initial weight.
$\pi \in \Pi$	π is some permutation of m points picked from set of all possible permutation Π.
r_{init}	Random initialization i.e. $\boldsymbol{w}_0$.
r_p	a Random permutation for m points.
$r = (r_{init}, r_p)$	Random string r having $\boldsymbol{w}_0$ and $\{\pi_i\}_{i=0}^{\tau-1}$ i.e., for all epochs.
L_l	Local parameter Lipschitz constant.
K_l	Local parameter Smoothness constant.
W^l	Weight matrix of l-th layer on NN.
$W^l_{j,:}$	The row of l-th layer weight of NN.
$w^{(l)}_{i,j}$	weight value of l-th layer from i^{th} neuron of l-th layer to j-th neuron of $l+1$-th layer.
$\mathcal{T}$	Size of test set.
$f(\boldsymbol{w}_t, z)$	Loss at t-th step of SGD computed on point z.
$f(r, S, z)$	Loss of NN trained on set S and evaluated on point z.

Table 2: Notation used in the body of the paper.

We first define a probabilistic weakening of bounded difference property.

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-bounded difference w.r.t. D* if

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

with probability at least $1 - \eta$. In case $\eta = 0$ we say that f satisfies *almost surely β-bounded difference 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- bounded difference functions.

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

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

Lemma A.2 follows directly from a result shown in (Combes, 2015, Proposition 2). Since the proof is available in Combes (2015) we omit it here.

B Almost Sure (a.s.) Support Stability of SGD Proved

Proof of Lemma 4.2. Let f be the partial function of $\boldsymbol{w}$ (i.e., assuming z is already given) 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 is valid for all the partial functions (i.e., for all $z \in \Omega$) and hence proves the theorem. ☐

Now we define the weight set for Test Lipschitz constant.

Definition B.1. The weight set for Test Lipschitz constant A is the set of weights encountered while training through SGD, for all possible permutation of points in training set S. Let the set of all permutations of S be Π and $\pi \in \Pi$ be a permutation of S, let $\boldsymbol{w}_{i,\pi}$ be the weight at i-th step of SGD where gradients are computed using first i points of S indexed using permutation π. So weights encountered across all permutations

$$A = \bigcup_{\pi \in \Pi} \{\boldsymbol{w}_{1,\pi}, \boldsymbol{w}_{2,\pi}, \cdots, \boldsymbol{w}_{m,\pi}\}.$$

Proof of Theorem 4.3. 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 $z \in \mathrm{supp}\,(D)$, 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 4.2 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

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

with probability 1 since from Lemma 4.2 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

$$\delta_m \leq 2\alpha_{I_0} L_S \prod_{t=I_0+1}^{m} (1 + \alpha_t K_S) \leq 2\alpha_{I_0} L_S \exp\left\{ \sum_{t=I_0+1}^{m} \alpha_t K_S \right\}, \tag{5}$$

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

$$\delta_{\tau m} \leq 2\alpha_{I_0} L_S (2)^{\tau-1} \exp\left\{ \sum_{t=I_0+1}^{\tau m} \alpha_t K_S \right\} \leq \alpha_0 L_S 2^{\tau} \frac{1}{I_0^{1-\rho(\tau,m)}} \exp\left\{ \frac{\alpha_0 K_S \left((\tau m)^{\rho(\tau,m)} - I_0^{\rho(\tau,m)} \right)}{\rho(\tau, m)} \right\}. \tag{6}$$

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 (6) 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

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

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

$$\mathrm{E}_{r_p}\left[L_g \delta_T | r_{init}\right] \leq 2^{\tau} \alpha_0 L_g L_S \cdot U\left(\alpha_0, K_S, \rho(\tau, m)\right) \cdot \frac{\exp\left\{\frac{\alpha_0 K_S}{\rho(\tau,m)} (\tau m)^{\rho(\tau,m)}\right\}}{m} \tag{7}$$

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

C An improved generalization result with "Expected" Lipschitz constants and an application

We present a more fine-grained analysis that improves the bounds of Theorem 4.3 and Theorem 5.1 by replacing the worst-case Lipschitz constant L_S with the square root of the second moment of the random Lipschitz constants encountered during training. With this new analysis we are able to present a concrete example where the applicability of the bounds can be established directly.

C.1 The improved result

A key issue with the use of the constant L_S is that it is a worst-case constants over the entire training trajectory. In fact our analysis allows for a more fine-grained analysis that may give better results in certain cases. Below we present versions of our key theorems where the bounds depend on an "expected" L_S which is in fact the square-root of the second moment of the random Lipschitz constants encountered during training. To understand what this means, let us note that as SGD training proceeds, and if it is able to converge to a minima, the Lipschitz constants encountered decrease. Now if we consider the index i of the training set that has been changed as per the definition of η-almost β- bounded difference, a random permutation of S will locate this index at a randomly selected position between 1 and m in the training sequence. If we look at a particular trace of SGD, the bounding constant L_S can actually be replaced with the constant encountered at this position. Hence, when the expectation is taken over the random choices, the bounding value is related to the expected value of the Lipschitz constants encountered. *The significance of this is that this expected value could be substantially smaller than the worst case value in many cases, especially when the training converges from a high value of the loss to a low value rapidly.*

We formalize this argument below. For simplicity we present the analysis for a single epoch of training.

For a given choice of initial paramters of SGD, we define $L_{S,i}$ to be the maximum over all possible permutations of S of the Lipschitz constant encountered at i-th step of the training. Note that $max_{i\in[m]}(L_{S,i}) = L_S$. And since we have shown in Lemma 4.2 that L_S is bounded so $L_{S,i}$ will also be bounded for every $i \in [m]$.

We now state the revised version of Theorem 4.3. Note that the random string r will be presented in two independent parts (r_{init}, r_p) with the former being the random choice of initial parameters and the latter being the random permutation used by SGD.

Theorem C.1. *Given the assumption S1, S2 and S3 for* $\tau = 1$, *we have bounded values* L_g, K_S *and* L_{S,I_0} *for each* $I_0 \in [m]$, *which are functions of the random string* $r = (r_{init}, r_p)$, *and we have with probability* 1 *over* z,

$$E_r\left[f(\boldsymbol{w}_m, z) - f(\boldsymbol{w}'_m, z)\right] \leq \alpha_0 \cdot E_{r_{init}}\left[\frac{L_g \sqrt{E_{r_p}\left[L^2_{S,I_0}\right]}\sqrt{U'(\alpha_0, K_S, m)}}{m^{\frac{1}{2} - \frac{\alpha_0 K_S(\log m - 1)}{\log\log m}}}\right] \tag{8}$$

Where $U'(\alpha_0, K_S, m) \leq 1 + \frac{1}{2\alpha_0 K_S}$ *and as* $\alpha_0 \to 0$, $U'(\alpha_0, K_S, m) \to 1 + \frac{\log(m)^2}{\log\log m}$ *and* $E_{r_p}\left[L_{S,I_0}\right]$ *is expectation of* L_{S,I_0} *over* $I_0 \in [m]$.

The key point to note here is that we now have $\sqrt{\mathrm{E}_{r_p}\left[L^2_{S,I_0}\right]}$.

Proof sketch We provide a proof sketch as it's very similar to the proof of Theorem 4.3. The main point is that in line 640, we can actually use L_{S,I_0} instead of L_S as it's the gradient at I_0-th step. The rest of the steps follow similarly till equation 5 so we get

$$\delta_m \leq 2\alpha_{I_0} L_{S,I_0} \exp\left\{\sum_{t=I_0+1}^{m} \alpha_t K_S\right\}$$

Now calculating $\mathrm{E}_r\left[L_g \delta_m\right]$, note that earlier (in line 622) L_S was constant w.r.t the permutation (r_p) but here L_{S,I_0} depends on the step, so taking expectation over permutation is equivalent to taking expectation over random variable I_0 which is picked uniformly from 1 to m.

$$\mathrm{E}_{r_p}\left[L_g \delta_m | r_{init}\right] \leq \alpha_0 L_g \cdot \mathrm{E}_{r_p}\left[\frac{L_{S,I_0}}{I_0^{1-\frac{\log\log m}{\log m}}} \exp\left\{\sum_{t=I_0+1}^{m} \alpha_t K_S\right\}\right]$$

So using Cauchy Schwarz inequality to separate expectation over the random variable L_{S,I_0}

$$\mathrm{E}_{r_p}\left[L_g \delta_m | r_{init}\right] \leq \alpha_0 L_g \cdot \sqrt{\mathrm{E}_{r_p}\left[L_{S,I_0}^2\right]} \sqrt{\frac{1}{m}\sum_{I_0=1}^{m} \frac{1}{I_0^{2-\frac{2\log\log m}{\log m}}} \exp\left\{\sum_{t=I_0+1}^{m} 2\alpha_t K_S\right\}}$$

Upper bounding summation inside exponent by integration exactly like we did in equation 6. we get the second square-root terms as

$$\leq \frac{1}{m}\sum_{I_0=1}^{m} \frac{1}{I_0^{2-\frac{2\log\log m}{\log m}}} \exp\left\{\frac{2\alpha_0 K_S\left((m)^{\frac{\log\log m}{\log m}} - I_0^{\frac{\log\log m}{\log m}}\right)\log m}{\log\log m}\right\}$$

Assuming $p = \frac{\log\log m}{\log m}$ to simplify the equation,

$$\leq \frac{1}{m}\exp\left\{\frac{-2\alpha_0 K_S}{p}\right\}\left(\exp\frac{-2\alpha_0 K_S}{p} + \sum_{I_0=2}^{m}\frac{1}{I_0^{(2-2p)}}\exp\left\{\frac{-2\alpha_0 K_S I_0^p}{p}\right\}\right)$$

Now we use integration to bound the summation term,

$$\sum_{I_0=2}^{m}\frac{1}{I_0^{(2-2p)}}\exp\left\{\frac{-2\alpha_0 K_S I_0^p}{p}\right\} = \int_{x=1}^{m} x^{-2(1-p)}\exp\left\{\frac{-2\alpha_0 K_S}{p}x^p\right\}$$

Substituting $u = \frac{x^p}{p}$, and in next step upper bounding $(u \cdot p)^{(1-1/p)}$ by 1 (as $p < 1$ and $u \cdot \geq 1$ throughout the limit) and then putting the limits.

$$\begin{aligned}\sum_{I_0=2}^{m}\frac{1}{I_0^{(2-2p)}}\exp\left\{\frac{-2\alpha_0 K_S I_0^p}{p}\right\} &= \int_{u=\frac{1}{p}}^{m^p/p} (u \cdot p)^{(1-1/p)}\exp\left\{-2\alpha_0 K_S u\right\} \\ &\leq p^{1-1/p}\int_{u=1/p}^{m^p/p}\exp\left\{-2\alpha_0 K_S u\right\} \\ &\leq \frac{p^{1-1/p}}{2\alpha_0 K_S}\exp\left\{\frac{-2\alpha_0 K_S}{p}\right\}\end{aligned}$$

Using this and putting the value of p and taking $p^{1-1/p}$ we get,

$$\mathrm{E}_{r_p}\left[L_g \delta_m | r_{init}\right] \leq \alpha_0 L_g \cdot \sqrt{U'\left(\alpha_0, K_S, m\right)}\sqrt{\mathrm{E}_{r_p}\left[L_{S,I_0}^2\right]} \cdot \frac{1}{m^{\frac{1}{2} - \frac{\alpha_0 K_S(\log m - 1)}{\log\log m}}} \tag{9}$$

where

$$U'(\alpha_0, K_S, m) = 1 + \frac{1}{2\alpha_0 K_S}$$

Now applying this to NNs we get a generalization error result which is an alternate version of Theorem 5.1 for the 1-epoch case.

Theorem C.2. *If N1, N2 and N3 hold and* $r = (r_{init}, r_p)$ *then* K_S *is constant w.r.t* m *for all* r *and* $E_{r_{init}}\left[L_g^2\right]$ *and* $E_r\left[L_{S,I_0}^2\right]$ *are also constants w.r.t.* m*, with* $c > 0$ *such that*

$$E_r\left[\|R(S,r) - R_e(S,r)\|\right] \leq c\left(\alpha_0 \cdot \sqrt{E_{r_{init}}\left[L_g^2\right] E_r\left[L_{S,I_0}^2\right] \cdot U'(\alpha_0, K_S, S)}\frac{\log(m)^2}{m^{\left(\frac{1}{2} - \frac{\alpha_0 K_S(\log m - 1)}{\log\log m}\right)}} + \sqrt{\frac{\log(m)}{m}}\right),$$

with probability at least $1 - 1/m$*, where* $U'(\alpha_0, K_S, m) \leq 1 + \frac{1}{2\alpha_0 K_S}$ *and as* $\alpha_0 \to 0$, $U'(\alpha_0, K_S, m) \to 1 + \frac{\log(m)^2}{\log\log m}$ *and* $E_r\left[L_{S,I_0}^2\right]$ *is expectation of* L_{S,I_0}^2 *over* $I_0 \in [m]$ *(i.e.,* r_p*) and also* r_{init}*.*

Note the presence of $\sqrt{\mathrm{E}_r\left[L_{S,I_0}^2\right]}$ on the R.H.S here as opposed to L_S in the statement of Theorem 5.1

Proof sketch It is easy to see that using C.1, Theorem 5.2 and Proposition 5.3 and putting the value of α_0 we get this result.

C.2 A framework for bounding the second moment of the Lipschitz constant

In order to bound$\mathrm{E}_r\left[L_{S,I_0}^2\right]$, we use result from (Bottou et al., 2018, Theorem 4.10) and modify it to get the result. This theorem provides a bound on the second moment of the random gradients computed during SGD. The bound is in terms of the initial and optimal (final) values of the loss function (and includes terms based on the learning rate). Therefore the bound is very useful in practice. The theorem requires the following condition which is a kind of version of a bound on the variance of the gradients encountered during training.

P1. (Assumption 4.3 (equation 4.9) of Bottou et al. (2018)). There exist constants $M \geq 0$ and $M_G \geq 0$ (independent of the size of training set) such that,

$$\mathrm{E}_{z \in S}\left[\|\nabla f(\boldsymbol{w}_t, z)\|^2\right] \leq M + M_G \|\mathrm{E}_{z \in S}\left[\nabla f(\boldsymbol{w}_t, z)\right]\|^2. \tag{10}$$

Theorem C.3. *(Bottou et al., 2018, Theorem 4.10) Suppose we run SGD under the assumption that* K_S *is constant for all* r *and P1 holds, for* $M \geq 0$, $M_G \geq 0$*. If* $\boldsymbol{w}_0$ *is the initialization weight,* $\boldsymbol{w}^*$ *is optimal value for* $E_{z \in S}\left[f(\boldsymbol{w}, z)\right]$ *and* α_t *is any diminishing step size then,*

$$\sum_{t=1}^{T} \alpha_t E_{z \in S}\left[\|\nabla f(\boldsymbol{w}_t, z)\|^2\right] \leq 2(E_{z \in S}\left[f(\boldsymbol{w}_0, z)\right] - E_{z \in S}\left[f(\boldsymbol{w}^*, z)\right]) + K_S M \sum_{t=1}^{T} \alpha_t^2 \tag{11}$$

Now using Theorem C.3 and assumption P1 we can get a bound on $\mathrm{E}_r\left[L_{S,I_0}^2\right]$, we state this in the next corollary

Corollary C.4. *If we run Stochastic Gradient Descent for* 1 *epoch under the assumption that* K_S *is constant for all* r *and P1 holds, for* $M \geq 0$, $M_G \geq 0$, $\boldsymbol{w}_0$ *is the initialization weight,* $\boldsymbol{w}^*$ *is optimal value for* $E_{z \in S}\left[f(\boldsymbol{w}, z)\right]$, $r = (\boldsymbol{w}_0, r_p)$ *is the random string for SGD and* α_t *is any diminishing learning rate then,*

$$\alpha_0 E_r\left[L_{S,I_0}^2\right] \leq \alpha_0 M + 2\alpha_0 M_G E_{\boldsymbol{w}_0, z \in S}\left[f(\boldsymbol{w}_0, z) - f(\boldsymbol{w}^*, z)\right] + E_{\boldsymbol{w}_0}\left[K_S\right] M_G M \sum_{t=0}^{T} \alpha_t^2 \tag{12}$$

The R.H.S of this equation is constant w.r.t. m f $\sum_{t=1}^{T} \alpha_t^2$ is constant w.r.t. m. In our analysis this is true (c.f. assumption S3). Hence this corollary becomes useful for us since it shows that under mild and reasonable assumptions we can bound the second moment of the Lipschitz constants encountered during training.

Proof of C.4. Writing assumption P1 averaged over T steps and multiplying both side by α_0 we get,

$$\frac{1}{T}\sum_{t=1}^{T}\alpha_0 \mathrm{E}_{z\in S}\left[L^2_{S,I_0}\right] \leq \alpha_0 M + \frac{\alpha_0 M_G}{T}\sum_{t=1}^{T}\|\mathrm{E}_{z\in S}\left[\nabla f(\boldsymbol{w}_t, z)\right]\|^2$$

L.H.S. could be written as $\alpha_0 \mathrm{E}_{r_p}\left[L^2_{S,I_0}\right]$ as r_p is the random permutation string of SGD. Using $\frac{\alpha_0}{T}\sum_{t=1}^{T}\|\mathrm{E}_{z\in S}\left[\nabla f(\boldsymbol{w}_t, z)\right]\|^2 \leq \sum_{t=1}^{T}\alpha_t \mathrm{E}_{z\in S}\left[\|\nabla f(\boldsymbol{w}_t, z)\|^2\right]$ we can use bound of theorem C.3, and taking expectation over $\boldsymbol{w}_0$ we get the result.

□

C.3 Application: Classification with a two-layer Neural Network

We now use CorollaryC.4 to establish a generalization bound for the case of two-class classification with a two-layer Neural Network. For simplicity of exposition we have assumed that the data points are taken from $\mathbb{R}^2$. We fully specify the problem through the following assumptions:

X1. We have a two class classification ($\mathcal{Y} = \{-1, 1\}$) in 2 dimension ($x = [x_1, x_2]$) such that for expectations of centers we have $\mathrm{E}\left[x_1|y=1\right] = \mathrm{E}\left[x_2|y=1\right] = 2$ and $\mathrm{E}\left[x_1|y=-1\right] = \mathrm{E}\left[x_2|y=-1\right] = -1$, for second moment for a constant $\sigma > 0$ we have $\mathrm{E}\left[x_1^2\right] = \mathrm{E}\left[x_2^2\right] = \sigma^2$ and also for a constant $\mu_p > 0$ $\mathrm{E}\left[|x_1|\right] = \mathrm{E}\left[|x_2|\right] = \mu_p$.

X2. We use a single hidden layer feed-forward neural network, with k as hidden layer size. Its parameters are initialized from $w^{(l)}_{i,j} = \mathcal{N}(0, 1)$. The total number of parameter values are $n = 2k + k$. For a loss function $f(\boldsymbol{w}, z) = |y - O(\boldsymbol{w}, x)|$, where $O(\boldsymbol{w}, x)$ is the output of the neural network, $\nabla f(\boldsymbol{w}, z)_j$ denotes the j-th partial derivative of function f and $w^{j_1}_{j_2,j_3}$ be the associated weight. Let α_t be any diminishing step size at step t of SGD.

X3. We assume the ratio of absolute values of the weights of the Neural Network are bounded by B where $B > 1$.

Theorem C.5. *If X1,X2,X3 hold, we have*

$$E_r\left[L^2_{S,I_0}\right] \leq \frac{16B^4\sigma^2\left(\pi + 4k\sigma^2\mu_p\right)}{(B-1)^2\pi} \tag{13}$$

Since the distance between the centers of the two classes is fixed at 2, the bound presented here satisfies our intuition by showing that if the variances of the two classes are small, i.e., the classes are well-separated, then the second moment of the Lipschitz constants encountered during training is small, and hence the generalization bound of Theorem C.2 is small.

Proof of Theorem C.5. First lets assume that assumption P1 holds so we need to show $\mathrm{E}_{\boldsymbol{w}_0, z\in S}\left[f(\boldsymbol{w}_0, z)\right]$ is bounded, so calculating the value for this,

$$\begin{aligned} f(\boldsymbol{w}_0, z) &= |y - O(\boldsymbol{w}, x)| \\ &= \left|y - \sum_{i=1}^{k} w^{(2)}_i w^{(1)}_{i,1} x_1 - \sum_{i=1}^{k} w^{(2)}_i w^{(1)}_{i,2} x_2\right| \\ &= |y| + \left|\sum_{i=1}^{k} w^{(2)}_i w^{(1)}_{i,1} x_1\right| + \left|\sum_{i=1}^{k} w^{(2)}_i w^{(1)}_{i,2} x_2\right| \end{aligned}$$

Let $o_1(\boldsymbol{w}_0) = \sum_{i=1}^{k} w^{(2)}_i w^{(1)}_{i,1}$ and $o_2(\boldsymbol{w}_0) = \sum_{i=1}^{k} w^{(2)}_i w^{(1)}_{i,2}$ for ease of writing, then take expectation over $\boldsymbol{w}_0$, we directly place $\mathrm{E}_{\boldsymbol{w}_0}\left[|o_1(\boldsymbol{w}_0)|\right] = \mathrm{E}_{\boldsymbol{w}_0}\left[|o_2(\boldsymbol{w}_0)|\right] \leq \frac{2k\sigma^2}{\pi}$ because of half normal distribution and i.i.d assumption ,i.e.,

$$\mathrm{E}_{\boldsymbol{w}_0}\left[f(\boldsymbol{w}_0, z)\right] \leq |y| + \frac{2k\sigma^2}{\pi}(|x_1| + |x_2|)$$

Taking expectation over z, we have

$$\mathrm{E}_{\boldsymbol{w}_0, z\in S}\left[f(\boldsymbol{w}_0, z)\right] = 1 + \frac{4k\sigma^2\mu_p}{\pi} \tag{14}$$

Now, we show the assumption P1 holds. Note that we ignore the case when weights are exactly zero or weights become exactly equal to other weights to avoid zero in the denominator. We take $M = 0$ from assumption P1. Now we take the upper bound of L.H.S. of assumption P1 (without expectation),

$$\|\nabla f(\boldsymbol{w}, z)\|^2 \leq n \cdot \max_j(\nabla f(\boldsymbol{w}, z)_j^2)$$

and we take the lower bound of R.H.S. using

$$n \cdot \min_i(\mathrm{E}_{z\in S}\left[\nabla f(\boldsymbol{w}, z)_i\right])^2 \leq \|\mathrm{E}_{z\in S}\left[\nabla f(\boldsymbol{w}, z)\right]\|^2$$

Using the above two inequalities and taking $M = 0$ in assumption P1, we get,

$$\left(\frac{\mathrm{E}_{z\in S}\left[\max_{j\in[n]}\{\nabla f(\boldsymbol{w}, z)_j^2\}\right]}{\min_{i\in n}\{\mathrm{E}_{z\in S}\left[\nabla f(\boldsymbol{w}, z)_i\right]^2\}}\right) \leq M_G$$

Now, calculating for numerator, we first write the max over the square of gradients,

$$\begin{aligned}
\max_j\{(\nabla f(\boldsymbol{w}, z)_j)^2\} &= \max_j\left\{\left(\frac{\partial O(\boldsymbol{w}, x)}{\partial w^{(j_1)}_{j_2,j_3}}\right)^2\right\} \\
&= \max\{(w^{(2)}_{j_2})x^2_{j_3}\}, && \text{if } j_1 = 1 \\
&= \max\{(w^{(1)}_{j_2,1}x_1 + w^{(1)}_{j_2,2}x_2)^2\}, && \text{if } j_1 = 2
\end{aligned}$$

Let w_h be the highest absolute value of weight(s) and w_l be the lowest absolute value of weight(s). To easily calculate expectation, we take out the max weights across all, we get an upper bound for the numerator,

$$\mathrm{E}_{z\in S}\left[\max_j\left\{(\nabla f(\boldsymbol{w}, z)_j)^2\right\}\right] \leq 2w_h^2\sigma^2 \tag{15}$$

Now lower bounding denominator, so square of expectation of partial derivative, i.e.,

$$\begin{aligned}
\mathrm{E}_{z\in S}\left[\nabla f(\boldsymbol{w}, z)_i\right]^2 &= \left(\mathrm{E}_{z\in S}\left[\frac{\partial O(\boldsymbol{w}, x)}{\partial w^{(i_1)}_{i_2,i_3}}\right]\right)^2 \\
&= (w^{(2)}_{i_2}\mathrm{E}\left[x_{i_3}\right])^2, && \text{if } i_1 = 1 \\
&= (w^{(1)}_{i_2,1}\mathrm{E}\left[x_1\right] + w^{(1)}_{i_2,2}\mathrm{E}\left[x_2\right])^2, && \text{if } i_1 = 2
\end{aligned}$$

For $i_1 = 2$ term, after taking expectation, we could write it as,

$$(w^{(1)}_{i_2,1}\mathrm{E}\left[x_1\right] + w^{(1)}_{i_2,2}\mathrm{E}\left[x_2\right])^2 = \frac{1}{4}\left(w^{(1)}_{i_2,1} + w^{(1)}_{i_2,1}\right)^2$$

Since we have B for all ratios of weights we could use this to bound below the absolute difference between any pair of weights (i.e., $|w_l' - w_l| \geq |w_l/B - w_l|$), and we get

$$(w^{(1)}_{i_2,1}\mathrm{E}\left[x_1\right] + w^{(1)}_{i_2,2}\mathrm{E}\left[x_2\right])^2 \geq w_l^2(B-1)^2/B^2$$

So we can bound the whole denominator by,

$$\min_i \left\{ (\mathrm{E}_{z\in S}\left[\nabla f(\boldsymbol{w}, z)_i\right])^2 \right\} \geq \frac{1}{4} \min \left\{ w_l^2, \frac{w_l^2 (B-1)^2}{B^2} \right\} \tag{16}$$

$$\geq \frac{w_l^2 (B-1)^2}{4B^2} \tag{17}$$

Using 15 and 16 we get,

$$M_G = \frac{8B^4\sigma^2}{(B-1)^2} \tag{18}$$

And from 14 and 18 we get the theorem statement. □

D Neural Networks: Characterization and Proofs

Symbol	**Explanation**
k_l	Number of neurons in l-th layer.
H	Depth of neural network.
$\mathsf{in}_{i,j}(\boldsymbol{w}, x)$	input to the i-th neuron of j-th layer.
$\mathsf{out}_{i,j}(\boldsymbol{w}, x)$	Output of the i-th neuron of j-th layer.
$\mathsf{out}_{H,1} = out(\boldsymbol{w}, x)$	Is the label returned by the neural network.
$\phi(\boldsymbol{w}, x)$	Polynomial associated with fully connected NN.
$\phi_{i,j}(\boldsymbol{w}, x)$	Base polynomial associated with j-th neuron of i-th layer.
$G_{\boldsymbol{w},x}$	The set of weights need to set to zero, to apply all closed ReLU gate in NN (with ReLU).
$\phi(\boldsymbol{w}, x)\{G_{\boldsymbol{w},x}\}$	A neural network with ReLU activation with closed ReLU gates set to 0.
$\mathsf{lab}(x)$	label of point x.
I_j	Position in permutation in j-th epoch when i-th training points (i.e. the replaced point) is encountered based on π_j.
δ_t	Norm of difference between weights for S and S^i at t-th step of SGD (i.e., $\lVert \boldsymbol{w}_t - \boldsymbol{w}'_t \rVert$).

Table 3: Notation used in section D

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 D.1 and the proof in Section D.2.

D.1 A Polynomial-based Characterization Neural Networks

D.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 $\mathcal{Y}$. For convenience we will denote this architecture as $\mathcal{N}$.

D.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, $p(x)\{_{x_i=0}\}$ is the polynomial $p(x)$ with all monomials containing x_i removed, and $p(x)\{_{x_i=1}\}$ 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 $k_0 = d$. Let us consider $\mathcal{N}$ defined in Sec. D.1.1 and create a version of it that replaced the ReLU activation at each node with the identity activation function. We will call this the *identity version* of $\mathcal{N}$ and denote it $I(\mathcal{N})$. We will say that $I(\mathcal{N})$ 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 $\mathcal{N}$. The base polynomial associated with the jth neuron in layer i can be derived from the base polynomial of the network, we express this in figure 4 and also write formally as follows

$$\phi_{i,j}(\boldsymbol{w}, x) = \frac{\phi(\boldsymbol{w}, x)\left\{w^{(i)}_{l_1,l_2}=0, w^{(l_3)}_{l_4,l_5}=1 : l_1\in[k_i]\setminus\{j\}, l_2\in[k_{i+1}], l_3>i, l_4\in[k_{l_3}], l_5\in[k_{l_3+1}]\right\}}{\prod_{p=i+1}^{H} k_i}. \tag{19}$$

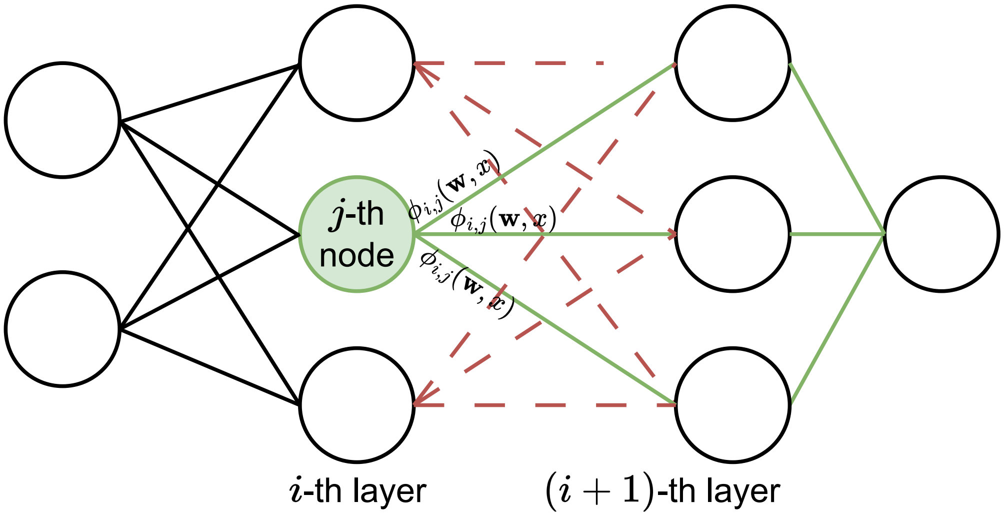

j-th node
$\phi_{i,j}(\mathbf{w}, \boldsymbol{x})$
$\phi_{i,j}(\mathbf{w}, \boldsymbol{x})$
$\phi_{i,j}(\mathbf{w}, \boldsymbol{x})$
i-th layer
$(i+1)$-th layer

Figure 4: The output of the j-th neuron of the i-th layer represented represented by the base polynomial. Here the weights along the dotted red lines are set to zero and the weights along the green lines are set to one. The output of the neuron is represented by the values on the connection. Notice that the output is scaled by the product of the number of intermediate nodes because of which we divide it later in 19.

Also we could describe a Network whose say i^{th} layer j^{th} neuron's gate is closed by $\phi(\boldsymbol{w}, x)\{w^{i}_{l_1,j} = 0, \forall l_1 \in k_{i-1}\}$, This is represented by the figure 5. 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,

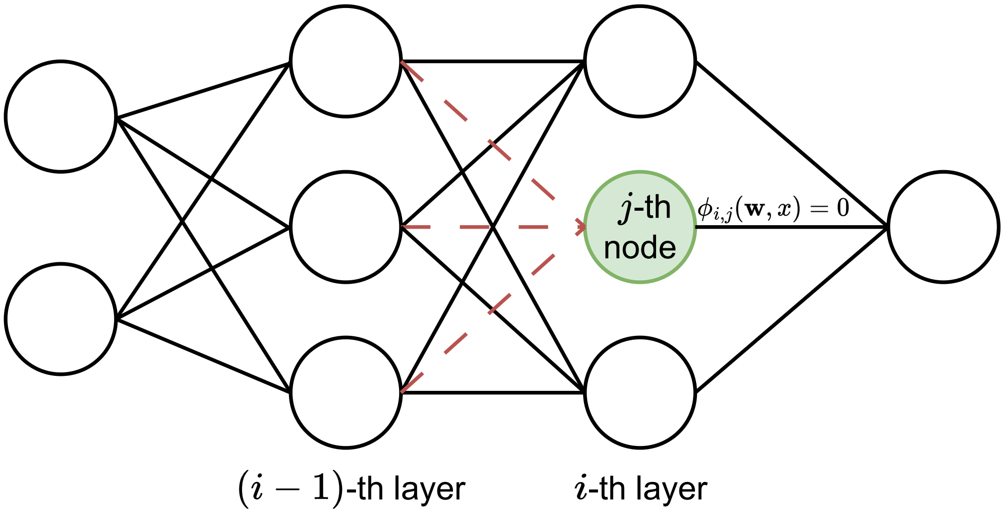


Figure 5: For a neural network with ReLU, if the i-th layers j-th neuron's ReLU gate is closed, this is represented by the base polynomial. Here the dotted red lines are set to zero.

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 D.1. *Given* $\boldsymbol{w} \in \mathbb{R}^n$ *and* $x \neq (0,\dots,0) \in \mathbb{R}^d, i \in [H], j \in [k_i]$ *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. ☐

D.2 Proof of Theorem 5.2

Proof of Theorem 5.2. The idea behind this proof is due to (Milne, 2019, Lemma 1) who used it for a different purpose. From Observation D.1 it follows that if we have $x \neq (0,\dots,0) \in \mathbb{R}^d$ 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 D.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 D.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 $x_1, \ldots, x^d_{\mathbb{R}}$. 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 $w^{(l)}_{i,j}$. For this let $I^{(l)}_{i,j}, A^{(l+1)}_j$ and $B^{(l-1)}_i$ be 3 matrices of size $W^{(l)}, W^{(l+1)}$ and $W^{(l-1)}$ respectively such that $I^{(l)}_{i,j}[i,i] = 1$ and reset all entries are 0, $A^{(l+1)}_j[k,j] = W^{(l+1)}[k,j], \forall k$ and rest all entries are 0 and $B^{(l-1)}_i[i,k] = W^{(l-1)}[i,k], \forall k$ and rest all entries are one. Using these 3 matrices and the weight matrices we can compute the gradient as

$$\frac{\partial \phi(\boldsymbol{w}, x)}{\partial w^{(l)}_{i,j}} = W^{(H)} \cdots W^{(l+2)} \cdot A^{(l+1)}_j \cdot I^{(l)}_{i,j} \cdot B^{(l-1)}_i \cdot W^{l-2} \cdots W^1 \cdot x \tag{20}$$

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

$$M'_{l,i,j} = A^{(l+1)}_j \cdot I^{(l)}_{i,j} \cdot B^{(l-1)}_i$$

Although we have scalar values taking spectral norm on both sides of eq 20 we get

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

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 $\boldsymbol{w}^{(l_1)}_{i_1,j_1}$ and $\boldsymbol{w}^{(l_2)}_{i_2,j_2}$ we have

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

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. ☐

E Rebuttal

We would like to thank all three reviewers for their close reading of our paper and their detailed comments and suggestions. We have tried to address as many of the comments and incorporate as many of the suggestions as possible. Below we have brought out lists of relevant statements from each review and addressed each of them. There are more than 50 such comments and their responses below.

We have marked the edited text in 2 colors. Red indicates that the changes are important and add some extra information/value to the paper. Blue indicates the changes are mainly to increase readability, and they do not affect the paper's claims. We request the reviewers to go through them and respond and help us further improve this work.

Thanks.

E.1 Reviewer 1,

ReviewComment E.1.1. **Reviewer Comment:** The write up is missing a clearly formulated target problem (neural network classification?) and explicitly formulated target question(s) corresponding to the problem (e.g. generalization). Note that targeting generalization requires explicit definitions such that it might be possible to judge the claims. For instance, I didn't see explicit description of the meaning of "generalization error" in the introduction. A short sentence saying that this is the gap between the expected (population) error and the empirical error rate would be sufficient to clarify what is meant, and better if a cross-reference to the formal definition in Section 3.1 is given, to avoid confusions.

Rebuttal: We have given a more nuanced introduction to the notion of "generalization error" at the beginning of para 1 of the intro [lines: 17-22] and also clarified the applicability of our techniques in terms of target problems at the bottom of Para 1 [lines: 39-41].1

ReviewComment E.1.2. **Reviewer Comment:** I flag for criticism the opening sentence "The low generalization error of Deep neural networks is now a well known empirical result" - if the meaning of generalization error is as I just described above, then this claim is unfounded. It might be that the authors (or the authors of the cited paper) are trying to refer in this claim to the gap between test error rate and training error rate. This needs clarification.

Rebuttal: We have clarified this. Please see the response to the previous pointE.1.1.

ReviewComment E.1.3. **Reviewer Comment:** Define formally the meaning of "generalization" such that readers may evaluate the claims regarding it.

Rebuttal: We have now added this in the first few lines of the paper as mentioned above.[line: 17-22]

ReviewComment E.1.4. **Reviewer Comment:** Clarify the setting of this work by formulating explicitly the target problem, which I think it is neural network classification.

Rebuttal: We have added in introduction at end of first paragraph [line: 39-40] that our result is valid for both classification and regression as there is no condition on some kind of separability for the data.1

ReviewComment E.1.5. **Reviewer Comment:** Accordingly, clearly restrict the loss function (s) under consideration. Is it not the cross-entropy as surrogate loss for training?

Rebuttal: We need a bounded value and twice differentiable loss function. We have added the condition on the loss function in Introduction at the end of first paragraph 1 and in Section 3.1 in paragraph "About the loss function" 3.1.

ReviewComment E.1.6. **Reviewer Comment:** The role of the set $\mathcal{R}$ and its elements (decision strings) is unclear. Needs discussion/illustration for adding clarity.

Rebuttal: We have added some discussion about the random string in Section 3.1 first paragraph [lines: 133-136] 3.1.

ReviewComment E.1.7. **Reviewer Comment:** The meaning of "loss function" needs to be specified before defining risks and other things. At least some discussion of the typical loss functions that are considered, and that one of those is referred to when we read "loss function" in the sequel.

Rebuttal: We have added some discussion on the loss function in Sec 3.1 [lines: 148-153;3.1].

ReviewComment E.1.8. **Reviewer Comment:** The notations x_z and y_z are unnecessary. Simply replace $z \sim D$ with $(x, y) \sim D$ and then you can write $l(A_{S,r}(x), y)$ in the definition of risk. Similarly, replace $z \in S$ with $(x, y) \in S$ in the sums for the empirical risk.

Rebuttal: We were trying to clearly specify the dataset point z, the label y_z and the input point x_z. We almost everywhere use z as a point of the dataset, this is the only reason we keep using z here as well. But if the reviewer still prefers it we can remove z from this argument.

ReviewComment E.1.9. **Reviewer Comment:** Reading sometimes $A_{S,r}$ and sometimes A_S or A caused confusion. Can the relation between these things be declared? Perhaps a formal definition of each of these things would help to clear the confusion.

Rebuttal: A_S didn't mean anything it was a typing mistake, we have corrected it (as $A_{S,r}$). We have also changed $R(A, S, r)$ to $R(A_{S,r})$ [line: 142; 3.1] while defining generalization error in Section 3.1 to improve clarity.

ReviewComment E.1.10. **Reviewer Comment:** When declaring S^i , after the definition perhaps add comments to the effect that S^i is formed by replacing the i th entry of S with an independent copy (which is taken as the i th entry from the "second set of size m "). By the way, this reminds of the double sample argument, which goes back to classical literature on statistical learning, which might be good to cite.

Rebuttal: We have added this now. Just above Definition 3.1 [lines: 155-157;3.2].

ReviewComment E.1.11. **Reviewer Comment:** Definition 3.1: Poor choices of terminology (η -almost support stability, a.s. support stability) which don't add much clarity. I suggest reformulating these things. The way I see it, the main idea being defined is that of "support stability" corresponding to the displayed condition, which may hold with some probability $1 - \eta$ or with probability 1 (almost surely). Then you could reformulate this definition writing "support stability β with probability $1 - \eta$ " and "support stability β almost surely" – the latter formulations are way more clear in conveying the meaning of what's being defined.

Rebuttal: We have modified Definition 3.1 and removed the η probability part as we are only using the almost surely part (as suggested by reviewer d5U1) now we only use a.s. support stability. 3.1

ReviewComment E.1.12. **Reviewer Comment:** I don't think the notation $[m]$ was declared. Just declare it somewhere before this definition.

Rebuttal: Done, above the Definition 3.1 [line: 159;3.2].

ReviewComment E.1.13. **Reviewer Comment:** If we choose a constant loss function $l = 1$, then this satisfies support stability β for any $\beta > 0$? This perhaps shows one of the problems in not restricting the meaning of "loss function" from the start.

Rebuttal: The reviewer is right. In Section 3.1 in the discussion on loss functions [lines: 148-153] we have clarified that we work with loss functions that can be minimized to some meaningful value. As we have mentioned now at the beginning of the intro, bounding what we call the generalization error is only useful when the empirical error is low.

ReviewComment E.1.14. **Reviewer Comment:** Also I'd like to flag the restriction "$\forall z \in supp(D)$" in this definition. It appears to contradict the statements "with probability at least $1 - \eta$" and "almost surely" so this needs clarification.

Rebuttal: Here the probability is over selecting the $2m$ points from D^{2m}, including the training points and their random replacements. The z mentioned here is the population data point on which the $A_{S,r}$ is applied after it has learned it's parameters.

ReviewComment E.1.15. **Reviewer Comment:** Actually, a few lines below it is stated that "This probability is defined over the random choices of $Z_1, \ldots, Z_2m$ " which then suggests that Definition 3.1 needs to be rewritten making this explicit. Since the distribution of the random sample (i.i.d. points) of size $2m$ is $D^{\otimes 2m}$, this would be written saying "$D^{\otimes 2m}$ -probability $1 - \eta$ " and "$D^{\otimes 2m}$ -almost surely"

Rebuttal: The reviewer is right. We have simplified the statement in Definition 3.1. [line: 160,3.1].

ReviewComment E.1.16. **Reviewer Comment:** Still, it is a very strong requirement that the inequality holds $\forall z \in supp(D)$ (and $\forall i \in [m]$). Could the authors give illustrative examples of cases for which it is possible to calculate or estimate this condition.

Rebuttal: The reviewer will recall that an even stricter notion of stability (Uniform stability) was defined in Bousquet & Elisseeff (2002). This was $\forall z$. Even for that definition the authors showed that it holds for some classical ML algorithms like soft margin SVM, bounded SVM regression and regularized least square regression. So, our weakening will hold for all these algorithms as well. We have added this in second paragraph after definition 3.1. [lines: 168-173;3.2]

ReviewComment E.1.17. **Reviewer Comment:** Theorem 3.2: Could the authors comment on the constant $c > 0$ please. The exact value of this constant can make all the difference between the bound being useful or it being useless.

Rebuttal: This constant comes from the result of Feldman and Vondrak (2019b). Since our analysis is asymptotic we only need this to be independent of m which it is. We have added this clarification just below Theorem 3.2. [line: 185;3.3]

ReviewComment E.1.18. **Reviewer Comment:** The meaning of "symmetric in distribution" needs to be specified (before Theorem 3.2).

Rebuttal: We have added a discussion in Section 3.3 [lines: 177-183; 3.3] before the statement of the theorem.

ReviewComment E.1.19. **Reviewer Comment:** Definitions 4.1 and 4.2: I don't know what meaning to map to "Given a set Ω defined over $\mathcal{Z}$.

Rebuttal: We meant Ω is a subset of $\mathcal{Z}$, we have changed this to "given a subset Ω of $\mathcal{Z}$", for clarity in Definition 4.1. Also please note that based on other reviews we have combined the Definitions 4.1 and 4.2 into a single definition for clarity. [line: 216;4.1]

ReviewComment E.1.20. **Reviewer Comment:** Theorem 4.4: Reading "We are given a labelled data set $\mathcal{Z}$ " is surprising. I thought $\mathcal{Z}$ was reserved for the space of all possible instances and labels, i.e. $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$; while a "labelled data set" should be a finite sequence in this space.

Rebuttal: We appreciate the reviewer's attention to detail. We have fixed this in Theorem 4.3 (earlier 4.4), $\mathcal{Z}$ is the space and D is the probability distribution defined over it. [lines: 250-252;4.3]

ReviewComment E.1.21. **Reviewer Comment:** The rest of the theorem statement is really (!) hard to parse. Could this be improved? Similar comment for Theorem 5.1.

Rebuttal: We have tried to simplify Theorems 4.3 (earlier 4.4), Corollary 4.4 and Theorem 5.1. We have moved the assumptions out of the statements and broken them in points. We hope this makes them more readable. [from line: 259; 4.3],[from line: 280;4.4],[from line: 312; 5.1]

ReviewComment E.1.22. **Reviewer Comment:** Note that Theorem 5.1 neglects many details of the neural network architecture. Does this mean that the theorem holds for any choice of architecture (e.g. any depth and and widths in the hidden layers) as long as the output layer is 1-dimensional?

Rebuttal: We currently show this for fully connected networks. In theorem 5.1 we need a bound on $E_r[L_g L_S]$ and K_S, in Proposition 5.3 we bound these terms for a fully connected network. We don't bound these quantities for other types of architectures like ResNet, GCN etc it's an open question to bound these quantities for these networks. We now have added the fully connected criteria in Theorem 5.1. [line: 306;5.1]

ReviewComment E.1.23. **Reviewer Comment:** Figure 2: The obvious question coming to mind is what is being plotted for "generalizaion error" here. If we take the definition stated earlier on in the paper that "generalization error" stands for the gap between risk and empirical risk, then I would ask how the authors obtained the values plotted here. Perhaps the plotted quantity is a proxy for the generalization error. I any case, this needs explanation.

Rebuttal: Here a validation set is used as a proxy for the population error (risk). This kind of approach to empirically studying what we call the generalization error has been taken before in the literature, an example being in the work of Kuzborskij & Lampert (2018). Our precise methodology is explained in Section 5.1.1 in the paragraph entitled Experiment 2. [line: 425;5.1.1]

ReviewComment E.1.24. **Reviewer Comment:** Another observation regarding Figure 2: The bound values appear to be loose. Definitely they are not nearly the best bound values for neural network classifiers reported in the literature. This raises the question about what is the take-home message that readers could get from reading this paper. If not tightness of the bound values, then it must be something else, but currently unclear.

Rebuttal: Undoubtedly the bounds are loose. We have mentioned at the bottom of Sec 5.1.1 paragraph “Experiment 2”[line: 430;5.1.1] that the use of an upper bound for L_S is a possible reason. Our goal in this paper was to build a framework and to establish that the study of gradient sizes can yield insights into generalization behaviour. This is the take-home message. We hope that more finegrained analysis on these lines will yield such bounds. The graphs in Figure 2 should be seen as a proof of concept rather than the best possible result obtainable from our framework.

ReviewComment E.1.25. **Reviewer Comment:** Bottom of page 11, bullet about NTK: I could not make sense of what’s written here. Please elaborate and clarify.

Rebuttal: NTK is highly technical and elaborating it would send us into a rabbithole of notation and lemmas that are not really relevant here. So, we have removed the bullet.

ReviewComment E.1.26. **Reviewer Comment:** I think "data-dependent" needs hyphenation (throughout the paper).

Rebuttal: Done throughout the paper.

E.2 Reviewer 2, d5U1

E.2.1 Weaknesses pointed out by the reviewer

ReviewComment E.2.1. **Reviewer Comment:** A notion of stability holding with any probability is introduced, only for its particular case of holding with probability one to be used later. In this case, introducing the almost sure version directly makes the paper easier to follow.

Rebuttal: Following the reviewer’s suggestion we have restricted the Definition 3.1 to a.s. Support stability. [line: 160][3.1]

ReviewComment E.2.2. **Reviewer Comment:** The terms L_l Lipschitzness and K_l smoothness allude to constants L_l and K_l , yet these constants might depend on z. Isn’t it more convenient to consider L_l and K_l as functions rather than constants in this case ?

Rebuttal: We have made a minor mistake, L_l and K_l are not only function of z rather they are a function of Ω from which z is picked. So we added it in the name of the property i.e. L_l Lipschitzness property w.r.t Ω in Definition 4.1 [line: 220] and everywhere else including assumption S1 [line: 251-252; 4.1].

ReviewComment E.2.3. **Reviewer Comment:** Thm 4.4: Error in a claim: from the proof of the theorem, we have $U(\alpha_0, K_S, \rho(\tau, m)) = 1 + \frac{1-\exp(-\alpha_0 K_S m^{\rho(\tau,m)}/\rho(\tau,m))}{\alpha_0 K_S}$, in which the second term converges to $\frac{m^{\rho(\tau,m)}}{\rho(\tau,m)}$ rather than to 0. Also, due to the concavity of the numerator of the second term, one can bound U by $1 + \frac{m^{\rho(\tau,m)}}{\rho(\tau,m)}$ which is more informative than the bound in the paper, as it corresponds also to the limit for very small α_0.

Rebuttal: Note that because of $m^{\rho(\tau,m)}$ the expression inside $\exp(\cdot)$ is unbounded and could $\to -\infty$. Therefore we bound the numerator by 1. In the case when $\alpha_0 \to 0$ we agree and we have fixed this in Theorem 4.3 (earlier 4.4)[line: 262; 4.3] Also please note in order to simplify the Theorem 4.3 (earlier 4.4) and 5.1 we have moved out the assumptions and stated them in bullet points before the respective theorems.

ReviewComment E.2.4. **Reviewer Comment:** Thm 4.4: Unsubstantiated claim: Just after the statement of Theorem 4.4, in the "Data dependence with Training Lipschitz constant and Test Lipschitz constant", there is the statement "we expect that if the unknown distribution D has a low variance then the Lipschitz constants will be small". Although this is a conjecture, some justification of motivation for it should be provided, experimentally at least (for example, by plotting the data-dependent Lipschitz constant bounds against the variance of a data set.

Rebuttal: This line was actually an artifact from an earlier draft that should have been removed. We have removed it now. To see the context in which this remark had earlier been made, please see the Discussion paragraph at the end of Sec 5.2.[lines: 458-464; 5.2]

ReviewComment E.2.5. **Reviewer Comment:** Unsubstiantiated claim and unclarity on the behaviour of β after Corollary 4.5: it is mentioned that $\beta = o(\frac{1}{m})$ with the conditions of Theorem 4.4 and Corollary 4.5. However, applying that to Theorem 3.2, one would expect a behaviour that is $\sqrt{\frac{\log(m)}{m}}$ as the second term in the generalization bounds dominates in this case. However, the bound given in the corollary still keeps a term growing in $\frac{1}{m^\epsilon}$. Also, I could not find the justification for this asymptotic behaviour of β in the appendix.

Rebuttal: We have corrected the $\beta = o(\frac{1}{m^\epsilon})$ and provided the exact value of ϵ to make the Corollary 4.4 (earlier 4.5) more clear.[lines: 284-285 ;4.4]

E.2.2 Requested changes, Other suggestions

ReviewComment E.2.6. **Reviewer Comment:** Constructing a simple theoretical example in which data-dependent Lipschitz and smoothness constants are small or moderately large, but absolute ones are too large or even infinite.

Rebuttal: We have added a Discussion at the end of section 5.1 just before the experiments where we discuss in detail a scenario in which data-dependent Lipschitz constants are significantly smaller than the absolute Lipschitz constants.[from line: 376;5.1]

ReviewComment E.2.7. **Reviewer Comment:** Adding a discussion on the interaction between the two terms in the bound in Corollary 4.5. Indeed, the first term divided by the second yields $\frac{(\log m^{3/2}}{m^{\epsilon-1/2}}$, which means that one gets the "usual" rate of $\tilde{O}(\frac{1}{\sqrt{m}}$ whenever $\epsilon \geq 1/2$ (where the tilde hides the logarithmic factors), and will be slower if $\epsilon < 1/2$. It would be interesting to analyze this behavior and to give an intuition of the slow rate, and when would it be beneficial to slow it for example. The overall rate can also be written as $\tilde{O}(m^{-\min(\epsilon,1/2)})$.

Rebuttal: We have added a discussion inside the Proof of Corollary 4.4.[lines: 286-289 ;4.2]

ReviewComment E.2.8. **Reviewer Comment:** In the proof of Proposition 5.3, the product of norms is used to bound the norm of products. I think it would have been fine to let the norm of the product of matrices without further bounding it. Indeed, besides the fact that it already only incorporates quantities that appear in the bound stated by the proposition, it would result in a much tighter bound.

Rebuttal: We agree that this will give a better bound, But we avoided directly writing this because then the final term becomes very hard to interpret. This is because in the proof of Theorem 5.3 in equation 8 (Appendix), for l^{th} layer we can at best break the product of matrices in just two parts i.e. $\|W^H \cdots W^{l+2}\| \cdot \|W^{l-2} \cdots W^1\|$, but now when we square and sum these partial derivatives across all layers this becomes very hard to interpret. So we keep this product of norm terms in proposition 5.3 but we add a statement [lines: 363-365; 5.1] that it's possible to get a better bound by not taking the bound on product of matrices however for the sake of clarity we use this bound. If the reviewer still thinks we should put norm of product variant we can change this.

ReviewComment E.2.9. **Reviewer Comment:** The McDiarmid inequality for functions satisfying the bounded differences with high probability can be related to the result of [1].

Rebuttal: We thank the reviewer for pointing us to the work of Combes. He has actually proved the same result before we did. We have now cited his work as the source of this result and removed the proof from the Appendix to ensure that readers do not think we are claiming to be the first to prove this result. [line: 187 and 587 ;A]

ReviewComment E.2.10. **Reviewer Comment:** In the discussion just after Theorem 5.1, it is written "then these values will be high and reflect a bad generalization", where "these values" refers to L_S and K_S. While the right-hand side of the generalization inequality of the theorem vanishes as L_S approaches 0 (since $F(\tau)$ is proportional to L_S), I do not think that seeing it holds for K_S is straightforward. Indeed,on the one hand, when K_S tends to 0, $m^{1-\frac{\alpha_0 K_S}{\rho(\tau,m)}}$ tends to m . On the other one, we have $U(\alpha_0, K_S, \rho(\tau, m) = 1 + \frac{1-\exp(-\alpha_0 K_S m^{\rho(\tau,m)}/\rho(\tau,m))}{\alpha_0 K_S}$ converges to its upper bound $1+\frac{m^{\rho(\tau,m)}}{\rho(\tau,m)}$ as K_S vanishes. However, since $\rho(\tau, m) = \frac{\log\log m}{\log m + \log\tau}$, we have $m^{\rho(\tau,m)} = \exp(\frac{\log m \log\log m}{\log m+\log\tau})$ behaves as $\log m$ as m grows, hence the bound on U behaves as $\log m$. In the end, we would have a generalization bound that behaves as $O(\frac{(\log m)^3}{m} + \sqrt{\frac{\log m}{m}}) = \tilde{O}(\sqrt{\frac{1}{m}})$. Hence, I think a more detailed explanation should be given.

Rebuttal: We thank the reviewer for this insight. We are adding a discussion around this near the end of paragraph after Theorem 5.1. We would like the reviewer to note that there is a slight change in the line identified. Another reviewer suggested a change so we have slightly modified the statement. [lines: 336-341;5]

E.2.3 To facilitate reading

ReviewComment E.2.11. **Reviewer Comment:** Unifying Lipschitzness and smoothness: Definitions 4.1 and 4.2 can be unified under the same definition of Lipschitzness of a vector-valued function. Then, local parameter-Lipschitzness and local parameter-smoothness can be stated as simple specializations of that definition to the function itself and to its gradient, respectively. The same holds for Lemma 4.3

Rebuttal: We have merged Definitions 4.1 and 4.2 and stated the two local properties as specializations of the definition. We have also updated the Lemma 4.3(now Lemma 4.2) accordingly. [lines: 290;4.2]

ReviewComment E.2.12. **Reviewer Comment:** At the beginning of Section 3.2, the sentence "Given the set S , we construct S_i via replacing the $i-$ th element of S by an independently generated element from D " would facilitate conveying the idea, along with the given formula

Rebuttal: We have now added this in first para of Section 3.2. [lines: 155-157;3.2]

ReviewComment E.2.13. **Reviewer Comment:** In the discussion just after the statement of Lemma 4.3, the expressions "set of weights encountered during training over all possible permutations" and "set of final parameter vectors produced by SGD for each of the possible permutations" would be much clearer if a formal definition is provided for the set A in each case. This definition can for example be only in the appendix, but I think it clarifies these quantities.

Rebuttal: We have written a formal definition of A in the Appendix (Definition B.1, line 597) and pointed towards it in the main paper at the end of section 4.2 after Discussion of "Global" properties. [4.1]

ReviewComment E.2.14. **Reviewer Comment:** Base polynomials, Adding interpretations:
Full NN: for example, "output of a network with the identity activation function, i.e. fully linear".
Specific neuron: obtained by setting... In this case, a figure illustrating the concerned neuron and the operation applied to obtain the polynomial can significantly quicken understanding.
Provide references for the base polynomial (if any)

Rebuttal: We have clarified the full NN point in Appendix section C.1.2 in second paragraph. We have added figures for ease of understanding [4, 5]. We would also like to point out that we have not taken this construction form anywhere, it is an original contribution of this paper. [line:647]

ReviewComment E.2.15. **Reviewer Comment:** Adding a notation table in the supplementary material

Rebuttal: We have added the 2 notation table on at start of appendix 2 and another for appendix section D for it. 3.

ReviewComment E.2.16. **Reviewer Comment:** Using the same letter to index layers, and neurons. I agree this is nitpicky, but I think it would facilitate reading more. For example, the index l for layers (including l_i where i is any index).

Rebuttal: We might have missed the point here but as we understand we are defining such neurons as l-th layer's i-th neuron, like on the first line [line: 715] in paragraph just above Observation D.1, we write i^{th} layer j^{th} neuron. Please let us know if we misinterpreted something.

ReviewComment E.2.17. **Reviewer Comment:** Specifying the exact source of inspiration for the proofs (precisely which results in the mentioned reference.)

Rebuttal: We would like to point out that we have tried to include all the source of inspirations and references which we used.

- In the paragraph just below Definition 3.1 on [line: 161] “this notion weakens the notion of uniform stability introduced by Bousquet & Elisseeff (2002)”. 3.2

- In proof outline of Theorem 3.2 on [lines: 186-188] “Our proof extends the proof of Feldman and Vondrak (Feldman & Vondrak (2019a)) to accommodate the generalization of McDiarmid’s Lemma A.2 from Combes (2015)”. 3.3
- In proof outline of Theorem 4.3 on [lines: 265-266] “The proof follows the lines of the argument presented by Hardt et al. (2016) with the difference that we allow for a probabilistic relaxation...”.4.2
- In proof outline of Theorem 5.2 on [lines: 351-352] “This proof is an adaption of an idea of Milne (2019) and can be found in Appendix C”. 5.1

If the reviewer feels we missed something please let use know we will definitely add them.

ReviewComment E.2.18. **Reviewer Comment:** In the second point of the list of contributions, a number of epochs of $c\log(m)$ is mentioned, but there is no explanation on the nature of constant c (e.g. a universal constant, a constant depending on some parameters ...). Alternatively, if it is just the $\log(m)$ growth rate that is to be highlighted, then for instance, writing "for a number of epochs proportional to $\log(m)$ " solves the issue

Rebuttal: The reviewer is right: we only needed to highlight the $\log(m)$ term. We have updated this in the 2nd bullet point of the last paragraph in the Intro. [line: 80; 1]

ReviewComment E.2.19. **Reviewer Comment:** In Theorem 4.4 $F(\tau)$’s expression can be directly incorporated in the bound, i.e. without introducing $F(\tau)$.

Rebuttal: We have directly written the value of $F(\tau)$ in Theorem 4.3 (earlier Theorem 4.4) and Theorem 5.1. Please also note in order to simplify the Theorem 4.3 (earlier 4.4) and 5.1 we have moved out the assumptions part and stated it in points just before the respective theorems.

ReviewComment E.2.20. **Reviewer Comment:** Avoiding long sentences, or adding commas at least: Corollary 4.5: The first sentence is very long, and needs a comma after....

Rebuttal: We have incorporated the requested changes in Corollary 4.5. [4.4]

ReviewComment E.2.21. **Reviewer Comment:** In proposition 5.3, it is written "Note that this equation holds for both Training Lipschitz ...". While It is understandable that the meant equation is the one providing a bound on L_g , attributing a number to it would be clearer

Rebuttal: Done, we have added this in proposition 5.3 in the last line of the proof. [line: 361;5.3]

ReviewComment E.2.22. **Reviewer Comment:** In the Definition $A.1$ in the appendix, the bounded differences property is called $\beta-$ Lipschitzness. However, Lipschitzness is a well-known property that is totally different from the bounded differences. This can be confusing for readers and I recommend calling it the bounded differences property.

Rebuttal: We have updated it’s name in Definition A.1.

E.2.4 Smaller changes

ReviewComment E.2.23. **Reviewer Comment:** Notation and terms issues: No definition of the term "symmetric in distribution" was given before its first use in Theorem 3.2.

Rebuttal: We have added the definition of "symmetric in distribution" in the first paragraph of section 3.3.[from line: 176]

ReviewComment E.2.24. **Reviewer Comment:** It should be indicated whether constant c in Theorem 3.2 is universal, and if not, what quantities it depends on.

Rebuttal: This constant comes from the proof of Feldman and Vondrak (2019b). Since our analysis is asymptotic we only need this to be independent of m which it is. We have added this clarification just below Theorem 3.2. [line: 185, 3.3]

ReviewComment E.2.25. **Reviewer Comment:** n Definitions 4.1 and 4.2, z is used in the Lipschitzness condition for the first time, without being introduced. Only later it is specified that the constant L_l or K_l depends on z. This can be fixed by introducing z at the moment when $\boldsymbol{w}$ is introduced.

Rebuttal: We have incorporated this suggestion in Def 4.1. [4.1]

ReviewComment E.2.26. **Reviewer Comment:** The logarithm is denoted "log " in all of the paper except for the expression of ρ. Homogeneity of notation is better.

Rebuttal: We have made the notation homogenous through the paper as suggested.

ReviewComment E.2.27. **Reviewer Comment:** For points on Formatting, Grammar and Typos...

Rebuttal: We thank the reviewer for reading the paper closely. We have incorporated all the changes of this kind that were requested.

E.3 Reviewer 3, 2QBs

E.3.1 Weakness

ReviewComment E.3.1. **Reviewer Comment:** However, I am unconvinced regarding the applicability of the obtained results. Compared to similar data-dependent bounds such as the ones in (Hardt et al. 2016) or (Kuzborskij and Lampert 2018), the constants L_S, K_S and L_g have a very complicated dependence in the data distribution and the specifics of the SGD dynamics (which can be very hard to analyze). Further, it seems to me like those bounds can increase arbitrarily in m (as in the random label example), and the conditions under which L_S, K_S or L_g are bounded seem very hard to verify compared to previous work.

Rebuttal: Although it appears that the dependence of the constants on the data distribution is complicated, there are useful and commonly encountered cases where this not so.On the reviewer's suggestion we have included a simple but widely applicable example [line: 376; 5.1] where we have shown that for a low-dimensional data set in a high-dimensional space, using constants such as ours highlights the fact that it is the dimension of the data rather than the dimension of the space that controls the generalization properties. This satisfies the intuition of ML practitioners.

We would like the reviewer to consider that the fact that the constants grow with m is actually a good thing in this case since it is clear that no ML model can generalize on ths example.

We understand that applicability is subjective and we appreciate the reviewer's concern, but we would like the reviewer to consider that our framework is a powerful one that can be used with good effect on specific examples like the one we have now included.

ReviewComment E.3.2. **Reviewer Comment:** Further, all of section 5.3 consists in unsubstantiated claims about extensions of the results to other settings, without much theoretical footing; either those are easy adaptations and the theorem statements could be adapted to include those changes, or there are significant challenges that should be underlined.

Rebuttal: The reviewer is right. Our language was casual and could easily be misinterpreted to imply that the possibilities we suggest are trivial to establish as results. This was not our intent. We have moderated the language significantly now and hope that we have been able to convey that a good amount of research is required on each of those directions. [from line: 466; 5.3],[from line: 472;5.3], [from line: 477; 5.3]

E.3.2 Requested changes

ReviewComment E.3.3. **Reviewer Comment:** In my opinion, the paper requires a much larger discussion on the scaling of the three constants L_S, K_S or L_g; the ReLU network example still consists in a norm supremum over all possible trajectories of SGD, and thus is still very delicate to bound. Even a toy model for which those constants can be easily bounded could already be a good addition to the paper.

Rebuttal: We thank the reviewer for this suggestion as it encouraged us to add an example. We have added it in a discussion just before section 5.1.1. For a more detailed response to this point we refer the reviewer back to our response to comment E.3.1.[line: 376; 5.1]

ReviewComment E.3.4. **Reviewer Comment:** At the very least, a discussion on what those constants imply on the training procedure is in order. For example, the authors state that "if the model can't fit the training set properly then these values will be high and reflect a bad generalization"; why would a high training loss imply high Lipschitz constants ?

Rebuttal: We thank the reviewer for this comment because it brought back a line of analysis we had earlier followed and then abandoned for simplicity of presentation. Appendix C presents an alternate line of analysis that shows that the *expected value* of the Lipschitz constants encountered during the training is the relevant value and not the worst case value encapsulated in L_S. We have added a detailed discussion on this below the statement of Theorem 5.1 [from line: 323]. Hopefully this will shed better light on the relationship between the training and these quantities as requested by the reviewer.

ReviewComment E.3.5. **Reviewer Comment:** Regarding section 5.1.1, under which conditions can we expect that L_S and L_g are close ?

Rebuttal: We have addressed this question in terms of our current and alternate analysis in the paragraphs below the statement of Theorem 5.1. In case the training converges early (as per the alternate analysis of Appendix C) or if the initial choice of weights is close to the final choice of parameters (in either the original or the alternate analysis) these two could be close.

ReviewComment E.3.6. **Reviewer Comment:** Minor remarks: I found the theorem statements quite heavy to read; a lot of the preambles are very similar, and should be broken down into separate assumptions. Less inline math should also be used. the notion of "symmetric in distribution" is never defined

Rebuttal: We thank the reviewer for these suggestions. To make the theorem statements easier to read we have moved the assumptions out for Theorem 4.3 (earlier Theorem 4.4) and Theorem 5.1. We have also made some changes based on other reviews, we hope this makes the paper a better read. If the reviewer finds some more improvements which we can make then please let us know.

We have added the definition of "Symmetric in distribution" on the first paragraph of section 3.3.

F Post-Rebuttal question

F.1 Reviewer 2, d5U1, dated: 14 Nov 2023, 18:37

Reviewer Comment: Typos in the newly added text

Rebuttal: We have fixed them now. Thank you.

F.1.1 Other remarks

Reviewer Comment: Line 39: It is mentioned that the results hold "for both classification and regression cases." However, assumption N2 requires that the output space is countably infinite, which does not agree with the regression setting. Can it be relaxed to incorporate the uncountable setting ? Otherwise, can the scope be restricted to classification ?

Rebuttal: The reviewer is right. We have now clarified on Line 39 that the scope is restricted to classification.

Reviewer Comment: Line 162: The expression "with a certain probability" is more convient to the previous version that considers an arbitrary probability $1-\eta$ in my opinion. I would replace "with a certain probability" to "almost everywhere" or more precisely D^{2m}- or $D^{\otimes 2m}$- almost everywhere.., as in the review of Reviewer FJNX.

Rebuttal: We have fixed this on Line 165 and also at the end of Definition 3.1.

Reviewer Comment: Line 169: $z \in D$ does not hold as D is not a set. Either $z \in \mathcal{Z}$ or $z \sim D$.

Rebuttal: We have fixed this now [Lines 171,173].

Reviewer Comment: Line 326: The η almost β bounded difference is mentioned. However, it is only defined in the appendix, so a remark (even between parentheses) referring to the appendix is necessary.

Rebuttal: This was a mistake and we have corrected it. We wanted to write a.s. support stability [Line 331].

Reviewer Comment: Line 400: It is mentioned that "we need bound on just the square of the expectation of each term." By the Cauchy-Schwarz inequality, we have $E[L_S.L_g] \leq \sqrt{E[L_S^2]E[L_g^2]}$. Hence, I think it is rather "the expectation of the square (or the second moment) of each term".

Rebuttal: The reviewer is correct. We have fixed this now [Line 404].

Reviewer Comment: In line 710, I appreciate adding the "NN with identity function" precision to facilitate the interpreation of the base polynomial. However, as I read it "a fully connected NN defined in ..., i.e. NN with the identity function", this implies that the neural network defined in section D.1.1. has an identity activation function which is not the case. Also, the formulation makes it look as if the base polynomial is defined for an NN with an identity activation function, whereas it can be defined as the output for any fully connected NN if we assume it has the identity activation, while its weights are kept.

Rebuttal: The reviewer is right. As suggested by the reviewer, we have now made a distinction between the NN as defined and a *different version* of the NN where the ReLU activation at each node is replaced by the identity activation. The base polynomial is then defined over this version.

Reviewer Comment: In my previous review, in the comment about the sources for inspiration in the proofs, I was meaning for example "[Reference R, Theorem T]" instead of just "[Refernce R]". It is not necessary but its role is to locate the result quickly in the mentioned references. I apologize for any lack of clarity.

Rebuttal: We had not understood the request earlier. It makes perfect sense and we have now given the specific theorem references throughout the paper.

F.1.2 Questions

Reviewer Comment: In line 251, in assumption S1, it is written: "L_l -LPL w.r.t. supp$D$ and "K_l -LPS w.r.t. D ". Is there a difference between saying "w.r.t. D " or supp$D$?

Rebuttal: This is a typing error, thank you for pointing it out. K_l will only be w.r.t S as it only depends on training set. We have fixed this now [Line 256].

Reviewer Comment: Is it possible to develop more on the advantages on the product of norms in Proposition 5.3, in terms of interpretability ?

Rebuttal: Several works in the literature have commented on the connection between spectral norms like the ones we have presented in Prop 5.3 (e.g. Bartlett et al. (2017), Lin et al. (2021)) and some works also use spectral regularizers to improve generalization Yoshida & Miyato (2017). However, there is no clear answer in the literature for how we can interpret such norms. The general feeling is that a low product of norms implies a distribution that is “easier” to learn and a high product of norms implies a distribution that is “harder” to learn. That may be one way of interpreting the message of works such as Rahaman et al. (2019). However, it is very hard to make such a statement mathematically rigorous. But this question of the reviewer is very thought provoking and could lead to interesting lines of research.

F.2 Reviewer 3, 2QBs, dated: 13 Nov 2023, 22:16

Reviewer Comment: I have read both the rebuttal and the new version of the paper. I welcome the additional clarity changes; however I am still unconvinced by the added example to illustrate the applicability of the theorems. Indeed, the example shows that the test Lipschitz constant L_g can be better bounded when W_1 ranges across the space of bounded vectors. However, as the training goes, I would expect W_1 to align with the structure of x and hence to be fully concentrated in its first D coordinates as well. This would imply that the upper bounds over Ω and supp(D) actually coincide...
This discrepancy is a good example of the (in my opinion) flaw in the results : the need to bound the Lipschitz constants over all possible training trajectories makes it extremely difficult to provide examples where there is a significant improvement over the previous results.

Rebuttal: The reviewer's observation that as the training proceeds W_1 aligns with the subspace of the training points is correct. And, in fact, this is the crux of our work. The reviewer appears to claim that previous works can easily adopt this idea into their own frameworks but, to the best of our understanding, it is not trivial to do so and requires the machinery we have carefully developed for this purpose.

Having said that, we appreciate the spirit of the reviewer's comment. This comment pushed us to further develop the material in Appendix C and we now believe that we have demonstrated the kind of concrete applicability that the reviewer has been concerned about from the beginning. In the first rebuttal we had added an alternate analysis that gives a generalization bound based on the square root of the second moment of the Lipschitz constants encountered during training: $\sqrt{\mathrm{E}_r\left[L^2_{S,I_0}\right]}$. In the current revision of the paper we show in Appendix C.2 how a theorem of (Bottou et. al., 2018) can be used to bound this quantity in a variety of settings. This result is presented as Corollary C.4. In Appendix C.3 we consider a specific two-class classification problem learned by a two-layer NN and show a generalization bound for this case in terms of the parameters of the problem (Theorem C.5).

We hope that the reviewer will agree that the framework of Appendix C.2 is quite useful, as illustrated by the concrete example in Sec C.3. We feel that this is indeed the missing piece of the puzzle and we are grateful to the reviewer for pushing us to develop it. The ease with which our alternate analysis connected with the theory developed by (Bottou et. al. 2018) was exciting for us as we felt that it shows that we have now reached a way of looking at the problem that has appropriate resonances in the works of others.

Reviewer Comment: New minor remarks:

Rebuttal: We have fixed the minor remarks. Thank you for the careful look over our paper.