Data-dependent Sample Complexity of Deep Neural Networks via Lipschitz Augmentation

Colin Wei∗ and Tengyu Ma†

May 10, 2019

Abstract

Existing Rademacher complexity bounds for neural networks rely only on norm control of the weight matrices and depend exponentially on depth via a product of the matrix norms. Lower bounds show that this exponential dependence on depth is unavoidable when no additional properties of the training data are considered. We suspect that this conundrum comes from the fact that these bounds depend on the training data only through the margin. In practice, many data-dependent techniques such as Batchnorm improve the generalization performance. We obtain tighter Rademacher complexity bounds by considering additional data-dependent properties of the network: the sizes of the hidden layers of the network, and the norms of the Jacobians of each layer with respect to the previous layers. Our bounds scale polynomially in depth when these empirical quantities are small, as is usually the case in practice. To obtain these bounds, we develop general tools for making a composition of functions Lipschitz by augmentation and then covering this augmented function. Inspired by our theory, we directly regularize the network’s Jacobians during training and empirically demonstrate that this improves test performance.

1 Introduction

Deep networks trained in practice typically use many more parameters than training examples, and therefore have the capacity to heavily overfit to the training set (Zhang et al., 2016). Fortunately, there are also many known (and unknown) sources of regularization during training: model capacity regularization such as simple weight decay, implicit or algorithmic regularization (Gunasekar et al., 2017, 2018b; Soudry et al., 2018; Li et al., 2018), and finally regularization that depends on the training data such as Batchnorm (Ioffe & Szegedy, 2015), layer normalization (Ba et al., 2016), group normalization (Wu & He, 2018), dropout (Srivastava et al., 2014; Wager et al., 2013), and regularizing the variance of activations (Littwin & Wolf, 2018).

In many cases, it remains unclear why data-dependent regularization can improve the final test error — for example, why Batchnorm empirically improves the generalization performance in practice (Ioffe & Szegedy, 2015; Zhang et al., 2019). Tools for analyzing data-dependent regularization are largely missing in the literature; existing bounds typically consider properties of the weights of the learned model but very little about their interactions with the training set. Formally, define a data-dependent property as any function of the learned model and the training data. We hope that a better understanding of how data-dependent properties relate to generalization can enable us to develop better regularization techniques in the future. In this work, we prove tighter bounds on generalization by considering additional data-dependent properties of the network.

The margin of the neural network is a very important data-dependent property for generalization; Bartlett et al. (2017) show that networks with larger normalized margins have better generalization guarantees. However, neural networks are extremely complex, so there remain many other data-dependent properties which could potentially lead to better generalization. We extend the bounds and techniques of Bartlett et al. (2017) by considering additional data-dependent properties: the hidden layer norms and Jacobian norms. Let \( F \) denote a neural network with smooth activation \( \phi \) parameterized by weight matrices \( \{ W^{(i)} \}_{i=1}^r \) that perfectly classifies the training data with margin \( \gamma > 0 \).

∗Stanford University, email: colinwei@stanford.edu
†Stanford University, email: tengyuma@stanford.edu
Our bounds depend on the model’s empirical hidden layer norm and Lipschitz constants. The dependency in our main Theorem 7.1 is a polynomial in these quantities; the bound below is a simplification for expositional purposes. Let \( t \) denote the maximum \( \ell_2 \) norm of any hidden layer or training datapoint, and \( \sigma \) the maximum operator norm of any interlayer Jacobian, where both quantities are evaluated only on the training data.

**Theorem 1.1** (Simplified version of Theorem 7.1). Suppose \( \sigma, t \geq 1 \). With probability \( 1 - \delta \) over the training data, we can bound the test error of \( F \) by

\[
L_{0,1}(F) \leq \tilde{O} \left( \frac{r^3 \sigma^2 \sigma}{\tilde{\kappa}^2} + \frac{\sum_i \| W(i) \|_2^{2/3}}{\sqrt{n}} + \frac{r^2 \sigma}{n} \right)
\]

The notation \( \tilde{O} \) hides log factors.

The degree of the dependencies on \( \sigma \) may look unconventional — this is mostly due to the dramatic simplification from our full Theorem 7.1 by using uniform upper bounds for the data-dependent norms. Our full Theorem 7.1 obtains a more natural bound that considers all interlayer Jacobian norms instead of only the maximum. As \( t \) and \( \sigma \) are typically small in practice (Arora et al., 2018; Nagarajan & Kolter, 2019), this simplified bound scales polynomially in the depth of the network.

In contrast, the bounds of Neyshabur et al. (2015); Bartlett et al. (2017); Neyshabur et al. (2017a); Golowich et al. (2017) all depend on a product of norms of weight matrices which scales exponentially in the network depth, and which can be thought of as a worst case Lipschitz constant of the network. In fact, lower bounds show that with only norm-based constraints on the hypothesis class, this product of norms is unavoidable for Rademacher complexity-based approaches (see for example Theorem 3.4 of Bartlett et al. (2017) and Theorem 7 of Golowich et al. (2017)).

We circumvent these lower bounds by additionally considering the model’s Jacobian norms – empirical Lipschitz constants which are much smaller than the product of norms because they are only computed on the training data.

The bound of Nagarajan & Kolter (2019) also depends polynomially on the Jacobian norms rather than exponentially in depth; however these bounds also require that the inputs to the activation layers are bounded away from 0, an assumption that does not hold in practice Nagarajan & Kolter (2019). The bound of Arora et al. (2018) depends on similar quantities related to noise stability but only holds for a compressed network and not the original.

In Figure 1, we plot the distribution over the sum of products of Jacobian and hidden layer norms for a WideResNet (Zagoruyko & Komodakis, 2016) trained with and without Batchnorm. Figure 1 shows that this sum blows up for networks trained without Batchnorm, indicating that the terms in our bound are empirically relevant for explaining data-dependent regularization.

An immediate issue in proving Theorem 1.1 is that typical tools for showing concentration inequalities require fixing the hypothesis class before looking at training data, whereas conditioning on data-dependent properties makes the hypothesis class a random object depending on the data. A natural attempt is to augment the loss with indicators on the intended data-dependent quantities \( \{ \gamma_i \} \), with desired bounds \( \{ \kappa_i \} \): we consider the augmented loss

\[
l_{\text{aug}} = (l_{\text{old}} - 1) \prod_{\text{properties } \gamma_i} \mathbb{1}(\gamma_i \leq \kappa_i) + 1
\]

which upper bounds the original loss \( l_{\text{old}} \in [0, 1] \). When all properties hold empirically for the training data, the training loss remains the same after augmentation. The main challenges in this approach are twofold: 1) designing the correct set of properties and 2) proving generalization of the final loss \( l_{\text{aug}} \), a complicated function of the network.

The main tool that we use is covering numbers: Lemma 4.1 shows that a composition of functions (i.e., a neural network) has low covering number if the output is worst-case Lipschitz at each level of the composition and internal layers are bounded in norm. Unfortunately, the standard neural net loss satisfies neither of these properties, but by augmenting with properties \( \gamma \), we can guarantee they hold. One technical challenge is that augmenting the loss makes it harder to reason about covering, as the indicators can introduce complicated dependencies between layers.

Our main technical contributions are: 1) We demonstrate how to augment compositions of functions to make it Lipschitz, and thus easy to cover, at all layers. Before this augmentation, the Lipschitz constant could scale exponentially in depth (Theorem 6.2). 2) We reduce covering a complicated sequence of operations to covering the individual quantities related to noise stability but only holds for a compressed network and not the original.
functions in these operations (Theorem 5.3). 3) By combining 1 and 2, it follows cleanly that our augmented loss on neural networks has low covering number and therefore has good generalization. Our bound scales polynomially in the depth of the network, not exponentially, when the network has good Lipschitz constants on the training data (Theorem 7.1).

Empirically, we show in Section 8 that directly regularizing our complexity measure can result in improved test performance. Inspired by Theorem 1.1, we add direct regularization of the Jacobian norms to networks with normalization layers in order to control the hidden layer and Jacobian norms simultaneously. We show that this technique improves test accuracy for models trained with low learning rate or without data augmentation or BatchNorm.

**Organization of the paper:** We survey related work in Section 2 and define basic notations in Section 3. In Section 4 we provide an overview of our core technical ideas. In order to formalize these ideas, we adapt the notion of a computational graph to our setting in Section 5. In Section 6, we show how to convert an arbitrary sequential composition graph into a Lipschitz one. In Section 7, we use these tools to derive our main generalization bound for neural nets (Theorem 7.1), and empirically evaluate regularization techniques motivated by this bound in Section 8.

2 Related Work

Zhang et al. (2016) and Neyshabur et al. (2017b) show that generalization in deep learning often disobeys conventional statistical wisdom. One of the approaches adopted towards explaining generalization is implicit regularization; numerous recent works have shown that the training method prefers minimum norm or maximum margin solutions (Soudry et al., 2018; Li et al., 2018; Ji & Telgarsky 2018; Gunasekar et al., 2017, 2018a,b; Wei et al., 2018). With the exception of (Wei et al., 2018), these papers analyze simplified settings and do not apply to larger neural networks.

This paper more closely follows a line of work related to Rademacher complexity bounds for neural networks (Neyshabur et al., 2015, 2018; Bartlett et al., 2017; Golowich et al., 2017). For a comparison, see the introduction. There has also been work on deriving PAC-Bayesian bounds for generalization (Neyshabur et al., 2017b,a; Nagarajan & Kolter, 2019, Dziugaite & Roy (2017) optimize a bound to compute non-vacuous bounds for generalization error. Another line of work analyzes neural nets via their behavior on noisy inputs. Neyshabur et al. (2017b) prove PAC-Bayesian generalization bounds for random networks under assumptions on the network’s empirical noise stability. Arora et al. (2018) develop a notion of noise stability that allows for compression of a network under an appropriate noise distribution. They additionally prove that the compressed network generalizes well. In comparison, our Lipschitzness construction also relates to noise stability, but our bounds hold for the original network and do not rely on the particular noise distribution.

Nagarajan & Kolter (2019) use PAC-Bayes bounds to prove a similar result as ours for generalization of a network with bounded hidden layer and Jacobian norms. The main difference is that their bounds only apply to relu networks and depend on the inverse relu preactivations, which are found to be large in practice (Nagarajan & Kolter, 2019); our bounds apply to smooth activations and avoid this dependence at the cost of an additional factor in the Jacobian norm (shown to be empirically small). We note that the choice of smooth activations is empirically justified (Clevert et al., 2015; Klambauer et al., 2017). We also work with Rademacher complexity and covering numbers instead of the PAC-Bayes framework. It is relatively simple to adapt our techniques to relu networks to produce a similar result to that of Nagarajan & Kolter (2019), by conditioning on large pre-activation values in our Lipschitz augmentation step (see Definition 6.1). In Section 2 we provide a sketch of this argument and obtain a bound for relu networks that is polynomial in hidden layer and Jacobian norms and inverse preactivations. However, it is not obvious how to adapt the argument of Nagarajan & Kolter (2019) to activation functions whose derivatives are not piecewise-constant.

There are also other perspectives on generalization: Hardt et al. (2015) show that models which train faster tend to generalize better; Keskar et al. (2016); Hoffer et al. (2017) study the effect of batch size on generalization. Brutzkus et al. (2017) analyze a neural network trained on hinge loss and linearly separable data and show that gradient descent recovers the exact separating hyperplane.

Finally, many recent papers have shown optimization results for neural nets (Allen-Zhu et al., 2018a,b; Li & Liang, 2018; Arora et al., 2019; Mei et al., 2018; Zou et al., 2018; Du et al., 2018a,b; Lee et al., 2019; Jacot et al., 2018; Cao & Gu, 2019). Generalization results also follow from the optimization analyses of Allen-Zhu et al. (2018a,b; Li & Liang,
4 Overview of Main Results and Proof Techniques

In this section, we give a general overview of the main technical results and outline how to prove them with minimal notation. We will point to later sections where many statements are formalized.

To simplify the core mathematical reasoning, we abstract feed-forward neural networks (including residual networks) as compositions of operations. Let $F_1, \ldots, F_k$ be a sequence of families of functions (which corresponds to families of single layer neural nets in the deep learning setting) and $\ell$ be a Lipschitz loss function. We study the family of compositions of $\ell$ and functions in $F_i$’s:

$$L \triangleq \ell \circ F_k \circ F_{k-1} \circ \cdots \circ F_1 = \{ \ell \circ f_k \circ f_{k-1} \circ \cdots \circ f_1 : \forall i, f_i \in F_i \}$$

Figure 1: Let $h_1, h_2, h_3$ denote the 1st, 2nd, and 3rd blocks of a 16-layer WideResNet and $J_i$ the Jacobian of the output w.r.t layer $i$. In log-scale we plot a histogram of the 100 largest values on the training set of $\sum_{i=1}^3 h_i \|J_i\|/\gamma$ for a WideResNet trained with and without Batchnorm on CIFAR10, where $\gamma$ is the example’s margin.
Textbook results \cite{BartlettMendelson2002} bound the generalization error by the Rademacher complexity (formally defined in Section \ref{A} of the family of losses $\mathcal{L}$, which in turn is bounded by the covering number of $\mathcal{L}$ through Dudley’s entropy integral theorem \cite{Dudley1967}. Modulo minor nuances, the key remaining question is to give a tight covering number bound for the family $\mathcal{L}$ for every target cover size $\epsilon$ in a certain range (often, considering $\epsilon \in [1/n^{O(1)}, 1]$ suffices).

As alluded to in the introduction, generalization error bounds obtained through this machinery only depend on the (training) data through the margin in the loss function, and our aim is to utilize more data-dependent properties. Towards understanding which data-dependent properties are useful to regularize, it is helpful to revisit the data-independent covering technique of \cite{BartlettEtal2017}, the skeleton of which is summarized below.

Recall that $\mathcal{N}(\epsilon, \mathcal{F}, s)$ denotes the covering number for arbitrary $n$ data points with norm less than $s$. The following lemma says that if the intermediate variable (or the hidden layer) $f_1 \circ \cdots \circ f_i(x)$ is bounded, and the composition of the rest of the functions $l \circ f_k \circ \cdots \circ f_{i+1}(x)$ is Lipschitz, then small covering number of local functions imply small covering number for the composition of functions.

**Lemma 4.1.** [abstraction of techniques in \cite{BartlettEtal2017}] In the context above, assume:

1. for any $x \in \text{supp}(P_n)$, $\|f_i \circ \cdots \circ f_1(x)\| \leq s_i$.
2. $\ell \circ f_k \circ \cdots \circ f_{i+1}$ is $\kappa_i$-Lipschitz for all $i$.

Then, we have the following covering number bound for $\mathcal{L}$ (for any choice $\epsilon_1, \ldots, \epsilon_k > 0$):

$$\log \mathcal{N}(\sum_{i=1}^{k} \kappa_i \epsilon_i, \mathcal{L}, s_0) \leq \sum_{i=1}^{k} \log \mathcal{N}(\epsilon_i, \mathcal{F}, s_{i-1})$$

The lemma says that the log covering number and the cover size scale linearly if the Lipschitzness parameters and norms remain constant. However, these two quantities, in the worst case, can easily scales exponentially in the number of layers, and they are the main sources of the dependency of product of spectral/Frobenius norms of layers \cite{GolowichEtal2017, BartlettEtal2017, NeyshaburEtal2017, NeyshaburEtal2015}. More precisely, the worst-case Lipschitzness over all possible data points can be exponentially bigger than the average/typical Lipschitzness for examples randomly drawn from the training or test distribution. We aim to bridge this gap by deriving a generalization error bound that only depends on the Lipschitzness and boundedness on the training examples.

Our general approach, partially inspired by margin theory, is to augment the loss function by soft indicators of the Lipschitzness and bounded-ness. Let $h_i$ be shorthand notation for $f_i \circ \cdots \circ f_1$ so that $h_i(x)$ denotes the $i$-th intermediate value, and let $z(x) \equiv \ell(h_k(x))$ be the original loss. Our first attempt considered:

$$\tilde{z}'(x) \equiv 1 + (z(x) - 1) \cdot \prod_{i=1}^{k} \mathbb{1}_{\leq s_i}(|h_i(x)|) \cdot \prod_{i=1}^{k} \mathbb{1}_{\leq \kappa_i}(|\partial z/\partial h_i|_{\text{op}}) \quad (2)$$

The hope was that the indicators would flatten those regions where $h_i$ is not bounded and where $z$ is not Lipschitz in $h_i$. However, there are two immediate issues. First, the soft indicators functions are themselves functions of $h_i$. It’s unclear whether the augmented function can be Lipschitz w.r.t $h_i$, and thus we cannot apply Lemma 4.1. Second, the augmented loss function becomes complicated and doesn’t fall into the sequential computation form of Lemma 4.1 and therefore even if Lipschitzness is not an issue, we need new covering techniques beyond Lemma 4.1.

We address the first issue by recursively augmenting the loss function by multiplying more soft indicators that bound the Jacobian of the current function. The final loss $\tilde{z}$, which upper bounds the original loss $z$, reads:

$$\tilde{z}(x) \equiv 1 + (z(x) - 1) \cdot \prod_{i=1}^{k} \mathbb{1}_{\leq s_i}(|h_i(x)|) \cdot \prod_{1 \leq i \leq j \leq k} \mathbb{1}_{\leq \kappa_{j\rightarrow i}}(|Df_j \circ \cdots \circ f_i[h_{i-1}]|_{\text{op}}) \quad (3)$$

\footnote{A priori, it’s also unclear what “Lipschitz in $h_i$” means since the $z'$ does not only depend on $x$ through $h_i$. We will formalize this in later section after defining proper language about dependencies between variables.}

\footnote{Unlike in equation (3), we don’t augment the Jacobian of the loss w.r.t the layers. This allows us to deal with non-differentiable loss functions such as ramp loss.}
where $\kappa_{j-e-i}$’s are user-defined parameters. For our application to neural nets, we instantiate $s_i$ as the maximum norm of layer $i$ and $\kappa_{j-e-i}$ as the maximum norm of the Jacobian between layer $j$ and $i$ across the training dataset. A polynomial in $\kappa$, $s$ can be shown to bound the worst-case Lipschitzness of the function w.r.t. the intermediate variables in the formula above. By our choice of $\kappa$, $s$, a) the training loss is unaffected by the augmentation and b) the worst-case Lipschitzness of the loss is controlled by a polynomial of the Lipschitzness on the training examples. Our recursive Lipschitz-fication procedure and its guarantees are formally presented in Definition 6.1 and Theorem 6.2.

The downside of the Lipschitz-fication is that it further complicates the loss function. Towards covering the loss function (assuming Lipschitz properties) efficiently, we extend Lemma 4.1 which works for sequential compositions of functions, to general families of formulas, or computational graphs. We will show that under certain Lipschitz conditions, the covering of the function classes representable by computation graphs can be reduced to covering all the local composition rules of the variables. See Theorem 5.3 and generally Section 5.

Combining the Lipschitz-fication and graphs covering results, we obtain a covering number bound of augmented loss. The theorem below is formally stated in Section 6.

**Theorem 4.2.** Let $\mathcal{L}$ be the family of augmented losses defined in (3). For cover resolutions $\epsilon_i$ and values $\hat{\kappa}_i$ that are polynomial in the parameters $s_i, \kappa_{j-e-i}$, we obtain the following covering number bound for $\mathcal{L}$:

$$\log \mathcal{N}(\sum_i \epsilon_i \hat{\kappa}_i, \hat{\mathcal{L}}, s_0) \leq \sum_i \log \mathcal{N}(\epsilon_i, \mathcal{F}_i, s_{i-1}) + \sum_i \log \mathcal{N}(\epsilon_i, D\mathcal{F}_i, s_{i-1})$$

where $D\mathcal{F}_i$ denotes the function class obtained from applying the total derivative operator to all functions in $\mathcal{F}_i$.

Now, following the standard technique of bounding Rademacher complexity via covering numbers, we can obtain generalization error bounds for augmented loss. For the demonstration of our technique, suppose that the following simplification holds: $\log \mathcal{N}(\epsilon_i, D\mathcal{F}_i, s_{i-1}) = \log \mathcal{N}(\epsilon_i, \mathcal{F}_i, s_{i-1}) = s_{i-1}^2/\epsilon_i^2$. Then after minimizing the covering number bound in $\epsilon_i$ via standard techniques, we obtain the below generalization error bound on the original loss for parameters $\hat{\kappa}_i$ alluded to in Theorem 4.2 and formally defined in Theorem 6.3. When the training examples satisfy the augmented indicators, $E_{\text{train}}[z] = E_{\text{train}}[\hat{z}]$, and because $\hat{z}$ bounds $z$ from above, we have

$$E_{\text{test}}[z] - E_{\text{train}}[z] \leq E_{\text{test}}[\hat{z}] - E_{\text{train}}[\hat{z}] \leq O \left( \left( \frac{\sum_i \hat{\kappa}_i^{2/3}}{s_{i-1}^{2/3}} \right)^{3/2} \sqrt{n} \frac{\log(1/\delta)}{n} \right)$$

5 Covering of Computational Graphs

In this section, we adapt the notion of a computational graph to our setting. In Section 5.1 we formalize the notion of a computational graph and demonstrate how neural networks fit under this framework. In Section 5.2, we define the notion of release-Lipschitzness that abstracts the sequential notion of Lipschitzness in Lemma 4.1. We show that when this release-Lipschitzness condition and a boundedness condition on the internal nodes hold, it is possible to cover a family of computational graphs by simply covering the function class at each vertex.

5.1 Formalization of computational graphs

When we augment the neural network loss with data-dependent properties, we introduce dependencies between the various layers, making it complicated to cover the augmented loss. We use the notion of computational graphs to abstractly model these dependencies.

Computational graphs are originally introduced by Bauer (1974) to represent computational processes and study error propagation. Formally, a computational graph $G(\mathcal{V}, \mathcal{E}, \{R_V\})$ is an acyclic directed graph with three components: the set of nodes $\mathcal{V}$ corresponds to variables, the set of edges $\mathcal{E}$ describes dependencies between these variables, and $\{R_V\}$ contains a list of composition rules indexed by the variables $V$’s, representing the process of computing $V$ from its direct predecessors.

\[\text{As mentioned in footnote 4 we will formalize the precise meaning of Lipschitzness later.}\]
More precisely, let \( I_G = \{I_1, \ldots, I_p\} \) be the subset of nodes with no predecessors and we call them the “input nodes” of the graph. For simplicity, we assume the graph contains a unique sink, denoted by \( O_G \), and we call it the “output node”. (It’s straightforward to generalize to scenarios with multiple output nodes.) For every variable \( V \in \mathcal{V} \), let \( \mathcal{D}_V \) be the space that \( V \) resides in. If \( V \) has \( t \) direct predecessors \( C_1, \ldots, C_t \), then the associated composition rule \( R_V \) is a function that maps \( \mathcal{D}_{C_1} \times \cdots \times \mathcal{D}_{C_t} \) to \( \mathcal{D}_V \). If \( V \) is an input node, then the composition rule \( R_V \) is not relevant. For any node \( V \), the computational graph defines/induces a function that computes the variable \( V \) from inputs, or, in mathematical words, that maps the inputs space \( \mathcal{D}_{I_1} \times \cdots \times \mathcal{D}_{I_m} \) to \( \mathcal{D}_V \). This associated function, denoted by \( \mathcal{V} \) again with slight abuse of notations, is defined recursively as follows: set \( \mathcal{V}(x_1, \ldots, x_p) \)

\[
\begin{cases}
    x_i & \text{if } V \text{ is the } i\text{-th input node } I_i \\
    R_V(C_1(x_1, \ldots, x_p), \ldots, C_t(x_1, \ldots, x_p)) & \text{if } V \text{ has } t \text{ direct predecessors } C_1, \ldots, C_t 
\end{cases}
\]

More succinctly, we can write \( V = R_V \circ (C_1 \times \cdots \times C_t) \). We also overload the notation \( \mathcal{O}_G \) to denote the function that the computational graph \( G \) finally computes (which maps \( \mathcal{D}_{I_1} \times \cdots \times \mathcal{D}_{I_m} \) to \( \mathcal{D}_O \)). For any set \( \mathcal{S} = \{V_1, \ldots, V_i\} \subseteq \mathcal{V} \), use \( \mathcal{D}_S \) to denote the space \( \mathcal{D}_{V_1} \times \cdots \times \mathcal{D}_{V_i} \). We use \( \mathcal{P}(G, \mathcal{V}) \) to denote the set of direct predecessors of \( V \) in graph \( G \), or simply \( \mathcal{P}(V) \) when the graph \( G \) is clear from context.

Example 5.1 (Feed-forward neural networks). For an activation function \( \phi \) and parameters \( \{W^{(1)}\} \) we compute a neural net \( F : \mathbb{R}^{d_I} \to \mathbb{R}^{d_O} \) as follows: \( F(x) = W^{(r)}(\phi(\cdots \phi(W^{(1)}x) \cdots )) \). Figure 2 depicts how this neural network fits into a computational graph with one input node, 2r − 1 internal nodes, and a single output. Here we treat matrix operations and activations as distinct layers, and map each layer to a node in the computational graph.

5.2 Reducing graph covering to local function covering

In this section we introduce the notion of a family of computational graphs, generalizing the sequential family of function compositions in (1). We define release-Lipschitzness, a condition which allows reduce covering the entire the graph family to covering the composition rules at each node. We formally state this reduction in Theorem 5.3.

Family of computational graphs: Let \( \mathcal{G} = \{G(\mathcal{V}, \mathcal{E}, \{R_V\}) : \{R_V\} \in \mathcal{R}\} \) be a family of computational graph with shared nodes and edges, where \( \mathcal{R} \) is a collection of lists of composition rules. This family of computational graphs defines a set of functions \( \mathcal{O}_G \) of \( G \in \mathcal{G} \). We’d like to cover this set of functions in \( \mathcal{O}_G \) with respect to some metric \( L(P_n, \|\cdot\|) \).

For a list of composition rules \( \{R_V\} \in \mathcal{R} \) and subset \( \mathcal{S} \subseteq \mathcal{V} \), we define the projection of composition rules onto \( \mathcal{S} \) by \( \{R_V\}_S = \{R_V : V \in \mathcal{S}\} \). Now let \( \mathcal{R}_S = \{\{R_V\}_S : \{R_V\} \in \mathcal{R}\} \) denote the marginal collection of the composition rules on node subset \( \mathcal{S} \).

For any computational graph \( G \) and a non-input node \( V \in \mathcal{V} \setminus \mathcal{I} \), we can define the following operation that “releases” \( V \) from its dependencies on its predecessors by cutting all the inward edges: Let \( GV \) be sub-graph of \( G \) where all the edges pointing towards \( V \) are removed from the graph. Thus, by definition, \( V \) becomes a new input node of the graph \( GV \): \( I_{GV} = \{V\} \cup I_G \). Moreover, we can “recover” the dependency by plugging the right value for \( V \) in the new graph \( GV \): Let \( V(x) \) be the function associated to the node \( V \) in graph \( G \), then we have

\[
\forall x \in \mathcal{D}_I, \quad \mathcal{O}_{GV}(V(x), x) = \mathcal{O}_G(x)
\] (4)

In our proofs, we will release variables in orders. Let \( \mathcal{S} = \{V_1, \ldots, V_m\} \) be an ordering of the intermediate variables \( \mathcal{V} \setminus (\mathcal{I} \cup \{O\}) \). We call \( \mathcal{S} \) a forest ordering if for any \( i \), in the original graph \( G \), \( V_1 \) only at most depends on
the input nodes and \( V_1, \ldots, V_{i-1} \). For any sequence of variables \( (V_1, \ldots, V_i) \), we can define the graph obtained by releasing the variables in order: \( G^{\setminus(V_1, \ldots, V_i)} = (\ldots(G^{\setminus V_1}) \ldots)^{\setminus V_i} \). We next define the release-Lipschitz condition, which states that the graph function remains Lipschitz when we sequentially release vertices in a forest ordering of the graph.

**Definition 5.2** (Release-Lipschitzness). A graph \( G \) is release-Lipschitz with parameters \( \{\kappa_V\} \) w.r.t a forest ordering of the internal nodes, denoted by \( (V_1, \ldots, V_m) \) if the following happens: upon releasing \( V_1, \ldots, V_m \) in order from any \( G \in \mathcal{G} \), for any \( 0 \leq i \leq m \), we have that the function defined by the released graph \( G^{\setminus(V_1, \ldots, V_i)} \) is \( \kappa_V \)-Lipschitz in the argument \( V_i \), for any values of the rest of the input nodes (\( \approx \{V_1, \ldots, V_{i-1}\} \cup \mathcal{I}_G \)). We also say graph \( G \) is release-Lipschitz if such a forest ordering exists.

Now we show that the release-Lipschitz condition allows us to cover any family of computational graphs whose output collapses when internal nodes are too large. For the augmented loss defined in (3), the function output collapses to 1 when internal computations are large. The proof is deferred to Section [B].

**Theorem 5.3.** Suppose \( G \) is a computational graph with the associated family of lists of composition rules \( \mathcal{R} \), as formally defined above. Let \( \mathcal{P}_n \) be a uniform distribution over \( n \) points in \( \mathcal{D}_Z \). Let \( \kappa_V, s_V, \epsilon_V \) be three families of fixed parameters indexed by \( V \setminus \mathcal{I} \) (whose meanings are defined below). Assume the following:

1. Every \( G \in \mathcal{G} \) is sequential-releasing Lipschitz with parameters \( \{\kappa_V\} \) w.r.t a forest ordering of the internal nodes \( (V_1, \ldots, V_m) \) (the parameter \( \kappa_V \)'s and ordering doesn’t depend on the choice of \( G \)).

2. For the same order as before, if \( (v, x) \in (\mathcal{D}_{V_1} \otimes \cdots \otimes \mathcal{D}_{V_i}) \otimes \mathcal{D}_Z \) is an input of the released graph satisfying \( \|v_j\| \geq s_{V_j} \) for some \( j \leq i \), then \( O_{G^{\setminus(V_1, \ldots, V_i)}}(v, x) = c \) for some constant \( c \).

Then, small covering numbers for all of the local composition rules of \( V \) with resolution \( \epsilon_V \) would imply small covering number for the family of computational graphs with resolution \( \sum_V \epsilon_V \kappa_V \):

\[
\log \mathcal{N}(\sum_{V \in V \setminus \mathcal{I}(O)} \kappa_V \epsilon_V + \epsilon_O, O_G, s_{\mathcal{I}}) \leq \sum_{V \in V \setminus \mathcal{I}} \log \mathcal{N}(\epsilon_V, \mathcal{R}_{(V)}, s_{\mathcal{R}(V)})
\]

(5)

### 6 Lipschitz Augmentation of Computational Graphs

The covering number bound for the computational graph family \( \mathcal{G} \) in Theorem 5.3 relies on the release-Lipschitzness condition (condition 1 of Theorem 5.3) and rarely holds for deep computational graphs such as deep neural networks. The conundrum is that the worst-case Lipschitzness as required in the release-Lipschitz condition is very likely to scale in the product of the worst-case Lipschitzness of each operations in the graph, which can easily be exponentially larger than the average Lipschitzness over typical examples.

In this section, we first define a model of sequential computational graphs, which captures the class of neural networks. Before Lipschitz augmentation, the worst-case Lipschitz constant of graphs in this family could scale exponentially in the depth of the graph. In Definition 6.4, we generalize the operation of \( \mathcal{B} \) to augment any family \( \mathcal{G} \) of sequential graphs and produce a family \( \tilde{\mathcal{G}} \) satisfying the release-Lipschitz condition. In Theorem 6.3, we combine this augmentation with the framework of 5.3 to produce general covering number bounds for the augmented graphs. For the rest of this section we will work with sequential families of computational graphs.

A sequential computational graph has nodes set \( V = \{I, V_1, \ldots, V_q, O\} \), where \( I \) is the single input node, and all the edges are \( \mathcal{E} = \{(I, V_1), (V_1, V_2), \ldots, (V_{q-1}, V_q)\} \cup \{(V_1, O), \ldots, (V_q, O)\} \). We often use the notation \( V_0 \) to refer to the input \( I \). See Figure 3 for a visualization.

---

4We say the Lipschitzness required is worst case because the release-Lipschitz condition requires the Lipschitzness of nodes for any possible choice of inputs.
Informal explanation of Lipschitz augmentation: To demonstrate the challenges of Lipschitz augmentation and our solution, we return to the example of Section 4 where we consider composing functions $f_k \circ \ldots \circ f_1$ with $h_i \triangleq f_i \circ \ldots \circ f_1$. Note that this setting can be modeled by a sequential computational graph. Recall the starting point we considered in (2):

$$
\tilde{z}'(x) \triangleq 1 + (z(x) - 1) \cdot \prod_{i=1}^{k} \mathbb{1}_{\leq \kappa_i} (\| h_i(x) \|) \cdot \prod_{i=1}^{k} \mathbb{1}_{\leq \kappa_i} (\| \partial z / \partial h_i \|_{op})
$$

We consider more closely how to control the change in the term $\| \partial z / \partial h_i \|_{op}$: by the chain rule (Claim F.2), we have the expansion $\frac{\partial z}{\partial h_i} = \frac{\partial z}{\partial h_k} \frac{\partial h_k}{\partial h_{k-1}} \ldots \frac{\partial h_{k+1}}{\partial h_i}$. From this equation, it is clear that $\frac{\partial z}{\partial h_i}$ is a complicated function in $x$ which depends on the internal computations $h_j$. It follows that to bound the error in $\frac{\partial z}{\partial h_i}(x)$ under perturbations $\nu$ to $x$, we must control the individual $| \frac{\partial h_i}{\partial h_{j-1}}(x) - \frac{\partial h_i}{\partial h_{j-1}}(x + \nu) |$, which is challenging and creates complicated dependencies between variables.

Our key insight is that we can still control Lipschitzness of the entire system by accounting for the derivative $\frac{\partial z}{\partial h_i}$ as an intended bound of the Jacobian between pairs of layers (variables). Figure 3 depicts the expansion $\frac{\partial h_i}{\partial h_{j-1}}$.

We note that the graph $G$ maps a vector in space $D_{V_i}$ into a product and then applying a telescoping argument to bound the difference in the product by differences in individual terms. Below we formally define the augmentation operation.

Definition 6.1 (Lipschitz augmentation of sequential graphs). Given a differentiable sequential computational graph $G$ with $q$ internal nodes $V_1, \ldots, V_q$, define its Lipschitz augmentation $\tilde{G}$ as follows. We first add $q$ nodes to the graph denoted by $J_1, \ldots, J_q$. The composition rules for original internal nodes remain the same, and the composition rule for $J_i$ is defined as

$$
\tilde{R}_{J_i} = DR_{V_i}
$$

Here $DR_{V_i}$ is the total derivative of the function $R_{V_i}$. In other words, the variable $J_i$, is a Jacobian for $R_{V_i}$, a linear operator that maps $D_{V_{i-1}}$ to $D_{V_i}$. (Note that if $V_i$'s are considered as vector variables, then $J_i$'s are matrix variables.) We equip the space of $J_i$ with operator norm, denoted by $\| \cdot \|_{op}$, induced by the original norms on spaces $V_{i-1}$ and $V_i$. The Lipschitz-ness w.r.t variable $J_i$ will be measured with operator norm.

We pre-determine a family of parameters $\kappa_{j-i}$ for all pairs $(i, j)$ with $i \leq j$. The final loss is augmented by a product of soft indicators that truncates the function when any of Jacobians is much larger than $\kappa_{j-i}$:

$$
\tilde{R}_{O}(x, v_1, \ldots, v_q, D_1, \ldots, D_q) \triangleq (R_{O}(x, v_1, \ldots, v_q) - 1) \prod_{i \leq j} \mathbb{1}_{\leq \kappa_{j-i}} (\| D_j \cdots D_i \|_{op}) + 1
$$

where $x \in \mathcal{D}_x$, $v_i \in \mathcal{D}_{V_i}$, and $D_i \in \mathcal{D}_{J_i}$. Note that $D_j \cdots D_i$ is the total derivative of $V_j$ w.r.t $V_i$, and thus the $\kappa_{j-i}$ has the interpretation as an intended bound of the Jacobian between pairs of layers (variables). Figure 3 depicts the augmentation.

Note that under these definitions, we finally get that the output function of $\tilde{G}$ computes

$$
O_{\tilde{G}}(x) = (O_G(x) - 1) \prod_{i \leq j} \mathbb{1}_{\leq \kappa_{j-i}} (\| D V_j(x) \cdots D V_i(x) \|_{op}) + 1
$$

We note that the graph $\tilde{G}$ contains the original $G$ as a subgraph. Furthermore, by Claim F.1, $O_{\tilde{G}}$ upper bounds $O_G$, which is desirable when $\tilde{G}$ computes loss functions. The below theorem formally proves release-Lipschitzness for $\tilde{G}$.

Theorem 6.2. [Lipschitz guarantees of augmented graphs] Let $G$ be a family of sequential computational graphs. Suppose for any $G \in \mathcal{G}$, the composition rule of the output node, $R_{O_G}$, is $c_i$-Lipschitz in variable $V_i$ for all $i$, and it only outputs value in $[0, 1]$. Suppose that $D R_{V_i}$ is $\kappa_i$-Lipschitz for each $i$. Let $\kappa_{j-i}$ (for $i \leq j$) be a set of parameters

Note that $D R_{V_i}$ maps a vector in space $D_{V_{i-1}}$ to an linear operator that maps $D_{V_{i-1}}$ to $D_{V_i}$.
that we intend to use to control Jacobians in the Lipschitz augmentation. With them, we apply Lipschitz augmentation as defined in Definition 5.2 to every graph in \( \mathcal{G} \) and obtain a new family of graphs, denoted by \( \mathcal{G}' \).

Then, the augmented family \( \mathcal{G}' \) is release-Lipschitz (Definition 5.2) with parameters \( \bar{\kappa}_V \)'s below:

\[
\bar{\kappa}_V \triangleq \sum_{i \leq j \leq q} 3c_j \kappa_{j-i+1} + 18 \sum_{1 \leq j \leq q} \sum_{j' = \max(i+1,j)} \frac{K_{j' - j} K_{j' - 1} - 1}{K_{j' - j}}, \quad \bar{\kappa}_J \triangleq \sum_{j \leq j' \leq q} 4K_{j' - j} \kappa_{j' - 1 - j}
\]

where for simplicity in the above expressions, we extend the definition of \( \kappa \)'s to \( \kappa_{j-1-j} = 1 \).

Finally, we combine Theorems 5.3 and Theorems 6.2 to derive covering number bounds for any Lipschitz augmentation of sequential computational graphs. The final covering bound in (7) can be easily computed given covering number bounds for each individual function class. In Section 7, we use this theorem to derive Rademacher complexity bounds for neural networks. The proof is deferred to Section C.

**Theorem 6.3.** Consider any family \( \mathcal{G} \) of sequential computational graphs satisfying the conditions of Theorem 6.2. By combining the augmentation of Definition 6.1 with additional indicators on the internal node norms, we can construct a new family \( \mathcal{G}' \) of computational graphs which output

\[
O_{\mathcal{G}}(x) = (O_{\mathcal{G}}(x) - 1) \prod_{i=1}^{q} \mathbb{1}_{s_{V_i}}(\|V_i(x)\|) \prod_{1 \leq i \leq j \leq q} \mathbb{1}_{\kappa_{j-1-j}}(\|DV_j(x) \cdots DV_i(x)\|_{op}) + 1
\]

The family \( \mathcal{G}' \) satisfies the following guarantees:

1. Each computational graph in \( \mathcal{G}' \) upper bounds its counterpart in \( \mathcal{G} \), i.e. \( O_{\mathcal{G}}(x) \geq O_{\mathcal{G}'}(x) \).

2. Define \( \bar{\kappa}_V' \triangleq \bar{\kappa}_V + \sum_{i \leq j \leq q} \kappa_{j-i+1} \cdot \kappa_{j-i+1}^{-1} \) and \( \bar{\kappa}_J' = \bar{\kappa}_J \) where \( \bar{\kappa}_V, \bar{\kappa}_J \) are defined as in Theorem 6.2. Then for any node-wise errors \( \{\epsilon_V\} \),

\[
\log N\left(\sum_{i \geq 1} \bar{\kappa}_V' \epsilon_{V_i} + \bar{\kappa}_J' \epsilon_J + \epsilon_O, O_{\mathcal{G}'}, s_I\right) \leq \sum_{i \geq 1} \log N(\epsilon_{V_i}, 2\mathcal{R}_{V_i}, 2s_{V_i-1}) + \log N(\epsilon_J, D\mathcal{R}_{V_i}, 2s_{V_i-1}) + \log N(\epsilon_O, \mathcal{R}_O, \{2s_{V_i}\}_{i=1}^q \cup \{I\})
\]

where \( D\mathcal{R}_{V_i} \) denotes the family of total derivatives of functions in \( \mathcal{R}_{V_i} \) and \( V_0 \) the input vertex.

### 7 Application to Neural Networks

In this section we provide our generalization bound for neural nets. Recall the definition of a neural network \( F \) parameterized by \( r \) weight matrices \( \{W^{(i)}\} \) in Example 5.4. We will use the convention that the \((i)\) superscript will index the weights of the network, and a subscript will index the individual layers, where activations and matrix multiplications are treated as distinct layers. Thus, the matrix \( W^{(i)} \) will be applied in layer \( 2i-1 \) of the network (see Figure 2 for a visualization of the layer indexing), and even indexed layers \( 2i \) will apply the activation \( \phi \).

For \( j < j' \), let \( F_{j' - j} \) denote the function computed between layers \( j \) and \( j' \), so \( F_{2j-2j} = \phi, F_{2j-1-2j-1} = h \mapsto W^{(i)}h \), and \( F_{j' - j} \) is recursively defined by \( F_{j' - j} = \phi \circ F_{j' - 1 - j} \) for \( j' > j \). Now for \( j < j' \), we define the layer \( j \)-to-\( j' \) Jacobian by \( Q_{j' - j} = DF_{j' - j} \circ F_{j' - 1 - j} \) where \( F_{0 - 1} \) denotes the identity mapping. For the below bound, \( P \) will denote a test distribution over examples \( x \) and labels \( y \), and \( F_n \) will denote the distribution on training examples. The proof is in Section A.
Theorem 7.1. Assume that the activation $\phi$ is 1-Lipschitz with a $\bar{\sigma}$-Lipschitz derivative. Fix reference matrices $\{A^{(i)}\}, \{B^{(i)}\}$. With probability $1 - \delta$ over the random draws of the data $P_n$, all neural networks $F$ with parameters $\{W^{(i)}\}$ and positive margin $\gamma$ will have the following generalization guarantee:

$$
\mathbb{E}_{(x,y) \sim P_n} [l_{0,1}(F(x), y)] \leq \tilde{O} \left( \left( \frac{\sum_i (\kappa_{\text{hidden}}(i) a^{(i)}(i-1))^2/3 + (\kappa_{\text{jacobian}}(i) b^{(i)}))^2/3}{\sqrt{n}} + r \frac{\log(1/\delta)}{n} \right) \right)
$$

where $\kappa_{\text{jacobian}}(i) \triangleq \sum_{1 \leq j \leq 2i-1 \leq j' \leq 2r-1} \frac{\sigma_{j' \rightarrow j}}{\sigma_{j' \rightarrow j}}$, and

$$
\kappa_{\text{hidden}}(i) \triangleq \frac{1}{\text{poly}(r)} + \frac{\sigma_{2i-1 \rightarrow 2i}}{\gamma} + \sum_{i \leq i' < r} \frac{\sigma_{2i' \rightarrow 2i}}{\gamma(i')} + \sum_{1 \leq j \leq j' \leq 2r-1} \frac{\sigma_{j' \rightarrow j}}{\gamma(j') \text{ even}}.
$$

In these expressions, we define $\sigma_{j-1 \rightarrow j} = 1$, and:

$$
a^{(i)} \triangleq \text{poly}(r)^{-1} + \|W^{(i)^T} - A^{(i)^T}\|_{2,1}, b^{(i)} \triangleq \text{poly}(r)^{-1} + \|W^{(i)} - B^{(i)}\|_{1,1}
$$

$$
t^{(0)} \triangleq \text{poly}(r)^{-1} + \max_{x \in P_n} ||x||, \quad t^{(i)} \triangleq \text{poly}(r)^{-1} + \max_{x \in P_n} ||F_{2i-1}(x)||
$$

$$
\sigma_{j' \rightarrow j} \triangleq \text{poly}(r)^{-1} + \max_{x \in P_n} ||Q_{j' \rightarrow j}(x)||_{\text{op}}, \quad \text{and} \quad \gamma \triangleq \min_{(x,y) \in P_n} [F(x)]_y - \max_{y' \neq y} [F(x)]_{y'} > 0
$$

Note that the training error here is 0 because of the existence of positive margin $\gamma$.

To obtain Theorem 7.1, we apply the result of Theorem 6.3 to the sequential computational graph interpretation of neural nets (Example 5.1) and bound the Rademacher complexity.

8 Experiments

Though the main purpose of the paper is to study the data-dependent generalization bounds from a theoretical perspective, we provide some preliminary experiments that demonstrate that the proposed complexity measure and generalization bounds are empirically relevant because regularizing the complexity measure leads to better test accuracy. Inspired by Theorem 7.1, we experiment with directly regularizing the Jacobian w.r.t outputs of normalization layers. Our reasoning is that normalization layers control the hidden layer norms, so additionally regularizing the Jacobians results in regularization of the product, which appears in our bound. We find that this is effective for improving test accuracy in a variety of settings. We note that Sokolíc et al. (2017) show positive experimental results for a similar regularization technique in data-limited settings.

For a WideResNet16 (Zagoruyko & Komodakis, 2016) architecture, we regularize the squared Frobenius norm of the gradient of the margin with respect to various hidden layers of the network. Inspired by the truncations in our augmented loss, we average the regularization over all examples in the batch whose squared Frobenius norm is above a certain threshold (in all our experiments, we choose 0.1) and add this cost to our objective. We tune the coefficient of this cost as a hyperparameter. For our experiments, we add Jacobian regularization w.r.t. layers appearing after BatchNorm in the standard architecture and use a baseline initial learning rate of 0.1. For all settings, we train for 200 epochs with learning rate decay by a factor of 0.2 at epochs 60, 120, and 150. Figure 4 shows the results for models trained and tested on CIFAR10 in low learning rate and no data augmentation settings. We also experiment with replacing BatchNorm layers with LayerNorm and additionally regularizing the Jacobian. We observe improvements in test error for all these settings.

9 Conclusion

In this paper, we tackle the question of how data-dependent properties affect generalization. We prove tighter generalization bounds that depend polynomially on the hidden layer sizes and norms of the interlayer Jacobians. To prove
| Setting                        | Normalization | Jacobian Reg | Test Error |
|-------------------------------|---------------|--------------|------------|
| Baseline                      | BatchNorm     | ×            | 4.43%      |
| Low learning rate (0.01)      | BatchNorm     | ×            | 5.98%      |
|                               |               | ✓            | 5.46%      |
| No data augmentation          | BatchNorm     | ×            | 10.44%     |
|                               |               | ✓            | 8.25%      |
| No BatchNorm                  | None          | ×            | 6.65%      |
|                               | LayerNorm     | ×            | 6.20%      |
|                               |               | ✓            | 5.57%      |

Figure 4: Test error for models trained on CIFAR10 in various settings.

these bounds, we work with the abstraction of computational graphs and develop general tools to augment any sequential family of computational graphs into a Lipschitz family and then cover this Lipschitz family. This augmentation and covering procedure applies to any sequence of function compositions. An interesting direction for future work is to generalize our techniques for Lipschitz augmentation to arbitrary computational graph structures, which will allow us to derive generalization bounds for arbitrary computational graph families. Additionally, there is the exciting empirical question of how to apply these bounds to develop better data-dependent regularization techniques.

Acknowledgments

Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

References

Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. arXiv preprint arXiv:1811.04918, 2018a.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. arXiv preprint arXiv:1811.03962, 2018b.

Sanjeev Arora, Ron Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. arXiv preprint arXiv:1802.05296, 2018.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. arXiv preprint arXiv:1901.08584, 2019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. Journal of Machine Learning Research, 3(Nov):463–482, 2002.

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

Friedrich L Bauer. Computational graphs and rounding error. SIAM Journal on Numerical Analysis, 11(1):87–96, 1974.

Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. Sgd learns over-parameterized networks that provably generalize on linearly separable data. arXiv preprint arXiv:1710.10174, 2017.
Yuan Cao and Quanquan Gu. A generalization theory of gradient descent for learning over-parameterized deep relu networks. *arXiv preprint arXiv:1902.01384*, 2019.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Simon S Du, Jason D Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. *arXiv preprint arXiv:1811.03804*, 2018a.

Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018b.

RM Dudley. The sizes of compact subsets of hilbert space and continuity of gaussian processes. *Journal of Functional Analysis*, 1(3):290–330, 1967.

Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. *arXiv preprint arXiv:1712.06541*, 2017.

Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems*, pp. 6151–6159, 2017.

Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry. *arXiv preprint arXiv:1802.08246*, 2018a.

Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Implicit bias of gradient descent on linear convolutional networks. *arXiv preprint arXiv:1806.00468*, 2018b.

Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pp. 1731–1741, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pp. 8571–8580, 2018.

Ziwei Ji and Matus Telgarsky. Risk and parameter convergence of logistic regression. *arXiv preprint arXiv:1803.07300*, 2018.

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

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pp. 971–980, 2017.

Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *arXiv preprint arXiv:1902.06720*, 2019.

Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pp. 8168–8177, 2018.
Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. In Conference On Learning Theory, pp. 2–47, 2018.

Etai Littwin and Lior Wolf. Regularizing by the variance of the activations’ sample-variances. In Advances in Neural Information Processing Systems, pp. 2115–2125, 2018.

Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layers neural networks. arXiv preprint arXiv:1804.06561, 2018.

Vaishnavh Nagarajan and Zico Kolter. Deterministic PAC-bayesian generalization bounds for deep networks via generalizing noise-resilience. In International Conference on Learning Representations, 2019. URL https://openreview.net/forum?id=Hygn2o0qKX

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In Conference on Learning Theory, pp. 1376–1401, 2015.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. arXiv preprint arXiv:1707.09564, 2017a.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In Advances in Neural Information Processing Systems, pp. 5947–5956, 2017b.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. arXiv preprint arXiv:1805.12076, 2018.

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

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. The Journal of Machine Learning Research, 19(1):2822–2878, 2018.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958, 2014.

Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. In Advances in neural information processing systems, pp. 351–359, 2013.

Colin Wei, Jason D Lee, Qiang Liu, and Tengyu Ma. On the margin theory of feedforward neural networks. arXiv preprint arXiv:1810.05369, 2018.

Wikipedia contributors. Chain rule — Wikipedia, the free encyclopedia, 2019.

Yuxin Wu and Kaiming He. Group normalization. arXiv preprint arXiv:1803.08494, 2018.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. arXiv preprint arXiv:1605.07146, 2016.

Chiuyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530, 2016.

Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Residual learning without normalization via better initialization. In International Conference on Learning Representations, 2019. URL https://openreview.net/forum?id=H1gsz30cKX

Difan Zou, Yuan Cao, Dongrui Zhou, and Quanquan Gu. Stochastic gradient descent optimizes over-parameterized deep relu networks. arXiv preprint arXiv:1811.08888, 2018.
A Missing Proofs for Section [7]

For a class of real-valued functions \( \mathcal{L} \) and dataset \( P_n \), define the empirical Rademacher complexity of this function class by

\[
\text{Rad}_n(\mathcal{L}) = \mathbb{E}_{\alpha_i} \left[ \sup_{l \in \mathcal{L}} \sum_i \alpha_i l(x_i) \right]
\]

where \( \alpha_i \) are independent uniform \( \pm 1 \) random variables. We will work in the neural network setting defined in Section 7. We will first state our generalization bound for neural networks.

**Theorem A.1.** Assume that the activation \( \phi \) is 1-Lipschitz with \( \bar{\sigma}_\phi \)-Lipschitz derivative. Fix parameters \( \sigma_{j' \rightarrow j}, t^{(i)}, a^{(i)}, b^{(i)}, \gamma \) and reference matrices \( \{A^{(i)}\}, \{B^{(i)}\} \). With probability \( 1 - \delta \) over the random draws of the distribution \( P_n \), all neural networks \( F \) with parameters \( \{W^{(i)}\} \) satisfying the following data-dependent conditions:

1. Hidden layers norms are controlled: \( \max_{x \in P_n} \|F_{2i-1}(x)\| \leq t^{(i)} \forall 1 \leq i \leq r \).
2. Jacobians are balanced: \( \max_{x \in P_n} \|Q_{j' \rightarrow j}(x)\|_{op} \leq \sigma_{j' \rightarrow j} \forall j < j' \).
3. The margin is large: \( \min_{(x,y) \in P_n} [F(x)]_y - \max_{y' \neq y} [F(x)]_{y'} \geq \gamma > 0 \).

and the additional data-independent condition

\[
\|W^{(i)} - A^{(i)}\|_{2,1} \leq a^{(i)}, \|W^{(i)} - B^{(i)}\|_{1,1} \leq b^{(i)}, \|W^{(i)}\|_{op} \leq \sigma_{2i-1 \rightarrow 2i-1}
\]

will have the following generalization to test data:

\[
\mathbb{E}_{(x,y) \sim \mathcal{D}} [l_{0,1}(F(x), y)] \leq \tilde{O}\left( \frac{\sum_i (\kappa_{\text{hidden},(i)} d^{(i)(1)}/2^{1/3} + (\kappa_{\text{jacobian},(i)} d^{(i)}/2^{1/3})^{3/2}}{\sqrt{n}} + \sqrt{\frac{\log(1/\delta)}{n}} \right)
\]

where

\[
\kappa_{\text{hidden},(i)} \triangleq \sum_{1 \leq j \leq 2i-1 \leq j' \leq 2r-1} \frac{4\sigma_{j' \rightarrow 2j} \sigma_{2i-2j}}{\sigma_{j' \rightarrow j}}
\]

\[
\kappa_{\text{jacobian},(i)} \triangleq \frac{\sigma_{2r-1 \rightarrow 2i}}{\gamma} + \sum_{i \leq i' < r} \frac{3\sigma_{2i' \rightarrow 2i}}{t^{(i')}}
\]

\[
+ \sum_{1 \leq j' \leq 2r-1 \leq j'' = \max\{2i,j\}, j'' \text{ even}} \frac{\sigma_{j' \rightarrow j'' + 1} \sigma_{j'' - 1 \rightarrow j} \sigma_{j' \rightarrow j}}{\sigma_{j' \rightarrow j}}
\]

Here we use the convention that \( \sigma_{j-1 \rightarrow j} = 1 \) and let \( t^{(0)} = \max_{x \in P_n} \|x\| \).

This generalization bound follows straightforwardly via the below Rademacher complexity bound for the augmented loss class:

**Theorem A.2.** Suppose that \( \phi \) is 1-Lipschitz with \( \bar{\sigma}_\phi \)-Lipschitz derivative. Define the following class of neural networks with norm bounds on its weight matrices with respect to reference matrices \( \{A^{(i)}\}, \{B^{(i)}\} \):\[
\mathcal{F} \triangleq \left\{ F \rightarrow F(x) : \|W^{(i)} - A^{(i)}\|_{2,1} \leq a^{(i)}, \|W^{(i)} - B^{(i)}\|_{1,1} \leq b^{(i)}, \|W^{(i)}\|_{op} \leq \sigma^{(i)} \right\}
\]
and let $\sigma_{j' \leftarrow j}$ be parameters that will bound the $j$ to $j'$ layerwise Jacobian for $j' \geq j$, where we set $\sigma_{2r-2i} = 1$ and $\sigma_{2r-2i+1} = \sigma^{(i)}$. Let $t^{(i)}$ be parameters bounding the layer norm after applying the $i$-th activation. (In particular, $t^{(i)}$ bounds $\max_{x \in P_i} \|x\|$.) Define the class of augmented losses

$$\mathcal{L}_{\text{aug}} \triangleq \left\{ (l_{\gamma} - 1) \circ F \prod_{i=1}^{r-1} \mathbb{1}_{\leq t^{(i)}}(\|F_{2i-1}\|) \prod_{1 \leq j' \leq 2r-1} \mathbb{1}_{\leq \sigma^{(i)}_{j' \leftarrow j}}(\|Q_{j' \leftarrow j}\|_{\text{op}}) + 1 : F \in \mathcal{F} \right\}$$

and define for $1 \leq i \leq r$, $\kappa_{\text{jacobian}}^{(i)}, \kappa_{\text{hidden}}^{(i)}$ meant to bound the influence of the matrix $W^{(i)}$ on the Jacobians and hidden variables, respectively as in [9], [10]. Then we can bound the empirical Rademacher complexity of the augmented loss class by

$$\text{Rad}_n(\mathcal{L}_{\text{aug}}) = \tilde{O}\left( \frac{\left( \sum_{i} (\kappa_{\text{hidden}}^{(i)}A^{(i)}2^{(i-1)})/3 + (\kappa_{\text{jacobian}}^{(i)}d^{(i)}2^{i/3})^{3/2} \right)}{\sqrt{n}} \right)$$

where we recall that the notation $\tilde{O}$ hides log factors in the arguments and the dimension of the weight matrices.

**Proof.** We associate the un-augmented loss class on neural networks $l_{\gamma} \circ \mathcal{F}$ with a family of sequential computation graphs $\mathcal{G}$ with depth $2r-1$. The composition rules are as follows: for internal node $V_{2i}$, $\mathcal{R}_{V_{2i}} = \{ \phi \}$, the set with only one element: the activation $\phi$. We also let $\mathcal{R}_{V_{2i-1}} = \{ h \mapsto W h : \|W^T - A^{(i)}\|_{2,1} \leq a^{(i)}, \|W - B^{(i)}\|_{1,1} \leq b^{(i)}, \|W\|_{\text{op}} \leq \sigma^{(i)} \}$. Finally, we choose $\mathcal{R}_{O}$ to be the singleton class $\{ l_{\gamma} \}$. Our collection of composition rules is then simply $\mathcal{R} = \mathcal{R}_V \otimes \cdots \otimes \mathcal{R}_{V_{2r-1}} \otimes \mathcal{R}_O$. Since $O_G$ takes values in $[0,1]$, we can apply Theorem 6.3 on this class $\mathcal{G}$ using $s_x = \max_{x \in P_i} \|x\|, s_{V_{2i}} = t^{(i)}, s_{V_{2i-1}} = \infty, \kappa_{2r-2i+1} = 1, \kappa_{2r-2i} = \sigma^{(i)}$, and $\kappa_{j' \leftarrow j} = \sigma_{j' \leftarrow j}$ for $j' > j$. Furthermore, we note that $\kappa_{2r} = \sigma^\prime$, and $\kappa_{2r-1} = 0$ as the Jacobian is constant for matrix multiplications. We thus obtain the class $\mathcal{G}$ where each augmented loss upper bounds the corresponding loss in $\mathcal{G}$. Recall that $J_i$ denote the additional nodes in our augmented computation graph. Note that under these choices of $s_{V_{2r-1}}, \kappa_{i \leftarrow i}$, we get that

$$\mathbb{1}_{\leq \kappa_{2r-2i+1}}(\|J_{2i}(x)\|_{\text{op}}) = \mathbb{1}_{\leq \kappa_{2r-2i-1}}(\|J_{2i-1}(x)\|_{\text{op}}) = 1 \quad \text{(as $|\phi'| \leq 1$)}$$

$$\mathbb{1}_{\leq \kappa_{2r-2i+1}}(\|J_{2i}(x)\|_{\text{op}}) = \mathbb{1}_{\leq \kappa_{2r-2i-1}}(\|J_{2i-1}(x)\|_{\text{op}}) = 1 \quad \text{(as $W^{(i)} \leq \sigma^{(i)}$)}$$

Furthermore, the other indicators in the augmented loss map to indicators in the outputs of our augmented graphs $O_G$, so therefore the families $\mathcal{L}_{\text{aug}}$ defined in the theorem statement and $\mathcal{G}$ are equivalent. Thus, it suffices to bound the Rademacher complexity of $\mathcal{G}$. To do this, we invoke covering numbers. By Theorem 6.3 we bound the covering number of $O_G$:

$$\log \mathcal{N}(\sum_{i \geq 1} (\tilde{\kappa}_{V_i} + \tilde{\kappa}_{J_i}) \epsilon_V + \epsilon_O, O_G, s_x) \leq$$

$$\sum_{i \geq 1} \log \mathcal{N}(\epsilon_V, \mathcal{R}_{V_i}, 2s_{V_{i-1}}) + \log \mathcal{N}(\epsilon_{J_i}, D\mathcal{R}_{V_i}, 2s_{V_{i-1}}) + \log \mathcal{N}(\epsilon_O, \mathcal{R}_{O}, \{2s_{V_i}\}_{i \geq 0})$$

(11)

where $\tilde{\kappa}_{V_i}, \tilde{\kappa}_{J_i}$ are defined in the statement of Theorem 6.3. After plugging in our values for $\tilde{\kappa}_j, s_{V_i}, \kappa_{j' \leftarrow j}$ in our application of Theorem 6.3 and noting that $c_{2i} = 1/t^{(i)}$, $c_{2i-1} = 0$ for $i < r$ and $1/\gamma$ for $i = r$ (as the margin loss is $1/\gamma$-Lipschitz), we obtain that

$$\tilde{\kappa}_{V_{2r-1}} = \kappa_{\text{hidden}}^{(i)}, \tilde{\kappa}_{J_{2r-1}} = \kappa_{\text{jacobian}}^{(i)}$$

We first note that the last term in (11) is simply 0 because there is exactly one output function in $\mathcal{R}_O$. Now for the other terms of (11), by definition $\mathcal{R}_{V_{2i}}, \mathcal{R}_{J_{2i}}$ consist of a singleton set and therefore have log cover size 0 for any error resolution $\epsilon$. Otherwise, to cover $\mathcal{R}_{V_{2i-1}}$ it suffices to bound $\log \mathcal{N}(\epsilon_{V_{2i-1}}, \{ h \mapsto W h : \|W^T - A^{(i)}\|_{2,1} \leq \sigma^{(i)} \}, \mathcal{R}_{V_{2i-1}}, \mathcal{R}_{J_{2i}}, s_{V_{i-1}}) \leq \log \mathcal{N}(\epsilon_{V_{2i-1}}, 2s_{V_{i-1}}) \leq$$

$$\log \mathcal{N}(\epsilon_{V_{2i-1}}, 2s_{V_{i-1}}) \leq \log \mathcal{N}(\epsilon_{V_{2i-1}}, s_{V_{i-1}})$$

(12)
under these choices of $\epsilon$. Thus, we can apply Lemma A.3 to obtain
\[
\log \mathcal{N}(\epsilon_{V_{2i-1}}, \mathfrak{R}_{V_{2i-1}}, 2s_{V_{2i-2}}) \leq \tilde{O}\left(\frac{(a(i) \epsilon^{(i-1)})^2}{\epsilon_{V_{2i-1}}^2}\right)
\]
Now we apply Dudley’s entropy theorem to obtain that $\epsilon_{V_{2i-1}}$, it suffices to cover $\{W : \|W - B^{(i)}\|_{1,1} \leq b^{(i)}\}$. The $\epsilon$-covering number of a $d_n^2$-dimensional $\ell_1$-ball with radius $b$ w.r.t. $\ell_2$ norm is $O\left(\frac{b^2}{\epsilon^2 \log d_n}\right)$. Thus,
\[
\log \mathcal{N}(\epsilon_{J_{2i-1}}, D \mathfrak{R}_{V_{2i-1}}, 2s_{V_{2i-2}}) \leq \tilde{O}\left(\frac{(b^{(i)})^2}{\epsilon_{J_{2i-1}}^2}\right)
\]
Now we define
\[
\beta^* \triangleq \left(\sum_i (\tilde{k}_{V_{2i-1}} a(i) \epsilon^{(i-1)})^{2/3} + (\tilde{k}_{J_{2i-1}} b^{(i)})^{2/3}\right)^{3/2}
\]
Now for a fixed error parameter $\epsilon$, we set $\epsilon_{V_i} = 0$, $\epsilon_{V_{2i-1}} = 0$, $\epsilon_{J_{2i-1}} = 0$ (as the log cover size is 0 anyways), and $\epsilon_{V_{2i-1}} = \epsilon_{V_{2i-1}} = \epsilon_{J_{2i-1}} = \epsilon$. Furthermore, under these choices of $\epsilon_{V_i}, \epsilon_{J_i}$, we end up with
\[
\sum_{i \geq 1} \log \mathcal{N}(\epsilon_{V_i}, \mathfrak{R}_{V_i}, 2s_{V_{i-1}}) + \log \mathcal{N}(\epsilon_{J_i}, D \mathfrak{R}_{V_i}, 2s_{V_{i-1}})
\]
\[
\leq \tilde{O}\left(\frac{1}{\epsilon^2} (\beta^*)^{4/3}\left(\sum_i (\tilde{k}_{\text{hidden},(i)} a(i) \epsilon^{(i-1)})^{2/3} + (\tilde{k}_{\text{jacobian},(i)} b(i))^{2/3}\right)^{3/2}\right) = \tilde{O}(\epsilon^{-2} (\beta^*)^2)
\]
Thus, substituting terms into (11) and collecting sums, we obtain that
\[
\log \mathcal{N}(\epsilon, O \tilde{g}, s) \leq \tilde{O}(\epsilon^{-2} (\beta^*)^2)
\]
Now we apply Dudley’s entropy theorem to obtain that
\[
\text{Rad}_n(\tilde{g}) = \tilde{O}\left(\frac{\left(\sum_i (\tilde{k}_{\text{hidden},(i)} a(i) \epsilon^{(i-1)})^{2/3} + (\tilde{k}_{\text{jacobian},(i)} b(i))^{2/3}\right)^{3/2}}{\sqrt{n}}\right)
\]
We now apply A.2 to prove Theorem A.1

**Proof of Theorem A.1** We start with Theorem A.2 which bounds the Rademacher complexity of the augmented loss class $\mathcal{L}_{\text{aug}}$. Using $l_{\text{aug}}(F, x, y)$ to denote the application of this augmented loss on the network $F$, its weights, and data $(x, y)$, we first note that $l_{0,1}(F(x), y) \leq l_{1}(F(x), y) \leq l_{\text{aug}}(F, x, y)$ for any datapoint $(x, y)$. We used the fact that margin loss upper bounds 0-1 loss, and $l_{\text{aug}}$ upper bounds margin loss by the construction in Theorem 6.3. Thus, applying the standard Rademacher generalization bound, with probability $1 - \delta$ over the training data, it holds that
\[
\mathbb{E}_{(x, y) \sim \mathcal{D}}[l_{0,1}(F(x), y)] \leq \mathbb{E}_{(x, y) \sim \mathcal{D}}[l_{\text{aug}}(F, x, y)] \leq \mathbb{E}_{(x, y) \sim \mathcal{D}}[l_{\text{aug}}(F, x, y)] + \text{Rad}_n(\mathcal{L}_{\text{aug}}) + \sqrt{\frac{\log(1/\delta)}{n}}
\]
\[
= \text{Rad}_n(\mathcal{L}_{\text{aug}}) + \sqrt{\frac{\log(1/\delta)}{n}} \quad \text{(by the data-dependent conditions)}
\]
Plugging in the bound on $\text{Rad}_n(\mathcal{L}_{\text{aug}})$ from Theorem A.2 gives the desired result. 

\[
\square
\]
Finally, to prove Theorems 7.1 and 1.1 we simply take a union bound over the choices of parameters \( \sigma_{j' \rightarrow j}, \ell(i), \alpha(i), b(i) \).

**Proof of Theorems 7.1 and 1.1** We will apply Theorem A.1 repeatedly over a grid of parameter choices \( \ell(i), \sigma_{j' \rightarrow j}, a(i), b(i) \) (following a technique of Bartlett et al. [2017]). For a collection \( \mathcal{M} \) of nonnegative integers \( m_t^{(j' \rightarrow j)}, m_a^{(j' \rightarrow j)}, m_b^{(j' \rightarrow j)}, m_r \), we apply Theorem A.1 choosing \( \ell(i) = \text{poly}(r)^{-1} 2^{m_t^{(j' \rightarrow j)}}, \sigma_{j' \rightarrow j} = \text{poly}(r)^{-1} 2^{m_a^{(j' \rightarrow j)}}, a(i) = \text{poly}(r)^{-1} 2^{m_b^{(j' \rightarrow j)}}, b(i) = \text{poly}(r)^{-1} 2^{m_r} \), \( \gamma = 2^{-m_r} \text{poly}(r) \max_i \sigma_{2r-1 \rightarrow 2i} \) and using error probability \( \delta_M = \frac{1}{\sum_{m_t^{(j' \rightarrow j)} \in \mathcal{M}} 2^{m_t^{(j' \rightarrow j)} \cdot m_a^{(j' \rightarrow j)} \cdot m_r}} \). First, we note that by union bound, using the fact that \( \sum_{\text{choices of } \mathcal{M}} 2^{m_t^{(j' \rightarrow j)} \cdot m_a^{(j' \rightarrow j)} \cdot m_r} = \delta \) where \( \mathcal{M} \) ranges over nonnegative integers, we get that the generalization bound of Theorem A.1 holds for choices of \( \mathcal{M} \) with probability \( 1 - \delta \).

Now for the network \( F \) at hand, there would have been some choice of \( \mathcal{M} \) for which the bound was applied using parameters \( \ell(i), \hat{\sigma}_{j' \rightarrow j}, \hat{\alpha}(i), \hat{b}(i), \gamma \) and

\[
\| W(i)^T - A(i)^T \|_{2,1} \leq \hat{\alpha}(i) = \text{poly}(r)^{-1} 2^{m_b^{(j' \rightarrow j)}} \leq \text{poly}(r)^{-1} + 2 \| W(i)^T - A(i)^T \|_{2,1}
\]

\[
\| W(i) - B(i) \|_{1,1} \leq \hat{b}(i) = \text{poly}(r)^{-1} 2^{m_b^{(j' \rightarrow j)}} \leq \text{poly}(r)^{-1} + 2 \| W(i) - B(i) \|_{1,1}
\]

\[
\max_{x \in P_n} \| F_{2i+1}(x) \| \leq \hat{i}(i) = \text{poly}(r)^{-1} 2^{m_a^{(j' \rightarrow j)}} \leq \text{poly}(r)^{-1} + 2 \max_{x \in P_n} \| F_{2i+1} \|
\]

\[
\max_{x \in P_n} \| Q_{j' \rightarrow j}(x) \|_{\text{op}} \leq \hat{\sigma}_{j' \rightarrow j} = \text{poly}(r)^{-1} 2^{m_a^{(j' \rightarrow j)}} \leq \text{poly}(r)^{-1} + 2 \max_{x \in P_n} \| Q_{j' \rightarrow j}(x) \|_{\text{op}}
\]

Furthermore, using \( \gamma \) to denote the true margin of the network, we also have \( \hat{\gamma} \leq \gamma \) and \( \frac{\hat{\gamma} 2r-1 - \hat{\sigma}_{j' \rightarrow j} - \hat{b}(i)}{n} \leq 4 \max_{x \in P_n} \| Q_{2r-1 \rightarrow 2i}(x) \|_{\text{op}} + \frac{1}{\text{poly}(r)} \). Furthermore, note that the cost we pay in \( \sqrt{\log(1/\delta_M)} = \hat{O} \left( r \sqrt{\log(1/\delta)} \right) \), where \( \hat{O} \) hides polylog factors in \( r \) and other parameters. Thus, the bound of Theorem 7.1 holds.

The proof of the simpler Theorem 1.1 follows the same above argument. The only difference is that we union bound over parameters \( \sigma, t \) and the matrix norms.

**Proposition A.1** (Dudley’s entropy theorem [Dudley, 1967]). Let \( s = \max_{x \in P_n} \| x \| \) be an upper bound on the largest norm of a datapoint. Then the following bound relates Rademacher complexity to covering numbers:

\[
\text{Rad}_n(\mathcal{L}) \leq \inf_{\alpha > 0} \left( \alpha + \int_0^\alpha \frac{\log N(\epsilon, \mathcal{L}, s)}{n} \right)
\]

**Lemma A.3.** For reference matrix \( A \in \mathbb{R}^{d_1 \times d_2} \), define the class of matrices mapping functions \( \mathcal{U} \triangleq \{ h \mapsto Uh : U \in \mathbb{R}^{d_1 \times d_2}, \| U^T - A^T \|_{2,1} \leq a \} \). Then

\[
\log N(\epsilon, \mathcal{U}, b) \leq \frac{2a^2 b^2}{\epsilon^2} \log(2d_1 d_2)
\]

**Proof.** By Lemma 3.2 of Bartlett et al. [2017], we can construct cover \( \mathcal{U} \) for the class \( \{ h \mapsto (U - A)h : U \in \mathbb{R}^{d_1 \times d_2}, \| U^T - A^T \|_{2,1} \leq a \} \) with the given cover size (Note that in our definition of empirical covering number, the resolution \( \epsilon \) is scaled by factor \( \frac{1}{n} \) versus theirs). To cover \( \mathcal{U} \) with the same cardinality set, we simply shift all functions in \( \mathcal{U} \) by \( A \).
Given a family of graphs \( \mathcal{G} \) (with shared edges \( \mathcal{E} \) and nodes \( \mathcal{V} \)), we assume the inductive hypothesis that “for any family of graphs with more than \( |\mathcal{Z}| \) input vertices, the theorem statement holds.” Under this hypothesis, we will show that the theorem statement holds for the graph family \( \mathcal{G} \).

We take node \( V_1 \) from the forest ordering \( (V_1, \ldots, V_m) \) assumed in the theorem. Suppose \( V_1 \) depends on \( C_1, \ldots, C_t \), which are assumed to be the input nodes by the definition of forest ordering. We release the node \( V_1 \) from the graph and obtain a new family \( \mathcal{G}^{V_1} = \{ G^{V_1} : G \in \mathcal{G} \} \) with a smaller number of edges than that of \( \mathcal{G} \).

Define \( u(h, x) \triangleq G_{G^{V_1}}(h, x) \) for \( h \in D_{V_1} \) and \( x \in D_{x} \), and \( u(x) = V_1(x) \). Then we can check that \( u(w(x), x) = O_G(x) \).

Lemma B.1. Let \( \mathcal{U} = \{ G_{G^{V_1}} : G \in \mathcal{G} \} \), and let \( \mathcal{W} = \mathcal{R}_{V_1} \). As each function in \( \mathcal{U} \) is \( \kappa_{V_1} \)-Lipschitz in \( V_1 \) because of condition 1, and it equals the fixed constant \( c \) if \( ||| \mathcal{V}_i ||| \geq s_V \) or \( ||| C_i ||| \geq s_C \), we have \( \mathcal{U}, \mathcal{W} \) satisfies the conditions of the composition lemma (see Lemma B.1). With the lemma, we conclude:

\[
\log \mathcal{N}(|\mathcal{U}|, \mathcal{G}, s) \leq \log \mathcal{N}(|\mathcal{V}_1|, \mathcal{U}, (s_{V_1}, s)) + \log \mathcal{N}(|\mathcal{U}|, \mathcal{G}, s) \]

Combining equation \( (14) \) and \( (15) \) above, we prove \( (5) \) for \( \mathcal{G} \), and complete the induction. \( \square \)

Below we provide the composition lemma necessary for Theorem 5.3.

**Lemma B.1.** Suppose

\[
\mathcal{U} \subseteq \{(h, x^{(1)}, \ldots, x^{(m)}) \in D_{h} \otimes D_{x}^{(1)} \otimes \cdots \otimes D_{x}^{(m)} \to D_{u}\}
\]

is a family of functions with two arguments and \( \mathcal{W} \subseteq \{ x^{(1)}, \ldots, x^{(m)} \in D_{x}^{(1)} \otimes \cdots \otimes D_{x}^{(m)} \to D_{h} \} \) is another family of functions. We overload notation and refer to \( x^{(i)} \) as \( x \). The spaces \( D_{h}, D_{x}, D_{u} \) all associate with some norms \( \| \cdot \| \) (the norms can potentially be different for each space, but we use the same notation for all of them.)

Assume the following:

1. All functions in \( \mathcal{U} \) are \( \kappa \)-Lipschitz in the argument \( h \) for any possible choice of \( x \): for any \( u \in \mathcal{U} \), \( x \in D_{x} \), and \( h, h' \in D_{h} \), we have \( \| u(h, x) - u(h', x) \| \leq \kappa \| h - h' \| \).

2. Any function \( u \in \mathcal{U} \) collapses on inputs with large norms: there exists a constant \( b \) such that \( u(h, x) = b \) if \( \| x^{(i)} \| \geq s_{x}^{(i)} \) for any \( i \).

Then, the family of the composition of \( u \) and \( w \), \( \mathcal{Z} = \{ z(x) = u(w(x), x) : u \in \mathcal{U}, w \in \mathcal{W} \} \), has covering number bound:

\[
\log \mathcal{N}(\kappa \epsilon_{u} + \epsilon_{u}, \mathcal{Z}, s) \leq \log \mathcal{N}(\epsilon_{u}, \mathcal{W}, s) + \log \mathcal{N}(\kappa \epsilon_{u}, \mathcal{G}, s)
\]

**Proof.** When it is clear from context, we let \( \| x \| \leq s_x \) denote the statement that \( \| x^{(i)} \| \leq s_{x}^{(i)} \). Suppose \( P_{n} \) is a uniform distribution over \( n \) data points \( \{ x_{1}, \ldots, x_{n} \} \subset D_{x} \) with norms not larger than \( s_{x} \). Given function \( u \in \mathcal{U} \) and \( w \in \mathcal{W} \), we will construct a pair of functions such that \( \tilde{u}(\tilde{w}(x), x) \) covers \( u(w(x), x) \). We will count (in a straightforward way) how many distinct pairs of functions we have construct for all the \((u, w)\) pairs at the end of the proof.

Let \( P' \) be the uniform distribution over \( \{ x_{1} : \| x_{i} \| \leq s_{x} \} \), and suppose \( \hat{W} \) is a \( \epsilon_{w} \sqrt{n / \text{supp}(P')} \) error cover of \( \mathcal{W} \) with respect to the metric \( L_2(P', \| \cdot \|) \). We note that \( \hat{W} \) has size at most \( \mathcal{N}(\epsilon_{w}, \mathcal{W}, s_{x}) \). We found \( \hat{w} \in \mathcal{W} \) such that \( \hat{w} \) is \( \epsilon_{w} \)-close to \( w \) in metric \( L_2(P', \| \cdot \|) \). Let \( \hat{h} \) be the \( w \)-closest \( h \). Let \( Q' \) be the uniform distribution over \( \{ (\hat{h}_{i}, x_{i}) : \| \hat{h}_{i} \| \leq s_{h}, \| x_{i} \| \leq s_{x} \} \), and let \( Q \) be the uniform distribution over all \( n \) points, \( \{ (\hat{h}_{1}, x_{1}), \ldots, (\hat{h}_{n}, x_{n}) \} \).
We first state the proofs of Theorem 6.2 and Theorem 6.3, which follow straightforwardly from the technical tools developed in Section 5.

**Proof of Theorem 6.2.** Fix any forest ordering \( S \) of \( \tilde{\mathcal{G}} \). Fix \( \tilde{\mathcal{G}} \subset \tilde{G} \). Let \( S' \) be the prefix sequence of \( S \) ending in \( V_i \). Note that \( S' \) will not contain any \( J_j \) or \( V_j \) for \( j > i \), as \( V_j \) and \( J_j \) will still depend on a non-input node (namely, \( V_{j-1} \)). Thus, we can fit \( \tilde{G}^{S'} \) under the framework of Lemma D.1, where we set \( k = q - i \) and identify \( f_j \) with \( R_{V_{j+1}} \). We set \( m = i \), and identify \( A_{m'} \) with \( J_i \cdots J_m \), where \( J_j \) may depend on input variables or itself be an input variable for \( 1 \leq j \leq i \), but this does not matter for our purposes. Then that to apply Lemma D.1 we set \( \tau_{j'i} = \kappa_{j'i' + i' + 1} \), \( \tau_{j'i = 1} = \kappa_{j'i' + i' - 1} \), and \( \tau_i = \kappa_i + 1 \). Now we can apply Lemma D.1 to conclude that \( \tilde{G}^S \) is \( \kappa_{V_i} \)-Lipschitz in \( V_i \) for any \( 1 \leq i \leq q \).

Now we prove release-Lipschitzness for a prefix sequence \( S' \) of \( S \) that ends in node \( V_i \). For all \( j \neq i \), fix \( D_j \in D_{J_i} \). It suffices to show that the function \( Q \) defined by

\[
Q(J_i) \triangleq \prod_{j \leq i} \mathbb{1}_{\leq \kappa_{j',i}}(\|D_j \cdots D_{i+1} J_i D_{i-1} \cdots D_j\|_{op}) \\
\times \prod_{j \geq i+1} \mathbb{1}_{\leq \kappa_{j',i' + i'}}(\|D_j' \cdots D_{i+1}\|_{op}) \times \prod_{j \leq i} \mathbb{1}_{\leq \kappa_{i',i}}(\|D_i' \cdots D_j\|_{op})
\]

\[\square\]
is \( \tilde{k}_{J_i} \)-Lipschitz in the value of \( J_i \). This is because after fixing all other inputs besides \( J_i \), we can write \( O_{\tilde{G},s'} \) in the form \( Q(J_i)a + 1 \), where \( a \) may depend on the other inputs but not \( J_i \) and \( |a| \leq 1 \). Now we simply apply Lemma D.8 to conclude that \( Q(J_i) \), and therefore \( O_{\tilde{G},s'} \), is \( \tilde{k}_{J_i} \)-Lipschitz.

**Proof of Theorem 6.3.** We first construct an augmented family of graphs \( G' \) sharing the same vertices and edges as \( G \). For \( G \in \mathcal{G} \), we add \( G' \) to \( G' \) computing

\[
O_{G'}(x) = (O_G(x) - 1) \prod_{i=1}^{q} \mathbb{1}_{s_{V_i}}(\|V_i(x)\|) + 1
\]

This is achieved by modifying the family of output rules as follows:

\[
R'_O(x, v_1, \ldots, v_q) = (R_O(x, v_1, \ldots, v_q) - 1) \prod_{i=1}^{q} \mathbb{1}_{s_{V_i}}(\|v_i\|) + 1
\]

where \( x \in D_x \) and \( v_i \in D_{V_i} \). We can also apply Claim F.1 to conclude that \( R'_O \) outputs values in \([0, 1]\). Furthermore, as the function \( \mathbb{1}_{s_{V_i}}(\|v_i\|) \) is \( s_{V_i}^{-1} \)-Lipschitz in \( v_i \), by the product property for Lipschitzness, \( R'_O \) is \((c_i + s_{V_i})^{-1}\)-Lipschitz in \( v_i \). Now we apply Theorem 6.2 to obtain a graph family \( \tilde{G} \) that is \( \{\tilde{k}_{V_i}\}\)-release-Lipschitz with respect to any forest ordering on \((\tilde{V}, \tilde{E})\) for parameters \( \{\tilde{k}_{V_i}\} \) defined in the theorem statement. Furthermore, by the construction of our augmentation and application of Claim F.1, it follows that

\[
\tilde{R}_O(x, v_1, \ldots, v_q, D_1, \ldots, D_q) =
(R_O(x, v_1, \ldots, v_q) - 1) \prod_{i=1}^{q} \mathbb{1}_{s_{V_i}}(\|v_i\|) \prod_{1 \leq i \leq q} \mathbb{1}_{s_{V_i}}(\|D_j \cdots D_i\|_{op}) + 1
\]

and in particular outputs the constant value 1 when \( \|v_i\| > 2s_{V_i} \) or \( \|D_i\| > 2\tilde{k}_{V_i} \). As this is a property of the output rule \( \tilde{R}_O \) itself, it is clear that condition 2 of Theorem 6.2 holds for any forest ordering on \((\tilde{V}, \tilde{E})\). Now we can apply Theorem 6.2.

\[
\log N(\sum_{i \geq 1} \tilde{k}_{V_i} + \tilde{k}_{J_i})\epsilon_{V} + \epsilon_{O}, \tilde{G}, s_{\tilde{x}}) \leq \sum_{i \geq 1} \log N(\epsilon_{V_i}, \tilde{G}_{V_i}, 2s_{V_i}) - 1
\]

\[
+ \log N(\epsilon_{J_i}, D\tilde{R}_{V_i}, 2s_{V_i}) + \log N(\epsilon_{O}, \tilde{R}_{O}, \{2s_{V_i}\} \cup \{I\} \cup \{2s_{J_i}\}_{i \geq 1})
\]

Now all terms match (7) except for the term \( \log N(\epsilon_{O}, \tilde{R}_{O}, \{2s_{V_i}\} \cup \{I\} \cup \{2s_{J_i}\}_{i \geq 1}) \). First, we note that all functions in \( \tilde{R}_{O} \) can be written in the form

\[
\tilde{R}_{O}(x, v_1, \ldots, v_q, D_1, \ldots, D_q) = (R_O(x, v_1, \ldots, v_q) - 1)Q(v_1, \ldots, v_q, D_1, \ldots, D_q) + 1
\]

where the function \( Q \) is the same for all \( \tilde{R}_{O} \in \tilde{R}_{O} \). It follows that to cover \( \tilde{R}_{O} \), we can first obtain a cover \( \tilde{R}_{O} \) of \( R_{O} \) and then apply the operation \( \hat{r} \mapsto (\hat{r} - 1)Q + 1 \) to each element in \( R_{O} \). Thus, we get the equivalence

\[
\log N(\epsilon_{O}, \tilde{R}_{O}, \{2s_{V_i}\}_{i \geq 0} \cup \{2s_{J_i}\}_{i \geq 1}) = \log N(\epsilon_{O}, R_{O}, \{2s_{V_i}\} \cup \{I\})
\]

This allows us to conclude (7). Finally, we note that as the augmentation operations are in the form of those considered in Claim F.1, it follows that \( O_{\tilde{G}} \) upper bounds \( O_{G} \). \( \square \)

## D Technical Tools for Lipschitz Augmentation

In this section, we develop the technical tools needed for proving Theorem 6.2. The main result in this section is our Lemma D.1 which essentially states that augmenting the loss with a product of Jacobians (plus additional matrices meant to model previous Jacobian nodes already released from the computational graph) will make the loss Lipschitz.
For this section, we say a function $J$ taking input $x \in D$ and outputting an operator mapping $D$ to $D'$ is $\kappa$-Lipschitz if $\|J(x) - J(x')\|_{op} \leq \kappa \|x - x'\|$ for any $x, x'$ in its input domain. We will consider functions $f_1, \ldots, f_k$, where $f_i : D_{i-1} \to D_i$ and $D_0$ is a compact subset of some normed space. For ease of notation, we use $\| \cdot \|$ to denote the (possibly distinct) norms on $D_0, \ldots, D_k$. For $1 \leq i \leq k$, Let $f_{j \leftarrow i} : D_{i-1} \to D_j$ denote the composition

$$f_{j \leftarrow i} \triangleq f_j \circ \cdots \circ f_i$$

For convenience in indexing, for $(i, j)$ with $i > j$, we will set $f_{j \leftarrow i} : D_{i-1} \to D_{i-1}$ to be the identity function.

Finally consider a function real-valued function $g : D_0 \otimes \cdots \otimes D_k \to [0, 1]$ and define the composition $\bar{z} : D_0 \mapsto [0, 1]$ by

$$\bar{z}(x) = g(x, f_{1 \leftarrow 1}(x), \ldots, f_{k \leftarrow 1}(x))$$

We will construct a “Lipschitz-fication” for the function $\bar{z}$.

Let $A_1, \ldots, A_m$ denote a collection of linear operators that map to the space $D_0$. We will furthermore use $J_{j \leftarrow i, m'}$ to denote the $i$-to-$j$ Jacobian, i.e.

$$J_{j \leftarrow i, m'} \triangleq Df_{j \leftarrow i} \circ f_{i-1 \leftarrow 1}$$

When $i = 1$ and $0 \leq j \leq k$, we will also consider products between 1-to-$j$ Jacobians and the matrices $A_{m'}$: define

$$J_{j \leftarrow 1, m'} \triangleq (Df_{j \leftarrow 1})A_{m'}$$

Note in particular that $J_{0 \leftarrow 1, m'} = A_{m'}$.

**Lemma D.1.** [Lipschitz-fication] Following the notation in this section, suppose that $g$ is $c_k$-Lipschitz in its $(k' + 1)$-th argument for $0 \leq k' \leq k$. Suppose that $Df_{j \leftarrow j}$ is $\tau_j$-Lipschitz for all $1 \leq j \leq k$. For any $(i, j)$ with $1 \leq i \leq j \leq k$, let $\tau_{j \leftarrow i}$ be parameters that intend to be a tight bound on $\|J_{j \leftarrow i}\|_{op}$, and also define $\tau_{j \leftarrow 1, m'}$ which will bound $\|J_{j \leftarrow 1, m'}\|_{op}$. Define the augmented function $\tilde{z} : D_0 \mapsto [0, 1]$ by

$$\tilde{z}(x) = (z(x) - 1) \prod_{2 \leq i \leq j} \mathbb{1}_{\leq \tau_{j \leftarrow i}((\|J_{j \leftarrow i}\|_{op}) \prod_{0 \leq j \leq k, m'} \mathbb{1}_{\leq \tau_{j \leftarrow 1, m'}((\|J_{j \leftarrow 1, m'}\|_{op}) + 1

Define $\tau^*$, a Lipschitz parameter for $\tilde{z}$, by

$$\tau^* \triangleq \sum_{0 \leq j \leq k} 3c_j \tau_{j \leftarrow 1}$$

$$+ 18 \sum_{1 \leq j \leq k} \sum_{1 \leq i \leq j} \frac{\tau_i \tau_{j \leftarrow i+1} \tau_{j \leftarrow 1} + \tau_i \tau_{j \leftarrow i+1} \tau_{j \leftarrow 1}}{\tau_{j \leftarrow i}}$$

$$+ 18 \sum_{1 \leq j \leq k, m'} \sum_{1 \leq i \leq j} \frac{\tau_i \tau_{j \leftarrow i+1} \tau_{j \leftarrow 1} + \tau_i \tau_{j \leftarrow i+1} \tau_{j \leftarrow 1}}{\tau_{j \leftarrow 1, m'}}$$

where for convenience we let $\tau_{j \leftarrow i} = 1$ when $j < i$. Then $\tilde{z}$ is $\tau^*$-Lipschitz in $x$.

**Proof.** For ease of notation, we will first define for any $(i, j)$ with $1 \leq i \leq j \leq k$, $Q_{j \leftarrow i} \triangleq \mathbb{1}_{\leq \tau_{j \leftarrow i}((\|J_{j \leftarrow i}\|_{op})$ and for $(j, m')$ with $0 \leq j \leq k$, $Q_{j \leftarrow 1, m'} \triangleq \mathbb{1}_{\leq \tau_{j \leftarrow 1, m'}((\|J_{j \leftarrow 1, m'}\|_{op})$. Note in particular that $Q_{0 \leftarrow 1, m'}$ is always a constant function. We will also let $\mathcal{Q}$ denote the collection of functions

$$\mathcal{Q} = \{Q_{j \leftarrow j} \}_{1 \leq j \leq k} \cup \{Q_{j \leftarrow 1, m'} \}_{0 \leq j \leq k, 1 \leq m' \leq m}$$

We define the following order $\succ \mathcal{Q}$ on this collection of functions:
To prove that \( \tilde{z} \) is \( \tau^* \)-Lipschitz, It suffices show that \( \forall x \) and sufficiently small \( \nu \), 
\[
|\tilde{z}(x) - \tilde{z}(x + \nu)| \leq \tau^* \|\nu\|.
\]
First, define for \( 0 \leq j \leq k \)
\[
\gamma_j(x, \nu) \triangleq g(f_{0-1}(x), \ldots, f_{j-1}(x), f_{j+1-1}(x + \nu), \ldots, f_{k-1}(x + \nu)) - g(f_{0-1}(x), \ldots, f_{j-1}(x), f_{j+1-1}(x + \nu), \ldots, f_{k-1}(x + \nu))
\]
Next, define the telescoping differences
\[
\delta_j(x, \nu) \triangleq \gamma_j(x, \nu) \prod_{Q \succeq Q_{j-1}, j} Q(x) \prod_{Q \succ Q_{j-1}, j} Q(x + \nu) \forall 0 \leq j \leq k
\]
\[
\Delta_{j-1, i}(x, \nu) \triangleq (Q_{j-1}(x) - Q_{j-1}(x + \nu)) \prod_{Q \succeq Q_{j-1}, j} Q(x) \prod_{Q \succ Q_{j-1}, j} Q(x + \nu) \forall 1 \leq i \leq j \leq k
\]
\[
\Delta_{j-1, m'}(x, \nu) \triangleq (Q_{j-1, m'}(x) - Q_{j-1, m'}(x + \nu)) \prod_{Q \succeq Q_{j-1, m'}, j} Q(x) \prod_{Q \succ Q_{j-1, m'}, j} Q(x + \nu) \forall 0 \leq j \leq k
\]
Now note that by Claim [D.7] we have the bound
\[
|\tilde{z}(x) - \tilde{z}(x + \nu)| \leq \sum_{0 \leq j \leq k} |\delta_j(x, \nu)| + \sum_{1 \leq i \leq j \leq k} |\Delta_{j-1, i}(x, \nu)| + \sum_{0 \leq j \leq k, m'} |\Delta_{j-1, m'}(x, \nu)|
\]
Define \( \bar{\tau} \) to be the Lipschitz constant of \( J_{1-1} \) on \( D_0 \) for all \( 1 \leq i \leq j \leq k \) guaranteed by Claim [D.6] First, note that \( \Delta_{0-1, m'} = 0 \) for all \( m' \). Thus, by Claims [D.4] and [D.5] it follows that
\[
|\tilde{z}(x) - \tilde{z}(x + \nu)| \leq \sum_{0 \leq j \leq k} c_j (2\tau_{j-1} + \frac{\bar{\tau}}{2} \|\nu\|)\|\nu\|
\]
\[
+ \sum_{1 \leq i \leq j \leq k} \|\nu\| \sum_{i' = i}^j (2\tau_{j-i'} + \frac{\bar{\tau}}{2} \|\nu\|)\tau_{i'} (2\tau_{j-i'-1} + \frac{\bar{\tau}}{2} \|\nu\|) 2\tau_{j-i-i'}
\]
\[
+ \sum_{1 \leq j \leq k, 1 \leq m' \leq m} \|\nu\| \sum_{i' = 1}^j (2\tau_{j-i'} + \frac{\bar{\tau}}{2} \|\nu\|)\tau_{i'} (2\tau_{j-i'-1} + \frac{\bar{\tau}}{2} \|\nu\|) 2\tau_{j-i'-1, m'}
\]
Now note that if \( \|\nu\| \leq 2 \min_{i \leq j} \tau_{i-j} \), then it follows that \( 2\tau_{j-i} + \frac{\bar{\tau}}{2} \|\nu\| \leq 3\tau_{j-i} \forall i \leq j \). Substituting into (21), we
get that \( \forall x, \| \nu \| \leq \frac{2 \min_{j \leq k} \tau_{j+i}}{\xi} \),

\[
|\tilde{z}(x) - \tilde{z}(x + \nu)| \leq \|\nu\| \sum_{0 \leq j \leq k} 3c_j \tau_{j+1}
\]

Furthermore, for \( i \leq j \leq k \), we can expand the error \( J_{j-i}(x) - J_{j-i}(x + \nu) \) as follows:

\[
J_{j-i}(x) - J_{j-i}(x + \nu) = \sum_{i' = i}^{j} J_{j-i'+1}(x + \nu)(J_{i'_{-i'}}(x) - J_{i'_{-i'}}(x + \nu))J_{i'_{-1-i'}}(x)
\]

\[
J_{j-i}(x) - J_{j-i}(x + \nu) = \sum_{i' = i}^{j} J_{j-i'+1}(x + \nu)(J_{i'_{-i'}}(x) - J_{i'_{-i'}}(x + \nu))J_{i'_{-1-i'}}(x)
\]

Proof. We will first show (22) by inducting on \( j - i \). The base case \( j = i \) follows by definition, as we can reduce \( J_{i_{-i+1}} \) and \( J_{i_{-1-i}} \) to constant-valued functions that output the identity matrix.

For the inductive step, we use Claim F.2 to expand

\[
J_{j-i}(x) - J_{j-i}(x + \nu) = J_{j-i+1}(x)J_{i-i}(x) - J_{j-i+1}(x + \nu)J_{i-i}(x + \nu)
\]

\[
= (J_{j-i+1}(x) - J_{j-i+1}(x + \nu))J_{i-i}(x)
\]

\[
+ J_{j-i+1}(x + \nu)(J_{i-i}(x) - J_{i-i}(x + \nu))
\]

\[
= \sum_{i' = i+1}^{j} J_{j-i'+1}(x + \nu)(J_{i'_{-i'}}(x) - J_{i'_{-i'}}(x + \nu))J_{i'_{-1-i'+1}}(x)J_{i_{-1-i'}}(x)
\]

(by the inductive hypothesis)

\[
+ J_{j-i+1}(x + \nu)(J_{i-i}(x) - J_{i-i}(x + \nu))
\]

\[
= \sum_{i' = i+1}^{j} J_{j-i'+1}(x + \nu)(J_{i'_{-i'}}(x) - J_{i'_{-i'}}(x + \nu))J_{i'_{-1-i'+1}}(x)
\]

(by Claim F.2)

\[
+ J_{j-i+1}(x + \nu)(J_{i-i}(x) - J_{i-i}(x + \nu))
\]

\[
= \sum_{i' = i}^{j} J_{j-i'+1}(x + \nu)(J_{i'_{-i'}}(x) - J_{i'_{-i'}}(x + \nu))J_{i'_{-1-i'+1}}(x)
\]

as desired.

It follows that \( \tilde{z} \) is \( \tau^* \)-Lipschitz. \( \square \)

Claim D.2. In the setting of Lemma D.1, for \( 1 \leq i \leq j \leq k \), we can expand the error \( J_{j-i}(x) - J_{j-i}(x + \nu) \) as follows:

\[
J_{j-i}(x) - J_{j-i}(x + \nu) = \sum_{i' = i}^{j} J_{j-i'+1}(x + \nu)(J_{i'_{-i'}}(x) - J_{i'_{-i'}}(x + \nu))J_{i'_{-1-i'}}(x)
\]

Furthermore, for \( 1 \leq j \leq k, m' \), we can expand the error \( J_{j-i,m'}(x) - J_{j-i,m'}(x + \nu) \) as follows:

\[
J_{j-i,m'}(x) - J_{j-i,m'}(x + \nu) = \sum_{i' = i}^{j} J_{j-i'+1}(x + \nu)(J_{i'_{-i'}}(x) - J_{i'_{-i'}}(x + \nu))J_{i'_{-1-i'}}(x)
\]
To prove (25), we first note that by definition, \( J_{j \leftarrow 1, m'}(x) = J_{j \leftarrow 1}(x) J_{0 \leftarrow 1, m'} \), so

\[
J_{j \leftarrow 1, m'}(x) - J_{j \leftarrow 1, m'}(x + \nu) = (J_{j \leftarrow 1}(x) - J_{j \leftarrow 1}(x + \nu)) J_{0 \leftarrow 1, m'}
\]  
(24)

\[
= \sum_{i' = 1}^{j} J_{j \leftarrow i' + 1}(x + \nu)(J_{i' \leftarrow i'}(x) - J_{i' \leftarrow i'}(x + \nu)) J_{i' \leftarrow 1 - 1, m'} J_{0 \leftarrow 1, m'}
\]  
(by (22))

\[
= \sum_{i' = 1}^{j} J_{j \leftarrow i' + 1}(x + \nu)(J_{i' \leftarrow i'}(x) - J_{i' \leftarrow i'}(x + \nu)) J_{i' \leftarrow 1 - 1, m'}(x) \quad \text{(since } J_{i' \leftarrow 1 - 1}(x) J_{0 \leftarrow 1, m'} = J_{i' \leftarrow 1 - 1, m'}(x))
\]

\[
\square
\]

**Claim D.3.** In the setting of Lemma [D.1], suppose that \( J_{i \leftarrow i} \) is \( \bar{\tau} \)-Lipschitz for all \( 1 \leq i \leq j \leq k \). Then we can bound the operator norm error in the Jacobian by

\[
\|J_{j \leftarrow i}(x) - J_{j \leftarrow i}(x + \nu)\|_{op} \leq \|\nu\| \sum_{i' = 1}^{j} (\|J_{j \leftarrow i' + 1}(x)\|_{op} + \bar{\tau} \|\nu\|) \bar{\tau}_{i'}(\|J_{i' \leftarrow 1 - 1}(x)\|_{op} + \frac{\bar{\tau}}{2} \|\nu\|) \|J_{i' \leftarrow 1 - 1}(x)\|_{op}
\]  
(25)

Likewise, we can bound the operator norm error in the product between Jacobian and auxiliary matrices by

\[
\|J_{j \leftarrow 1, m'}(x) - J_{j \leftarrow 1, m'}(x + \nu)\|_{op} \leq \|\nu\| \sum_{i' = 1}^{j} (\|J_{j \leftarrow i' + 1}(x)\|_{op} + \bar{\tau} \|\nu\|) \bar{\tau}_{i'}(\|J_{i' \leftarrow 1 - 1}(x)\|_{op} + \frac{\bar{\tau}}{2} \|\nu\|) \|J_{i' \leftarrow 1 - 1, m'}(x)\|_{op}
\]  
(26)

**Proof.** We will first prove (25), as the proof of (26) is nearly identical. Starting from (22) of Claim D.2, we have

\[
J_{j \leftarrow i}(x) - J_{j \leftarrow i}(x + \nu) = \sum_{i' = 1}^{j} J_{j \leftarrow i' + 1}(x + \nu)(J_{i' \leftarrow i'}(x) - J_{i' \leftarrow i'}(x + \nu)) J_{i' \leftarrow 1 - 1}(x)
\]

By triangle inequality and the fact that \( J_{i' \leftarrow i'} \) is \( \bar{\tau} \)-Lipschitz \( \forall i' \leq j' \), it follows that

\[
\|J_{j \leftarrow i}(x) - J_{j \leftarrow i}(x + \nu)\|_{op} \leq \sum_{i' = 1}^{j} \|J_{j \leftarrow i' + 1}(x + \nu)\|_{op}\|J_{i' \leftarrow i'}(x) - J_{i' \leftarrow i'}(x + \nu)\|_{op}\|J_{i' \leftarrow 1 - 1}(x)\|_{op}
\]  
(27)

\[
\leq \sum_{i' = 1}^{j} (\|J_{j \leftarrow i' + 1}(x + \nu)\|_{op} + \bar{\tau} \|\nu\|)\|J_{i' \leftarrow 1 - 1}(x)\|_{op}
\]

\[
(28)
\]

Next, we note that

\[
\|J_{i' \leftarrow i'}(x) - J_{i' \leftarrow i'}(x + \nu)\|_{op} = D f_{i' \leftarrow i'}[f_{i' \leftarrow 1 - 1}(x)] - D f_{i' \leftarrow i'}[f_{i' \leftarrow 1 - 1}(x + \nu)]
\]

\[
\leq \bar{\tau}_{i'}\|f_{i' \leftarrow 1 - 1}(x) - f_{i' \leftarrow 1 - 1}(x + \nu)\|
\]

\[
\leq \bar{\tau}_{i'}(\|J_{i' \leftarrow 1 - 1}(x)\|_{op} + \frac{\bar{\tau}}{2} \|\nu\|) \|\nu\|
\]  
(applying Claim F.4)

Plugging the above into (28), we get (25). To prove (26), we start from (25) and follow the same steps as above. \( \square \)
Claim D.4. In the setting of Lemma D.1, suppose that $J_{j-i}$ is $\tau$-Lipschitz for all $1 \leq i \leq j \leq k$. Then we can upper bound the error terms corresponding to the indicators by

$$|\Delta_{j-i}(x, \nu)| \leq \|\nu\| \sum_{i' = i}^{j} (2\tau_{j-i'+1} + \bar{\tau}\|\nu\|)\bar{\nu}(2\tau_{i'-1+i} + \frac{\bar{\tau}}{2}\|\nu\|)2\tau_{i'-1+i}$$

(29)

Likewise, the following upper bound holds for all $(j, m')$ with $1 \leq j \leq k$, $1 \leq m' \leq m$:

$$|\Delta_{j-1,m'}(x, \nu)| \leq \|\nu\| \sum_{i = 1}^{j} (2\tau_{j-i'+1} + \bar{\tau}\|\nu\|)\bar{\nu}(2\tau_{i'-1+i} + \frac{\bar{\tau}}{2}\|\nu\|)2\tau_{i'-1+i,m'}$$

(30)

Proof. We will prove (29) as the proof of (30) is analogous. Note that as $1 \leq_{\tau_{j-i}}$ is $\frac{1}{\tau_{j-i}}$-Lipschitz in its argument, we have

$$|Q_{j-i}(x) - Q_{j-i}(x + \nu)| = \|\frac{1}{\tau_{j-i}}(\|J_{j-i}(x)\|_{op} - 1)\|_{op} - \|J_{j-i}(x + \nu)\|_{op}|$$

$$\leq \frac{1}{\tau_{j-i}}\|J_{j-i}(x)\|_{op} - \|J_{j-i}(x + \nu)\|_{op}$$

$$\leq \frac{1}{\tau_{j-i}}\|J_{j-i}(x) - J_{j-i}(x + \nu)\|_{op}$$

Plugging this into our definition for $\Delta_{j-i}$ (19), it follows that

$$|\Delta_{j-i}(x, \nu)| \leq \frac{1}{\tau_{j-i}}\|J_{j-i}(x) - J_{j-i}(x + \nu)\|_{op} \prod_{Q \supseteq Q_{j-i}} Q(x) \prod_{Q_{j-i'} \supseteq Q} Q(x + \nu)$$

(31)

Now we define the set $E$ by

$$E = \cap_{i \leq i'} \{x : \|J_{j-i'+1}(x)\|_{op} \leq 2\tau_{j-i'+1}, \|J_{i'-1+i}(x)\|_{op} \leq 2\tau_{i'-1+i}, \text{and } \|J_{i'-1+i}(x)\|_{op} \leq 2\tau_{i'-1+i}\}$$

Note that if $x \notin E$, then $\exists i' < j'$ such that $Q_{j-i'}(x) = 0$ and $Q_{j'-i'}(Q_{j-i})$ by definition of the order $\succ_{Q}$. It follows that if $x \notin E$, $\prod_{h \succ_{Q} Q_{j-i}} h(x) = 0$, so $|\Delta_{j-i}(x, \nu)| = 0$. Otherwise, if $x \in E$, by Claim D.3 we have

$$\|J_{j-i}(x) - J_{j-i}(x + \nu)\|_{op} \leq \|\nu\| \sum_{i = 1}^{j} (2\tau_{j-i'+1} + \bar{\tau}\|\nu\|)\bar{\nu}(2\tau_{i'-1+i} + \frac{\bar{\tau}}{2}\|\nu\|)2\tau_{i'-1+i}$$

where we recall that $\tau_{j-1-i} = 1$. Plugging this into (31) and using the fact that all functions $h \in Q$ are bounded by 1 gives the desired statement.

To prove (30), we simply apply the above argument with (26).

Claim D.5. In the setting of Lemma D.1, fix index $j$ with $0 \leq j \leq k$ and suppose that $J_{j-i}$ is $\bar{\tau}$-Lipschitz. Then we can bound the error due to function composition by

$$|\delta_{j}(x, \nu)| \leq c_{j}(2\tau_{j-1} + \frac{\bar{\tau}}{2}\|\nu\|)\|\nu\|$$

Proof. Starting from (18), we can first express $\delta_{i}(x, \nu)$ by

$$\delta_{j}(x, \nu) = \gamma_{j}(x, \nu)Q_{j-i}(x) \prod_{Q \supseteq Q_{j-i}, Q \neq Q_{j-i}} Q(x) \prod_{Q_{j-i'} \supseteq Q} Q(x + \nu)$$

as $Q_{j-i} \succ_{Q} Q_{j-i-1}$. First we note that by definition, $|\gamma_{j}(x, \nu)| \leq \|f_{j-i}(x) - f_{j-i}(x + \nu)\|_{op}$, as the function $g$ is $c_{j}$-Lipschitz in its $j$-th argument. Thus, since all functions $Q \in Q$ are bounded by 1, it follows that

$$|\delta_{j}(x, \nu)| \leq |\gamma_{j}(x, \nu)|Q_{j-i}(x)$$

$$\leq c_{j}(2\tau_{j-1} + \frac{\bar{\tau}}{2}\|\nu\|)\|\nu\|$$

(by Claim F.4)
Claim D.6. In the setting of Lemma D.1, there exists \( \exists \bar{\tau} \) such that \( \forall i \leq j, J_{j+1} \) is \( \bar{\tau} \)-Lipschitz on a compact domain \( D_0 \).

Proof. We first show inductively that \( f_{i+1} \) is Lipschitz for all \( i \). The base case \( f_{1+1} \) follows by definition, as \( f_{1+1} \) is continuously differentiable and \( D_0 \) is a compact set.

Now we show the inductive step: first write \( f_{i+1} = f_i \circ f_{i-1+1} \). By continuity, \( \{ f_{i-1+1}(x) : x \in D_0 \} \) is compact. Furthermore, \( f_i \) is continuously differentiable under the assumptions of Lemma D.1. Thus, \( f_i \) is Lipschitz on domain \( \{ f_{i-1+1}(x) : x \in D_0 \} \). As \( f_{i+1} = f_i \circ f_{i-1+1} \) is the composition of Lipschitz functions by the inductive hypothesis, \( f_{i+1} \) is itself Lipschitz.

Now it follows that \( \forall i, J_{j+1} \) is Lipschitz on \( D_0 \), as it is the composition of \( Df_{i+1} \) and \( f_{i-1+1} \), both of which are Lipschitz. Finally, by the chain rule (Claim F.3), we have that \( J_{j+1} = J_{j+2} \cdots J_{j+1} \) is the product of Lipschitz functions, and therefore Lipschitz for all \( i < j \). We simply take \( \bar{\tau} \) to be the maximum Lipschitz constant of \( J_{j+1} \) over all \( i \leq j \).

Claim D.7. In the setting of Lemma D.1,

\[
|\tilde{z}(x) - \tilde{z}(x + \nu)| \leq \sum_{0 \leq j \leq k} |\delta_j(x, \nu)| + \sum_{1 \leq i \leq j \leq k} |\Delta_j(x, \nu)| + \sum_{0 \leq j \leq k, m'} |\Delta_{j+1, m'}(x, \nu)|
\]

Proof. For \( 0 \leq j \leq k + 1 \), define \( z_j(x, \nu) \) by

\[
z_j(x, \nu) \triangleq g(f_0(x), \ldots, f_{j-1+1}(x), f_{j+1}(x + \nu), f_{j+1}(x + \nu), \ldots, f_{k+1}(x + \nu))
\]

Thus, \( z_j(x, \nu) \) denotes \( g \circ (f_0 \circ \cdots \circ f_{k+1}) \) with the last \( k + 1 - j \) inputs given by \( g \) depending on \( x + \nu \) instead of \( x \).

Now we claim that by a telescoping argument (Claim F.3),

\[
\tilde{z}(x) - \tilde{z}(x + \nu) = \sum_{0 \leq j \leq k} \delta_j(x, \nu) + \sum_{1 \leq i \leq j \leq k} (z_k(j, \nu) - 1) \Delta_j + \sum_{0 \leq j \leq k, m'} (z_j(x, \nu) - 1) \Delta_{j+1, m'}
\]

(32)

To see this, compute the sum in the order the following sequence of terms, which corresponds to a traversal of \( Q \) in least-to-greatest order:

\[
\delta_k, (z_k(x, \nu) - 1) \Delta_{k+1, m'}, \ldots, (z_k(x, \nu) - 1) \Delta_{k+1, m'}, (z_k(x, \nu) - 1) \Delta_{k+1, m'}, \ldots, (z_k(x, \nu) - 1) \Delta_{k+1, m'}
\]

Now we simply apply triangle inequality on (32) and use the fact that \( z_j(x, \nu) - 1 \in [-1, 0] \forall 0 \leq j \leq k + 1 \) to obtain the desired statement.

Lemma D.8. In the setting of Theorem 6.2, fix \( 1 \leq i \leq p \) and define

\[
Q(J_i) \triangleq \prod_{j \leq i \leq j'} \mathbb{I}_{k_{j'-1}}(\|D_j \cdots D_{i+1} J_i D_{i-1} \cdots D_j\|_{op}) \times \prod_{j' \geq i+1} \mathbb{I}_{k_{j'+1}}(\|D_j \cdots D_{i+1}\|_{op}) \times \prod_{j \leq i} \mathbb{I}_{k_{i-1}}(\|D_{i-1} \cdots D_j\|_{op})
\]

Then \( Q \) is \( \tilde{k}_{J_i} \)-Lipschitz in \( J_i \), where

\[
\tilde{k}_{J_i} \triangleq \sum_{j \leq i \leq j'} \frac{4k_{j'-i+1}k_{i-1-j}}{k_{j'-j}}
\]

Here for convenience we use the convention that \( k_{i-1-i} = 1 \).

Proof. There are two cases: the condition \( \|D_j \cdots D_{i+1}\|_{op} \leq 2k_{j'-i+1} \) and \( \|D_{i-1} \cdots D_j\|_{op} \leq 2k_{i-1-j} \) for all \( j' \geq i + 1, j \leq i - 1 \) either holds or does not hold. In the case that it does not hold, \( Q \) is the constant function at 0, and is certainly \( \tilde{k}_{J_i} \)-Lipschitz. In the case that the condition does hold, \( \mathbb{I}_{k_{j'-i+1}}(\|D_j \cdots D_{i+1} J_i D_{i-1} \cdots D_j\|_{op}) \) is \( \frac{k_{j'+1}k_{i-1-j}}{k_{j'-j}} \)-Lipschitz for all \( j' \leq i \leq j \), and therefore their product is \( \tilde{k}_{J_i} \)-Lipschitz. As the remaining indicators do not depend on \( J_i \) are constants in \([0, 1]\), it follows that \( Q \) is \( \tilde{k}_{J_i} \)-Lipschitz.
E ReLU Networks

In this section, we apply our augmentation technique to relu networks to produce a generalization bound similar to that of Nagarajan & Kolter (2019), which is polynomial in the Jacobian norms, hidden layer norms, and inverse pre-activations.

Recall the definition of neural nets in Example 5.1: the neural net with parameters \( \{W^{(i)}\} \) and activation \( \phi \) is defined by

\[
F(x) = W^{(r)} \phi(\cdots \phi(W^{(1)} x) \cdots)
\]

For this section, we will set \( \phi \) to be the relu activation. We also use the same notation for layers and indexing as Section 7.

We first state our generalization bound for relu networks:

**Theorem E.1.** Fix reference matrices \( \{A^{(i)}\}, \{B^{(i)}\} \). With probability \( 1 - \delta \) over the random draws of the data \( P_n \), all neural networks \( F \) with relu activations parameterized by \( \{W^{(i)}\} \) will have the following generalization guarantee

\[
\mathbb{E}_{(x,y) \sim P} [l_{0:1}(F(x), y)] \leq \tilde{O}\left( \left( \sum_{i} (\kappa_{relu-hidden, (i)} a^{(i)(i-1)} / 2 + (\kappa_{relu-jacobian, (i)} t^{(i)}) / 2) + r \sqrt{\log(1/\delta) \over n} \right) \right)
\]

where

\[
\begin{align*}
\kappa_{relu-jacobian, (i)} & \triangleq \sum_{1 \leq j \leq 2i-1 \leq j' \leq 2r-1} \sigma_{j' \leftarrow j} / \sigma_{2i-2j} \\
\kappa_{relu-hidden, (i)} & \triangleq \frac{1}{\text{poly}(r)} + \frac{\sigma_{2i-2j}}{\gamma} + \sum_{i \leq i' < r} \frac{\sigma_{2i'-2i}}{t(i')} + \frac{\sigma_{2i'-1-2i}}{\gamma(i')}
\end{align*}
\]

(33)

In these expressions, we define \( \sigma_{j \leftarrow j} = 1 \), \( \gamma^{(i)} \) to be the minimum pre-activation after the \( i \)-th weight matrix over all coordinates in the \( i \)-th layer and all datapoints:

\[
\gamma^{(i)} \triangleq \min_{x \in P_n} \min_{j} |[F_{2i-1-1}(x)]_j|
\]

where \( [F_{2i-1-1}(x)]_j \) indexes the \( j \)-th coordinate of \( F_{2i-1-1}(x) \), and additionally use

\[
\begin{align*}
a^{(i)} & \triangleq \text{poly}(r)^{-1} + \|W^{(i)^\top} - A^{(i)^\top}\|_{1,1}, b^{(i)} \triangleq \text{poly}(r)^{-1} + \|W^{(i)} - B^{(i)}\|_{1,1} \\
t^{(0)} & \triangleq \text{poly}(r)^{-1} + \max_{x \in P_n} \|x\|, t^{(i)} \triangleq \text{poly}(r)^{-1} + \max_{x \in P_n} \|F_{2i-1-1}(x)\| \\
\sigma_{j \leftarrow j} & \triangleq \text{poly}(r)^{-1} + \max_{x \in P_n} \|Q_{j \leftarrow j}(x)\|_{\text{op}}, \quad \gamma \triangleq \min_{(x,y) \in P_n} \|F(x)_y - \max_{y' \neq y} |F(x)|_{y'}| > 0
\end{align*}
\]

Note that we assume the existence of a positive margin, so the training error here is 0.

We note that compared to Theorem 7.1, \( \kappa_{relu-jacobian, (i)} = \kappa_{jacobian, (i)} \), but \( \kappa_{relu-hidden, (i)} \) now has a dependence on the preactivations \( \gamma^{(i)} \), as in Nagarajan & Kolter (2019).

We provide a proof sketch of Theorem E.1 here. We first bound the Rademacher complexity some family of augmented losses, specified precisely in Theorem E.2. The rest of the argument then follows the same way as the proof of Theorem 7.1 using Rademacher complexity to argue that the augmented losses generalize, applying the fact that the augmented losses upper-bound the 0-1 loss, and then union bounding over all choices of parameters.

**Theorem E.2.** Following the definitions in Theorem A.2 let \( \mathcal{F} \) denote the class of neural networks, \( \sigma_{j \leftarrow j} \) be parameters intended to bound the spectral norm of the \( j \) to \( j \) layerwise Jacobian, and \( t^{(i)} \) be parameters bounding the layer norm after applying the \( i \)-th activation. Define \( \gamma^{(i)} \) as parameters intended to lower bound the minimum preactivations after the \( i \)-th linear layer. Define the class of augmented losses

\[
\mathcal{L}_{\text{relu-aug}} \triangleq \left\{ (l_\gamma - 1) \circ F \prod_{i=1}^{r-1} \mathbb{I}_{\leq \gamma^{(i)}}(\|F_{2i-1}\|_1) \mathbb{I}_{\geq \gamma^{(i)}}(\min_j |[F_{2i-1-1}]_j|) \prod_{1 \leq j \leq 2r-1} \mathbb{I}_{\leq \sigma_{j \leftarrow j}}(\|Q_{j \leftarrow j}\|_{\text{op}}) + 1 : F \in \mathcal{F} \right\}
\]

28
where \( I_{\geq \gamma(i)} \triangleq 1 - I_{\leq \gamma(i)/2} \). Define for \( 1 \leq i \leq r \), \( \kappa_{\text{relu-jacobian},(i)}, \kappa_{\text{relu-hidden},(i)} \) meant to bound the influence of the matrix \( W^{(i)} \) on the Jacobians and hidden variables, respectively, as in \( [33] \). Then the augmented loss class \( L_{\text{relu-aug}} \) has empirical Rademacher complexity upper bound

\[
\text{Rad}_n(L_{\text{relu-aug}}) = \tilde{O} \left( \frac{(\sum_i (\kappa_{\text{relu-hidden},(i)} \alpha(i) t(i-1)) 2^{3/4} + (\kappa_{\text{relu-jacobian},(i)} \beta(i)) 2^{3/4})^{3/2}}{\sqrt{n}} \right)
\]

Note the differences with Theorem A.2: the augmented loss class now includes the additional indicators \( I_{\geq \gamma(i)} \min_j \| [F_{2i-1}^{-1}]_j \| \), and we use the Lipschitz constants \( \kappa_{\text{relu-hidden},(i)}, \kappa_{\text{relu-jacobian},(i)} \) defined in Theorem E.1.

**Proof sketch.** As in the proof of Theorem A.2, associate the loss class \( L_{\text{relu-aug}} \) with a family \( \mathcal{G} \) of computation graphs on internal nodes \( V_1, \ldots, V_{2r-1}, J_1, \ldots, J_{2r-1} \) as follows: define the graph structure to be identical to the Lipschitz augmentation of a sequential computation graph family (Figure 3) and define the composition rules

\[
\mathcal{R}_{V_1} = \{ \phi \}
\]

\[
\mathcal{R}_{V_{2i-1}} = \{ h \mapsto Wh : \| W^T - A^{(i)} \|_{2,1} \leq a^{(i)}, \| W - B^{(i)} \|_{1,1} \leq b^{(i)}, \| W \|_{\text{op}} \leq \sigma^{(i)} \}
\]

Assign to the \( J_i \) nodes composition rule \( R_{J_i} = DR_{V_1} \), and finally, assign to the output node \( O \) the composition rule

\[
R_O(x, v_1, \ldots, v_{2r-1}, D_1, \ldots, D_{2r-1}) \triangleq \left( l_1(v_{2r-1}) - 1 \right) \prod_{i=1}^{r-1} I_{\leq \kappa(i)} \left( \| v_{2i} \| I_{\geq \gamma(i)} \min_j \| [V_{2i-1}]_j \| \right) \prod_{1 \leq j \leq 2r-1} I_{\leq \sigma_j^{(i)}} \| D_j \ldots D_{J_i} \|_{\text{op}} + 1
\]

The resulting family of computation graphs will compute \( L_{\text{relu-aug}} \). Now we claim that \( \mathcal{G} \) is \( \kappa_{\text{relu-hidden},(i)} \)-release-Lipschitz in nodes \( V_{2i-1} \) and \( \kappa_{\text{relu-jacobian},(i)} \)-release-Lipschitz in nodes \( J_{2i-1} \). (Note that the Lipschitzness of nodes \( V_{2i}, J_{2i} \) will not matter because the associated function classes and singletons and therefore have a covering number of 0 anyways).

The argument for the \( \kappa_{\text{relu-jacobian},(i)} \)-release-Lipschitzness of \( J_{2i-1} \) follows analogously to the argument of Lemma D.1 and Theorem A.2.

To see the \( \kappa_{\text{relu-hidden},(i)} \)-release-Lipschitzness of \( V_{2i-1} \), we first note that we can account for the instantaneous change in the graph output given a change to \( V_{2i-1} \) as a sum of the following: 1) the change in \( l_1(V_{2i-1}) - 1 \) multiplied by the other indicators, 2) the change in the term \( I_{\leq \kappa(i)} \| v_{2i} \| I_{\geq \gamma(i)} \min_j \| [V_{2i-1}]_j \| \) multiplied by the other indicators, and 3) the change in \( I_{\leq \sigma_j^{(i)}} \| D_j \ldots D_{J_i} \|_{\text{op}} \) multiplied by the other indicators. The term 1) can be computed as \( \frac{2r-1-2i}{\gamma} \), term 2) can be accounted for by \( \frac{2r-1-2i}{\gamma} + \frac{2r-1-2i}{\gamma} \), and finally the term 3) is 0 because as relu is piecewise-linear, the instantaneous change in the Jacobian is 0 if all preactivations are bounded away from 0, and in the case that the preactivations are not bounded away from 0, the indicator \( I_{\geq \gamma(i)} \min_j \| [V_{2i-1}]_j \| \) takes value 0. The same steps as Lemma D.1 can be used to formalize this argument.

Finally, to conclude the desired Rademacher complexity bounds given the release-Lipschitzness, we apply the same reasoning as in Theorem A.2.

\[\Box\]

### F Toolbox

**Claim F.1.** Consider the function \( u : [0, 1] \times [0, 1] \rightarrow \mathbb{R} \) defined as follows: \( u(x_1, x_2) = (x_1 - 1)x_2 + 1 \). Then the following statements hold:

1. The function \( u \) outputs values in \([0, 1]\).
2. \( u(x_1, x_2) \geq x_1 \).
3. \( u(u(x_1, x_2), x_3) = u(x_1, x_2 x_3) \).

\[29\]
Proof. First, we note that \( u(x_1, x_2) = x_1 x_2 + 1 - x_2 \leq x_2 + 1 - x_2 = 1 \). Furthermore, \( u(x_1, x_2) \geq x_1 x_2 + x_1 (1 - x_2) = x_1 \), which completes the proof of statements 1 and 2. To prove the third statement, we note that
\[
\begin{aligned}
u(x_1, x_2, x_3) &= (x_1 x_2 + 1 - x_2) x_3 + 1 - x_3 = x_1 x_2 x_3 + 1 - x_2 x_3 = u(x_1, x_2 x_3).
\end{aligned}
\]

Claim F.2 (Chain rule Wikipedia contributors (2019)). The Jacobian of a composition of a sequence of functions \( f_1, \ldots, f_k \) satisfies
\[
D f_{k-1}(x) = D f_k(f_{k-1}(x)) \cdot D f_{k-1}(f_{k-2}(x)) \cdots D f_2(f_1(x)) \cdot D f_1(x)
\] (34)
where the \( \cdot \) notations are standard matrix multiplication. For simplicity, we also write in the function form:
\[
D f_{k-1}(x) = (D f_k \circ f_{k-1}) \cdot (D f_{k-1} \circ f_{k-2}) \cdots (D f_2 \circ f_1) \cdot D f_1
\] (35)

Claim F.3 (Telescoping sum). Let \( p_1, \ldots, p_m \) and \( q_1 \ldots q_m \) be two sequence of functions from \( \mathbb{R}^d \) to \( \mathbb{R} \). Then,
\[
p_1 p_2 \cdot p_m - q_1 q_2 \cdot q_m = (p_1 - q_1)p_2 \cdots p_m + q_1 (p_2 - q_2) p_3 \cdots p_m + \cdots + q_1 \cdots q_{m-1} (p_m - q_m)
\] (36)

Claim F.4 (Bounding function differences). Let \( f : \mathcal{D} \to \mathcal{D}' \), and consider the total derivative \( Df \) operator mapping \( \mathcal{D} \) to a linear operator between normed spaces \( \mathcal{D} \) to \( \mathcal{D}' \). Suppose that \( Df(x) \) is \( \kappa \)-Lipschitz in \( x \), in the sense that
\[
\|Df(x) - Df(y)\|_{op} \leq \kappa \|x - y\|
\]
where \( \| \cdot \|_{op} \) is the operator norm induced by \( \mathcal{D} \) and \( \mathcal{D}' \). Then
\[
\|f(x) - f(x + \nu)\| \leq (\|Df(x)\|_{op} + \frac{\kappa}{2} \|\nu\|) \|\nu\|
\] (37)

Furthermore,
\[
\|f(x) - f(x + \nu)\|_{\mathbb{I}_{\leq \tau_f}(\|Df(x)\|_{op})} \leq (2\tau_f + \frac{\tau_f^2}{2} \|\nu\|) \|\nu\|
\] (38)

Proof. We write \( f(x + \nu) - f(x) = \left( \int_{t=0}^1 Df[x + t\nu] dt \right) \nu \). Now we note that
\[
\| \int_{t=0}^1 Df[x + t\nu] dt \|_{op} \leq \int_{t=0}^1 \|Df[x + t\nu]\|_{op} dt \quad \text{(by triangle inequality)}
\]
\[
\leq \int_{t=0}^1 (\|Df[x]\|_{op} + t\kappa \|\nu\|) dt \quad \text{(by Lipschitzness of } Df)
\]
\[
\leq \|Df[x]\|_{op} + \frac{\kappa}{2} \|\nu\|
\] (39)

Thus,
\[
\|f(x + \nu) - f(x)\| \leq \| \int_{t=0}^1 Df[x + t\nu] dt \|_{op} \|\nu\|
\]
\[
\leq \left( \|Df[x]\|_{op} + \frac{\kappa}{2} \|\nu\| \right) \|\nu\| \quad \text{(by } 39)\]
which proves (37).

To prove (38), we consider two cases. first, if \( \|Df[x]\|_{op} > 2\tau_f \), then \( \mathbb{I}_{\leq \tau_f}(\|Df[x]\|_{op}) = 0 \) so (38) immediately holds. Otherwise, if \( \|Df[x]\|_{op} \leq 2\tau_f \), we can plug this into (37) to obtain (38), as desired. \( \square \)