

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

Colin Wei* and Tengyu Ma†

May 31, 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. For feedforward neural nets as well as RNNs, we obtain tighter Rademacher complexity bounds by considering additional data-dependent properties of the network: the norms 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 augmenting a sequence of functions to make their composition Lipschitz and then covering the augmented functions. 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 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 and Szegedy, 2015], layer normalization [Ba et al., 2016], group normalization [Wu and He, 2018], path normalization [Neyshabur et al., 2015a], dropout [Srivastava et al., 2014, Wager et al., 2013], and regularizing the variance of activations [Littwin and 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 and Szegedy, 2015, Zhang et al., 2019]. We do not have many tools for analyzing data-dependent regularization in the literature; with the exception of [Arora et al., 2018] and [Nagarajan and Kolter, 2019] (with which we compare later in more detail), existing bounds typically consider properties of the weights of the learned model but 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. In this work, we prove tighter generalization bounds by considering additional data-dependent properties of the network. Optimizing these bounds leads to data-dependent regularization techniques that empirically improve performance.

One well-understood and important data-dependent property is the training margin: Bartlett et al. [2017] show that networks with larger normalized margins have better generalization guarantees. However,

*Stanford University, email: colinwei@stanford.edu

†Stanford University, email: tengyuma@stanford.edu

neural nets are 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 properties: the hidden layer norms and interlayer Jacobian norms. Our final generalization bound (Theorem 7.1) is a polynomial in the hidden layer norms and Lipschitz constants on the training data. We give a simplified version below for expositional purposes. Let F denote a neural network with smooth activation ϕ parameterized by weight matrices $\{W^{(i)}\}_{i=1}^r$ that perfectly classifies the training data with margin $\gamma > 0$. Let t denote the maximum ℓ_2 norm of any hidden layer or training datapoint, and σ 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*

$$L_{0-1}(F) \leq \tilde{O} \left(\frac{\left(\frac{\sigma}{\gamma} + r^3 \sigma^2 \right) t \left(1 + \sum_i \|W^{(i)}\|_{2,1}^{2/3} \right)^{3/2} + r^2 \sigma \left(1 + \sum_i \|W^{(i)}\|_{1,1}^{2/3} \right)^{3/2}}{\sqrt{n}} + r \sqrt{\frac{\log(\frac{1}{\delta})}{n}} \right)$$

The notation \tilde{O} hides logarithmic factors in d, r, σ, t and the matrix norms. The $\|\cdot\|_{2,1}$ norm is formally defined in Section 3.

The degree of the dependencies on σ may look unconventional — this is mostly due to the dramatic simplification from our full Theorem 7.1, which obtains a more natural bound that considers all interlayer Jacobian norms instead of only the maximum. Our bound is polynomial in t, σ , and network depth, but independent of width. In practice, t and σ have been observed to be much smaller than the product of matrix norms [Arora et al., 2018, Nagarajan and Kolter, 2019]. We remark that our bound is not homogeneous because the smooth activations are not homogeneous and can cause a second order effect on the network outputs.

In contrast, the bounds of Neyshabur et al. [2015b], 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 Arora et al. [2018] depends on similar quantities related to noise stability but only holds for a compressed network and not the original. The bound of Nagarajan and 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 and Kolter, 2019].

In Section E, we additionally present a generalization bound for recurrent neural nets that scales polynomially in the same quantities as our bound for standard neural nets. Prior generalization bounds for RNNs either require parameter counting [Koiran and Sontag, 1997] or depend exponentially on depth [Zhang et al., 2018, Chen et al., 2019].

In Figure 1, we plot the distribution over the sum of products of Jacobian and hidden layer norms (which is the leading term of the bound in our full Theorem 7.1) for a WideResNet [Zagoruyko and 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 bottleneck in proving Theorem 1.1 is that standard tools 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

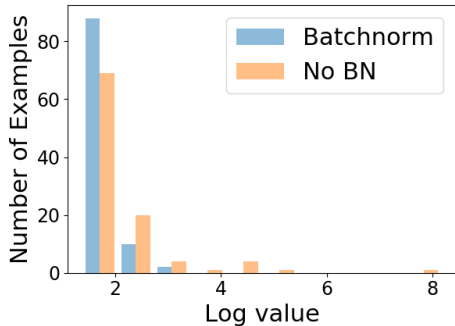


Figure 1: Let h_1, h_2, h_3 denote the 1st, 2nd, and 3rd blocks of a 16-layer WideResNet and J_i the Jacobian of the output w.r.t layer i . In log-scale we plot a histogram of the 100 largest values on the training set of $\sum_{i=1}^3 \|h_i\| \|J_i\| / \gamma$ for a WideResNet trained with and without Batchnorm on CIFAR10, where γ is the example’s margin.

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 for the training data, the training loss remains the same after augmentation. The augmentation lets us reason about a hypothesis class that is independent of the data by directly conditioning on data-dependent properties in the loss. The main challenges with this approach are twofold: 1) designing the correct set of properties and 2) proving generalization of the final loss l_{aug} , a complicated function of the network.

Our main tool 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 (without exponential dependencies on depth). However, by augmenting with properties γ , 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 a composition of functions to make it Lipschitz at all layers, and thus easy to cover. Before this augmentation, the Lipschitz constant could scale exponentially in depth (Theorem 4.4). 2) We reduce covering a complicated sequence of operations to covering the individual operations (Theorem 4.3). 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, not exponentially, in the depth of the network when the network has good Lipschitz constants on the training data (Theorem 7.1).

As a complement to the main theoretical results in this paper, we show empirically in Section 8 that directly regularizing our complexity measure can result in improved test performance.

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 and 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., 2015b, 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 and Kolter, 2019]. Dziugaite and 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 and 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 depend on the inverse relu preactivations, which are found to be large in practice [Nagarajan and 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 and Kolter [2019], by conditioning on large pre-activation values in our Lipschitz augmentation step (see Section 4.2). In Section F, 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 and 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.

3 Notation

Let $\mathbb{1}(E)$ be the indicator function of event E . Let l_{0-1} denote the standard 0-1 loss. For $\kappa \geq 0$, Let $\mathbb{1}_{\leq \kappa}(\cdot)$ be the softened indicator function defined as

$$\mathbb{1}_{\leq \kappa}(t) = \begin{cases} 1 & \text{if } t \leq \kappa \\ 2 - t/\kappa & \text{if } \kappa \leq t \leq 2\kappa \\ 0 & \text{if } 2\kappa \leq t \end{cases}$$

Note that $\mathbb{1}_{\leq \kappa}$ is κ^{-1} -Lipschitz. Define the norm $\|\cdot\|_{p,q}$ by $\|A\|_{p,q} \triangleq \left(\sum_j \left(\sum_i A_{i,j}^p \right)^{q/p} \right)^{1/q}$. Let P_n be a uniform distribution over n points $\{x_1, \dots, x_n\} \subset \mathcal{D}_x$. Let f be a function that maps \mathcal{D}_x to some output space \mathcal{D}_f , and assume both spaces are equipped with some norms $\|\cdot\|$ (these norms can be different but we use the same notations for them). Then the $L_2(P_n, \|\cdot\|)$ norm of the function f is defined as $\|f\|_{L_2(P_n, \|\cdot\|)} \triangleq \left(\frac{1}{n} \sum_i \|f(x_i)\|^2 \right)^{1/2}$. We use D to denote total derivative operator, and thus $Df(x)$ represents the Jacobian of f at x .

Suppose \mathcal{F} is a family of functions from \mathcal{D}_x to \mathcal{D}_f . Let $\mathcal{C}(\epsilon, \mathcal{F}, \rho)$ be the covering number of the function class \mathcal{F} w.r.t. metric ρ with cover size ϵ . In many cases, the covering number depends on the examples through the norms of the examples, and in this paper we only work with these cases. Thus, we let $\mathcal{N}(\epsilon, \mathcal{F}, s)$ be the maximum covering number for any possible n data points with norm not larger than s . Precisely, if we define $\mathcal{P}_{n,s}$ to be the set of all possible uniform distributions supported on n data points with norms not larger than s , then $\mathcal{N}(\epsilon, \mathcal{F}, s) \triangleq \sup_{P_n \in \mathcal{P}_{n,s}} \mathcal{C}(\epsilon, \mathcal{F}, L_2(P_n, \|\cdot\|))$. Suppose \mathcal{F} contains functions with m inputs that map from a tensor product m Euclidean space to Euclidean space, then we define $\mathcal{N}(\epsilon, \mathcal{F}, (s_1, \dots, s_m)) \triangleq \sup_{P: \forall (x_1, \dots, x_m) \in \text{supp}(P) \atop \|x_i\| \leq s_i} \mathcal{C}(\epsilon, \mathcal{F}, L_2(P))$.

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 $\mathcal{F}_1, \dots, \mathcal{F}_k$ be a sequence of families of functions (corresponding to families of single layer neural nets in the deep learning setting) and ℓ be a Lipschitz loss function taking values in $[0, 1]$. We study the compositions of ℓ and functions in \mathcal{F}_i 's:

$$\mathcal{L} \triangleq \ell \circ \mathcal{F}_k \circ \mathcal{F}_{k-1} \cdots \circ \mathcal{F}_1 = \{\ell \circ f_k \circ f_{k-1} \circ \cdots \circ f_1 : \forall i, f_i \in \mathcal{F}_i\} \quad (1)$$

Textbook results [Bartlett and Mendelson, 2002] bound the generalization error by the Rademacher complexity (formally defined in Section 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 [Dudley, 1967]. 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 ϵ 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 [Bartlett et al., 2017], 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) $f_i \circ \cdots \circ f_1(x)$ is bounded, and the composition of the rest of the functions $\ell \circ f_k \circ \cdots \circ f_{i+1}(x)$ 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 [Bartlett et al., 2017]] 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 κ_i -Lipschitz for all i .*

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

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

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 scale exponentially in the number of layers, and they are the main sources of the dependency of product of spectral/Frobenius norms of layers in [Golowich et al., 2017, Bartlett et al., 2017, Neyshabur et al., 2017a, 2015b]. 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 boundedness. 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 $z(x) \triangleq \ell(h_k(x))$ be the original loss. Our first attempt considered:

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

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 with a small constant w.r.t h_i , and thus we cannot apply Lemma 4.1.¹ 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:²

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

where $\kappa_{j \leftarrow i}$'s are user-defined parameters. For our application to neural nets, we instantiate s_i as the maximum norm of layer i and $\kappa_{j \leftarrow i}$ as the maximum norm of the Jacobian between layer j and i across the training dataset. A polynomial in κ, s can be shown to bound the worst-case Lipschitzness of the function w.r.t. the intermediate variables in the formula above.³ By our choice of κ, 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. We provide an informal overview of our augmentation procedure in Section 4.2 and formally state definitions and guarantees in Section 6. The downside of the Lipschitz augmentation 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 informally overview this extension in Section 4.1 using a minimal set of notations, and in Section 5, we give a formal presentation of these results.

Combining the Lipschitz augmentation and graphs covering results, we obtain a covering number bound of augmented loss. The theorem below is formally stated in Theorem 6.3 of Section 6.

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

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

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 ϵ_i via standard techniques, we obtain the below generalization error bound on the original loss for parameters $\tilde{\kappa}_i$ alluded to in Theorem 4.2 and formally defined in Theorem 6.2. When the training examples satisfy the augmented indicators, $\mathbb{E}_{\text{train}}[\tilde{z}] = \mathbb{E}_{\text{train}}[z]$, and because \tilde{z} bounds z from above, we have

$$\mathbb{E}_{\text{test}}[z] - \mathbb{E}_{\text{train}}[z] \leq \mathbb{E}_{\text{test}}[\tilde{z}] - \mathbb{E}_{\text{train}}[\tilde{z}] \leq \tilde{O}\left(\frac{\left(\sum_i \tilde{\kappa}_i^{2/3} s_{i-1}^{2/3}\right)^{3/2}}{\sqrt{n}} + \sqrt{\frac{\log(1/\delta)}{n}}\right) \quad (4)$$

4.1 Overview of Computational Graph Covering

To obtain the augmented \tilde{z} defined in (3), we needed to condition on data-dependent properties which introduced dependencies between the various layers. Because of this, Lemma 4.1 is no longer sufficient to

¹A priori, it's also unclear what "Lipschitz in h_i " means since the \tilde{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.

²Unlike in equation (2), 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.

³As mentioned in footnote 1, we will formalize the precise meaning of Lipschitzness later.

cover \tilde{z} . In this section, we informally overview how to extend Lemma 4.1 to cover more general functions via the notion of computational graphs. Formal definitions and theorem statements are provided in Section 5.

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. For simplicity, we assume the graph contains a unique sink, denoted by O_G , and we call it the “output node”. We also overload the notation O_G to denote the function that the computational graph G finally computes. Let $\mathcal{I}_G = \{I_1, \dots, I_p\}$ be the subset of nodes with no predecessors, which we call the “input nodes” of the graph.

The notion of a family of computational graphs generalize the sequential family of function compositions in (1). Let $\mathcal{G} = \{G(\mathcal{V}, \mathcal{E}, \{R_V\})\}$ be a family of computational graphs with shared nodes, edges, output node, and input nodes (denoted by \mathcal{I}). Let \mathfrak{R}_V the collection of all possible composition rules used for node V by the graphs in the family \mathcal{G} . This family \mathcal{G} defines a set of functions $O_G \triangleq \{O_G : G \in \mathcal{G}\}$.

The theorem below extends Lemma 4.1. In the computational graph interpretation, Lemma 4.1 applies to a sequential family of computational graphs with k internal nodes V_1, \dots, V_k , where each V_i computes the function f_i , and the output computes the composition $O_G = \ell \circ f_k \cdots \circ f_1 = z$. However, the augmented loss \tilde{z} no longer has this sequential structure, requiring the below theorem for covering generic families of computational graphs. We show that covering a general family of computational graphs can be reduced to covering all the local composition rules.

Theorem 4.3 (Informal and weaker version of Theorem 5.3). *Suppose that there is an ordering (V_1, \dots, V_m) of the nodes, so that after cutting out nodes V_1, \dots, V_{i-1} , the node V_i becomes a leaf node and the output O_G is κ_{V_i} -Lipschitz w.r.t to V_i for all $G \in \mathcal{G}$. In addition, assume that for all $G \in \mathcal{G}$, the node V 's value has norm at most s_V . Let $\text{pr}(V)$ be all the predecessors of V and $s_{\text{pr}(V)}$ be the list of norm upper bounds of the predecessors of V .*

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

$$\log \mathcal{N}\left(\sum_{V \in \mathcal{V} \setminus \mathcal{I} \cup \{O\}} \kappa_V \epsilon_V + \epsilon_O, O_G, s_{\mathcal{I}}\right) \leq \sum_{V \in \mathcal{V} \setminus \mathcal{I}} \log \mathcal{N}(\epsilon_V, \mathfrak{R}_V, s_{\text{pr}(V)}) \quad (5)$$

In Section 5 we formalize the notion of “cutting” nodes from the graph. The condition that node V 's value has norm at most s_V is a simplification made for expositional purposes; our full Theorem 5.3 also applies if O_G collapses to a constant whenever node V 's value has norm greater than s_V . This allows for the softened indicators $\mathbb{1}_{\leq s_i}(\|h_i(x)\|)$ used in (3).

4.2 Lipschitz Augmentation of Computational Graphs

The covering number bound of Theorem 4.3 relies on Lipschitzness w.r.t internal nodes of the graph under a worst-case choice of inputs. For deep networks, this can scale exponentially in depth via the product of weight norms and easily be larger than the average Lipschitz-ness over typical inputs. In this section, we explain a general operation to augment sequential graphs (such as neural nets) into graphs with better worst-case Lipschitz constants, so tools such as Theorem 4.3 can be applied. We provide formal definitions and results in Section 6.

The augmentation relies on introducing terms such as the soft indicators in equation (2) and (3) which condition on data-dependent properties. As outlined in Section 4, they will translate to the data-dependent properties in the generalization bounds. We also require the augmented function to upper bound the original.

We will present a generic approach to augment function compositions such as $z \triangleq \ell \circ f_k \circ \dots \circ f_1$, whose Lipschitz constants are potentially exponential in depth, with only properties involving the norms of the inter-layer Jacobians. We will produce \tilde{z} , whose worst-case Lipschitzness w.r.t. internal nodes can be polynomial in depth.

Informal explanation of Lipschitz augmentation: In the same setting of Section 4, recall that in (2), our first unsuccessful attempt to smooth out the function was by multiplying indicators on the norms of the

derivatives of the output: $\prod_{i=1}^k \mathbb{1}_{\leq \kappa_i}(\|\partial z / \partial h_i\|_{\text{op}})$. The difficulty lies in controlling the Lipschitzness of the new terms $\|\partial z / \partial h_i\|_{\text{op}}$ that we introduce: by the chain rule, we have the expansion $\frac{\partial z}{\partial h_i} = \frac{\partial z}{\partial h_k} \frac{\partial h_k}{\partial h_{k-1}} \dots \frac{\partial h_{i+1}}{\partial h_i}$, where each $h_{j'}$ is itself a function of h_j for $j' > j$. This means $\frac{\partial z}{\partial h_i}$ is a complicated function in the intermediate variables h_j for $1 \leq j \leq k$. Bounding the Lipschitzness of $\frac{\partial z}{\partial h_i}$ requires accounting for the Lipschitzness of every term in its expansion, which is challenging and creates complicated dependencies between variables.

Our key insight is that by considering a more complicated augmentation which conditions on the derivatives between all intermediate variables, we can still control Lipschitzness of the system, leading to the more involved augmentation presented in (3). Our main technical contribution is Theorem 4.4, which we informally state below.

Theorem 4.4 (Informal version of Theorem 6.2). *The functions \tilde{z} (defined in (3)) can be computed by a family of computational graphs $\tilde{\mathcal{G}}$ illustrated in Figure 2. This family has internal nodes V_i and J_i computing h_i and $Df_i[h_{i-1}]$, respectively, and computes a modified output rule that augments the original with soft indicators. These soft indicators condition that the norms of the Jacobians and h_i are bounded by parameters $\kappa_{j \leftarrow i}, s_i$.*

Importantly, the output $O_{\tilde{\mathcal{G}}}$ is $\tilde{\kappa}_{V_i}, \tilde{\kappa}_{J_i}$ -Lipschitz w.r.t. V_i, J_i , respectively, after cutting nodes $V_1, J_1, \dots, V_{i-1}, J_{i-1}$, for parameters $\tilde{\kappa}_{V_i}, \tilde{\kappa}_{J_i}$ that are polynomials in $\kappa_{j \leftarrow i}, s_i$.

In addition, the augmented function \tilde{z} will upper bound the original with equality when all the indicators are satisfied. The crux of the proof is leveraging the chain rule to decompose $\frac{\partial z}{\partial h_i}$ into a product and then applying a telescoping argument to bound the difference in the product by differences in individual terms. In Section 6 we present a formal version of this result and also apply Theorem 4.3 to produce a covering number bound for $\tilde{\mathcal{G}}$.

5 Covering of Computational Graphs

This section is a formal version of Section 4.1 with full definition and theorem statements. 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.

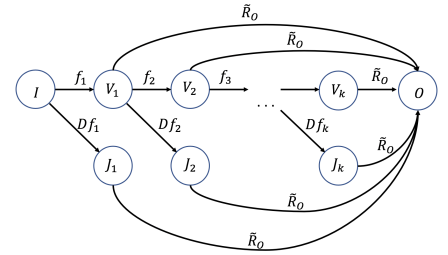
5.1 Formalization of computational graphs

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. Recall the notation $G(\mathcal{V}, \mathcal{E}, \{R_V\})$ introduced for a computational graph in Section 4.1, with input nodes $\mathcal{I}_G = \{I_1, \dots, I_p\}$ and output node denoted by O_G . (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, \dots, C_t , then the associated composition rule R_V is a function that maps $\mathcal{D}_{C_1} \otimes \dots \otimes \mathcal{D}_{C_t}$ 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

Figure 2: Lipschitz augmentation (informally defined).



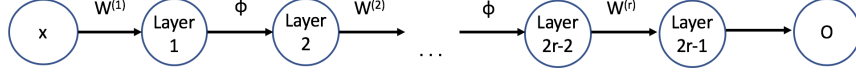


Figure 3: The computational graph corresponding to a neural network with r weight matrices. Odd-indexed layers multiply matrices and even-indexed layers apply the activation ϕ .

$\mathcal{D}_{I_1} \otimes \cdots \otimes \mathcal{D}_{I_p}$ to \mathcal{D}_V . This associated function, denoted by V again with slight abuse of notations, is defined recursively as follows: set $V(x_1, \dots, x_p)$ to

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

More succinctly, we can write $V = R_V \circ (C_1 \otimes \cdots \otimes C_t)$. We also overload the notation O_G to denote the function that the computational graph G finally computes (which maps $\mathcal{D}_{I_1} \otimes \cdots \otimes \mathcal{D}_{I_p}$ to \mathcal{D}_O). For any set $\mathcal{S} = \{V_1, \dots, V_t\} \subseteq \mathcal{V}$, use $\mathcal{D}_{\mathcal{S}}$ to denote the space $\mathcal{D}_{V_1} \otimes \cdots \otimes \mathcal{D}_{V_t}$. We use $\text{pr}(G, V)$ to denote the set of direct predecessors of V in graph G , or simply $\text{pr}(V)$ when the graph G is clear from context.

Example 5.1 (Feed-forward neural networks). *For an activation function ϕ and parameters $\{W^{(i)}\}$ we compute a neural net $F : \mathbb{R}^{d_I} \rightarrow \mathbb{R}^{d_O}$ as follows: $F(x) = W^{(r)}\phi(\cdots\phi(W^{(1)}x)\cdots)$. Figure 3 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 $\mathcal{G} = \{G(\mathcal{V}, \mathcal{E}, \{R_V\}) : \{R_V\} \in \mathfrak{R}\}$ be a family of computational graph with shared nodes and edges, where \mathfrak{R} is a collection of lists of composition rules. This family of computational graphs defines a set of functions $O_{\mathcal{G}} \triangleq \{O_G : G \in \mathcal{G}\}$. We'd like to cover this set of functions in $O_{\mathcal{G}}$ with respect to some metric $L(P_n, \|\cdot\|)$.

For a list of composition rules $\{R_V\} \in \mathfrak{R}$ and subset $\mathcal{S} \subseteq \mathcal{V}$, we define the projection of composition rules onto \mathcal{S} by $\{R_V\}_{\mathcal{S}} = \{R_V : V \in \mathcal{S}\}$. Now let $\mathfrak{R}_{\mathcal{S}} = \{\{R_V\}_{\mathcal{S}} : \{R_V\} \in \mathfrak{R}\}$ 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 $G^{\setminus V}$ 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 $G^{\setminus V}$: $\mathcal{I}_{G^{\setminus V}} = \{V\} \cup \mathcal{I}_G$. Moreover, we can “recover” the dependency by plugging the right value for V in the new graph $G^{\setminus V}$: Let $V(x)$ be the function associated to the node V in graph G , then we have

$$\forall x \in \mathcal{D}_{\mathcal{I}}, \quad O_{G^{\setminus V}}(V(x), x) = O_G(x) \quad (6)$$

In our proofs, we will release variables in orders. Let $\mathcal{S} = (V_1, \dots, V_m)$ 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_i at most depends on the input nodes and V_1, \dots, V_{i-1} . For any sequence of variables (V_1, \dots, V_t) , we can define the graph obtained by releasing the variables in order: $G^{\setminus (V_1, \dots, V_t)} \triangleq (\cdots (G^{\setminus V_1}) \cdots)^{\setminus V_t}$. 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, \dots, V_m) if the following happens: upon releasing V_1, \dots, 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 $G^{(V_1, \dots, V_i)}$ is κ_{V_i} -Lipschitz in the argument V_i , for any values of the rest of the input nodes $(=\{V_1, \dots, V_{i-1}\} \cup \mathcal{I}_G)$. 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. The below is a formal and complete version of Theorem 4.3. 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 \mathcal{G} is a computational graph with the associated family of lists of composition rules \mathfrak{R} , as formally defined above. Let P_n be a uniform distribution over n points in $\mathcal{D}_{\mathcal{I}}$. Let κ_V , s_V , and ϵ_V be three families of fixed parameters indexed by $\mathcal{V} \setminus \mathcal{I}$ (whose meanings are defined below). Assume the following:*

1. *Every $G \in \mathcal{G}$ is release-Lipschitz with parameters $\{\kappa_V\}$ w.r.t a forest ordering of the internal nodes (V_1, \dots, V_m) (the parameter κ_V 's and ordering doesn't depend on the choice of G .)*
2. *For the same order as before, if $(v, x) \in (\mathcal{D}_{V_1} \otimes \dots \otimes \mathcal{D}_{V_i}) \otimes \mathcal{D}_{\mathcal{I}}$ is an input of the released graph satisfying $\|v_j\| \geq s_{V_j}$ for some $j \leq i$, then $O_{G^{(V_1, \dots, V_i)}}(v, x) = c$ for some constant c .*

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

$$\log \mathcal{N}\left(\sum_{V \in \mathcal{V} \setminus \mathcal{I} \cup \{O\}} \kappa_V \epsilon_V + \epsilon_O, O_{\mathcal{G}}, s_{\mathcal{I}}\right) \leq \sum_{V \in \mathcal{V} \setminus \mathcal{I}} \log \mathcal{N}(\epsilon_V, \mathfrak{R}_{\{V\}}, s_{\text{pr}(V)}) \quad (7)$$

6 Lipschitz Augmentation of Computational Graphs

In this section, we provide a more thorough and formal presentation of the augmentation framework of Section 4.2.

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 condition⁴ 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 6.1, we generalize the operation of (3) 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 $\mathcal{V} = \{I, V_1, \dots, V_q, O\}$, where I is the single input node, and all the edges are $\mathcal{E} = \{(I, V_1), (V_1, V_2), \dots, (V_{q-1}, V_q)\} \cup \{(V_1, O), \dots, (V_q, O)\}$. We often use the notation V_0 to refer to the input I . 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, \dots, V_q , define its Lipschitz augmentation \tilde{G} as follows. We first add q nodes*

⁴We say the Lipschitzness required is worst case because the release-Lipschitz condition requires the Lipschitzness of nodes for any possible choice of inputs

to the graph denoted by J_1, \dots, 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 $\mathcal{D}_{V_{i-1}}$ to \mathcal{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 $\|\cdot\|_{\text{op}}$, 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 $\kappa_{j \leftarrow i}$ 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 the Jacobians is much larger than $\kappa_{i \leftarrow j}$:

$$\tilde{R}_O(x, v_1, \dots, v_q, D_1, \dots, D_q) \triangleq (R_O(x, v_1, \dots, v_q) - 1) \prod_{i \leq j} \mathbb{1}_{\leq \kappa_{j \leftarrow i}}(\|D_j \cdots D_i\|_{\text{op}}) + 1$$

where $x \in \mathcal{D}_{\mathcal{I}}$, $v_i \in \mathcal{D}_{V_i}$, 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 $\kappa_{j \leftarrow i}$ has the interpretation as an intended bound of the Jacobian between pairs of layers (variables). Figure 4 depicts the augmentation.

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

$$O_{\tilde{G}}(x) = (O_G(x) - 1) \prod_{i \leq j} \mathbb{1}_{\leq \kappa_{j \leftarrow i}}(\|DV_j(x) \cdots DV_i(x)\|_{\text{op}}) + 1 \quad (8)$$

which matches (3) for the example in Section 4. We note that the graph \tilde{G} contains the original G as a subgraph. Furthermore, by Claim H.1, $O_{\tilde{G}}$ upper bounds O_G , which is desirable when G computes loss functions. The below theorem, which formalizes Theorem 4.4, proves release-Lipschitzness for \tilde{G} .

Theorem 6.2. [Lipschitz guarantees of augmented graphs] Let \mathcal{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 DR_{V_i} is $\bar{\kappa}_i$ -Lipschitz for each i .⁵ Let $\kappa_{j \leftarrow i}$ (for $i \leq j$) be a set of parameters that we intend to use to control Jacobians in the Lipschitz augmentation. With them, we apply Lipschitz augmentation as defined in Definition 6.1 to every graph in \mathcal{G} and obtain a new family of graphs, denoted by $\tilde{\mathcal{G}}$.

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

$$\begin{aligned} \tilde{\kappa}_{V_i} &\triangleq \sum_{i \leq j \leq q} 3c_j \kappa_{j \leftarrow i+1} + 18 \sum_{1 \leq j \leq j' \leq q} \sum_{i' = \max\{i+1, j\}}^{j'} \frac{\bar{\kappa}_{i'} \kappa_{j' \leftarrow i'+1} \kappa_{i'-1 \leftarrow i+1} \kappa_{i'-1 \leftarrow j}}{\kappa_{j' \leftarrow j}}, \\ \tilde{\kappa}_{J_i} &\triangleq \sum_{j \leq i \leq j'} \frac{4\kappa_{j' \leftarrow i+1} \kappa_{i-1 \leftarrow j}}{\kappa_{j' \leftarrow j}} \end{aligned}$$

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

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 (9) 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. In Section E, we also use these tools to derive Rademacher complexity bounds for RNNs.

⁵Note that DR_{V_i} maps a vector in space $\mathcal{D}_{V_{i-1}}$ to an linear operator that maps $\mathcal{D}_{V_{i-1}}$ to \mathcal{D}_{V_i} .

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 $\tilde{\mathcal{G}}$ of computational graphs which output

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

The family $\tilde{\mathcal{G}}$ satisfies the following guarantees:

1. Each computational graph in $\tilde{\mathcal{G}}$ upper bounds its counterpart in \mathcal{G} , i.e. $O_{\tilde{\mathcal{G}}}(x) \geq O_G(x)$.
2. Define $\tilde{\kappa}'_{V_i} \triangleq \tilde{\kappa}_{V_i} + \sum_{i \leq j \leq q} s_{V_j}^{-1} \cdot \kappa_{j \leftarrow i+1}$ and $\tilde{\kappa}'_{J_i} = \tilde{\kappa}_{J_i}$ where $\tilde{\kappa}_{V_i}, \tilde{\kappa}_{J_i}$ are defined as in Theorem 6.2. Then for any node-wise errors $\{\epsilon_V\}$,

$$\begin{aligned} & \log \mathcal{N}(\sum_{i \geq 1} \tilde{\kappa}'_{V_i} \epsilon_{V_i} + \tilde{\kappa}_{J_i} \epsilon_{J_i} + \epsilon_O, O_{\tilde{\mathcal{G}}}, s_{\mathcal{I}}) \\ & \leq \sum_{i \geq 1} \log \mathcal{N}(\epsilon_{V_i}, \mathfrak{R}_{V_i}, 2s_{V_{i-1}}) + \log \mathcal{N}(\epsilon_{J_i}, D\mathfrak{R}_{V_i}, 2s_{V_{i-1}}) + \log \mathcal{N}(\epsilon_O, \mathfrak{R}_O, \{2s_{V_j}\}_{j=1}^q \cup \{I\}) \end{aligned} \quad (9)$$

where $D\mathfrak{R}_{V_i}$ denotes the family of total derivatives of functions in \mathfrak{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, which was obtained using machinery from Section 4.1. Define a neural network F parameterized by r weight matrices $\{W^{(i)}\}$ by $F(x) = W^{(r)} \phi(\cdots \phi(W^{(1)}(x)) \cdots)$. We use the convention that activations and matrix multiplications are treated as distinct layers indexed with a subscript, with odd layers applying a matrix multiplication and even layers applying ϕ (see Example 5.1 for a visualization). Additional notation details and the proof are in Section A.

The below result follows from modeling the neural net loss as a sequential computational graph and using our augmentation procedure to make it Lipschitz in its nodes with parameters $\kappa^{\text{hidden},(i)}, \kappa^{\text{jacobian},(i)}$. Then we cover the augmented loss to bound its Rademacher complexity.

Theorem 7.1. Assume that the activation ϕ is 1-Lipschitz with a $\bar{\sigma}_\phi$ -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 γ satisfy:

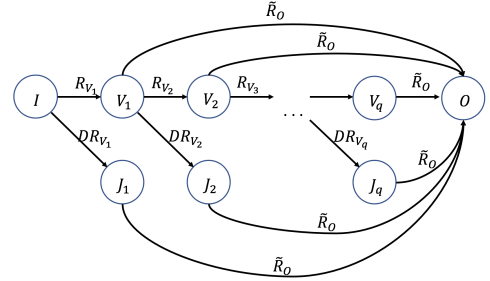
$$\mathbb{E}_{(x,y) \sim P} [l_{0-1}(F(x), y)] \leq \tilde{O} \left(\frac{\left(\sum_i (\kappa^{\text{hidden},(i)} a^{(i)} t^{(i-1)})^{2/3} + (\kappa^{\text{jacobian},(i)} b^{(i)})^{2/3} \right)^{3/2}}{\sqrt{n}} + r \sqrt{\frac{\log(1/\delta)}{n}} \right)$$

where $\kappa^{\text{jacobian},(i)} \triangleq \sum_{1 \leq j \leq 2i-1 \leq j' \leq 2r-1} \frac{\sigma_{j' \leftarrow 2i} \sigma_{2i-2 \leftarrow j}}{\sigma_{j' \leftarrow j}}$, and $\kappa^{\text{hidden},(i)} \triangleq \xi + \frac{\sigma_{2r-1 \leftarrow 2i}}{\gamma} + \sum_{i \leq i' < r} \frac{\sigma_{2i' \leftarrow 2i}}{t^{(i')}} + \sum_{1 \leq j \leq j' \leq 2r-1} \sum_{\substack{j'' = \max\{2i, j\} \\ j'' \text{ even}}}^{j'} \frac{\bar{\sigma}_\phi \sigma_{j' \leftarrow j''+1} \sigma_{j''-1 \leftarrow 2i} \sigma_{j''-1 \leftarrow j}}{\sigma_{j' \leftarrow j}}$.

In these expressions, we define $\sigma_{j-1 \leftarrow j} = 1$, $\xi = \text{poly}(r)^{-1}$, and:

$$a^{(i)} \triangleq \|W^{(i)\top} - A^{(i)\top}\|_{2,1} + \xi, b^{(i)} \triangleq \|W^{(i)} - B^{(i)}\|_{1,1} + \xi$$

Figure 4: Lipschitz augmentation (formally defined).



$$t^{(0)} \triangleq \max_{x \in P_n} \|x\| + \xi, \quad t^{(i)} \triangleq \max_{x \in P_n} \|F_{2i \leftarrow 1}(x)\| + \xi$$

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

where $Q_{j' \leftarrow j}$ computes the Jacobian of layer j' w.r.t. layer j . Note that the training error here is 0 because of the existence of positive margin γ .

The reference matrices $\{A^{(i)}\}, \{B^{(i)}\}$ are useful if there is some prior belief before training about what weight matrices are learned and also appear in the bounds of Bartlett et al. [2017]. In Section E, we also show that our techniques can easily be extended to provide generalization bounds for RNNs scaling polynomially in depth via the same quantities $t^{(i)}, \sigma_{j' \leftarrow j}$.

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 of the classification margin 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 Sokolić et al. [2017] show positive experimental results for a similar regularization technique in data-limited settings.

Suppose that $m(F(x), y) = [F(x)]_y - \max_{j \neq y} [t]_j$ denotes the margin of the network for example (x, y) . Letting $h^{(i)}$ denote some hidden layer of the network, we define the notation

$$J^{(i)} \triangleq \frac{\partial}{\partial h^{(i)}} m(F(x), y)$$

Then our training objective is

$$\hat{L}_{\text{reg}}[F] \triangleq \mathbb{E}_{(x,y) \sim P_n} \left[l(x, y) + \lambda \left(\sum_i \mathbb{1}(\|J^{(i)}(x)\|_F^2 \geq \sigma) \|J^{(i)}(x)\|_F^2 \right) \right]$$

where l denotes the standard cross entropy loss, and λ, σ are hyperparameters. Note the Jacobian is taken with respect to a scalar output and therefore is a vector, so it is easy to compute.

For a WideResNet16 [Zagoruyko and Komodakis, 2016] architecture, we train using the above objective. The threshold on the Frobenius norm in the regularization is inspired by the truncations in our augmented loss (in all our experiments, we choose $\sigma = 0.1$). We tune the coefficient of this cost as a hyperparameter. In our experiments, we took the regularized indices i to be all the layers following a BatchNorm in the standard WideResNet16 architecture. In the LayerNorm setting, we simply replaced BatchNorm layers with LayerNorm. The remaining hyperparameter settings are standard for WideResNet; additional details are provided in Section G.

Figure 1 shows the results for models trained and tested on CIFAR10 in low learning rate and no data augmentation settings, which are settings where generalization typically suffers. 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 norms and norms of the interlayer

Table 1: Test error for a model trained on CIFAR10 in various settings.

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 [Ba et al., 2016]	×	6.20%
		✓	5.57%

Jacobians. To prove 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 to arbitrary computational graph structures. Additionally, encouraged by our promising preliminary results, we believe there is the exciting empirical direction of applying these bounds to develop better data-dependent regularization.

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

- Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018.
- Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- Peter L Bartlett and Shahr 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*, pages 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.
- Minshuo Chen, Xingguo Li, and Tuo Zhao. On generalization bounds of a family of recurrent neural networks, 2019. URL <https://openreview.net/forum?id=Skf-oo0qt7>.
- 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.
- 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*, pages 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*, pages 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.
- 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*, pages 971–980, 2017.
- Pascal Koiran and Eduardo D Sontag. Vapnik-chervonenkis dimension of recurrent neural networks. In *European Conference on Computational Learning Theory*, pages 223–237. Springer, 1997.
- 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*, pages 2–47, 2018.
- Etai Littwin and Lior Wolf. Regularizing by the variance of the activations’ sample-variances. In *Advances in Neural Information Processing Systems*, pages 2115–2125, 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, Ruslan Salakhutdinov, and Nathan Srebro. Data-dependent path normalization in neural networks. *arXiv preprint arXiv:1511.06747*, 2015a.
- Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pages 1376–1401, 2015b.

- 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*, pages 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*, pages 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.
- Chiyuan 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>.
- Jiong Zhang, Qi Lei, and Inderjit S Dhillon. Stabilizing gradients for deep neural networks via efficient svd parameterization. *arXiv preprint arXiv:1803.09327*, 2018.

A Missing Proofs for Section 7

We first elaborate more on the notations introduced in Section 7. First, by our indexing, matrix $W^{(i)}$ will be applied in layer $2i - 1$ of the network, and even layers $2i$ apply ϕ . We let $F_{j' \leftarrow j}$ denote the function computed between layers j and j' and $Q_{j' \leftarrow j} = DF_{j' \leftarrow j} \circ F_{j'-1 \leftarrow j-1}$ denote the layer j -to- j' Jacobian. By our definition of $F_{j' \leftarrow j}$, $F_{2j \leftarrow 2j} = \phi$, $F_{2j-1 \leftarrow 2j-1} = h \mapsto W^{(j)}h$, and $F_{j' \leftarrow j}$ is recursively computed by $F_{j' \leftarrow j'} \circ F_{j'-1 \leftarrow j}$ for $j' > j$. We will use the convention that $F_{j-1 \leftarrow j}$ computes the identity mapping for $i \leq j$.

P will denote a test distribution over examples x and labels y , and P_n will denote the distribution on training examples.

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

$$\text{Rad}_n(\mathcal{L}) = \frac{1}{n} \mathbb{E}_{\alpha_i} \left[\sup_{l \in \mathcal{L}} \sum_i \alpha_i l(x_i) \right] \quad (10)$$

where α_i are independent uniform ± 1 random variables. Let $m(t, y) \triangleq [t]_y - \max_{j \neq y} [t]_j$ denote the margin operator for label y , and $l_\gamma(t, y) \triangleq \mathbb{1}(m(t, y) \leq 0) - \mathbb{1}(0 < m(t, y) \leq \gamma) \cdot m(t, y)/\gamma$ denote the standard ramp loss, which is $1/\gamma$ -Lipschitz. 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 ϕ is 1-Lipschitz with $\bar{\sigma}_\phi$ -Lipschitz derivative. Fix parameters $\sigma_{j' \leftarrow j}$, $t^{(i)}$, $a^{(i)}$, $b^{(i)}$, γ 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:* $\max_{x \in P_n} \|F_{2i \leftarrow 1}(x)\| \leq t^{(i)} \quad \forall 1 \leq i \leq r$.
2. *Jacobians are balanced:* $\max_{x \in P_n} \|Q_{j' \leftarrow j}(x)\|_{\text{op}} \leq \sigma_{j' \leftarrow j} \quad \forall j < j'$.
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

$$\|W^{(i)\top} - A^{(i)\top}\|_{2,1} \leq a^{(i)}, \|W^{(i)} - B^{(i)}\|_{1,1} \leq b^{(i)}, \|W^{(i)}\|_{\text{op}} \leq \sigma_{2i-1 \leftarrow 2i-1}$$

will have the following generalization to test data:

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

where

$$\kappa^{\text{jacobian}, (i)} \triangleq \sum_{1 \leq j \leq 2i-1 \leq j' \leq 2r-1} \frac{4\sigma_{j' \leftarrow 2i} \sigma_{2i-2 \leftarrow j}}{\sigma_{j' \leftarrow j}} \quad (11)$$

$$\begin{aligned} \kappa^{\text{hidden}, (i)} &\triangleq \frac{\sigma_{2r-1 \leftarrow 2i}}{\gamma} + \sum_{i \leq i' < r} \frac{3\sigma_{2i' \leftarrow 2i}}{t^{(i')}} \\ &+ \sum_{1 \leq j \leq j' \leq 2r-1} \sum_{\substack{j'' = \max\{2i, j\}, \\ j'' \text{ even}}}^{j'} \frac{\bar{\sigma}_\phi \sigma_{j' \leftarrow j''+1} \sigma_{j''-1 \leftarrow 2i} \sigma_{j''-1 \leftarrow j}}{\sigma_{j' \leftarrow j}} \end{aligned} \quad (12)$$

Here we use the convention that $\sigma_{j-1 \leftarrow j} = 1$ 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 ϕ 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 $\{A^{(i)}\}, \{B^{(i)}\}$:*

$$\mathcal{F} \triangleq \left\{ x \mapsto F(x) : \|W^{(i)\top} - A^{(i)\top}\|_{2,1} \leq a^{(i)}, \|W^{(i)} - B^{(i)}\|_{1,1} \leq b^{(i)}, \|W^{(i)}\|_{\text{op}} \leq \sigma^{(i)} \right\}$$

and let $\sigma_{j' \leftarrow j}$ be parameters that will bound the j to j' layerwise Jacobian for $j' \geq j$, where we set $\sigma_{2i \leftarrow 2i} = 1$ and $\sigma_{2i-1 \leftarrow 2i-1} = \sigma^{(i)}$. Let $t^{(i)}$ be parameters bounding the layer norm after applying the i -th activation. (In particular, $t^{(0)}$ bounds $\max_{x \in P_n} \|x\|$.) Define the class of augmented losses

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

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 (11), (12). Then we can bound the empirical Rademacher complexity of the augmented loss class by

$$\text{Rad}_n(\mathcal{L}_{\text{aug}}) = \tilde{O} \left(\frac{(\sum_i (\kappa^{\text{hidden},(i)} a^{(i)} t^{(i-1)})^{2/3} + (\kappa^{\text{jacobian},(i)} b^{(i)})^{2/3})^{3/2}}{\sqrt{n}} \right)$$

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} , $\mathfrak{R}_{V_{2i}} = \{\phi\}$, the set with only one element: the activation ϕ . We also let $\mathfrak{R}_{V_{2i-1}} = \{h \mapsto Wh : \|W^\top - A^{(i)\top}\|_{2,1} \leq a^{(i)}, \|W - B^{(i)}\|_{1,1} \leq b^{(i)}, \|W\|_{\text{op}} \leq \sigma^{(i)}\}$. Finally, we choose \mathfrak{R}_O to be the singleton class $\{l_\gamma\}$. Our collection of computation rules is then simply $\mathfrak{R} = \mathfrak{R}_{V_1} \otimes \cdots \otimes \mathfrak{R}_{V_{2r-1}} \otimes \mathfrak{R}_O$. Since $O_{\mathcal{G}}$ takes values in $[0, 1]$, we can apply Theorem 6.3 on this class \mathcal{G} using $s_{\mathcal{I}} = \max_{x \in P_n} \|x\|$, $s_{V_{2i}} = t^{(i)}$, $s_{V_{2i-1}} = \infty$, $\kappa_{2i \leftarrow 2i} = 1$, $\kappa_{2i-1 \leftarrow 2i-1} = \sigma^{(i)}$, and $\kappa_{j' \leftarrow j} = \sigma_{j' \leftarrow j}$ for $j' > j$. Furthermore, we note that $\bar{\kappa}_{2i} = \bar{\sigma}_\phi$, and $\bar{\kappa}_{2i-1} = 0$ as the Jacobian is constant for matrix multiplications. We thus obtain the class $\tilde{\mathcal{G}}$ 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 $s_{V_{2i-1}}, \kappa_{i \leftarrow i}$, we get that

$$\begin{aligned} \mathbb{1}_{\leq \kappa_{2i \leftarrow 2i}}(\|J_{2i}(x)\|_{\text{op}}) &= \mathbb{1}_{\leq 1}(\|D\phi \circ V_{2i-1}(x)\|_{\text{op}}) = 1 & (\text{as } |\phi'| \leq 1) \\ \mathbb{1}_{\leq \kappa_{2i-1 \leftarrow 2i-1}}(\|J_{2i-1}(x)\|_{\text{op}}) &= \mathbb{1}_{\leq \sigma^{(i)}}(\|W^{(i)} \circ V_{2i-2}(x)\|_{\text{op}}) = 1 & (\text{as } W^{(i)} \leq \sigma^{(i)}) \\ \mathbb{1}_{\leq s_{V_{2i-1}}}(\|V_{2i-1}(x)\|) &= \mathbb{1}_{\leq \infty}(\|V_{2i-1}(x)\|) = 1 \end{aligned}$$

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

$$\begin{aligned} \log \mathcal{N} \left(\sum_{i \geq 1} (\tilde{\kappa}_{V_i} + \tilde{\kappa}_{J_i}) \epsilon_V + \epsilon_O, O_{\tilde{\mathcal{G}}}, s_{\mathcal{I}} \right) &\leq \\ \sum_{i \geq 1} \log \mathcal{N}(\epsilon_{V_i}, \mathfrak{R}_{V_i}, 2s_{V_{i-1}}) + \log \mathcal{N}(\epsilon_{J_i}, D\mathfrak{R}_{V_i}, 2s_{V_{i-1}}) + \log \mathcal{N}(\epsilon_O, \mathfrak{R}_O, \{2s_{V_i}\}_{i \geq 0}) & \quad (13) \end{aligned}$$

where $\tilde{\kappa}_{V_i}, \tilde{\kappa}_{J_i}$ are defined in the statement of Theorem 6.3. After plugging in our values for $\bar{\kappa}_j, s_{V_j}, \kappa_{j' \leftarrow j}$ 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

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

We first note that the last term in (13) is simply 0 because there is exactly one output function in \mathfrak{R}_O . Now for the other terms of (13): by definition $\mathfrak{R}_{V_{2i}}, \mathfrak{R}_{J_{2i}}$ consist of a singleton set and therefore have log cover size 0 for any error resolution ϵ . Otherwise, to cover $\mathfrak{R}_{V_{2i-1}}$ it suffices to bound $\log \mathcal{N}(\epsilon_{V_{2i-1}}, \{h \mapsto Wh : \|W^\top - A^{(i)\top}\|_{2,1} \leq a^{(i)}, 2t^{(i-1)}\})$. Thus, we can apply Lemma A.3 to obtain

$$\log \mathcal{N}(\epsilon_{V_{2i-1}}, \mathfrak{R}_{V_{2i-1}}, 2s_{V_{2i-2}}) \leq \tilde{O} \left(\frac{(a^{(i)}t^{(i-1)})^2}{\epsilon_{V_{2i-1}}^2} \right)$$

Now to cover $D\mathfrak{R}_{V_{2i-1}}$, it suffices to cover $\{W : \|W - B^{(i)}\|_{1,1} \leq b^{(i)}\}$. The ϵ -covering number of a d_h^2 -dimensional ℓ_1 -ball with radius b w.r.t. ℓ_2 norm is $O(\frac{b^2}{\epsilon^2} \log d_h)$. 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

$$\begin{aligned} \beta^\star &\triangleq \left(\sum_i (\tilde{\kappa}_{V_{2i-1}} a^{(i)} t^{(i-1)})^{2/3} + (\tilde{\kappa}_{J_{2i-1}} b^{(i)})^{2/3} \right)^{3/2} \\ &= \left(\sum_i (\kappa^{\text{hidden},(i)} a^{(i)} t^{(i-1)})^{2/3} + (\kappa^{\text{jacobian},(i)} b^{(i)})^{2/3} \right)^{3/2} \end{aligned}$$

Now for a fixed error parameter ϵ , we set $\epsilon_O = 0, \epsilon_{V_{2i}} = 0, \epsilon_{J_{2i}} = 0$ (as the log cover size is 0 anyways), and $\epsilon_{V_{2i-1}} = \epsilon \frac{\tilde{\kappa}_{V_{2i-1}}^{-1/3} (a^{(i)} t^{(i-1)})^{2/3}}{(\beta^\star)^{2/3}}, \epsilon_{J_{2i-1}} = \epsilon \frac{\tilde{\kappa}_{J_{2i-1}}^{-1/3} (b^{(i)})^{2/3}}{(\beta^\star)^{2/3}}$. Now it follows that $\sum_j \epsilon_{V_j} \tilde{\kappa}_{V_i} + \epsilon_{J_j} \tilde{\kappa}_{J_j} = \epsilon$. Furthermore, under these choices of $\epsilon_{V_i}, \epsilon_{J_i}$, we end up with

$$\begin{aligned} &\sum_{i \geq 1} \log \mathcal{N}(\epsilon_{V_i}, \mathfrak{R}_{V_i}, 2s_{V_{i-1}}) + \log \mathcal{N}(\epsilon_{J_i}, D\mathfrak{R}_{V_i}, 2s_{V_{i-1}}) \\ &\leq \tilde{O} \left(\frac{1}{\epsilon^2} (\beta^\star)^{4/3} \left(\sum_i (\kappa^{\text{hidden},(i)} a^{(i)} t^{(i-1)})^{2/3} + (\kappa^{\text{jacobian},(i)} b^{(i)})^{2/3} \right)^{3/2} \right) = \tilde{O}(\epsilon^{-2} (\beta^\star)^2) \end{aligned}$$

Thus, substituting terms into (13) and collecting sums, we obtain that

$$\log \mathcal{N}(\epsilon, O_{\tilde{\mathcal{G}}}, s_{\mathcal{I}}) \leq \tilde{O}(\epsilon^{-2} (\beta^\star)^2)$$

Now we apply Dudley's entropy theorem to obtain that

$$\text{Rad}_n(\tilde{\mathcal{G}}) = \tilde{O} \left(\frac{(\sum_i (\kappa^{\text{hidden},(i)} a^{(i)} t^{(i-1)})^{2/3} + (\kappa^{\text{jacobian},(i)} b^{(i)})^{2/3})^{3/2}}{\sqrt{n}} \right)$$

□

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}_{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 $l_{0-1}(F(x), y) \leq l_{\gamma}(F(x), y) \leq l_{\text{aug}}(F, x, y)$ for any datapoint (x, y) . We used the fact that margin loss upper bounds 0-1 loss, and l_{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

$$\mathbb{E}_{(x,y) \sim \mathcal{D}} [l_{0-1}(F(x), y)] \leq \mathbb{E}_{(x,y) \sim \mathcal{D}} [l_{\text{aug}}(F, x, y)] \quad (14)$$

$$\leq \mathbb{E}_{(x,y) \sim \mathcal{D}_n} [l_{\text{aug}}(F, x, y)] + \text{Rad}_n(\mathcal{L}_{\text{aug}}) + \sqrt{\frac{\log(1/\delta)}{n}} \quad (15)$$

$$= \text{Rad}_n(\mathcal{L}_{\text{aug}}) + \sqrt{\frac{\log(1/\delta)}{n}} \quad (\text{by the data-dependent conditions})$$

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 $\sigma_{j' \leftarrow j}, t^{(i)}, a^{(i)}, b^{(i)}$.

Proof of Theorems 7.1 and 1.1. We will apply Theorem A.1 repeatedly over a grid of parameter choices $t^{(i)}, \sigma_{j' \leftarrow j}, a^{(i)}, b^{(i)}$ (following a technique of Bartlett et al. [2017]). For a collection \mathcal{M} of nonnegative integers $m_t^{(i)}, m_{\sigma}^{(j' \leftarrow j)}, m_a^{(i)}, m_b^{(i)}, m_{\gamma}$, we apply Theorem A.1 choosing $t^{(i)} = \text{poly}(r)^{-1} 2^{m_t^{(i)}}$, $\sigma_{j' \leftarrow j} = \text{poly}(r)^{-1} 2^{m_{\sigma}^{(j' \leftarrow j)}}$, $a^{(i)} = \text{poly}(r)^{-1} 2^{m_a^{(i)}}$, $b^{(i)} = \text{poly}(r)^{-1} 2^{m_b^{(i)}}$, $\gamma = 2^{-m_{\gamma}} \text{poly}(r) \max_i \sigma_{2r-1 \leftarrow 2i}$ and using error probability $\delta_{\mathcal{M}} \triangleq \frac{\delta}{2^{\sum_{m \in \mathcal{M}} m+1}}$. First, we note that by union bound, using the fact that $\sum_{\text{choices of } \mathcal{M}} \frac{\delta}{2^{\sum_{m \in \mathcal{M}} m+1}} = \delta$ 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 $\hat{t}^{(i)}, \hat{\sigma}_{j' \leftarrow j}, \hat{a}^{(i)}, \hat{b}^{(i)}, \hat{\gamma}$ and

$$\begin{aligned} \|W^{(i)\top} - A^{(i)\top}\|_{2,1} &\leq \hat{a}^{(i)} = \text{poly}(r)^{-1} 2^{m_a^{(i)}} \leq \text{poly}(r)^{-1} + 2\|W^{(i)\top} - A^{(i)\top}\|_{2,1} \\ \|W^{(i)} - B^{(i)}\|_{1,1} &\leq \hat{b}^{(i)} = \text{poly}(r)^{-1} 2^{m_b^{(i)}} \leq \text{poly}(r)^{-1} + 2\|W^{(i)} - B^{(i)}\|_{1,1} \\ \max_{x \in P_n} \|F_{2i \leftarrow 1}(x)\| &\leq \hat{t}^{(i)} = \text{poly}(r)^{-1} 2^{m_t^{(i)}} \leq \text{poly}(r)^{-1} + 2 \max_{x \in P_n} \|F_{2i \leftarrow 1}\| \\ \max_{x \in P_n} \|Q_{j' \leftarrow j}(x)\|_{\text{op}} &\leq \hat{\sigma}_{j' \leftarrow j} = \text{poly}(r)^{-1} 2^{m_{\sigma}^{(j' \leftarrow j)}} \leq \text{poly}(r)^{-1} + 2 \max_{x \in P_n} \|Q_{j' \leftarrow j}\|_{\text{op}} \end{aligned}$$

Furthermore, using γ to denote the true margin of the network, we also have $\hat{\gamma} \leq \gamma$ and $\frac{\hat{\sigma}_{2r-1 \leftarrow 2i}}{\hat{\gamma}} \leq 4 \frac{\max_{x \in P_n} \|Q_{2r-1 \leftarrow 2i}(x)\|_{\text{op}}}{\gamma} + \frac{1}{\text{poly}(r)}$. Furthermore, note that the cost we pay in $\sqrt{\frac{\log(1/\delta_{\mathcal{M}})}{n}}$ is $\tilde{O}\left(r \sqrt{\frac{\log(1/\delta)}{n}}\right)$,

where \tilde{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 σ, t and the matrix norms. \square

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:*

$$\text{Rad}_n(\mathcal{L}) \leq \inf_{\alpha > 0} \left(\alpha + \int_{\alpha}^{\infty} \sqrt{\frac{\log \mathcal{N}(\epsilon, \mathcal{L}, s)}{n}} \right)$$

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

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

Proof. By Lemma 3.2 of Bartlett et al. [2017], we can construct cover $\widehat{\mathcal{U}}$ for the class $\{h \mapsto (U - A)h : U \in \mathbb{R}^{d_1 \times d_2}, \|U^\top - A^\top\|_{2,1} \leq a\}$ with the given cover size (Note that in our definition of empirical covering number, the resolution ϵ is scaled by factor $\frac{1}{n}$ versus theirs). To cover \mathcal{U} with the same cardinality set, we simply shift all functions in $\widehat{\mathcal{U}}$ by A . \square

B Missing Proofs in Section 5

We first state the proof of Theorem 5.3.

Proof of Theorem 5.3. We prove the theorem by induction on the number of non-input vertices in the vertex set \mathcal{V} . The statement is true if O is the only non-input node in the graph: to cover the graph output with error ϵ_O , we simply cover \mathfrak{R}_O .

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 $|\mathcal{I}|$ 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, \dots, V_m) assumed in the theorem. Suppose V_1 depends on C_1, \dots, 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 $\mathcal{G}^{\setminus V_1} = \{G^{\setminus V_1} : G \in \mathcal{G}\}$ with a smaller number of edges than that of \mathcal{G} .

Define $u(h, x) \triangleq O_{G^{\setminus V_1}}(h, x)$ for $h \in \mathcal{D}_{V_1}$ and $x \in \mathcal{D}_{\mathcal{I}}$, and $w(x) = V_1(x)$. Then we can check that $u(w(x), x) = O_G(x)$. Let $\mathcal{U} = \{O_{G^{\setminus V_1}} : G \in \mathcal{G}\}$, and let $\mathcal{W} = \mathfrak{R}_{V_1}$. As each function in \mathcal{U} is κ_{V_1} -Lipschitz in V_1 because of condition 1, and it equals the fixed constant c if $\|V_1\| \geq s_V$ or $\|C_i\| \geq s_{C_i}$, we have \mathcal{U}, \mathcal{W} satisfies the conditions of the composition lemma (see Lemma B.1). With the lemma, we conclude:

$$\log \mathcal{N}(\kappa_{V_1} \epsilon_{V_1} + \epsilon_u, \mathcal{G}, s_{\mathcal{I}}) \leq \log \mathcal{N}(\epsilon_u, \mathcal{U}, (s_{V_1}, s_{\mathcal{I}})) + \log \mathcal{N}(\epsilon_{V_1}, \mathfrak{R}_{V_1}, s_{\text{pr}(V_1)}) \quad (16)$$

Note that by the definition of forest ordering, we have that (V_2, \dots, V_m) is a forest ordering of $G^{\setminus V_1}$ and by the assumption 1 of the theorem, we have that (V_2, \dots, V_m) satisfies the condition 1 for the graph family $\mathcal{G}^{\setminus V_1}$. $\mathcal{G}^{\setminus V_1}$ has one more input node than \mathcal{G} , so we can invoke the inductive hypothesis on $\mathcal{G}^{\setminus V_1}$ and obtain

$$\log \mathcal{N}\left(\sum_{V \in \mathcal{V} \setminus (\{V_1, O\} \cup \mathcal{I})} \kappa_V \cdot \epsilon_V + \epsilon_O, \mathcal{U}, (s_{V_1}, s_{\mathcal{I}})\right) \leq \sum_{V \in \mathcal{V} \setminus (\{V_1\} \cup \mathcal{I})} \log \mathcal{N}(\epsilon_V, \mathfrak{R}_V, s_{\text{pr}(V)}) \quad (17)$$

Combining equation (16) and (17) above, we prove (7) for \mathcal{G} , and complete the induction. \square

Below we provide the composition lemma necessary for Theorem 5.3.

Lemma B.1. Suppose

$$\mathcal{U} \subseteq \{(h, x^{(1)}, \dots, x^{(m)}) \in \mathcal{D}_h \otimes \mathcal{D}_x^{(1)} \otimes \dots \otimes \mathcal{D}_x^{(m)} \mapsto \mathcal{D}_u\}$$

is a family of functions with two arguments and $\mathcal{W} \subseteq \{x^{(1)}, \dots, x^{(m)} \in \mathcal{D}_x^{(1)} \otimes \dots \otimes \mathcal{D}_x^{(m)} \mapsto \mathcal{D}_h\}$ is another family of functions. We overload notation and refer to $x^{(1)}, \dots, x^{(m)}$ as x . The spaces $\mathcal{D}_h, \mathcal{D}_x, \mathcal{D}_u$ 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 κ -Lipschitz in the argument h for any possible choice of x : for any $u \in \mathcal{U}$, $x \in \mathcal{D}_x$, and $h, h' \in \mathcal{D}_h$, 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 $\|h\| \geq s_h$ or $\|x^{(i)}\| \geq s_x^{(i)}$ 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:

$$\log \mathcal{N}(\kappa \epsilon_w + \epsilon_u, \mathcal{Z}, s_x) \leq \log \mathcal{N}(\epsilon_w, \mathcal{W}, s_x) + \log \mathcal{N}(\epsilon_u, \mathcal{U}, (s_h, s_x))$$

Proof. When it is clear from context, we let $\|x\| \leq s_x$ denote the statement that $\|x^{(i)}\| \leq s_x^{(i)} \forall i$. Suppose P_n is a uniform distribution over n data points $\{x_1, \dots, x_n\} \subset \mathcal{D}_x$ 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 $\hat{u}(\hat{w}(x), x)$ 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 $\{x_i : \|x_i\| \leq s_x\}$, and suppose $\hat{\mathcal{W}}$ is a $\epsilon_w \sqrt{\frac{n}{|\text{supp}(P')|}}$ error cover of \mathcal{W} with respect to the metric $L_2(P', \|\cdot\|)$. We note that $\hat{\mathcal{W}}$ has size at most $\mathcal{N}(\epsilon_w, \mathcal{W}, s_x)$. We found $\hat{w} \in \hat{\mathcal{W}}$ such that \hat{w} is ϵ_w -close to w in metric $L_2(P', \|\cdot\|)$. Let \hat{h}_i denote $\hat{w}(x_i)$. 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, $\{(\hat{h}_1, x_1), \dots, (\hat{h}_n, x_n)\}$. Now we construct a intermediate cover $\hat{\mathcal{U}}'$ (that depends on \hat{w} implicitly) that covers \mathcal{U} with $\epsilon_u \sqrt{\frac{n}{|\text{supp}(Q')|}}$ error with respect to the metric $L_2(Q', \|\cdot\|)$. We augment this to a cover $\hat{\mathcal{U}}$ that covers \mathcal{U} with respect to metric $L_2(Q, \|\cdot\|)$ as follows: for every $\hat{u}' \in \hat{\mathcal{U}}'$, add the function \hat{u} to $\hat{\mathcal{U}}$ with

$$\hat{u}(h, x) = \begin{cases} \hat{u}'(h, x) & \text{if } \|h\| \leq s_h, \|x\| \leq s_x \\ b & \text{otherwise} \end{cases}$$

Note that by construction, the size of $\hat{\mathcal{U}}$ is at most $\mathcal{N}(\epsilon_u, \mathcal{U}, (s_h, s_x))$. Now let $\hat{u}' \in \hat{\mathcal{U}}'$ be the cover element for u w.r.t. $L_2(Q, \|\cdot\|)$, and \hat{u} be the corresponding cover element in $\hat{\mathcal{U}}$. Because $\hat{u}(\hat{h}, x) = b = u(\hat{h}, x)$ when $\|\hat{h}\| \geq s_h$ or $\|x^{(i)}\| \geq s_x^{(i)}$ for some i ,

$$\mathbb{E}_{\hat{h}, x \sim Q} [\|\hat{u}(\hat{h}, x) - u(\hat{h}, x)\|^2] = \frac{|\text{supp}(Q')|}{n} \mathbb{E}_{\hat{h}, x \sim Q'} [\|\hat{u}'(\hat{h}, x) - u(\hat{h}, x)\|^2] \leq \epsilon_u^2 \quad (18)$$

Then we bound the difference between $u(\hat{h}, x)$ and $u(h, x)$ by Lipschitzness; since $u(\hat{h}, x) = u(h, x) = b$ when $\|x\| > s_x$,

$$\mathbb{E}_{\hat{h}, x \sim Q} [\|u(\hat{h}, x) - u(h, x)\|^2] \leq \kappa^2 \frac{|\text{supp}(P')|}{n} \mathbb{E}_{\hat{h}, x \sim P'} [\|\hat{h} - h\|^2] \leq \kappa^2 \epsilon_w^2 \quad (19)$$

where in the last step we used the property of the cover \mathcal{G} . Finally, by triangle inequality, we get that

$$\begin{aligned} & \|\hat{u}(\hat{w}(x), x) - u(w(x), x)\|_{L_2(P_n, \|\cdot\|)} \\ & \leq \|\hat{u}(\hat{w}(x), x) - u(\hat{w}(x), x)\|_{L_2(P_n, \|\cdot\|)} + \|u(\hat{w}(x), x) - u(w(x), x)\|_{L_2(P_n, \|\cdot\|)} \\ & \leq \kappa \epsilon_w + \epsilon_u \quad (\text{by equation (18) and (19) and definition of } h_i, \hat{h}_i) \end{aligned}$$

Finally we count how many (\hat{w}, \hat{u}) we have constructed: $\hat{\mathcal{W}}$ is of size at most $\mathcal{N}(\epsilon_w, \mathcal{W}, s_x)$. and for every $\hat{w} \in \hat{\mathcal{W}}$, we've constructed a family of functions $\hat{\mathcal{U}}$ (that depends on \hat{w}) of size at most $\mathcal{N}(\epsilon_u, \mathcal{U}, (s_h, s_x))$. Therefore, the total size of the cover is at most $\mathcal{N}(\epsilon_w, \mathcal{W}, s_x) \cdot \mathcal{N}(\epsilon_u, \mathcal{U}, (s_h, s_x))$. \square

C Missing Proofs in Section 6

We first state the proofs of Theorem 6.2 and Theorem 6.3, which follow straightforwardly from the technical tools developed in Section D.

Proof of Theorem 6.2. Fix any forest ordering \mathcal{S} of $\tilde{\mathcal{G}}$. Fix $\tilde{G} \in \tilde{\mathcal{G}}$. Let \mathcal{S}' be the prefix sequence of \mathcal{S} ending in V_i . Note that \mathcal{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 $\tilde{G}^{\setminus \mathcal{S}'}$ under the framework of Lemma D.1, where we set $k = q - i$ and identify f_j with $R_{V_{i+j}}$. 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 $\tau_{j' \leftarrow i'} = \kappa_{j'+i \leftarrow i'+i}$, $\tau_{j' \leftarrow 1, m'} = \kappa_{j'+i \leftarrow m'}$, and $\bar{\tau}_j = \bar{\kappa}_{i+j}$. Now we can apply Lemma D.1 to conclude that $\tilde{G}^{\setminus \mathcal{S}'}$ is $\tilde{\kappa}_{V_i}$ -Lipschitz in V_i for any $1 \leq i \leq q$.

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

$$\begin{aligned} Q(J_i) &\triangleq \prod_{j \leq i \leq j'} \mathbb{1}_{\leq \kappa_{j' \leftarrow j}} (\|D_{j'} \cdots D_{i+1} J_i D_{i-1} \cdots D_j\|_{\text{op}}) \\ &\times \prod_{j' \geq i+1} \mathbb{1}_{\leq \kappa_{j' \leftarrow i+1}} (\|D_{j'} \cdots D_{i+1}\|_{\text{op}}) \times \prod_{j \leq i-1} \mathbb{1}_{\leq \kappa_{i-1 \leftarrow j}} (\|D_{i-1} \cdots D_j\|_{\text{op}}) \end{aligned}$$

is $\tilde{\kappa}_{J_i}$ -Lipschitz in the value of J_i . This is because after fixing all other inputs besides J_i , we can write $O_{\tilde{G}^{\setminus \mathcal{S}'}}$ 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 $O_{\tilde{G}^{\setminus \mathcal{S}'}}$, is $\tilde{\kappa}_{J_i}$ -Lipschitz. \square

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

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

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

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

where $x \in \mathcal{D}_{\mathcal{I}}$ and $v_i \in \mathcal{D}_{V_i}$. We can also apply Claim H.1 to conclude that R'_O outputs values in $[0, 1]$. Furthermore, as the function $\mathbb{1}_{\leq s_{V_i}} (\|v_i\|)$ 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 $\tilde{\mathcal{G}}$ that is $\{\tilde{\kappa}_V\}$ -release-Lipschitz with respect to any forest ordering on $(\tilde{\mathcal{V}}, \tilde{\mathcal{E}})$ for parameters $\{\tilde{\kappa}_V\}$ defined in the theorem statement. Furthermore, by the construction of our augmentation and application of Claim H.1, it follows that

$$\begin{aligned} \tilde{R}_O(x, v_1, \dots, v_q, D_1, \dots, D_q) = \\ (R_O(x, v_1, \dots, v_q) - 1) \prod_{i=1}^q \mathbb{1}_{\leq s_{V_i}} (\|v_i\|) \prod_{1 \leq i \leq j \leq q} \mathbb{1}_{\leq \kappa_{j \leftarrow i}} (\|D_j \cdots D_i\|_{\text{op}}) + 1 \end{aligned}$$

and in particular outputs the constant value 1 when $\|v_i\| > 2s_{V_i}$ or $\|D_i\| > 2\kappa_{i \leftarrow i}$. 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 $(\tilde{\mathcal{V}}, \tilde{\mathcal{E}})$. Now we can apply Theorem 6.2:

$$\begin{aligned} \log \mathcal{N}(\sum_{i \geq 1} (\tilde{\kappa}_{V_i} + \tilde{\kappa}_{J_i}) \epsilon_V + \epsilon_O, O_{\tilde{\mathcal{G}}}, s_{\mathcal{I}}) &\leq \sum_{i \geq 1} \log \mathcal{N}(\epsilon_{V_i}, \mathfrak{R}_{V_i}, 2s_{V_{i-1}}) \\ &+ \log \mathcal{N}(\epsilon_{J_i}, D\mathfrak{R}_{V_i}, 2s_{V_{i-1}}) + \log \mathcal{N}(\epsilon_O, \tilde{\mathfrak{R}}_O, \{2s_{V_i}\} \cup \{I\} \cup \{2s_{J_i}\}_{i \geq 1}) \end{aligned}$$

Now all terms match (9) except for the term $\log \mathcal{N}(\epsilon_O, \tilde{\mathfrak{R}}_O, \{2s_{V_i}\} \cup \{I\} \cup \{2s_{J_i}\}_{i \geq 1})$. First, we note that all functions in $\tilde{\mathfrak{R}}_O$ can be written in the form

$$\tilde{R}_O(x, v_1, \dots, v_q, D_1, \dots, D_q) = (R_O(x, v_1, \dots, v_q) - 1)Q(v_1, \dots, v_q, D_1, \dots, D_q) + 1$$

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

$$\log \mathcal{N}(\epsilon_O, \tilde{\mathfrak{R}}_O, \{2s_{V_i}\}_{i \geq 0} \cup \{2s_{J_i}\}_{i \geq 1}) = \log \mathcal{N}(\epsilon_O, \mathfrak{R}_O, \{2s_{V_i}\} \cup \{I\})$$

This allows us to conclude (9). Finally, we note that as the augmentation operations are in the form of those considered in Claim H.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 $x \in \mathcal{D}$ and outputting an operator mapping \mathcal{D} to \mathcal{D}' is κ -Lipschitz if $\|J(x) - J(x')\|_{\text{op}} \leq \kappa \|x - x'\|$ for any x, x' in its input domain. We will consider functions f_1, \dots, f_k , where $f_i : \mathcal{D}_{i-1} \rightarrow \mathcal{D}_i$ and \mathcal{D}_0 is a compact subset of some normed space. For ease of notation, we use $\|\cdot\|$ to denote the (possibly distinct) norms on $\mathcal{D}_0, \dots, \mathcal{D}_k$. For $1 \leq i \leq j \leq k$, Let $f_{j \leftarrow i} : \mathcal{D}_{i-1} \rightarrow \mathcal{D}_j$ denote the composition

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

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

Finally consider a function real-valued function $g : \mathcal{D}_0 \otimes \dots \otimes \mathcal{D}_k \rightarrow [0, 1]$ and define the composition $z : \mathcal{D}_0 \mapsto [0, 1]$ by

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

We will construct a “Lipschitz-fication” for the function z .

Let A_1, \dots, A_m denote a collection of linear operators that map to the space \mathcal{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 $\bar{\tau}_j$ -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 $\|J_{j \leftarrow i}\|_{\text{op}}$, and also define $\tau_{j \leftarrow 1, m'}$ which will bound $\|J_{j \leftarrow 1, m'}\|_{\text{op}}$. Define the augmented function $\tilde{z} : \mathcal{D}_0 \mapsto [0, 1]$ by*

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

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

$$\begin{aligned}\tau^* &\triangleq \sum_{0 \leq j \leq k} 3c_j \tau_{j \leftarrow 1} \\ &+ 18 \sum_{1 \leq i \leq j \leq k} \frac{\sum_{i'=i}^j \bar{\tau}_{i'} \tau_{j \leftarrow i' + 1} \tau_{i' - 1 \leftarrow 1} \tau_{i' - 1 \leftarrow i}}{\tau_{j \leftarrow i}} \\ &+ 18 \sum_{1 \leq j \leq k, m'} \frac{\sum_{i'=1}^j \bar{\tau}_{i'} \tau_{j \leftarrow i' + 1} \tau_{i' - 1 \leftarrow 1} \tau_{i' - 1 \leftarrow 1, m'}}{\tau_{j \leftarrow 1, m'}}\end{aligned}$$

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

Proof. For ease of notation, we will first define for any (i, j) with $1 \leq i \leq j \leq k$, $Q_{j \leftarrow i} \triangleq \mathbb{1}_{\leq \tau_{j \leftarrow i}}(\|J_{j \leftarrow i}\|_{\text{op}})$ and for (j, m') with $0 \leq j \leq k$, $Q_{j \leftarrow 1, m'} \triangleq \mathbb{1}_{\leq \tau_{j \leftarrow 1, m'}}(\|J_{j \leftarrow 1, m'}\|_{\text{op}})$. 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

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

We define the following order $\succ_{\mathcal{Q}}$ on this collection of functions:

$$\begin{aligned} &Q_{0 \leftarrow 1, m} \succ_{\mathcal{Q}} \cdots \succ_{\mathcal{Q}} Q_{0 \leftarrow 1, 1} \\ &\succ_{\mathcal{Q}} Q_{1 \leftarrow 1} \succ_{\mathcal{Q}} Q_{1 \leftarrow 1, m} \succ_{\mathcal{Q}} \cdots \succ_{\mathcal{Q}} Q_{1 \leftarrow 1, 1} \\ &\succ_{\mathcal{Q}} Q_{2 \leftarrow 2} \succ_{\mathcal{Q}} Q_{2 \leftarrow 1} \succ_{\mathcal{Q}} Q_{2 \leftarrow 1, m} \succ_{\mathcal{Q}} \cdots \succ_{\mathcal{Q}} Q_{2 \leftarrow 1, 1} \\ &\vdots \\ &\succ_{\mathcal{Q}} Q_{k \leftarrow k} \succ_{\mathcal{Q}} \cdots \succ_{\mathcal{Q}} Q_{k \leftarrow 1} \succ_{\mathcal{Q}} Q_{k \leftarrow 1, m'} \succ_{\mathcal{Q}} \cdots \succ_{\mathcal{Q}} Q_{k \leftarrow 1, 1} \end{aligned}$$

To prove that \tilde{z} is τ^* -Lipschitz, It suffices show that $\forall x$ and sufficiently small ν , $|\tilde{z}(x) - \tilde{z}(x + \nu)| \leq \tau^* \|\nu\|$. First, define for $0 \leq j \leq k$

$$\begin{aligned}\gamma_j(x, \nu) &\triangleq g(f_{0 \leftarrow 1}(x), \dots, f_{j-1 \leftarrow 1}(x), f_{j \leftarrow 1}(x), f_{j+1 \leftarrow 1}(x + \nu), \dots, f_{k \leftarrow 1}(x + \nu)) \\ &- g(f_{0 \leftarrow 1}(x), \dots, f_{j-1 \leftarrow 1}(x), f_{j \leftarrow 1}(x + \nu), f_{j+1 \leftarrow 1}(x + \nu), \dots, f_{k \leftarrow 1}(x + \nu))\end{aligned}$$

Next, define the telescoping differences

$$\delta_j(x, \nu) \triangleq \gamma_j(x, \nu) \prod_{Q \succeq_{\mathcal{Q}} Q_{j \leftarrow 1, 1}} Q(x) \prod_{Q_{j \leftarrow 1, 1} \succ_{\mathcal{Q}} Q} Q(x + \nu) \quad \forall 0 \leq j \leq k \quad (20)$$

$$\Delta_{j \leftarrow i}(x, \nu) \triangleq (Q_{j \leftarrow i}(x) - Q_{j \leftarrow i}(x + \nu)) \prod_{Q \succ_{\mathcal{Q}} Q_{j \leftarrow i}} Q(x) \prod_{Q_{j \leftarrow i} \succ_{\mathcal{Q}} Q} Q(x + \nu) \quad \forall 1 \leq i \leq j \leq k \quad (21)$$

$$\begin{aligned}\Delta_{j \leftarrow 1, m'}(x, \nu) &\triangleq (Q_{j \leftarrow 1, m'}(x) - Q_{j \leftarrow 1, m'}(x + \nu)) \cdot \\ &\prod_{Q \succ_{\mathcal{Q}} Q_{j \leftarrow 1, m'}} Q(x) \prod_{Q_{j \leftarrow 1, m'} \succ_{\mathcal{Q}} Q} Q(x + \nu) \quad \forall 0 \leq j \leq k \quad (22)\end{aligned}$$

Now note that by Claim D.7, we have the bound

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

Define $\bar{\tau}$ to be the Lipschitz constant of $J_{j \leftarrow i}$ on \mathcal{D}_0 for all $1 \leq i \leq j \leq k$ guaranteed by Claim D.6. First, note that $\Delta_{0 \leftarrow 1, m'} = 0$ for all m' . Thus, by Claims D.4 and D.5, it follows that

$$\begin{aligned}
|\tilde{z}(x) - \tilde{z}(x + \nu)| &\leq \sum_{0 \leq j \leq k} c_j (2\tau_{j \rightarrow 1} + \frac{\bar{\tau}}{2} \|\nu\|) \|\nu\| \\
&+ \sum_{1 \leq i \leq j \leq k} \|\nu\| \frac{\sum_{i'=i}^j (2\tau_{j \leftarrow i'+1} + \bar{\tau} \|\nu\|) \bar{\tau}_{i'} (2\tau_{i'-1 \leftarrow 1} + \frac{\bar{\tau}}{2} \|\nu\|) 2\tau_{i'-1 \leftarrow i}}{\tau_{j \leftarrow i}} \\
&+ \sum_{1 \leq j \leq k, 1 \leq m' \leq m} \|\nu\| \frac{\sum_{i'=1}^j (2\tau_{j \leftarrow i'+1} + \bar{\tau} \|\nu\|) \bar{\tau}_{i'} (2\tau_{i'-1 \leftarrow 1} + \frac{\bar{\tau}}{2} \|\nu\|) 2\tau_{i'-1 \leftarrow 1, m'}}{\tau_{j \leftarrow 1, m'}}
\end{aligned} \tag{23}$$

Now note that if $\|\nu\| \leq \frac{2 \min_{i \leq j} \tau_{j \leftarrow i}}{\bar{\tau}}$, then it follows that $2\tau_{j \leftarrow i} + \frac{\bar{\tau}}{2} \|\nu\| \leq 3\tau_{j \leftarrow i} \forall i \leq j$. Substituting into (23), we get that $\forall x, \|\nu\| \leq \frac{2 \min_{i \leq j} \tau_{j \leftarrow i}}{\bar{\tau}}$,

$$\begin{aligned}
|\tilde{z}(x) - \tilde{z}(x + \nu)| &\leq \|\nu\| \sum_{0 \leq j \leq k} 3c_j \tau_{j \leftarrow 1} \\
&+ \|\nu\| 18 \sum_{1 \leq i \leq j \leq k} \frac{\sum_{i'=i}^j \bar{\tau}_{i'} \tau_{j \leftarrow i'+1} \tau_{i'-1 \leftarrow 1} \tau_{i'-1 \leftarrow i}}{\tau_{j \leftarrow i}} \\
&+ \|\nu\| 18 \sum_{1 \leq j \leq k, m'} \frac{\sum_{i'=1}^j \bar{\tau}_{i'} \tau_{j \leftarrow i'+1} \tau_{i'-1 \leftarrow 1} \tau_{i'-1 \leftarrow 1, m'}}{\tau_{j \leftarrow 1, m'}} \\
&= \tau^* \|\nu\|
\end{aligned}$$

It follows that \tilde{z} is τ^* -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 $J_{j \leftarrow i}(x) - J_{j \leftarrow i}(x + \nu)$ as follows:*

$$J_{j \leftarrow i}(x) - J_{j \leftarrow i}(x + \nu) = \sum_{i'=i}^j J_{j \leftarrow i'+1}(x + \nu) (J_{i' \leftarrow i'}(x) - J_{i' \leftarrow i'}(x + \nu)) J_{i'-1 \leftarrow i}(x) \tag{24}$$

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

$$J_{j \leftarrow 1, m'}(x) - J_{j \leftarrow 1, m'}(x + \nu) = \sum_{i'=1}^j J_{j \leftarrow i'+1}(x + \nu) (J_{i' \leftarrow i'}(x) - J_{i' \leftarrow i'}(x + \nu)) J_{i'-1 \leftarrow 1, m'}(x) \tag{25}$$

Proof. We will first show (24) by inducting on $j - i$. The base case $j = i$ follows by definition, as we can reduce $J_{i \leftarrow i+1}$ and $J_{i-1 \leftarrow i}$ to constant-valued functions that output the identity matrix.

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

$$\begin{aligned}
J_{j \leftarrow i}(x) - J_{j \leftarrow i}(x + \nu) &= J_{j \leftarrow i+1}(x)J_{i \leftarrow i}(x) - J_{j \leftarrow i+1}(x + \nu)J_{i \leftarrow i}(x + \nu) \\
&= (J_{j \leftarrow i+1}(x) - J_{j \leftarrow i+1}(x + \nu))J_{i \leftarrow i}(x) \\
&\quad + J_{j \leftarrow i+1}(x + \nu)(J_{i \leftarrow i}(x) - J_{i \leftarrow i}(x + \nu)) \\
&= \sum_{i'=i+1}^j J_{j \leftarrow i'+1}(x + \nu)(J_{i' \leftarrow i'}(x) - J_{i' \leftarrow i'}(x + \nu))J_{i'-1 \leftarrow i+1}(x)J_{i \leftarrow i}(x) \\
&\quad \text{(by the inductive hypothesis)} \\
&\quad + J_{j \leftarrow i+1}(x + \nu)(J_{i \leftarrow i}(x) - J_{i \leftarrow i}(x + \nu)) \\
&= \sum_{i'=i+1}^j J_{j \leftarrow i'+1}(x + \nu)(J_{i' \leftarrow i'}(x) - J_{i' \leftarrow i'}(x + \nu))J_{i'-1 \leftarrow i}(x) \quad \text{(by Claim H.2)} \\
&\quad + J_{j \leftarrow i+1}(x + \nu)(J_{i \leftarrow i}(x) - J_{i \leftarrow i}(x + \nu)) \\
&= \sum_{i'=i}^j J_{j \leftarrow i'+1}(x + \nu)(J_{i' \leftarrow i'}(x) - J_{i' \leftarrow i'}(x + \nu))J_{i'-1 \leftarrow i}(x)
\end{aligned}$$

as desired.

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

$$\begin{aligned}
&J_{j \leftarrow 1, m'}(x) - J_{j \leftarrow 1, m'}(x + \nu) \\
&= (J_{j \leftarrow 1}(x) - J_{j \leftarrow 1}(x + \nu))J_{0 \leftarrow 1, m'} \\
&= \sum_{i'=1}^j J_{j \leftarrow i'+1}(x + \nu)(J_{i' \leftarrow i'}(x) - J_{i' \leftarrow i'}(x + \nu))J_{i'-1 \leftarrow 1}(x)J_{0 \leftarrow 1, m'} \quad \text{(by (24))} \\
&= \sum_{i'=1}^j J_{j \leftarrow i'+1}(x + \nu)(J_{i' \leftarrow i'}(x) - J_{i' \leftarrow i'}(x + \nu))J_{i'-1 \leftarrow 1, m'}(x) \quad \text{(since } J_{i'-1 \leftarrow 1}(x)J_{0 \leftarrow 1, m'} = J_{i'-1 \leftarrow 1, m'}(x))
\end{aligned} \tag{26}$$

□

Claim D.3. *In the setting of Lemma D.1, suppose that $J_{j \leftarrow i}$ 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*

$$\begin{aligned}
&\|J_{j \leftarrow i}(x) - J_{j \leftarrow i}(x + \nu)\|_{\text{op}} \leq \\
&\|\nu\| \sum_{i'=i}^j (\|J_{j \leftarrow i'+1}(x)\|_{\text{op}} + \bar{\tau}\|\nu\|)\bar{\tau}_{i'}(\|J_{i'-1 \leftarrow 1}(x)\|_{\text{op}} + \frac{\bar{\tau}}{2}\|\nu\|)\|J_{i'-1 \leftarrow i}(x)\|_{\text{op}}
\end{aligned} \tag{27}$$

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

$$\begin{aligned}
&\|J_{j \leftarrow 1, m'}(x) - J_{j \leftarrow 1, m'}(x + \nu)\|_{\text{op}} \leq \\
&\|\nu\| \sum_{i'=1}^j (\|J_{j \leftarrow i'+1}(x)\|_{\text{op}} + \bar{\tau}\|\nu\|)\bar{\tau}_{i'}(\|J_{i'-1 \leftarrow 1}(x)\|_{\text{op}} + \frac{\bar{\tau}}{2}\|\nu\|)\|J_{i'-1 \leftarrow 1, m'}(x)\|_{\text{op}}
\end{aligned} \tag{28}$$

Proof. We will first prove (27), as the proof of (28) is nearly identical. Starting from (24) of Claim D.2, we have

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

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

$$\|J_{j \leftarrow i}(x) - J_{j \leftarrow i}(x + \nu)\|_{\text{op}} \quad (29)$$

$$\begin{aligned} &\leq \sum_{i'=i}^j \|J_{j \leftarrow i'+1}(x + \nu)\|_{\text{op}} \|J_{i' \leftarrow i'}(x) - J_{i' \leftarrow i'}(x + \nu)\|_{\text{op}} \|J_{i'-1 \leftarrow i}(x)\|_{\text{op}} \\ &\leq \sum_{i'=i}^j (\|J_{j \leftarrow i'+1}(x)\|_{\text{op}} + \bar{\tau}\|\nu\|) \|J_{i' \leftarrow i'}(x) - J_{i' \leftarrow i'}(x + \nu)\|_{\text{op}} \|J_{i'-1 \leftarrow i}(x)\|_{\text{op}} \end{aligned} \quad (30)$$

Next, we note that

$$\begin{aligned} \|J_{i' \leftarrow i'}(x) - J_{i' \leftarrow i'}(x + \nu)\|_{\text{op}} &= \|Df_{i' \leftarrow i'}[f_{i'-1 \leftarrow 1}(x)] - Df_{i' \leftarrow i'}[f_{i'-1 \leftarrow 1}(x + \nu)]\|_{\text{op}} \\ &\leq \bar{\tau}_{i'} \|f_{i'-1 \leftarrow 1}(x) - f_{i'-1 \leftarrow 1}(x + \nu)\| \\ &\leq \bar{\tau}_{i'} (\|J_{i'-1 \leftarrow 1}(x)\|_{\text{op}} + \frac{\bar{\tau}}{2}\|\nu\|) \|\nu\| \quad (\text{applying Claim H.4}) \end{aligned}$$

Plugging the above into (30), we get (27). To prove (28), we start from (25) and follow the same steps as above. \square

Claim D.4. *In the setting of Lemma D.1, suppose that $J_{j \leftarrow i}$ is $\bar{\tau}$ -Lipschitz for all $1 \leq i \leq j \leq k$. Then we can upper bound the error terms corresponding to the indicators by*

$$|\Delta_{j \leftarrow i}(x, \nu)| \leq \|\nu\| \frac{\sum_{i'=i}^j (2\tau_{j \leftarrow i'+1} + \bar{\tau}\|\nu\|) \bar{\tau}_{i'} (2\tau_{i'-1 \leftarrow 1} + \frac{\bar{\tau}}{2}\|\nu\|) 2\tau_{i'-1 \leftarrow i}}{\tau_{j \leftarrow i}} \quad (31)$$

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

$$|\Delta_{j \leftarrow 1, m'}(x, \nu)| \leq \|\nu\| \frac{\sum_{i'=1}^j (2\tau_{j \leftarrow i'+1} + \bar{\tau}\|\nu\|) \bar{\tau}_{i'} (2\tau_{i'-1 \leftarrow 1} + \frac{\bar{\tau}}{2}\|\nu\|) 2\tau_{i'-1 \leftarrow 1, m'}}{\tau_{j \leftarrow 1, m'}} \quad (32)$$

Proof. We will prove (31) as the proof of (32) is analogous. Note that as $\mathbb{1}_{\leq \tau_{j \leftarrow i}}$ is $\frac{1}{\tau_{j \leftarrow i}}$ -Lipschitz in its argument, we have

$$\begin{aligned} |Q_{j \leftarrow i}(x) - Q_{j \leftarrow i}(x + \nu)| &= |\mathbb{1}_{\leq \tau_{j \leftarrow i}}(\|J_{j \leftarrow i}(x)\|_{\text{op}}) - \mathbb{1}_{\leq \tau_{j \leftarrow i}}(\|J_{j \leftarrow i}(x + \nu)\|_{\text{op}})| \\ &\leq \frac{1}{\tau_{j \leftarrow i}} \|\|J_{j \leftarrow i}(x)\|_{\text{op}} - \|J_{j \leftarrow i}(x + \nu)\|_{\text{op}}\| \\ &\leq \frac{1}{\tau_{j \leftarrow i}} \|J_{j \leftarrow i}(x) - J_{j \leftarrow i}(x + \nu)\|_{\text{op}} \end{aligned}$$

Plugging this into our definition for $\Delta_{j \leftarrow i}$ (21), it follows that

$$|\Delta_{j \leftarrow i}(x, \nu)| \leq \frac{1}{\tau_{j \leftarrow i}} \|J_{j \leftarrow i}(x) - J_{j \leftarrow i}(x + \nu)\|_{\text{op}} \prod_{Q \succ_{\mathcal{Q}} Q_{j \leftarrow i}} Q(x) \prod_{Q_{j \leftarrow i} \succ_{\mathcal{Q}} Q} Q(x + \nu) \quad (33)$$

Now we define the set \mathcal{E} by

$$\begin{aligned} \mathcal{E} &= \cap_{i \leq i' \leq j} \{x : \|J_{j \leftarrow i'+1}(x)\|_{\text{op}} \leq 2\tau_{j \leftarrow i'+1}, \|J_{i'-1 \leftarrow 1}(x)\|_{\text{op}} \leq 2\tau_{i'-1 \leftarrow 1}, \\ &\quad \text{and } \|J_{i'-1 \leftarrow i}(x)\|_{\text{op}} \leq 2\tau_{i'-1 \leftarrow i}\} \end{aligned}$$

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

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

where we recall that $\tau_{i-1 \leftarrow i} = 1$. Plugging this into (33) and using the fact that all functions $h \in \mathcal{Q}$ are bounded by 1 gives the desired statement.

To prove (32), we simply apply the above argument with (28). \square

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

$$|\delta_j(x, \nu)| \leq c_j(2\tau_{j \rightarrow 1} + \frac{\bar{\tau}}{2}\|\nu\|)\|\nu\|$$

Proof. Starting from (20), we can first express $\delta_i(x, \nu)$ by

$$\delta_j(x, \nu) = \gamma_j(x, \nu) Q_{j \leftarrow 1}(x) \prod_{Q \succeq_{\mathcal{Q}} Q_{j \leftarrow 1, 1}, Q \neq Q_{j \leftarrow 1}} Q(x) \prod_{Q_{j \leftarrow 1, 1} \succ_{\mathcal{Q}} Q} Q(x + \nu)$$

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

$$\begin{aligned} |\delta_j(x, \nu)| &\leq |\gamma_j(x, \nu)| Q_{j \leftarrow 1}(x) \\ &\leq c_j \|f_{j \leftarrow 1}(x) - f_{j \leftarrow 1}(x + \nu)\| \mathbb{1}_{\leq \tau_{j \leftarrow 1}}(\|J_{j \leftarrow 1}(x)\|_{\text{op}}) \\ &\leq c_j(2\tau_{j \rightarrow 1} + \frac{\bar{\tau}}{2}\|\nu\|)\|\nu\| \end{aligned} \quad (\text{by Claim H.4})$$

\square

Claim D.6. *In the setting of Lemma D.1, $\exists \bar{\tau}$ such that $\forall i \leq j$, $J_{j \leftarrow i}$ is $\bar{\tau}$ -Lipschitz on a compact domain \mathcal{D}_0 .*

Proof. We first show inductively that $f_{i \leftarrow 1}$ is Lipschitz for all i . The base case $f_{1 \leftarrow 1}$ follows by definition, as $f_{1 \leftarrow 1}$ is continuously differentiable and \mathcal{D}_0 is a compact set.

Now we show the inductive step: first write $f_{i \leftarrow 1} = f_i \circ f_{i-1 \leftarrow 1}$. By continuity, $\{f_{i-1 \leftarrow 1}(x) : x \in \mathcal{D}_0\}$ is compact. Furthermore, f_i is continuously differentiable under the assumptions of Lemma D.1. Thus, f_i is Lipschitz on domain $\{f_{i-1 \leftarrow 1}(x) : x \in \mathcal{D}_0\}$. As $f_{i \leftarrow 1} = f_i \circ f_{i-1 \leftarrow 1}$ is the composition of Lipschitz functions by the inductive hypothesis, $f_{i \leftarrow 1}$ is itself Lipschitz.

Now it follows that $\forall i$, $J_{i \leftarrow i}$ is Lipschitz on \mathcal{D}_0 , as it is the composition of $Df_{i \leftarrow i}$ and $f_{i-1 \leftarrow 1}$, both of which are Lipschitz. Finally, by the chain rule (Claim H.2), we have that $J_{j \leftarrow i} = J_{j \leftarrow j} \cdots J_{i \leftarrow i}$ 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 $J_{j \leftarrow i}$ over all $i \leq j$. \square

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

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

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

$$z_j(x, \nu) \triangleq g(f_{0 \leftarrow 1}(x), \dots, f_{j-1 \leftarrow 1}(x), f_{j \leftarrow 1}(x + \nu), f_{j+1 \leftarrow 1}(x + \nu), \dots, f_{k \leftarrow 1}(x + \nu))$$

Thus, $z_j(x, \nu)$ denotes $g \circ (f_{0 \leftarrow 1} \otimes \dots \otimes f_{k \leftarrow 1})$ with the last $k+1-j$ inputs to g depending on $x + \nu$ instead of x . Now we claim that by a telescoping argument (Claim H.3),

$$\begin{aligned} \tilde{z}(x) - \tilde{z}(x + \nu) &= \\ \sum_{0 \leq j \leq k} \delta_j(x, \nu) &+ \sum_{1 \leq i \leq j \leq k} (z_k(j, \nu) - 1) \Delta_{j \leftarrow i} + \sum_{0 \leq j \leq k, m'} (z_j(x, \nu) - 1) \Delta_{j \leftarrow 1, m'} \end{aligned} \quad (34)$$

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

$$\begin{aligned} & \delta_k, (z_k(x, \nu) - 1)\Delta_{k \leftarrow 1, 1}, \dots, (z_k(x, \nu) - 1)\Delta_{k \leftarrow 1, m'}, (z_k(x, \nu) - 1)\Delta_{k \leftarrow 1}, \dots, (z_k(x, \nu) - 1)\Delta_{k \leftarrow k} \\ & \vdots \\ & \delta_1, (z_1(x, \nu) - 1)\Delta_{1 \leftarrow 1, 1}, \dots, (z_1(x, \nu) - 1)\Delta_{1 \leftarrow 1, m'}, (z_1(x, \nu) - 1)\Delta_{1 \leftarrow 1} \\ & \delta_0, (z_0(x, \nu) - 1)\Delta_{0 \leftarrow 1, 1}, \dots, (z_0(x, \nu) - 1)\Delta_{0 \leftarrow 1, m'} \end{aligned}$$

Now we simply apply triangle inequality on (34) 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. \square

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

$$\begin{aligned} Q(J_i) &\triangleq \prod_{j \leq i \leq j'} \mathbb{1}_{\leq \kappa_{j' \leftarrow j}} (\|D_{j'} \cdots D_{i+1} J_i D_{i-1} \cdots D_j\|_{\text{op}}) \\ &\quad \times \prod_{j' \geq i+1} \mathbb{1}_{\leq \kappa_{j' \leftarrow i+1}} (\|D_{j'} \cdots D_{i+1}\|_{\text{op}}) \times \prod_{j \leq i-1} \mathbb{1}_{\leq \kappa_{i-1 \leftarrow j}} (\|D_{i-1} \cdots D_j\|_{\text{op}}) \end{aligned}$$

Then Q is $\tilde{\kappa}_{J_i}$ -Lipschitz in J_i , where

$$\tilde{\kappa}_{J_i} \triangleq \sum_{j \leq i \leq j'} \frac{4\kappa_{j' \leftarrow i+1} \kappa_{i-1 \leftarrow j}}{\kappa_{j' \leftarrow j}}$$

Here for convenience we use the convention that $\kappa_{i-1 \leftarrow i} = 1$.

Proof. There are two cases: the condition $\|D_{j'} \cdots D_{i+1}\|_{\text{op}} \leq 2\kappa_{j' \leftarrow i+1}$ and $\|D_{i-1} \cdots D_j\|_{\text{op}} \leq 2\kappa_{i-1 \leftarrow j}$ 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 $\tilde{\kappa}_{J_i}$ -Lipschitz. In the case that the condition does hold, $\mathbb{1}_{\leq \kappa_{j' \leftarrow j}} (\|D_{j'} \cdots D_{i+1} J_i D_{i-1} \cdots D_j\|_{\text{op}})$ is $\frac{\kappa_{j' \leftarrow i+1} \kappa_{i-1 \leftarrow j}}{\kappa_{j' \leftarrow j}}$ -Lipschitz for all $j' \leq i \leq j$, and therefore their product is $\tilde{\kappa}_{J_i}$ -Lipschitz. As the remaining indicators that do not depend on J_i are constants in $[0, 1]$, it follows that Q is $\tilde{\kappa}_{J_i}$ -Lipschitz. \square

E Application to Recurrent Neural Networks

In this section, we will apply our techniques to recurrent neural networks. Suppose that we are in a classification setting. For simplicity, we will assume that the hidden layer and input dimensions are d . We will define a recurrent neural network with $r - 1$ activation layers as follows using parameters W, U, Y , activation ϕ and input sequence $x = (x^{(0)}, \dots, x^{(r-2)})$:

$$\begin{aligned} F(x) &= Y h^{(2r-2)}(x) \\ h^{(2i)}(x) &= \phi(h^{(2i-1)}(x) + u^{(i-1)}(x)) \\ h^{(2i-1)}(x) &= W h^{(2i-2)}(x) \\ u^{(i-1)}(x) &= U x^{(i-1)} \end{aligned}$$

where $h^{(0)}$ is set to be 0. Now following the convention of Section 7, we will define the interlayer Jacobians. For odd indices $2i - 1, i \leq r - 1$, we simply set $Q_{2i-1 \leftarrow 2i-1}$ to the constant function $x \mapsto W$. For even indices $2i, i \leq r - 1$, we set $Q_{2i \leftarrow 2i}(x) \triangleq D\phi[h^{(2i-1)}(x) + u^{(i-1)}(x)]$, the Jacobian of the activation applied to the input of $h^{(2i)}(x)$. Finally, we set $Q_{2r-1 \leftarrow 2r-1}$ to be the constant function $x \mapsto Y$. Now for $i' > i$, we set $Q_{i' \leftarrow i}(x) = Q_{i' \leftarrow i'}(x) \cdots Q_{i \leftarrow i}(x)$. If $i' < i$, we set $Q_{i' \leftarrow i}$ to the identity matrix.

With this notation in place, we can state our generalization bound for RNN's:

Theorem E.1. Assume that the activation ϕ is 1-Lipschitz with a $\bar{\sigma}_\phi$ -Lipschitz derivative. With probability $1 - \delta$ over the random draws of P_n , all RNNs F will satisfy the following generalization guarantee:

$$\begin{aligned} & \mathbb{E}_{(x,y) \sim P} [l_{0-1}(F(x), y)] \leq \\ & \tilde{O} \left(\frac{\left((\kappa^{\text{rnn-hidden},(r)} a_Y t^{(r-1)})^{2/3} + \sum_{i=1}^{r-1} \kappa^{\text{rnn-hidden},(i)2/3} ((a_W t^{(i-1)})^{2/3} + (a_U t^{\text{data}})^{2/3}) + \sum_{i=1}^r (\kappa^{\text{rnn-jacobian},(i)b})^{2/3} \right)^{3/2}}{\sqrt{n}} \right) \\ & + \tilde{O} \left(r \sqrt{\frac{\log(1/\delta)}{n}} \right) \end{aligned}$$

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

$$\kappa^{\text{hidden},(i)} \triangleq \frac{1}{\text{poly}(r)} + \frac{\sigma_{2r-1 \leftarrow 2i}}{\gamma} + \sum_{i \leq i' < r} \frac{\sigma_{2i' \leftarrow 2i}}{t^{(i')}} + \sum_{1 \leq j \leq j' \leq 2r-1} \sum_{\substack{j'' = \max\{2i, j\}, \\ j'' \text{ even}}}^{j'} \frac{\bar{\sigma}_\phi \sigma_{j' \leftarrow j''+1} \sigma_{j''-1 \leftarrow 2i} \sigma_{j''-1 \leftarrow j}}{\sigma_{j' \leftarrow j}}$$

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

$$\begin{aligned} a_W &\triangleq \text{poly}(r)^{-1} + \|W^\top\|_{2,1}, a_U \triangleq \text{poly}(r)^{-1} + \|U^\top\|_{2,1} \\ a_Y &\triangleq \text{poly}(r)^{-1} + \|Y^\top\|_{2,1}, b \triangleq \text{poly}(r)^{-1} + \|W\|_{1,1} \\ t^{(0)} &= 0, t^{\text{data}} \triangleq \max_{x \in P_n} \max_i \|x^{(i)}\|, t^{(i)} \triangleq \text{poly}(r)^{-1} + \max_{x \in P_n} \|h^{(2i)}(x)\| \\ \sigma_{j' \leftarrow j} &\triangleq \text{poly}(r)^{-1} + \max_{x \in P_n} \|Q_{j' \leftarrow j}(x)\|_{\text{op}}, \text{ and } \gamma \triangleq \min_{(x,y) \in P_n} [F(x)]_y - \max_{y' \neq y} [F(x)]_{y'} > 0 \end{aligned}$$

Note that the training error here is 0 because of the existence of positive margin γ .

Our proof follows the template of Theorem 7.1: we bound the Rademacher complexity of some augmented RNN loss. We then argue for generalization of the augmented loss and perform a union bound over all the choices of parameters. As the latter steps are identical to those in the proof of Theorem 7.1, we omit these and focus on bounding the Rademacher complexity of an augmented RNN loss.

Theorem E.2. Suppose that ϕ is 1-Lipschitz with $\bar{\sigma}_\phi$ -Lipschitz derivative. Define the following class of RNNs with bounded weight matrices:

$$\mathcal{F} \triangleq \left\{ x \mapsto F(x) : \|W^\top\|_{2,1} \leq a_Y, \|U^\top\|_{2,1} \leq a_U, \|Y^\top\|_{2,1} \leq a_Y, \|W\|_{1,1} \leq b, \|W\|_{\text{op}} \leq \sigma_W, \|Y\|_{\text{op}} \leq \sigma_Y \right\}$$

and let $\sigma_{j' \leftarrow j}$ be parameters that will bound the j to j' layerwise Jacobian for $j' \geq j$, where we set $\sigma_{2i \leftarrow 2i} = 1$ and $\sigma_{2i-1 \leftarrow 2i-1} = \sigma_W$ for $i \leq r-1$, $\sigma_{2r-1 \leftarrow 2r-1} = \sigma_Y$. Let $t^{(i)}$ be parameters bounding the layer norm after applying the i -th activation, and let $t^{(0)} = 0, t^{\text{data}} = \max_{x \in P_n} \max_i \|x^{(i)}\|$. Define the class of augmented losses

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

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 (11), (12). Then we can bound the empirical Rademacher complexity of the augmented loss class by

$$\begin{aligned} & \text{Rad}_n(\mathcal{L}_{\text{rnn-aug}}) = \\ & \tilde{O} \left(\frac{\left((\kappa^{\text{rnn-hidden},(r)} a_Y t^{(r-1)})^{2/3} + \sum_{i=1}^{r-1} \kappa^{\text{rnn-hidden},(i)2/3} ((a_W t^{(i-1)})^{2/3} + (a_U t^{\text{data}})^{2/3}) + \sum_{i=1}^r (\kappa^{\text{rnn-jacobian},(i)b})^{2/3} \right)^{3/2}}{\sqrt{n}} \right) \end{aligned}$$

where $\kappa^{\text{rnn-hidden},(i)}, \kappa^{\text{rnn-jacobian},(i)}$ are defined in Theorem E.1.

Proof. We will associate the family of losses $\mathcal{L}_{\text{rnn-aug}}$ with a computational graph structure on internal nodes $H_1, H_2, \dots, H_{2r-1}$, J_1, \dots, J_{2r-1} , K_0, \dots, K_{r-2} , input nodes H_0, I_0, \dots, I_{r-2} , and output node O with the following edges:

1. Nodes H_i, J_i will point towards the output O .
2. Node H_i will point towards nodes H_{i+1} and J_{i+1} .
3. Node K_{i-1} will point towards node H_{2i} and node J_{2i} .
4. Node I_i will point towards node K_i .

We now define the composition rules at each node:

$$\begin{aligned}\mathfrak{R}_{H_{2i}} &= \{(h, k) \mapsto \phi(h + k)\} \\ \mathfrak{R}_{H_{2i-1}} &= \{h \mapsto Wh : \|W^\top\|_{2,1} \leq a_W, \|W\|_{\text{op}} \leq \sigma\} \text{ for } 2 \leq i \leq r-1 \\ \mathfrak{R}_{H_{2r-1}} &= \{h \mapsto Yh : \|Y^\top\|_{2,1} \leq a_Y, \|Y\|_{\text{op}} \leq \sigma_Y\} \\ \mathfrak{R}_{J_{2i}} &= \{(h, k) \mapsto D\phi[h + k]\} \\ \mathfrak{R}_{K_i} &= \{x \mapsto Ux : \|U^\top\|_{2,1} \leq a_U\}\end{aligned}$$

Finally, nodes J_{2i-1} will have composition rule $R_{J_{2i-1}} = DR_{H_{2i-1}}$. Finally, the output node O will have composition rule

$$R_O(x, h_1, \dots, h_{2r-1}, D_1, \dots, D_{2r-1}) \triangleq (l_\gamma(h_{2r-1}) - 1) \prod_{i=1}^{r-1} \mathbb{1}_{\leq t^{(i)}}(\|h_{2i}\|) \prod_{1 \leq j < j' \leq 2r-1} \mathbb{1}_{\leq \sigma_{j' \leftarrow j}}(\|D_{j'} \cdots D_j\|_{\text{op}}) + 1$$

Note that the family of functions computed by this computation graph family is a strict superset of $\mathcal{L}_{\text{rnn-aug}}$ (as we technically allow $R_{H_{2i-1}}, R_{H_{2i'-1}}$ to use different matrices W). We will refer to this resulting family as $\tilde{\mathcal{G}}$.

First, we claim that $\tilde{\mathcal{G}}$ satisfies the release-Lipschitz condition, with Lipschitz constants $\kappa^{\text{rnn-hidden},(i)}$ for nodes H_{2i-1} and K_{i-1} , and $\kappa^{\text{rnn-jacobian},(i)}$ for nodes J_{2i-1} . (As we will see later, the Lipschitzness of nodes V_{2i}, J_{2i} will not matter because the composition rules are function classes with log covering number 0.)

To see this, we note that if we release K_0, \dots, K_{r-2} from the graph and set them to fixed values, the resulting induced graph family is simply the Lipschitz augmentation of Section 6 for the sequential graph family on nodes H_0, \dots, H_{2r-1} and an un-augmented output. Thus, the machinery of Theorem 6.2 applies here, and we can conclude that this reduced graph family is $\kappa^{\text{rnn-hidden},(i)}$ -release-Lipschitz for nodes H_{2i-1} and $\kappa^{\text{rnn-jacobian},(i)}$ -release-Lipschitz for nodes J_{2i-1} . Since this holds for any choice of K_0, \dots, K_{r-2} , we can draw the same conclusion about $\tilde{\mathcal{G}}$, the augmented family that is not reduced. However, by nature of the composition rules in $\tilde{\mathcal{G}}$, the Lipschitzness of H_{2i-1} and K_{i-1} must be identical (as $f(x + y)$ must have the same worst-case Lipschitz constant in x and y for any function f). Thus, we get that $\tilde{\mathcal{G}}$ satisfies release-Lipschitzness with constants $\kappa^{\text{rnn-hidden},(i)}$ for nodes H_{2i-1}, K_{i-1} , and $\kappa^{\text{rnn-jacobian},(i)}$ for nodes J_{2i-1} .

With this condition established, we can complete the proof via the same covering number argument as in Theorem A.2. \square

Now as in the proof of Theorem 7.1, we first observe that the augmented loss upper bounds the 0-1 classification loss, giving us a 0-1 test error bound. We then apply the same union bound technique over parameters $\gamma, t^{(i)}, \sigma_{j' \leftarrow j}, a_W, a_U, a_Y$, as in the proof of Theorem 7.1.

F ReLU Networks

In this section, we apply our augmentation technique to relu networks to produce a generalization bound similar to that of Nagarajan and 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 ϕ is defined by

$$F(x) = W^{(r)}\phi(\dots\phi(W^{(1)}x)\dots)$$

For this section, we will set ϕ 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 F.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*

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

where

$$\begin{aligned} \kappa^{\text{relu-jacobian},(i)} &\triangleq \sum_{1 \leq j \leq 2i-1 \leq j' \leq 2r-1} \frac{\sigma_{j' \leftarrow 2i} \sigma_{2i-2 \leftarrow j}}{\sigma_{j' \leftarrow j}} \\ \kappa^{\text{relu-hidden},(i)} &\triangleq \frac{1}{\text{poly}(r)} + \frac{\sigma_{2r-1 \leftarrow 2i}}{\gamma} + \sum_{i \leq i' < r} \frac{\sigma_{2i' \leftarrow 2i}}{t^{(i')}} + \frac{\sigma_{2i' - 1 \leftarrow 2i}}{\gamma^{(i')}} \end{aligned} \quad (35)$$

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

$$\gamma^{(i)} \triangleq \min_{x \in P_n} \min_j |F_{2i-1 \leftarrow 1}(x)|_j$$

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

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

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, $\kappa^{\text{relu-jacobian},(i)} = \kappa^{\text{jacobian},(i)}$, but $\kappa^{\text{relu-hidden},(i)}$ now has a dependence on the preactivations $\gamma^{(i)}$, as in Nagarajan and Kolter [2019].

We provide a proof sketch of Theorem F.1 here. We first bound the Rademacher complexity some family of augmented losses, specified precisely in Theorem F.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 F.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

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

where $\mathbb{1}_{\geq \gamma^{(i)}} \triangleq 1 - \mathbb{1}_{\leq \gamma^{(i)}/2}$. 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 (35). Then the augmented loss class $\mathcal{L}_{\text{relu-aug}}$ has empirical Rademacher complexity upper bound

$$\text{Rad}_n(\mathcal{L}_{\text{relu-aug}}) = \tilde{O} \left(\frac{(\sum_i (\kappa^{\text{relu-hidden},(i)} a^{(i)} t^{(i-1)})^{2/3} + (\kappa^{\text{relu-jacobian},(i)} b^{(i)})^{2/3})^{3/2}}{\sqrt{n}} \right)$$

Note the differences with Theorem A.2: the augmented loss class $\mathcal{L}_{\text{relu-aug}}$ now includes the additional indicators $\mathbb{1}_{\geq \gamma^{(i)}}(\min_j \|F_{2i-1 \leftarrow 1}\|_j)$, and we use the Lipschitz constants $\kappa^{\text{relu-hidden},(i)}$, $\kappa^{\text{relu-jacobian},(i)}$ defined in Theorem F.1.

Proof sketch. As in the proof of Theorem A.2, associate the loss class $\mathcal{L}_{\text{relu-aug}}$ with a family $\tilde{\mathcal{G}}$ of computation graphs on internal nodes $V_1, \dots, V_{2r-1}, J_1, \dots, J_{2r-1}$ 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

$$\begin{aligned} \mathfrak{R}_{V_{2i}} &= \{\phi\} \\ \mathfrak{R}_{V_{2i-1}} &= \{h \mapsto Wh : \|W^\top - A^{(i)}\|_{2,1} \leq a^{(i)}, \|W - B^{(i)}\|_{1,1} \leq b^{(i)}, \|W\|_{\text{op}} \leq \sigma^{(i)}\} \end{aligned}$$

Assign to the J_i nodes composition rule $R_{J_i} = DR_{V_i}$, and finally, assign to the output node O the composition rule

$$\begin{aligned} R_O(x, v_1, \dots, v_{2r-1}, D_1, \dots, D_{2r-1}) &\triangleq \\ (l_\gamma(v_{2r-1}) - 1) \prod_{i=1}^{r-1} \mathbb{1}_{\leq t^{(i)}}(\|v_{2i}\|) \mathbb{1}_{\geq \gamma^{(i)}}(\min_j \|v_{2i-1}\|_j) &\prod_{1 \leq j < j' \leq 2r-1} \mathbb{1}_{\leq \sigma_{j' \leftarrow j}}(\|D_{j'} \cdots D_j\|_{\text{op}}) + 1 \end{aligned}$$

The resulting family of computation graphs will compute $\mathcal{L}_{\text{relu-aug}}$. Now we claim that $\tilde{\mathcal{G}}$ 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 log 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.8 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 $l_\gamma(V_{2r-1}) - 1$ multiplied by the other indicators, 2) the change in the term $\mathbb{1}_{\leq t^{(i')}}(\|V_{2i}\|) \mathbb{1}_{\geq \gamma^{(i)}}(\min_j \|V_{2i-1}\|_j)$ multiplied by the other indicators, and 3) the change in $\mathbb{1}_{\leq \sigma_{j' \leftarrow j}}(\|J_{j'} \cdots J_j\|_{\text{op}})$ multiplied by the other indicators. The term 1) can be computed as $\frac{\sigma_{2r-1 \leftarrow 2i}}{\gamma}$, term 2) can be accounted for by $\frac{\sigma_{2i' \leftarrow 2i}}{t^{(i')}} + \frac{\sigma_{2i' - 1 \leftarrow 2i}}{\gamma^{(i')}}$, 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 $\mathbb{1}_{\geq \gamma^{(i)}}(\min_j \|V_{2i-1}\|_j)$ 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. \square

G Experiment Details

For all settings, we train for 200 epochs with learning rate decay by a factor of 0.2 at epochs 60, 120, and 150. We additionally tuned the value of λ from values $\{0.1, 0.05, 0.01\}$ for each setting: for the experiments displayed in Figure 1, we used the following values:

1. Low learning rate: $\lambda = 0.1$
2. No data augmentation: $\lambda = 0.1$
3. No BatchNorm: $\lambda = 0.05$

For all other hyperparameters, we use the defaults in the PyTorch WideResNet implementation: <https://github.com/xternalz/WideResNet-pytorch>, and we base our code off of this implementation. We report results from a single run as the improvement with Jacobian regularization is statistically significant. We train on a single NVIDIA TitanXp GPU.

H Toolbox

Claim H.1. *Consider the function $u : [0, 1] \times [0, 1] \mapsto \mathbb{R}$ 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. *$u(u(x_1, x_2), x_3) = u(x_1, x_2x_3)$.*

Proof. First, we note that $u(x_1, x_2) = x_1x_2 + 1 - x_2 \leq x_2 + 1 - x_2 = 1$. Furthermore, $u(x_1, x_2) \geq x_1x_2 + x_1(1 - x_2) = x_1$, which completes the proof of statements 1 and 2. To prove the third statement, we note that $u(u(x_1, x_2), x_3) = (x_1x_2 + 1 - x_2)x_3 + 1 - x_3 = x_1x_2x_3 + 1 - x_2x_3 = u(x_1, x_2x_3)$. \square

Claim H.2 (Chain rule Wikipedia contributors [2019]). *The Jacobian of a composition of a sequence of functions f_1, \dots, f_k satisfies*

$$Df_{k \leftarrow 1}(x) = Df_k(f_{(k-1) \leftarrow 1}(x)) \cdot Df_{k-1}(f_{(k-2) \leftarrow 1}(x)) \cdots Df_2(f_1(x)) \cdot Df_1(x) \quad (36)$$

where the \cdot notations are standard matrix multiplication. For simplicity, we also write in the function form:

$$Df_{k \leftarrow 1} = (Df_k \circ f_{(k-1) \leftarrow 1}) \cdot (Df_{k-1} \circ f_{(k-2) \leftarrow 1}) \cdots (Df_2 \circ f_1) \cdot Df_1 \quad (37)$$

Claim H.3 (Telescoping sum). *Let p_1, \dots, p_m and $q_1 \dots q_m$ be two sequence of functions from \mathbb{R}^d to \mathbb{R} . Then,*

$$p_1p_2 \cdots p_m - q_1q_2 \cdots q_m = (p_1 - q_1)p_2 \cdots p_m + q_1(p_2 - q_2)p_3 \cdots p_m + \cdots + q_1 \cdots q_{m-1}(p_m - q_m) \quad (38)$$

Claim H.4 (Bounding function differences). *Let $f : \mathcal{D} \rightarrow \mathcal{D}'$, 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 κ -Lipschitz in x , in the sense that $\|Df[x] - Df[x + \nu]\|_{\text{op}} \leq \kappa\|\nu\|$, where $\|\cdot\|_{\text{op}}$ is the operator norm induced by \mathcal{D} and \mathcal{D}' . Then*

$$\|f(x) - f(x + \nu)\| \leq (\|Df[x]\|_{\text{op}} + \frac{\kappa}{2}\|\nu\|)\|\nu\| \quad (39)$$

Furthermore,

$$\|f(x) - f(x + \nu)\| \mathbb{1}_{\leq \tau_f}(\|Df[x]\|_{\text{op}}) \leq (2\tau_f + \frac{\bar{\tau}}{2}\|\nu\|)\|\nu\| \quad (40)$$

Proof. We write $f(x + \nu) - f(x) = \left(\int_{t=0}^1 Df[x + t\nu] dt \right) \nu$. Now we note that

$$\begin{aligned}
\left\| \int_{t=0}^1 Df[x + t\nu] dt \right\|_{\text{op}} &\leq \int_{t=0}^1 \|Df[x + t\nu]\|_{\text{op}} dt && \text{(by triangle inequality)} \\
&\leq \int_{t=0}^1 (\|Df[x]\|_{\text{op}} + t\kappa\|\nu\|) dt && \text{(by Lipschitzness of } Df) \\
&\leq \|Df[x]\|_{\text{op}} + \frac{\kappa}{2}\|\nu\| && (41)
\end{aligned}$$

Thus,

$$\begin{aligned}
\|f(x + \nu) - f(x)\| &\leq \left\| \int_{t=0}^1 Df[x + t\nu] dt \right\|_{\text{op}} \|\nu\| \\
&\leq \left(\|Df[x]\|_{\text{op}} + \frac{\kappa}{2}\|\nu\| \right) \|\nu\| && \text{(by (41))}
\end{aligned}$$

which proves (39).

To prove (40), we consider two cases. first, if $\|Df[x]\|_{\text{op}} > 2\tau_f$, then $\mathbb{1}_{\leq \tau_f}(\|Df[x]\|_{\text{op}}) = 0$ so (40) immediately holds. Otherwise, if $\|Df[x]\|_{\text{op}} \leq 2\tau_f$, we can plug this into (39) to obtain (40), as desired. \square