

A PATH-NORM TOOLKIT FOR MODERN NETWORKS: CONSEQUENCES, PROMISES AND CHALLENGES

Antoine Gonon, Nicolas Brisebarre, Elisa Riccietti & Rémi Gribonval

Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France

ABSTRACT

This work introduces the first toolkit around path-norms that fully encompasses general DAG ReLU networks with biases, skip connections and any operation based on the extraction of order statistics: max pooling, GroupSort etc. This toolkit notably allows us to establish generalization bounds for modern neural networks that are not only the most widely applicable path-norm based ones, but also recover or beat the sharpest known bounds of this type. These extended path-norms further enjoy the usual benefits of path-norms: ease of computation, invariance under the symmetries of the network, and improved sharpness on layered fully-connected networks compared to the product of operator norms, another complexity measure most commonly used.

The versatility of the toolkit and its ease of implementation allow us to challenge the concrete promises of path-norm-based generalization bounds, by numerically evaluating the sharpest known bounds for ResNets on ImageNet.

1 INTRODUCTION

Developing a thorough understanding of theoretical properties of neural networks is key to achieve central objectives such as efficient and trustworthy training, robustness to adversarial attacks (e.g. via Lipschitz bounds), or statistical soundness guarantees (via so-called generalization bounds).

The so-called path-norms and path-lifting are promising concepts to theoretically analyze neural networks: the L^1 path-norm has been used to derive generalization guarantees (Neysshabur et al., 2015; Barron & Klusowski, 2019), and the path-lifting has led for example to identifiability guarantees (Bona-Pellissier et al., 2022; Stock & Gribonval, 2023) and characterizations of properties of the dynamics of training algorithms (Marcotte et al., 2023).

Yet, current definitions of path-norms and of path-lifting are severely limited: they only cover simple models unable to combine in a single framework pooling layers, skip connections, biases, or even multi-dimensional output (Neysshabur et al., 2015; Kawaguchi et al., 2017; Bona-Pellissier et al., 2022; Stock & Gribonval, 2023). Thus, the promises of existing theoretical guarantees based on these tools are currently out of reach: they cannot even be tested on standard modern networks.

Due to the lack of versatility of these tools, known results have only been tested on toy examples. This prevents us from both understanding the reach of these tools and from diagnosing their strengths and weaknesses, which is necessary to either improve them in order to make them actually operational, if possible, or to identify without concession the gap between theory and practice, in particular for generalization bounds.

This work addresses the challenge of making these tools fully compatible with modern networks, and to concretely assess them on standard real-world examples. First, it formalizes a definition of path-lifting (and path-norms) adapted to very generic ReLU networks, covering any DAG architecture (in particular with skip connections), including in the presence of max/average-pooling (and even more generally k -max-pooling, which extracts the k -th largest coordinate, recovering max-pooling for $k = 1$) and/or biases. This covers a wide variety of modern networks (notably ResNets, VGGs, U-nets, ReLU MobileNets, Inception nets, Alexnet)¹, and recovers previously known definitions of these tools in simpler settings such as layered fully-connected networks.

¹The conclusion discusses networks not covered by the framework and adaptations needed to cover them.

The immediate interests of these tools are: 1) *path-norms are easy to compute*² on modern networks via a single forward-pass; 2) *path-norms are invariant under neuron permutations and parameter rescalings that leave the network invariant*; and 3) *the path-norms yield Lipschitz bounds*. These properties were known (but scattered in the literature) in the restricted case of layered fully-connected ReLU networks primarily without biases (and without average/ k -max-pooling nor skip connections) (Neyshabur et al., 2015; Neyshabur, 2017; Furusho, 2020; Jiang et al., 2020; Dziugaite et al., 2020; Bona-Pellissier et al., 2022; Stock & Gribonval, 2023). They are generalized here for generic DAG ReLU networks with most of the standard ingredients of modern networks (pooling, skip connections...), with the notable exception of the attention mechanism.

Moreover, *path-norms tightly lower bound products of operator norms*, another complexity measure that does not enjoy the same invariances as path-norms, despite being widely used for Lipschitz bounds (e.g., to control adversarial robustness) (Neyshabur et al., 2018; Gonon et al., 2023) or generalization bounds (Neyshabur et al., 2015; Bartlett et al., 2017; Golowich et al., 2018). This bound, which was only known for *scalar-valued* layered fully-connected ReLU networks without biases (Neyshabur et al., 2015), is again generalized here to generic *vector-valued* DAG ReLU networks. This requires introducing so-called *mixed* path-norms and extending products of operator norms.

Second, **this work also establishes a new generalization bound for modern ReLU networks based on their corresponding L^1 path-norm**. This bound covers arbitrary output dimension (while previous work focused on scalar dimension, see Table 1), generic DAG ReLU network architectures with average/ k -max-pooling, skip connections and biases. The achieved generalization bound recovers or beats the sharpest known ones of this type, that were so far only available in simpler restricted settings, see Table 1 for an overview. Among the technical ingredients used in the proof of this generalization bound, *the new contraction lemmas and the new peeling argument are among the main theoretical contributions of this work*. *The first new contraction lemma extends the classical ones with scalar $t_i \in \mathbb{R}$, and contractions f_i of the form $\mathbb{E}_e g(\sup_{t \in T} \sum_{i \in I} \varepsilon_i f_i(t_i)) \leq \mathbb{E}_e g(\sup_{t \in T} \sum_{i \in I} \varepsilon_i t_i)$ (Ledoux & Talagrand, 1991, Theorem 4.12), with convex non-decreasing g , to situations where there are multiples independent copies indexed by $z \in Z$ of the latter: $\mathbb{E}_e \max_{z \in Z} g(\sup_{t \in T^z} \sum_{i \in I} \varepsilon_{i,z} f_{i,z}(t_i)) \leq \mathbb{E}_e \max_{z \in Z} g(\sup_{t \in T^z} \sum_{i \in I} \varepsilon_{i,z} t_i)$. The second new contraction lemma deals with vector-valued $t_i \in \mathbb{R}^W$, and functions f_i that compute the k -th largest input's coordinate, to cope with k -max-pooling neurons, and it also handles multiple independent copies indexed by $z \in Z$. The most closely related lemma we could find is the vector-valued one in Maurer (2016) established with a different technique, and that holds only for $g = \text{id}$ with a single copy ($|Z| = 1$). The peeling argument reduces the Rademacher complexity of the whole model to that of the inputs by peeling off the neurons of the model one by one. This is inspired by the peeling argument of Golowich et al. (2018), which is however specific to layered fully-connected ReLU networks with layer-wise constraints on the weights. Substantial additional ingredients are developed to handle *arbitrary DAG ReLU networks* (as there is no longer such a thing as a *layer* to peel), *with not only ReLU but also k -max-pooling and identity neurons* (where Golowich et al. (2018) has only ReLU neurons), and leveraging *only a global constraint through the path-norm* (instead of layerwise constraints on operator norms). The analysis notably makes use of the rescaling invariance of the proposed generalized path-lifting.*

The versatility of the proposed tools **enables us to compute for the first time generalization bounds based on the L^1 path-norm on networks really used in practice**. This is the opportunity to assess the current state of the gap between theory and practice, and to diagnose possible room for improvements. As a concrete example, we demonstrate that on ResNet18 trained on ImageNet: 1) the proposed generalization bound can be numerically computed; 2) for a (dense) ResNet18 trained with standard tools, roughly *30 orders of magnitude* would need to be gained for this path-norm based bound to match practically observed generalization error; 3) the same bound evaluated on a *sparse* ResNet18 (trained with standard sparsification techniques) is decreased by up to 13 orders of magnitude. We conclude the paper by discussing promising leads to reduce this gap.

Paper structure. Section 2 introduces the ReLU networks being considered, and generalizes to this model the central definitions and results related to the path-lifting, the path-activations and path-norms. Section 3 state a versatile generalization bound for such networks based on path-norm, and sketches its proof. Section 4 reports numerical experiments on ImageNet and ResNets. Related works are discussed along the way, and we refer to Appendix K for more details.

²Code for reproducibility is in Gonon et al. (2024).

Table 1: Generalization bounds (up to universal multiplicative constants) for a ReLU network estimator with parameters restricted to belong to some subset $\Theta \subseteq \mathbb{R}^G$ (Definition 2.2), learned from n iid training points when 1) the loss $\hat{y} \in (\mathbb{R}^{d_{\text{out}}}, \|\cdot\|_2) \mapsto \ell(\hat{y}, y) \in \mathbb{R}$ is L -Lipschitz for every y , and 2) inputs are bounded in L^∞ -norm by $B \geq 1$. Here, $d_{\text{in}}/d_{\text{out}}$ are the input/output dimensions, $K = \max_{v \in N_{*-\text{pool}}} |\text{ant}(v)|$ is the maximum kernel size (Definition 2.2) of the $*$ -max-pooling neurons, M_d is the matrix of layer d for a *layered fully-connected* network (LFCN) without bias $R_\theta(x) = M_D \text{ReLU}(M_{D-1} \dots \text{ReLU}(M_1 x))$, D is the depth. Note that having r in the bound is more desirable than having R since $r \leq R$ and R can be arbitrarily large even when $r = 0$ (Theorem C.1 and Figure 2 in appendix). This is because R decouples the layers without taking into account rescaling invariances.

	Architecture	Parameter set Θ	Generalization bound
(Kakade et al., 2008, Eq. (5)) (Bach, 2024, Sec. 4.5.3)	LFCN with depth $D = 1$, no bias, $d_{\text{out}} = 1$ (linear regression)	$\ \theta\ _1 =$ $\ \Phi(\theta)\ _1 \leq r$	$\frac{LB}{\sqrt{n}} \times r \sqrt{\ln(d_{\text{in}})}$
(E et al., 2022, Thm. 6) (Bach, 2017, Proposition 7)	LFCN with $D = 2$, no bias, $d_{\text{out}} = 1$ (two-layer network)	$\ \Phi(\theta)\ _1 \leq r$	$\frac{LB}{\sqrt{n}} \times r \sqrt{\ln(d_{\text{in}})}$
(Neyshabur et al., 2015, Corollary 7)	DAG, no bias, $d_{\text{out}} = 1$	$\ \Phi(\theta)\ _1 \leq r$	$\frac{LB}{\sqrt{n}} \times 2^D r \sqrt{\ln(d_{\text{in}})}$
(Golowich et al., 2018, Theorem 3.2)	LFCN with arbitrary D , no bias, $d_{\text{out}} = 1$	$\prod_{d=1}^D \ M_d\ _{1,\infty} \leq R$	$\frac{LB}{\sqrt{n}} \times R \sqrt{D + \ln(d_{\text{in}})}$
(Barron & Klusowski, 2019, Corollary 2)	LFCN with arbitrary D , no bias, $d_{\text{out}} = 1$	$\ \Phi(\theta)\ _1 \leq r$	$\frac{LB}{\sqrt{n}} \times r \sqrt{D + \ln(d_{\text{in}})}$
Here, Theorem 3.1	DAG, with biases, arbitrary d_{out} , with ReLU, identity and k -max-pooling neurons for $k \in \{k_1, \dots, k_P\} \subset \{1, \dots, K\}$	$\ \Phi(\theta)\ _1 \leq r$	$\frac{LB}{\sqrt{n}} \times r \sqrt{D \ln(PK) + \ln(d_{\text{in}} d_{\text{out}})}$

2 RELU MODEL AND PATH-LIFTING

Section 2.1 defines a general DAG ReLU model that covers modern architectures. Section 2.2 then introduces the so-called path-norms and extends related known results to this general model.

2.1 RELU MODEL THAT COVERS MODERN NETWORKS

Definition 2.2 introduces the model considered here (extending, e.g., DeVore et al. (2021)).

Definition 2.1. The ReLU function is defined as $\text{ReLU}(x) := x \mathbb{1}_{x \geq 0}$ for $x \in \mathbb{R}$. The k -max-pooling function $k\text{-pool}(x) := x_{(k)}$ returns the k -th largest coordinate of $x \in \mathbb{R}^d$.

Definition 2.2. Consider a Directed Acyclic Graph (DAG) $G = (N, E)$ with edges E , and vertices N called neurons. For a neuron v , the sets $\text{ant}(v)$, $\text{suc}(v)$ of antecedents and successors of v are $\text{ant}(v) := \{u \in N, u \rightarrow v \in E\}$, $\text{suc}(v) := \{u \in N, v \rightarrow u \in E\}$. Neurons with no antecedents (resp. no successors) are called input (resp. output) neurons, and their set is denoted N_{in} (resp. N_{out}). Input and output dimensions are respectively $d_{\text{in}} := |N_{\text{in}}|$ and $d_{\text{out}} := |N_{\text{out}}|$.

• A **ReLU neural network architecture** is a tuple $(G, (\rho_v)_{v \in N \setminus N_{\text{in}}})$ composed of a DAG $G = (N, E)$ with attributes $\rho_v \in \{\text{id}, \text{ReLU}\} \cup \{k\text{-pool}, k \in \mathbb{N}_{>0}\}$ for $v \in N \setminus (N_{\text{out}} \cup N_{\text{in}})$ and $\rho_v = \text{id}$ for $v \in N_{\text{out}}$. We will again denote the tuple $(G, (\rho_v)_{v \in N \setminus N_{\text{in}}})$ by G , and it will be clear from context whether the results depend only on $G = (N, E)$ or also on its attributes. Define $N_\rho := \{v \in N, \rho_v = \rho\}$ for an activation ρ , and $N_{*-\text{pool}} := \cup_{k \in \mathbb{N}_{>0}} N_{k\text{-pool}}$. A neuron in $N_{*-\text{pool}}$ is called a $*$ -max-pooling neuron. For $v \in N_{*-\text{pool}}$, its kernel size is defined as being $|\text{ant}(v)|$.

• **Parameters** associated with this architecture are vectors³ $\theta \in \mathbb{R}^G := \mathbb{R}^{E \cup N \setminus N_{\text{in}}}$. We call bias $b_v := \theta_v$ the coordinate associated with a neuron v (input neurons have no bias), and denote $\theta^{u \rightarrow v}$ the weight associated with an edge $u \rightarrow v \in E$. We will often denote $\theta^{\rightarrow v} := (\theta^{u \rightarrow v})_{u \in \text{ant}(v)}$ and $\theta^{v \rightarrow} := (\theta^{u \rightarrow v})_{u \in \text{suc}(v)}$.

• The **realization** of a neural network with parameters $\theta \in \mathbb{R}^G$ is the function $R_\theta^G : \mathbb{R}^{N_{\text{in}}} \rightarrow \mathbb{R}^{N_{\text{out}}}$ (simply denoted R_θ when G is clear from the context) defined for every input $x \in \mathbb{R}^{N_{\text{in}}}$ as

$$R_\theta(x) := (v(\theta, x))_{v \in N_{\text{out}}},$$

³For an index set I , denote $\mathbb{R}^I = \{(\theta_i)_{i \in I}, \theta_i \in \mathbb{R}\}$.

where we use the same symbol v to denote a neuron $v \in N$ and the associated function $v(\boldsymbol{\theta}, x)$, defined as $v(\boldsymbol{\theta}, x) := x_v$ for an input neuron v , and defined by induction otherwise

$$v(\boldsymbol{\theta}, x) := \begin{cases} \rho_v(b_v + \sum_{u \in \text{ant}(v)} u(\boldsymbol{\theta}, x)\boldsymbol{\theta}^{u \rightarrow v}) & \text{if } \rho_v = \text{ReLU or } \rho_v = \text{id}, \\ k\text{-pool}((b_v + u(\boldsymbol{\theta}, x)\boldsymbol{\theta}^{u \rightarrow v})_{u \in \text{ant}(v)}) & \text{if } \rho_v = k\text{-pool}. \end{cases} \quad (1)$$

Such a model indeed encompasses modern networks via the following implementations:

- *Max-pooling*: set $\rho_v = k\text{-pool}$ for $k = 1$, $b_v = 0$ and $\boldsymbol{\theta}^{u \rightarrow v} = 1$ for every $u \in \text{ant}(v)$.
- *Average-pooling*: set $\rho_v = \text{id}$, $b_v = 0$ and $\boldsymbol{\theta}^{u \rightarrow v} = 1/|\text{ant}(v)|$ for every $u \in \text{ant}(v)$.
- *GroupSort/Top-k operator*: use the DAG structure and $*$ -max-pooling neurons (Anil et al., 2019; Sander et al., 2023).
- *Batch normalization*: set $\rho_v = \text{id}$ and weights accordingly. Batch normalization layers only differ from standard affine layers by the way their parameters are updated during training.
- *Skip connections*: via the DAG structure, the outputs of any past layers can be added to the pre-activation of any neuron by adding connections from these layers to the given neuron.
- *Convolutional layers*: consider them as (doubly) circulant/Toeplitz fully connected layers.

2.2 PATH-LIFTING, PATH-ACTIVATIONS AND PATH-NORMS

Given a general DAG ReLU network G as in Definition 2.2, it is possible to define a set of paths \mathcal{P}^G , a path-lifting Φ^G and path-activations \mathbf{A}^G , see Definition A.3 in the supplementary. The L^q path-norm is simply $\|\Phi^G(\boldsymbol{\theta})\|_q$. Some bounds we later establish also exploit so-called mixed path-norms $\|\Phi^G(\boldsymbol{\theta})\|_{q,r}$ ($= \|\Phi^G(\boldsymbol{\theta})\|_q$ when $q = r$) introduced in Definition A.4. Superscript G is omitted when obvious from the context. The interest is that (mixed) path-norms, which are easy to compute, can be interpreted as Lipschitz bounds of the network, smaller than another Lipschitz bound based on products of operator norms. We give here a high-level overview of the definitions and properties, and refer to Appendix A for formal definitions, proofs and technical details. We highlight that the definitions and properties coincide with previously known ones in classical simpler settings.

Path-lifting and path-activations: fundamental properties. The path-lifting Φ and the path-activations \mathbf{A} are defined to ensure the next fundamental properties: 1) for each parameter $\boldsymbol{\theta}$, the path-lifting $\Phi(\boldsymbol{\theta}) \in \mathbb{R}^{\mathcal{P}}$ is independent of the inputs x , and polynomial in the parameters $\boldsymbol{\theta}$ in a way that it is invariant under all neuron-wise rescaling symmetries⁴ networks.; 2) $\mathbf{A}(\boldsymbol{\theta}, x) \in \mathbb{R}^{\mathcal{P} \times (d_{in}+1)}$ takes a finite number of values and is piece-wise constant as a function of $(\boldsymbol{\theta}, x)$; and 3) denoting $\Phi^{\rightarrow v}$ and $\mathbf{A}^{\rightarrow v}$ to be the same objects but associated with the graph deduced from G by keeping only the largest subgraph of G with the same inputs as G and with single output v , the output of every neuron v can be written as

$$v(\boldsymbol{\theta}, x) = \left\langle \Phi^{\rightarrow v}(\boldsymbol{\theta}), \mathbf{A}^{\rightarrow v}(\boldsymbol{\theta}, x) \begin{pmatrix} x \\ 1 \end{pmatrix} \right\rangle. \quad (2)$$

Compared to previous definitions given in simpler models (no k -max-pooling even for a single given k , no skip connections, no biases, one-dimensional output and/or layered network) (Kawaguchi et al., 2017; Bona-Pellissier et al., 2022; Stock & Gribonval, 2023), the main novelty is essentially to properly define the path-activations $\mathbf{A}(\boldsymbol{\theta}, x)$ in the presence of $*$ -max-pooling neurons: when going through a k -max-pooling neuron, a path stays active only if the previous neuron of the path is the first in lexicographic order to be the k -th largest input of this pooling neuron.

Path-norms are easy to compute. It is mentioned in Dziugaite et al. (2020, Appendix C.6.5) and Jiang et al. (2020, Equation (44)) (without proof) that for layered fully-connected ReLU networks without biases, the L^2 path-norm can be computed in a single forward pass with the formula: $\|\Phi(\boldsymbol{\theta})\|_2^2 = \|R_{|\boldsymbol{\theta}|^2}(\mathbf{1})\|_1$, where $|\alpha|^q$ is the vector α with $x \mapsto |x|^q$ applied coordinate-wise and where $\mathbf{1}$ is the constant input equal to one. This can be proved in a straightforward way, using Equation (2), and extended to mixed path-norm: $\|\Phi(\boldsymbol{\theta})\|_{q,r} = \| |R_{|\boldsymbol{\theta}|^q}(\mathbf{1})|^{1/q} \|_r$. However, Appendix A

⁴Because of positive-homogeneity of the considered activations functions, the realized function is preserved (Stock & Gribonval, 2023) when the incoming weights and the bias of a neuron are multiplied by $\lambda > 0$, while its outgoing weights are divided by λ . Path-norms inherit such symmetries and are further invariant to certain neuron permutations, typically within each layer in the case of layered fully-connected

shows that this formula is *false* as soon as there is at least one *-max-pooling neuron, and that easy computation remains possible by first replacing the activation function of *-max-pooling neurons with the identity before doing the forward pass. Average-pooling neurons also need to be explicitly modeled as described after Definition 2.2, to apply $x \mapsto |x|^q$ to their weights.

The mixed $L^{1,r}$ path-norm is a Lipschitz bound. Equation (2) is fundamental to understand the role of path-norms. It shows that Φ contains information about the slopes of the function realized by the network on each region where \mathcal{A} is constant. A formal result that goes in this direction is the Lipschitz bound $\|R_\theta(x) - R_\theta(x')\|_r \leq \|\Phi(\theta)\|_{1,r} \|x - x'\|_\infty$, previously only known in the specific case $r = 1$, for *scalar-valued* layered fully-connected ReLU networks (Neyshabur, 2017, before Section 3.4) (Furusho, 2020, Theorem 5), which does not require mixed path-norms since $\|\Phi(\theta)\|_{q,r} = \|\Phi(\theta)\|_q$ for *scalar-valued* networks. This is generalized to the more general case of Definition 2.2 in Lemma A.2. This allows for leveraging generic generalization bounds that apply to the set of all L -Lipschitz functions $f : [0, 1]^{d_{in}} \rightarrow [0, 1]$, however these bounds suffer from the curse of dimensionality (von Luxburg & Bousquet, 2004), unlike the bounds established in Section 3.

Path-norms tightly lower bound products of operator norms. For *layered fully-connected* ReLU networks (LFCN) with no bias, *i.e.*, $R_\theta(x) = M_D \text{ReLU}(M_{D-1} \dots \text{ReLU}(M_1 x))$, with matrices M_1, \dots, M_D , another known Lipschitz bound is (Neyshabur et al., 2018; Gonon et al., 2023) $\prod_{d=1}^D \|M_d\|_{q,\infty}$ with $q = 1$, where $\|M\|_{q,\infty}$ is the maximum L^q norm of a row of matrix M . This product is also used to derive generalization guarantees (Neyshabur et al., 2015; Bartlett et al., 2017; Golowich et al., 2018). So which one of path-norm and products of operator norms should be used? There are at least three reasons to consider the path-norm. **First**, it holds $\|\Phi(\theta)\|_{q,\infty} \leq \prod_{d=1}^D \|M_d\|_{q,\infty}$ (Theorem C.1), with equality if the parameters are properly rescaled. This is known for *scalar-valued* LFCNs without biases (Neyshabur et al., 2015, Theorem 5). Appendix A generalizes it to DAGs as in Definition 2.2. The difficulty is to define the equivalent of the product of operator norms with an arbitrary DAG and in the presence of biases. Apart from that, the proof is essentially the same as in Neyshabur et al. (2015), with a similar rescaling that leads to equality of both measures, see Algorithm 1. **Second**, there are cases where the product of operator norms is arbitrarily large while the path-norm is zero (see Figure 2 in Appendix C). Thus, it is *not desirable* to have a generalization bound that depends on this product of operator norms since, compared to the path-norm, it fails to capture the complexity of the network end-to-end by *decoupling the layers of neurons* one from each other. **Third**, it has been empirically observed that products of operator norms *negatively* correlate with the empirical generalization error while the path-norm *positively* correlates (Jiang et al., 2020, Table 2)(Dziugaite et al., 2020, Figure 1).

3 GENERALIZATION BOUND

The generalization bound of this section is based on path-norm for general DAG ReLU network. It encompasses modern networks, recovers or beats the sharpest known bounds of this type, and applies to the cross-entropy loss. The top-one accuracy loss is not directly covered, but can be controlled via a bound on the margin-loss, as detailed at the end of this section.

3.1 MAIN RESULT

To state the main result let us recall the definition of the generalization error.

Definition 3.1. (*Generalization error*) Consider an architecture G (Definition 2.2) with input and output dimensions d_{in} and d_{out} , and a so-called loss function $\ell : \mathbb{R}^{d_{out}} \times \mathbb{R}^{d_{out}} \rightarrow \mathbb{R}$. The ℓ -generalization error of parameters θ on a collection Z of $n \in \mathbb{N}_{>0}$ pairs of input/output $z_i = (x_i, y_i) \in \mathbb{R}^{d_{in}} \times \mathbb{R}^{d_{out}}$ and with respect to a probability measure μ on $\mathbb{R}^{d_{in}} \times \mathbb{R}^{d_{out}}$ is:

$$\ell\text{-generalization error}(\theta, Z, \mu) := \underbrace{\mathbb{E}_{(\mathbf{x}_0, \mathbf{y}_0) \sim \mu} (\ell(R_\theta(\mathbf{X}_0), \mathbf{Y}_0))}_{\text{test error}} - \underbrace{\frac{1}{n} \sum_{i=1}^n \ell(R_\theta(x_i), y_i)}_{\text{training error when trained on } Z} .$$

While all the other results hold for arbitrary biases, the next theorem holds **only if the *-max-pooling neurons have null biases**. For simplicity, we state the result when *all* the biases are null. We then explain how to extend the result when neurons $u \notin N_{*\text{-pool}}$ may have $b_u \neq 0$.

Theorem 3.1. Consider a ReLU neural network architecture G (Definition 2.2) with null biases, with input/output dimensions d_{in}/d_{out} . Denote D its depth (the maximal length of a path from an

input to an output), $P := |\{k \in \mathbb{N}_{>0}, \exists u \in N_{k\text{-pool}}\}|$ the number of distinct types of $*$ -max-pooling neurons in G , $K := \max_{u \in N_{*\text{-pool}}} |\text{ant}(u)|$ its maximal kernel size ($K := 1$ if $P = 0$). Assume $N_{\text{in}} \cap N_{\text{out}} = \emptyset$. Consider a so-called loss function $\ell : \mathbb{R}^{d_{\text{out}}} \times \mathbb{R}^{d_{\text{out}}} \rightarrow \mathbb{R}$ and $L > 0$ such that

$$\ell(\hat{y}_1, y) - \ell(\hat{y}_2, y) \leq L \|\hat{y}_1 - \hat{y}_2\|_2, \quad \forall y, \hat{y}_1, \hat{y}_2 \in \text{support}(\mathbf{Y}_1). \quad (3)$$

Consider $n + 1$ iid random variables $\mathbf{Z}_i = (\mathbf{X}_i, \mathbf{Y}_i) \sim \mu$, $0 \leq i \leq n$, with μ a probability measure on input/output pairs in $\mathbb{R}^{d_{\text{in}}} \times \mathbb{R}^{d_{\text{out}}}$, and denote $\mathbf{Z} = (\mathbf{Z}_i)_{i=1, \dots, n}$. Define $\sigma := (\mathbb{E}_{\mathbf{X}} \max(n, \sum_{i=1}^n \|\mathbf{X}_i\|_{\infty}^2))^{1/2}$.

For any set of parameters Θ and any estimator $\hat{\theta} : \mathbf{Z} \mapsto \hat{\theta}(\mathbf{Z}) \in \Theta$ it holds⁵⁶:

$$\mathbb{E}_{\mathbf{Z}} \ell\text{-generalization error}(\hat{\theta}(\mathbf{Z}), \mathbf{Z}, \mu) \leq \frac{4\sigma}{n} LC \sup_{\theta \in \Theta} \|\Phi(\theta)\|_1$$

with (log being the natural logarithm)

$$C := \left(D \log((3 + 2P)K) + \log \left(\frac{3 + 2P}{1 + P} d_{\text{in}} d_{\text{out}} \right) \right)^{1/2}.$$

Any neural network with nonzero biases can be transformed into an equivalent network (with same path-norms and same R_{θ}) with null biases for every $v \notin N_{*\text{-pool}}$: add an input neuron v_{bias} with constant input equal to one, add edges between this input neuron and every neuron $v \notin N_{*\text{-pool}}$ with parameter $\theta^{v_{\text{bias}} \rightarrow v} := b_v$, and set $b_v = 0$. Thus the same result holds with $b_v \neq 0$ for $v \notin N_{*\text{-pool}}$, with d_{in} replaced by $d_{\text{in}} + 1$ in the definition of C , and with an additional constant input coordinate equal to one in the definition of σ so that $\sigma = (\mathbb{E}_{\mathbf{X}} \max(n, \max_{u=1, \dots, d_{\text{in}}} \sum_{i=1}^n (\mathbf{X}_i)_u^2))^{1/2} \geq \sqrt{n}$. The proof in Appendix F is directly given for networks with nonzero biases (except $*$ -max-pooling neurons), using this construction.

Theorem 3.1 applies to the cross-entropy loss with $L = \sqrt{2}$ (see Appendix G) if the labels y are one-hot encodings⁷. A final softmax layer can be incorporated for free to the model by putting it in the loss. This does not change the bound since it is 1-Lipschitz with respect to the L^2 -norm (this is a simple consequence of the computations made in Appendix G).

On ImageNet, it holds $1/\sqrt{n} \leq \sigma/n \leq 2.6/\sqrt{n}$ (Section 4). This yields a bounds that decays in $\mathcal{O}(n^{-1/2})$ which is better than the generic $\mathcal{O}(n^{-1/d_{\text{in}}})$ generalization bound for Lipschitz functions (von Luxburg & Bousquet, 2004, Thm. 18) that suffer from the curse of dimensionality. Besides its wider range of applicability, this bounds also recovers or beats the sharpest known ones based on path-norm, see Table 1.

Finally, note that Theorem 3.1 can be tightened: the same bound holds without counting the identity neurons when computing D . Indeed, for any neural network and parameters θ , it is possible to remove all the neurons $v \in N_{\text{id}}$ by adding a new edge $u \rightarrow w$ for any $u \in \text{ant}(v), w \in \text{suc}(v)$ with new parameter $\theta^{u \rightarrow v} \theta^{v \rightarrow w}$ (if this edge already exists, just add the latter to its already existing parameter). This still realizes the same function, with the same path-norm, but with less neurons, and thus with D possibly decreased. The proof technique would also yield a tighter bound but not by much: the occurrences of 3 in C would be replaced by 2.

Sketch of proof for Theorem 3.1. The proof idea is explained below. Details are in Appendix F.

Already known ingredients. Classical arguments (Shalev-Shwartz & Ben-David, 2014, Theorem 26.3)(Maurer, 2016), that are valid for any model, bound the expected generalization error by the Rademacher complexity of the model. It remains to bound the latter, and this gets specific to neural

⁵The definition of the generalization error (Definition 3.1) has been given for deterministic θ to keep things simple. The careful reader will have noted that the term corresponding to the test error in Definition 3.1 has to be modified when θ is a function of \mathbf{Z} . Indeed, the expectation has to be taken on an iid copy \mathbf{Z}_0 conditionally on each $\mathbf{Z}_i, i = 1, \dots, n$: the test error should be defined as $\mathbb{E}_{\mathbf{Z}_0 \sim \mu} (\ell(R_{\hat{\theta}(\mathbf{Z})}(\mathbf{X}_0), \mathbf{Y}_0) | \mathbf{Z})$. This correct definition will be used in the proof, but it has no importance here to understand the statement of the theorem.

⁶Classical concentration results (Boucheron et al., 2013) can be used to deduce a bound that holds with high probability under additional mild assumptions on the loss.

⁷A vector y is a one-hot encoding of a class c if $y = (\mathbb{1}_{c'=c})_{c' \in \{1, \dots, d_{\text{out}}\}}$.

networks. In the case of a layered fully-connected ReLU neural network with no biases and scalar output (and no skip connections nor k -max-pooling even for a single given k), Golowich et al. (2018) proved that it is possible to bound this Rademacher complexity with no exponential factor in the depth, by peeling, one by one, each layer off the Rademacher complexity. To get more specific, for a class of functions \mathbf{F} and a function $\Psi : \mathbb{R} \rightarrow \mathbb{R}$, denote $\text{Rad} \circ \Psi(\mathbf{F}) = \mathbb{E}_{\varepsilon} \Psi(\sup_{f \in \mathbf{F}} \sum_{i=1}^n \varepsilon_i f(x_i))$ the Rademacher complexity of \mathbf{F} associated with n inputs x_i and Ψ , where the ε_i are iid Rademacher variables ($\varepsilon_i = 1$ or -1 with equal probability). The goal for a generalization bound is to bound this in the case $\Psi(x) = \text{id}(x) = x$. In the specific case where \mathbf{F}_D is the class of functions that correspond to layered fully-connected ReLU networks with depth D , assuming that some operator norm of each layer d is bounded by r_d , Golowich et al. (2018) basically guarantees $\text{Rad} \circ \Psi_{\lambda}(\mathbf{F}_D) \leq 2 \text{Rad} \circ \Psi_{\lambda r_D}(\mathbf{F}_{D-1})$ for every $\lambda > 0$, where $\Psi_{\lambda}(x) = \exp(\lambda x)$. Compared to previous works of Golowich et al. (2018) that were directly working with $\Psi = \text{id}$ instead of Ψ_{λ} , the important point is that working with Ψ_{λ} gets the 2 outside of the exponential. Iterating over the depth D , optimizing over λ , and taking a logarithm at the end yields (by Jensen’s inequality) a bound on $\text{Rad} \circ \text{id}(\mathbf{F}_D)$ with a dependence on D that grows as $\sqrt{D \log(2)}$ instead of 2^D for previous approaches.

Novelties for general DAG ReLU networks. Compared to the setup of Golowich et al. (2018), there are at least three difficulties to do something similar here. First, *the neurons are not organized in layers* as the model can be an arbitrary DAG. So what should be peeled off one by one? Second, *the neurons are not necessarily ReLU neurons* as their activation function might be the identity (average-pooling) or $*$ -max-pooling. Finally, Golowich et al. (2018) has a constraint on the weights of each layer, which makes it possible to pop out the constant r_d when layer d is peeled off. *Here, the only constraint is global*, since it constrains the paths of the network through $\|\Phi(\theta)\|_1 \leq r$. In particular, due to rescalings, the weights of a given neuron could be arbitrarily large or small under this constraint.

The first difficulty is primarily addressed using a new peeling lemma (Appendix E) that exploits a new contraction lemma (Appendix D). The second difficulty is resolved by splitting the ReLU, k -max-pooling and identity neurons in different groups before each peeling step. This makes the $\log(2)$ term in Golowich et al. (2018) a $\log(3 + 2P)$ term here (P being the number of different k ’s for which k -max-pooling neurons are considered). Finally, the third obstacle is overcome by *rescaling the parameters* to normalize the vector of incoming weights of each neuron. This type of rescaling has also been used in Neyshabur et al. (2015); Barron & Klusowski (2019). \square

Remark 3.1 (Improved bound with assumptions on $*$ -max-pooling neurons). *In the specific case where there is a single type of k -max-pooling neurons ($P = 1$), assuming that these k -max-pooling neurons are grouped in layers, and that there are no skip connections going over these k -max-pooling layers (satisfied by ResNets, not satisfied by U-nets), a sharpened peeling argument can yield the same bound but with C replaced by $C_{\text{sharpened}} = (D \log(3) + M \log(K) + \log((d_{\text{in}} + 1)d_{\text{out}}))^{1/2}$ with M being the number of k -max-pooling layers (cf. Appendix E). The details are tedious so we only mention this result without proof. This basically improves $\sqrt{D \log(5K)}$ into $\sqrt{D \log(3) + M \log(K)}$. For Resnet152, $K = 9$, $D = 152$ and $M = 1$, $\sqrt{D \log(5K)} \simeq 24$ while $\sqrt{D \log(3) + M \log(K)} \simeq 13$.*

3.2 HOW TO DEAL WITH THE TOP-1 ACCURACY LOSS?

Theorem 3.1 does not apply to the top-1 accuracy loss as Equation (3) cannot be satisfied for any finite $L > 0$ in general (see Appendix H). It is still possible to bound the expected (test) top-1 accuracy by the so-called *margin loss* achieved at training (Bartlett et al., 2017, Lemma A.4). The margin-loss is a relaxed definition of the top-1 accuracy loss. A corollary of Theorem 3.1 is the next result proved in Appendix I.

Theorem 3.2 (Bound on the probability of misclassification). *Consider the setting of Theorem 3.1. Assume that the labels are indices $y \in \{1, \dots, d_{\text{out}}\}$. For any $\gamma > 0$, it holds*

$$\mathbb{P} \left(\arg \max_c R_{\theta}(\mathbf{X}_1)_c \neq \mathbf{Y}_1 \right) \leq \frac{1}{n} \sum_{i=1}^n \mathbb{1}_{(R_{\hat{\theta}(z)}(\mathbf{X}_i))_{\mathbf{Y}_i} \leq \gamma + \max_{c \neq \mathbf{Y}_i} (R_{\hat{\theta}(z)}(\mathbf{X}_i))_c} + \frac{8\sigma}{n} C \frac{\sup_{\theta} \|\Phi(\theta)\|_1}{\gamma}. \quad (4)$$

Table 2: Numerical evaluations on ResNets and ImageNet1k with 2 significant digits. Multiplying by the Lipschitz constant L of the loss and the path-norm gives the bound in Theorem 3.1. The second line reports the values when the analysis is sharpened for max-pooling neurons, see Remark 3.1.

ResNet	18	34	50	101	152
$\frac{4}{\sqrt{n}}CB =$	0.088	0.11	0.14	0.19	0.23
$\frac{4}{\sqrt{n}}C_{\text{sharpened}}B =$	0.060	0.072	0.082	0.11	0.13

Table 3: Path-norms of pretrained ResNets available on PyTorch, computed in float32.

ResNet	18	34	50	101	152
$\ \Phi(\theta)\ _1$	1.3×10^{30}	overflow	overflow	overflow	overflow
$\ \Phi(\theta)\ _2$	2.5×10^2	1.1×10^2	2.0×10^8	2.9×10^9	8.9×10^{10}
$\ \Phi(\theta)\ _4$	7.2×10^{-6}	4.9×10^{-6}	6.7×10^{-4}	3.0×10^{-4}	1.5×10^{-4}

Note that the result is homogeneous: scaling both the outputs of the model and γ by the same scalar leaves the classifier *and the corresponding bound* unchanged.

4 EXPERIMENTS

Theorem 3.1 gives the first path-norm generalization bound that can be applied to modern networks (with average/ k -max-pooling, skip connections etc.). This bound is also the sharpest known bound of this type (Table 1). Since this bound is also easy to compute, the goal of this section is to numerically challenge for the first time the sharpest generalization bounds based on path-norm on modern networks. Note also that path-norms tightly lower bound products of operator norms (Appendix C) so that this also challenges the latter.

When would the bound be informative? For ResNets trained on ImageNet, the training error associated with cross-entropy is typically between 1 and 2, and the top-1 training error is typically less than 0.30. The same orders of magnitude apply to the empirical generalization error. To ensure that the test error (either for cross-entropy or top-1 accuracy) is of the same order as the training error, *the bound should basically be of order 1*.

For parameters θ learned from training data, Theorem 3.1 and Theorem 3.2 allow to bound the expected loss in terms of a performance measure (that depends on a free choice of $\gamma > 0$ for the top-1 accuracy) on training data plus a term bounded by $\frac{4\sigma}{n}C \times L \times \|\Phi(\theta)\|_1$. The Lipschitz constant L is $\sqrt{2}$ for cross-entropy, and $2/\gamma$ for the top-1 accuracy.

Evaluation of $\frac{4\sigma}{n}C$ for ResNets on ImageNet. We further bound σ/n by B/\sqrt{n} , where $B \simeq 2.6$ is the maximum L^∞ -norm of the images of ImageNet normalized for inference. We at most lose a factor B compared to the bound directly involving σ since it also holds $\sigma/n \geq 1/\sqrt{n}$ by definition of σ . We train on 99% of ImageNet so that $n = 1268355$. Moreover, recall that $C = (D \log((3 + 2P)K) + \log(\frac{3+2P}{1+P}(d_{\text{in}} + 1)d_{\text{out}}))^{1/2}$. For ResNets, $P = 1$ (as there are only classical max-pooling neurons, corresponding to k -max-pooling with $k = 1$), the kernel size is $K = 9$, $d_{\text{in}} = 224 \times 224 \times 3$, $d_{\text{out}} = 1000$, and the depth is $D = 2 + \# \text{ basic blocks} \times \# \text{ conv per basic block}$, with the different values available in Appendix J. The values for $4BC/\sqrt{n}$ are reported in Table 2. Given these results and the values of the Lipschitz constant L , on ResNet18, *the bound would be informative only when $\|\Phi(\theta)\|_1 \lesssim 10$ or $\|\Phi(\theta)\|_1/\gamma \lesssim 10$ respectively for the cross-entropy and the top-1 accuracy*.

We now compute the path-norms of trained ResNets, both dense and sparse, using the simple formula proved in Theorem A.1 in appendix.

L^1 path-norm of pretrained ResNets are 30 orders of magnitude too large. Table 3 shows that the L^1 path-norm is 30 orders of magnitude too large to make the bound informative for the cross-entropy loss. The choice of γ is discussed in Appendix J, where we observe that there is no possible choice that leads to an informative bound for top-1 accuracy in this situation.

Sparse ResNets can decrease the bounds by 13 orders of magnitude. We just saw that pretrained ResNets have very large L^1 path-norm. Does every network with a good test top-1 accuracy have

such a large L^1 path-norm? Since any zero in the parameters θ leads to many zero coordinates in $\Phi(\theta)$, we now investigate whether sparse versions of ResNet18 trained on ImageNet have a smaller path-norm. Sparse networks are obtained with iterative magnitude pruning plus rewinding, with hyperparameters similar to the one in Frankle et al. (2021, Appendix A.3). Results show that the L^1 path-norm decreases from $\simeq 10^{30}$ for the dense network to $\simeq 10^{17}$ after 19 pruning iterations, basically losing between a half and one order of magnitude per pruning iteration. Moreover, the test top-1 accuracy is better than with the dense network for the first 11 pruning iterations, and after 19 iterations, the test top-1 accuracy is still way better than what would be obtained by guessing at random, so this is still a non-trivial matter to bound the generalization error for the last iteration. Details are in Appendix J. This shows that there are indeed practically trainable networks with much smaller L^1 path-norm that perform well. It remains open whether alternative training techniques, possibly with path-norm regularization, could lead to networks combining good performance and informative generalization bounds.

Additional observations: increased depth and train size. In practice, increasing the size of the network (*i.e.* the number of parameters) or the number of training samples can improve generalization. We can, again, assess for the first time whether the bounds based on path-norms follows the same trend for standard modern networks. Table 3 shows that path-norms of pretrained ResNets available on PyTorch roughly increase with depth. This is complementary to Dziugaite et al. (2020, Figure 1) where it is empirically observed on *simple layered fully-connected* models that path-norm has difficulty to correlate positively with the generalization error when the depth grows. For increasing training sizes, we did not observe a clear trend for the L^1 path-norm, which seems to mildly evolve with the number of epochs rather than with the train size, see Appendix J for details.

5 CONCLUSION

Contribution. To the best of our knowledge, this work is the first to introduce path-norm related tools for general DAG ReLU networks (with average/*-max-pooling, skip connections), and Theorem 3.1 is the first generalization bound valid for such networks based on path-norm. This bound recovers or beats the sharpest known ones of the same type. Its ease of computation leads to the first experiments on modern networks that assess the promises of such approaches. A gap between theory and practice is observed for a dense version of ResNet18 trained with standard tools: the bound is 30 orders of magnitude too large on ImageNet.

Possible leads to close the gap between theory and practice. 1) Without changing the bound of Theorem 3.1, sparsity seems promising to reduce the path-norm by several orders of magnitude without changing the performance of the network. 2) Theorem 3.1 results from the worst situation (that can be met) where all the inputs activate all the paths of the network simultaneously. Bounds involving the expected path-activations could be tighter. The coordinates of $\Phi(\theta)$ are elementary bricks that can be summed to get the slopes of R_θ on the different region where R_θ is affine (Arora et al., 2017), $\|\Phi(\theta)\|_1$ is the sum of all the bricks in absolute value, resulting in a worst-case uniform bound for all the slopes. Ideally, the bound should rather depend on the expected slopes over the different regions, weighted by the probability of falling into these regions. 3) Weight sharing may leave room for sharpened analysis (Pitas et al., 2019; Galanti et al., 2023). 4) A k -max-pooling neuron with kernel size K only activates $1/K$ of the paths, but the bound sums the coordinates of Φ related to these K paths. This may lead to a bound K times too large in general (or even more in the presence of multiple maxpooling layers). 5) Possible bounds involving the L^q path-norm for $q > 1$ deserve a particular attention, since numerical evaluations show that they are several orders of magnitude below the L^1 norm.

Extensions to other architectures. Despite its applicability to a wide range of standard modern networks, the generalization bound in Theorem 3.1 does not cover networks with other activations than ReLU, identity, and *-max-pooling. The same proof technique could be extended to new activations that: 1) are positively homogeneous, so that the weights can be rescaled without changing the associated function; and 2) satisfy a contraction lemma similar to the one established here for ReLU and max neurons (typically requiring the activation to be Lipschitz). A plausible candidate is Leaky ReLU. For smooth approximations of the ReLU, such as the SiLU (for Efficient Nets) and the Hardswish (for MobileNet-V3), parts of the technical lemmas related to contraction may extend since they are Lipschitz, but these activations are not positively homogeneous.

ACKNOWLEDGMENTS

This work was supported in part by the AllegroAssai ANR-19-CHIA-0009 and NuSCAP ANR-20-CE48-0014 projects of the French Agence Nationale de la Recherche.

The authors thank the Blaise Pascal Center for the computational means. It uses the SIDUS (Quemener & Corvellec, 2013) solution developed by Emmanuel Quemener.

REFERENCES

- Pierre Alquier. User-friendly introduction to PAC-Bayes bounds. *CoRR*, abs/2110.11216, 2021. URL <https://arxiv.org/abs/2110.11216>.
- Cem Anil, James Lucas, and Roger B. Grosse. Sorting out Lipschitz function approximation. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 291–301. PMLR, 2019. URL <http://proceedings.mlr.press/v97/anil19a.html>.
- Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. *Electron. Colloquium Comput. Complex.*, 24:98, 2017. URL <https://eccc.weizmann.ac.il/report/2017/098>.
- Francis Bach. Learning from first principles, 2024. URL https://www.di.ens.fr/~fbach/lftp_book.pdf.
- Francis R. Bach. Breaking the curse of dimensionality with convex neural networks. *J. Mach. Learn. Res.*, 18:19:1–19:53, 2017. URL <http://jmlr.org/papers/v18/14-546.html>.
- Andrew R. Barron and Jason M. Klusowski. Complexity, statistical risk, and metric entropy of deep nets using total path variation. *CoRR*, abs/1902.00800, 2019. URL <http://arxiv.org/abs/1902.00800>.
- Peter L. Bartlett and Shahar Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *J. Mach. Learn. Res.*, 3:463–482, 2002. URL <http://jmlr.org/papers/v3/bartlett02a.html>.
- Peter L. Bartlett, Dylan J. Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6240–6249, 2017. URL <https://proceedings.neurips.cc/paper/2017/hash/b22b257ad0519d4500539da3c8bcf4dd-Abstract.html>.
- Joachim Bona-Pellissier, François Malgouyres, and François Bachoc. Local identifiability of deep relu neural networks: the theory. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/b0ae046e198a5e43141519868a959c74-Abstract-Conference.html.
- Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration inequalities*. Oxford University Press, Oxford, 2013. ISBN 978-0-19-953525-5. doi: 10.1093/acprof:oso/9780199535255.001.0001. URL <https://doi-org.acces.bibliotheque-diderot.fr/10.1093/acprof:oso/9780199535255.001.0001>. A nonasymptotic theory of independence, With a foreword by Michel Ledoux.
- Patrick L. Combettes and Jean-Christophe Pesquet. Lipschitz certificates for layered network structures driven by averaged activation operators. *SIAM J. Math. Data Sci.*, 2(2):529–557, 2020. doi: 10.1137/19M1272780. URL <https://doi.org/10.1137/19M1272780>.

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. ISBN 978-0-262-03384-8. URL <http://mitpress.mit.edu/books/introduction-to-algorithms>.
- Ronald A. DeVore, Boris Hanin, and Guergana Petrova. Neural network approximation. *Acta Numer.*, 30:327–444, 2021. doi: 10.1017/S0962492921000052. URL <https://doi.org/10.1017/S0962492921000052>.
- Gintare Karolina Dziugaite. *Revisiting Generalization for Deep Learning: PAC-Bayes, Flat Minima, and Generative Models*. PhD thesis, Department of Engineering University of Cambridge, 2018.
- Gintare Karolina Dziugaite and Daniel M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In Gal Elidan, Kristian Kersting, and Alexander Ihler (eds.), *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017. URL <http://auai.org/uai2017/proceedings/papers/173.pdf>.
- Gintare Karolina Dziugaite, Alexandre Drouin, Brady Neal, Nitarshan Rajkumar, Ethan Caballero, Linbo Wang, Ioannis Mitliagkas, and Daniel M. Roy. In search of robust measures of generalization. In Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL <https://proceedings.neurips.cc/paper/2020/hash/86d7c8a08b4aaa1bc7c599473f5ddda-Abstract.html>.
- Weinan E, Chao Ma, and Lei Wu. The Barron space and the flow-induced function spaces for neural network models. *Constr. Approx.*, 55(1):369–406, 2022. ISSN 0176-4276. doi: 10.1007/s00365-021-09549-y. URL <https://doi-org.acces.bibliotheque-diderot.fr/10.1007/s00365-021-09549-y>.
- Jonathan Frankle, David J. Schwab, and Ari S. Morcos. The early phase of neural network training. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL <https://openreview.net/forum?id=Hk1liRNFwS>.
- Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Pruning neural networks at initialization: Why are we missing the mark? In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL <https://openreview.net/forum?id=Ig-VyQc-MLK>.
- Yasutaka Furusho. *Analysis of Regularization and Optimization for Deep Learning*. PhD thesis, Nara Institute of Science and Technology, 2020.
- Tomer Galanti, Mengjia Xu, Liane Galanti, and Tomaso A. Poggio. Norm-based generalization bounds for compositionally sparse neural networks. *CoRR*, abs/2301.12033, 2023. doi: 10.48550/arXiv.2301.12033. URL <https://doi.org/10.48550/arXiv.2301.12033>.
- Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet (eds.), *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, volume 75 of *Proceedings of Machine Learning Research*, pp. 297–299. PMLR, 2018. URL <http://proceedings.mlr.press/v75/golowich18a.html>.
- Antoine Gonon, Nicolas Brisebarre, Rémi Gribonval, and Elisa Riccietti. Approximation speed of quantized versus unquantized relu neural networks and beyond. *IEEE Trans. Inf. Theory*, 69(6): 3960–3977, 2023. doi: 10.1109/TIT.2023.3240360. URL <https://doi.org/10.1109/TIT.2023.3240360>.
- Antoine Gonon, Nicolas Brisebarre, Elisa Riccietti, and Rémi Gribonval. Code for reproducible research - A path-norm toolkit for modern networks: consequences, promises and challenges, March 2024. URL <https://hal.science/hal-04498597>. It is the code tagged with v1.0.0 at https://github.com/agonon/pathnorm_toolkit, and any updates will be available directly on that git repository.

- Benjamin Guedj. A primer on PAC-Bayesian learning. *CoRR*, abs/1901.05353, 2019. URL <http://arxiv.org/abs/1901.05353>.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL <https://doi.org/10.1109/CVPR.2016.90>.
- Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL <https://openreview.net/forum?id=SJgIPJBFvH>.
- Sham M. Kakade, Karthik Sridharan, and Ambuj Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou (eds.), *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pp. 793–800. Curran Associates, Inc., 2008. URL <https://proceedings.neurips.cc/paper/2008/hash/5b69b9cb83065d403869739ae7f0995e-Abstract.html>.
- Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *CoRR*, abs/1710.05468, 2017. URL <http://arxiv.org/abs/1710.05468>.
- Michel Ledoux and Michel Talagrand. *Probability in Banach spaces*, volume 23 of *Ergebnisse der Mathematik und ihrer Grenzgebiete (3) [Results in Mathematics and Related Areas (3)]*. Springer-Verlag, Berlin, 1991. ISBN 3-540-52013-9. doi: 10.1007/978-3-642-20212-4. URL <https://doi.org/10.1007/978-3-642-20212-4>. Isoperimetry and processes.
- Sibylle Marcotte, Rémi Gribonval, and Gabriel Peyré. Abide by the law and follow the flow: Conservation laws for gradient flows. *CoRR*, abs/2307.00144, 2023. doi: 10.48550/arXiv.2307.00144. URL <https://doi.org/10.48550/arXiv.2307.00144>.
- Andreas Maurer. A vector-contraction inequality for rademacher complexities. In Ronald Ortner, Hans Ulrich Simon, and Sandra Zilles (eds.), *Algorithmic Learning Theory - 27th International Conference, ALT 2016, Bari, Italy, October 19-21, 2016, Proceedings*, volume 9925 of *Lecture Notes in Computer Science*, pp. 3–17, 2016. doi: 10.1007/978-3-319-46379-7_1. URL https://doi.org/10.1007/978-3-319-46379-7_1.
- Vaishnavh Nagarajan and J. Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d’Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 11611–11622, 2019. URL <https://proceedings.neurips.cc/paper/2019/hash/05e97c207235d63ceb1db43c60db7bbb-Abstract.html>.
- Behnam Neyshabur. Implicit regularization in deep learning. *CoRR*, abs/1709.01953, 2017. URL <http://arxiv.org/abs/1709.01953>.
- Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In Peter Grünwald, Elad Hazan, and Satyen Kale (eds.), *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, volume 40 of *JMLR Workshop and Conference Proceedings*, pp. 1376–1401. JMLR.org, 2015. URL <http://proceedings.mlr.press/v40/Neyshabur15.html>.
- Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5947–5956, 2017. URL <https://proceedings.neurips.cc/paper/2017/hash/10ce03aled01077e3e289f3e53c72813-Abstract.html>.

- Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=Skz_WfbCZ.
- Guillermo Valle Pérez and Ard A. Louis. Generalization bounds for deep learning. *CoRR*, abs/2012.04115, 2020. URL <https://arxiv.org/abs/2012.04115>.
- Konstantinos Pitas, Andreas Loukas, Mike Davies, and Pierre Vandergheynst. Some limitations of norm based generalization bounds in deep neural networks. *CoRR*, abs/1905.09677, 2019. URL <http://arxiv.org/abs/1905.09677>.
- E. Quemener and M. Corvellec. SIDUS—the Solution for Extreme Deduplication of an Operating System. *Linux Journal*, 2013.
- Michael Eli Sander, Joan Puigcerver, Josip Djolonga, Gabriel Peyré, and Mathieu Blondel. Fast, differentiable and sparse top-k: a convex analysis perspective. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 29919–29936. PMLR, 2023. URL <https://proceedings.mlr.press/v202/sander23a.html>.
- Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014. ISBN 978-1-10-705713-5. URL <http://www.cambridge.org/de/academic/subjects/computer-science/pattern-recognition-and-machine-learning/understanding-machine-learning-theory-algorithms>.
- Pierre Stock and Rémi Gribonval. An embedding of ReLU networks and an analysis of their identifiability. *Constr. Approx.*, 57(2):853–899, 2023. ISSN 0176-4276,1432-0940. doi: 10.1007/s00365-022-09578-1. URL <https://doi.org/10.1007/s00365-022-09578-1>.
- Ulrike von Luxburg and Olivier Bousquet. Distance-based classification with Lipschitz functions. *J. Mach. Learn. Res.*, 5:669–695, 2004. URL <http://jmlr.org/papers/volume5/luxburg04b/luxburg04b.pdf>.
- Martin J. Wainwright. *High-dimensional statistics*, volume 48 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, Cambridge, 2019. ISBN 978-1-108-49802-9. doi: 10.1017/9781108627771. URL <https://doi-org.acces.bibliotheque-diderot.fr/10.1017/9781108627771>. A non-asymptotic viewpoint.
- Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM*, 64(3):107–115, 2021. doi: 10.1145/3446776. URL <https://doi.org/10.1145/3446776>.
- Shuxin Zheng, Qi Meng, Huishuai Zhang, Wei Chen, Nenghai Yu, and Tie-Yan Liu. Capacity control of ReLU neural networks by basis-path norm. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 5925–5932. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33015925. URL <https://doi.org/10.1609/aaai.v33i01.33015925>.

Supplementary material

A MODEL’S BASICS

Definition A.3 introduces the path-lifting and the path-activations associated with the general model described in Definition 2.2. We choose to call it path-lifting as in Bona-Pellissier et al. (2022) (and not an embedding as initially named in Stock & Gribonval (2023), since it is not injective). Formally, a lifting in the sense of category theory would factorize the mapping $\theta \mapsto R_\theta$, which is not the case of $\theta \mapsto \Phi(\theta)$, but is the case with the extended mapping $\theta \rightarrow (\text{sgn}(\theta), \Phi(\theta))$. We chose the name lifting for $\Phi(\theta)$ for the sake of brevity.

Definition A.1 (Paths and depth in a DAG). *Consider a DAG $G = (N, E)$ as in Definition 2.2. A path of G is any sequence of neurons v_0, \dots, v_d such that each $v_i \rightarrow v_{i+1}$ is an edge in G . Such a path is denoted $p = v_0 \rightarrow \dots \rightarrow v_d$. This includes paths reduced to a single $v \in N$, denoted $p = v$. The length of a path is $\text{length}(p) = d$ (the number of edges). We will denote $p_\ell := v_\ell$ the ℓ -th neuron for a general $\ell \in \{0, \dots, \text{length}(p)\}$ and use the shorthand $p_{\text{end}} = v_{\text{length}(p)}$ for the last neuron. The depth of the graph G is the maximum length over all of its paths. If $v_{d+1} \in \text{suc}(p_{\text{end}})$ then $p \rightarrow v_{d+1}$ denotes the path $v_0 \rightarrow \dots \rightarrow v_d \rightarrow v_{d+1}$. We denote by \mathcal{P}^G (or simply \mathcal{P}) the set of paths ending at an output neuron of G .*

Definition A.2 (Sub-graph ending at a given neuron). *Given a neuron v of a DAG G , we denote $G^{\rightarrow v}$ the graph deduced from G by keeping only the largest subgraph with the same inputs as G and with v as a single output: every neuron u with no path to reach v through the edges of G is removed, as well as all its incoming and outgoing edges. We will use the shorthand $\mathcal{P}^{\rightarrow v} := \mathcal{P}^{G^{\rightarrow v}}$ to denote the set of paths in G ending at v .*

Definition A.3 (Path-lifting and path-activations). *Consider a ReLU neural network architecture G as in Definition 2.2 and parameters $\theta \in \mathbb{R}^G$ associated with G . For $p \in \mathcal{P}$, define*

$$\Phi_p(\theta) := \begin{cases} \prod_{\ell=1}^{\text{length}(p)} \theta^{v_{\ell-1} \rightarrow v_\ell} & \text{if } p_0 \in N_{\text{in}}, \\ b_{p_0} \prod_{\ell=1}^{\text{length}(p)} \theta^{v_{\ell-1} \rightarrow v_\ell} & \text{otherwise,} \end{cases}$$

where an empty product is equal to 1 by convention. The path-lifting $\Phi^G(\theta)$ of θ is

$$\Phi^G(\theta) := (\Phi_p(\theta))_{p \in \mathcal{P}^G}.$$

This is often denoted Φ when the graph G is clear from the context. We will use the shorthand $\Phi^{\rightarrow v} := \Phi^{G^{\rightarrow v}}$ to denote the path-lifting associated with $G^{\rightarrow v}$ (Definition A.2).

Consider an input x of G . The activation of an edge $u \rightarrow v$ on (θ, x) is defined to be $a_{u \rightarrow v}(\theta, x) := 1$ when v is an identity neuron; $a_{u \rightarrow v}(\theta, x) := \mathbb{1}_{v(\theta, x) > 0}$ when v is a ReLU neuron; and when v is a k -max-pooling neuron, define $a_{u \rightarrow v}(\theta, x) := 1$ if the neuron u is the first in $\text{ant}(v)$ in lexicographic order to satisfy $u(\theta, x) := k\text{-pool}((w(\theta, x))_{w \in \text{ant}(v)})$ and $a_{u \rightarrow v}(\theta, x) := 0$ otherwise. The activation of a neuron v on (θ, x) is defined to be $a_v(\theta, x) := 1$ if v is an input neuron, an identity neuron, or a k -max-pooling neuron, and $a_v(\theta, x) := \mathbb{1}_{v(\theta, x) > 0}$ if v is a ReLU neuron. We then define the activation of a path $p \in \mathcal{P}$ with respect to input x and parameters θ as: $a_p(\theta, x) := a_{p_0}(\theta, x) \prod_{\ell=1}^{\text{length}(p)} a_{v_{\ell-1} \rightarrow v_\ell}(\theta, x)$ (with an empty product set to one by convention). Consider a new symbol v_{bias} that is not used for denoting neurons. The path-activations matrix $\mathbf{A}(\theta, x)$ is defined as the matrix in $\mathbb{R}^{\mathcal{P} \times (N_{\text{in}} \cup \{v_{\text{bias}}\})}$ such that for any path $p \in \mathcal{P}$ and neuron $u \in N_{\text{in}} \cup \{v_{\text{bias}}\}$

$$(\mathbf{A}(\theta, x))_{p, u} := \begin{cases} a_p(\theta, x) \mathbb{1}_{p_0 = u} & \text{if } u \in N_{\text{in}}, \\ a_p(\theta, x) & \text{otherwise when } u = v_{\text{bias}}. \end{cases}$$

The next lemma shows how the path-lifting and the path-activations offer an equivalent way to define the model of Definition 2.2.

Lemma A.1. *Consider a ReLU network as in Definition 2.2. For every neuron v , every input x and every parameters θ :*

$$v(\theta, x) = \left\langle \Phi^{\rightarrow v}(\theta), \mathbf{A}^{\rightarrow v}(\theta, x) \begin{pmatrix} x \\ 1 \end{pmatrix} \right\rangle. \quad (5)$$

Proof of Lemma A.1. For any neuron v , recall that $\mathcal{P}^{\rightarrow v}$ is the set of paths ending at neuron v (Definition A.2). We want to prove

$$\begin{aligned} v(\boldsymbol{\theta}, x) &= \left\langle \Phi^{\rightarrow v}(\boldsymbol{\theta}), \mathbf{A}^{\rightarrow v}(\boldsymbol{\theta}, x) \begin{pmatrix} x \\ 1 \end{pmatrix} \right\rangle \\ &= \sum_{p \in \mathcal{P}^{\rightarrow v}} \Phi_p(\boldsymbol{\theta}) a_p(\boldsymbol{\theta}, x) x_{p_0}. \end{aligned} \quad (6)$$

where we recall that p_0 denotes the first neuron of a path p (Definition A.1). For paths starting at an input neuron, x_{p_0} is simply the corresponding coordinate of x . For other paths, we use the convention $x_u := 1$ for any neuron $u \in N \setminus N_{\text{in}}$.

The proof of Equation (6) goes by induction on a topological sorting Cormen et al. (2009) of the neurons. We start with input neurons since by Definition 2.2, these are the ones without antecedents so they are the first to appear in a topological sorting. Consider an input neuron v . The only path in $\mathcal{P}^{\rightarrow v}$ is $p = v$. By Definition A.3, it holds $\Phi_p(\boldsymbol{\theta}) = 1$ (empty product) and $a_p(\boldsymbol{\theta}, x) = a_v(\boldsymbol{\theta}, x) = 1$. Moreover, we have $v(\boldsymbol{\theta}, x) = x_v$ (Definition 2.2). This proves Equation (6) for input neurons.

Now, consider $v \notin N_{\text{in}}$ and assume that Equation (6) holds true for every neuron $u \in \text{ant}(v)$. We first prove by cases that

$$v(\boldsymbol{\theta}, x) = a_v(\boldsymbol{\theta}, x) b_v + \sum_{u \in \text{ant}(v)} u(\boldsymbol{\theta}, x) a_{u \rightarrow v}(\boldsymbol{\theta}, x) \boldsymbol{\theta}^{u \rightarrow v}. \quad (7)$$

Case of an identity neuron. If v is an identity neuron, then by Definition 2.2

$$v(\boldsymbol{\theta}, x) = b_v + \sum_{u \in \text{ant}(v)} u(\boldsymbol{\theta}, x) \boldsymbol{\theta}^{u \rightarrow v}.$$

Moreover, by Definition A.3 we have $a_v(\boldsymbol{\theta}, x) = 1$ and $a_{u \rightarrow v}(\boldsymbol{\theta}, x) = 1$, since v is an identity neuron, so that the previous equality can also be written as

$$v(\boldsymbol{\theta}, x) = a_v(\boldsymbol{\theta}, x) b_v + \sum_{u \in \text{ant}(v)} u(\boldsymbol{\theta}, x) a_{u \rightarrow v}(\boldsymbol{\theta}, x) \boldsymbol{\theta}^{u \rightarrow v}.$$

This shows Equation (7) in this case.

Case of a ReLU neuron. If v is a ReLU neuron then similarly

$$\begin{aligned} v(\boldsymbol{\theta}, x) &= \text{ReLU} \left(b_v + \sum_{u \in \text{ant}(v)} u(\boldsymbol{\theta}, x) \boldsymbol{\theta}^{u \rightarrow v} \right) = \mathbb{1}_{v(\boldsymbol{\theta}, x) > 0} \left(b_v + \sum_{u \in \text{ant}(v)} u(\boldsymbol{\theta}, x) \boldsymbol{\theta}^{u \rightarrow v} \right) \\ &= \underbrace{\mathbb{1}_{v(\boldsymbol{\theta}, x) > 0}}_{=a_v(\boldsymbol{\theta}, x)} b_v + \sum_{u \in \text{ant}(v)} u(\boldsymbol{\theta}, x) \underbrace{\mathbb{1}_{v(\boldsymbol{\theta}, x) > 0}}_{=a_{u \rightarrow v}(\boldsymbol{\theta}, x)} \boldsymbol{\theta}^{u \rightarrow v} \end{aligned}$$

This shows again Equation (7).

Case of a k -max-pooling neuron. When v is a k -max-pooling neuron, it holds:

$$\begin{aligned} v(\boldsymbol{\theta}, x) &= k\text{-pool} \left((b_v + u(\boldsymbol{\theta}, x) \boldsymbol{\theta}^{u \rightarrow v})_{u \in \text{ant}(v)} \right) \\ &= b_v + \sum_{u \in \text{ant}(v)} \underbrace{\mathbb{1}_{u \text{ is the first to realize this } k\text{-pool}}}_{=a_{u \rightarrow v}(\boldsymbol{\theta}, x) \text{ (Definition A.3)}} u(\boldsymbol{\theta}, x) \boldsymbol{\theta}^{u \rightarrow v} \quad (\text{Definition 2.1}) \\ &= \underbrace{a_v(\boldsymbol{\theta}, x)}_{=1 \text{ (Definition A.3)}} b_v + \sum_{u \in \text{ant}(v)} u(\boldsymbol{\theta}, x) a_{u \rightarrow v}(\boldsymbol{\theta}, x) \boldsymbol{\theta}^{u \rightarrow v}. \end{aligned}$$

This finishes proving Equation (7) in every case. Using the induction hypothesis on the antecedents of v , Equation (7) implies

$$v(\boldsymbol{\theta}, x) = a_v(\boldsymbol{\theta}, x) b_v + \sum_{u \in \text{ant}(v)} \left(\sum_{p \in \mathcal{P}^{\rightarrow u}} \Phi_p(\boldsymbol{\theta}) a_p(\boldsymbol{\theta}, x) x_{p_0} \right) a_{u \rightarrow v}(\boldsymbol{\theta}, x) \boldsymbol{\theta}^{u \rightarrow v}.$$

We want to prove that this is equal to

$$\sum_{p \in \mathcal{P}^{\rightarrow v}} \Phi_p(\boldsymbol{\theta}) a_p(\boldsymbol{\theta}, x) x_{p_0}.$$

A path $\tilde{p} \in \mathcal{P}^{\rightarrow v}$ is either the path $\tilde{p} = v$ starting and ending at v , or it can be written in a unique way as $\tilde{p} = p \rightarrow v$ where $p \in \mathcal{P}^{\rightarrow u}$ is a path ending at an antecedent u of v . For the simple path $\tilde{p} = v$, it holds by definition (Definition A.3): $a_{\tilde{p}}(\boldsymbol{\theta}, x) = a_v(\boldsymbol{\theta}, x)$, $\Phi_{\tilde{p}}(\boldsymbol{\theta}) = b_v$ and by the convention used in this proof we have $x_{\tilde{p}_0} = x_v = 1$ since v is not an input neuron. This shows:

$$a_v(\boldsymbol{\theta}, x) b_v = \Phi_{\tilde{p}}(\boldsymbol{\theta}) a_{\tilde{p}}(\boldsymbol{\theta}, x) x_{\tilde{p}_0}.$$

When $\tilde{p} = p \rightarrow v$ as above, it holds by definition: $x_{p_0} = x_{\tilde{p}_0}$, $\Phi_{\tilde{p}}(\boldsymbol{\theta}) = \Phi_p(\boldsymbol{\theta}) \boldsymbol{\theta}^{u \rightarrow v}$ and $a_{\tilde{p}}(\boldsymbol{\theta}, x) = a_p(\boldsymbol{\theta}, x) a_{u \rightarrow v}(\boldsymbol{\theta}, x)$. It then holds:

$$\Phi_p(\boldsymbol{\theta}) a_p(\boldsymbol{\theta}, x) x_{p_0} a_{u \rightarrow v}(\boldsymbol{\theta}, x) \boldsymbol{\theta}^{u \rightarrow v} = \Phi_{\tilde{p}}(\boldsymbol{\theta}) a_{\tilde{p}}(\boldsymbol{\theta}, x) x_{\tilde{p}_0}.$$

This concludes the induction and proves the result. \square

A straightforward consequence of Lemma A.1 is that path-norms can be used to bound from above the Lipschitz constant of $x \mapsto R_{\boldsymbol{\theta}}(x)$.

Definition A.4. (Mixed path-norm) For $0 < q, r \leq \infty$, define:

$$\|\Phi(\boldsymbol{\theta})\|_{q,r} := \left\| \left(\|\Phi^{\rightarrow v}(\boldsymbol{\theta})\|_q \right)_{v \in N_{\text{out}}} \right\|_r. \quad (8)$$

Lemma A.2. Consider $0 < r \leq \infty$. For every parameters $\boldsymbol{\theta}$ and inputs x, x' :

$$\|R_{\boldsymbol{\theta}}(x) - R_{\boldsymbol{\theta}}(x')\|_r \leq \|\Phi(\boldsymbol{\theta})\|_{1,r} \|x - x'\|_{\infty}.$$

For $r = 1$, we simply have $\|\Phi(\boldsymbol{\theta})\|_{1,r} = \|\Phi(\boldsymbol{\theta})\|_1$. For this *specific value of r* , and in the *specific case* of layered fully-connected neural networks without biases, the conclusion of Lemma A.2 is already mentioned in Neyshabur (2017, before Section 3.4), and proven in Furusho (2020, Theorem 5). Lemma A.2 extends it to arbitrary DAG ReLU networks, and to arbitrary $r \in (0, \infty]$. This requires the introduction of *mixed* path-norms, which, to the best of our knowledge, have not been considered before in the literature.

Proof of Lemma A.2. Consider parameters $\boldsymbol{\theta}$. Consider inputs x, x' with the same path-activations with respect to $\boldsymbol{\theta}$: $\mathbf{A}(\boldsymbol{\theta}, x) = \mathbf{A}(\boldsymbol{\theta}, x')$. For $0 < r < \infty$:

$$\begin{aligned} \|R_{\boldsymbol{\theta}}(x) - R_{\boldsymbol{\theta}}(x')\|_r^r &\stackrel{\text{Lemma A.1}}{=} \sum_{v \in N_{\text{out}}} \left| \left\langle \Phi^{\rightarrow v}(\boldsymbol{\theta}), \mathbf{A}^{\rightarrow v}(\boldsymbol{\theta}, x) \begin{pmatrix} x \\ 1 \end{pmatrix} - \mathbf{A}^{\rightarrow v}(\boldsymbol{\theta}, x') \begin{pmatrix} x' \\ 1 \end{pmatrix} \right\rangle \right|^r \\ &\leq \sum_{v \in N_{\text{out}}} \|\Phi^{\rightarrow v}(\boldsymbol{\theta})\|_1^r \left\| \mathbf{A}^{\rightarrow v}(\boldsymbol{\theta}, x) \begin{pmatrix} x \\ 1 \end{pmatrix} - \mathbf{A}^{\rightarrow v}(\boldsymbol{\theta}, x') \begin{pmatrix} x' \\ 1 \end{pmatrix} \right\|_{\infty}^r \\ &\stackrel{\mathbf{A}(\boldsymbol{\theta}, x) = \mathbf{A}(\boldsymbol{\theta}, x')}{\leq} \sum_{v \in N_{\text{out}}} \|\Phi^{\rightarrow v}(\boldsymbol{\theta})\|_1^r \left\| \mathbf{A}^{\rightarrow v}(\boldsymbol{\theta}, x) \left(\begin{pmatrix} x \\ 1 \end{pmatrix} - \begin{pmatrix} x' \\ 1 \end{pmatrix} \right) \right\|_{\infty}^r \\ &\leq \sum_{v \in N_{\text{out}}} \|\Phi^{\rightarrow v}(\boldsymbol{\theta})\|_1^r \|x - x'\|_{\infty}^r \\ &= \|\Phi(\boldsymbol{\theta})\|_{1,r}^r \|x - x'\|_{\infty}^r. \end{aligned}$$

When $0 < r < \infty$, we just proved the claim locally on each region where the path-activations $\mathbf{A}(\boldsymbol{\theta}, \cdot)$ are constant. This stays true on the boundary of these regions by continuity. It then expands to the whole domain by triangular inequality because on any segment joining any pair of inputs, there is a finite number of times when the region changes. The proof can be easily adapted to $r = \infty$. \square

Another straightforward but important consequence of Lemma A.1 is that mixed path-norms can be computed in a single forward pass, up to replacing $*$ -max-pooling neurons with linear ones.

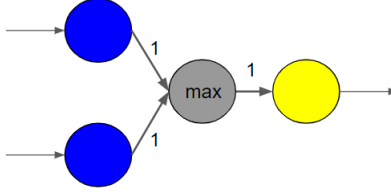


Figure 1: Example of a network where one must replace the max-pooling neuron to compute the path-norm with a single forward pass as in Equation (9).

Theorem A.1. Consider an architecture $G = (N, E, (\rho_v)_{v \in N \setminus N_{in}})$ as in Definition 2.2. Consider the architecture $\tilde{G} := (N, E, (\tilde{\rho}_v)_{v \in N \setminus N_{in}})$ with $\tilde{\rho}_v := \text{id}$ if $v \in N_{*-\text{pool}}$, and $\tilde{\rho}_v := \rho_v$ otherwise. Consider $q \in (0, \infty)$, $r \in (0, \infty]$ and arbitrary parameters $\theta \in \mathbb{R}^G = \mathbb{R}^{\tilde{G}}$. For a vector α , denote $|\alpha|^q$ the vector deduced from α by applying $x \mapsto |x|^q$ coordinate-wise. Denote by $\mathbf{1}$ the input full of ones. It holds:

$$\|\Phi(\theta)\|_{q,r} = \| |\mathbb{R}_{|\theta|^q}^{\tilde{G}}(\mathbf{1})|^{1/q} \|_r. \quad (9)$$

Moreover, the formula is false in general if the $*$ -max-pooling neurons have not been replaced with identity ones (i.e. if the forward pass is done on G rather than \tilde{G}).

Proof of Theorem A.1. Figure 1 shows that Equation (9) is false if the $*$ -max-pooling neurons have not been replaced with identity ones as the forward pass with input $\mathbf{1}$ yields output $\mathbf{1}$ while the L^1 path-norm is 2.

We now establish Equation (9). By continuity in θ of both sides of Equation (9), it is sufficient to prove it when every coordinate of θ in E is nonzero. Note that by Definition A.3, the path-lifting Φ^G and $\Phi^{\tilde{G}}$ associated respectively with G and \tilde{G} are the same since G and \tilde{G} have the same underlying DAG. Denote by $\Phi = \Phi^G = \Phi^{\tilde{G}}$ the common path-lifting, and by $a^{\tilde{G}}$ the path-activations associated with \tilde{G} . Denote by $\mathcal{P}^{\rightarrow v}$ the set of paths ending at a given neuron v of \tilde{G} . According to Lemma A.1, it holds for every output neuron v of \tilde{G} with $x = \mathbf{1}$, using the convention $x_u := 1$ if $u \notin N_{in}$

$$(R_{|\theta|^q}^{\tilde{G}}(\mathbf{1}))_v = \sum_{p \in \mathcal{P}^{\rightarrow v}} \Phi_p(|\theta|^q) a_p^{\tilde{G}}(|\theta|^q, x_{p_0}) = \sum_{p \in \mathcal{P}^{\rightarrow v}} \Phi_p(|\theta|^q) a_p^{\tilde{G}}(|\theta|^q, \mathbf{1}).$$

Since we restricted to θ_E with nonzero coordinates, $|\theta_E|^q$ has positive coordinates. As \tilde{G} has only identity or ReLU neurons, it follows by a simple induction on the neurons that $u(|\theta|^q, \mathbf{1}) > 0$ for every neuron u of \tilde{G} . Thus, every path $p \in \mathcal{P}^{\tilde{G}}$ is active, i.e., $a_p^{\tilde{G}}(|\theta|^q, \mathbf{1}) = 1$. Since by Definition A.3, $\Phi_p(|\theta|^q) = |\Phi_p(\theta)|^q$, we obtain:

$$(R_{|\theta|^q}^{\tilde{G}}(\mathbf{1}))_v = \sum_{p \in \mathcal{P}^{\rightarrow v}} |\Phi_p(\theta)|^q = \|\Phi^{\rightarrow v}(\theta)\|_q^q.$$

The claim follows by Definition A.4. \square

B NORMALIZED PARAMETERS

In the sequel it will be useful to restrict the analysis to *normalized* parameters.

Definition B.1. Consider $0 < q \leq \infty$. Parameters θ are said q -normalized if for every $v \in N \setminus (N_{out} \cup N_{in})$:

$$\|\Phi^{\rightarrow v}(\theta)\|_q = \left\| \begin{pmatrix} \theta^{\rightarrow v} \\ b_v \end{pmatrix} \right\|_q \in \{0, 1\},$$

and if this is 0 then it also holds $\theta^{v \rightarrow} = 0$ (*outgoing* edges of v).

Thanks to the rescaling-invariance of ReLU neural network parameterizations, Algorithm 1 allows to rescale *any* parameters θ into a normalized version $\hat{\theta}$ such that $R_{\hat{\theta}} = R_{\theta}$ and $\Phi(\theta) = \Phi(\hat{\theta})$

(Lemma B.1). The rescaling-invariance are due to the positive-homogeneity of every activation function $\rho \in \{\text{id}, \text{ReLU}, k\text{-pool}\}$: $\rho(x) = \frac{1}{\lambda}\rho(\lambda x)$ for every $\lambda > 0$ and $x \in \mathbb{R}$.

Algorithm 1 The first line of the algorithm considers a topological sorting of the neurons (Cormen et al., 2009, Section 22.4), *i.e.* an order on the neurons such that if $u \rightarrow v$ is an edge then u comes before v in this ordering. Such an order always exists (and it can be computed in linear time). Moreover, note that a classical max-pooling neuron v (corresponding to a k -max-pooling neuron with $k = 1$, constant incoming weights all equal to one, and biases set to zero) has not anymore its incoming weights equal to one after normalization, in general. This has no incidence on the validity of the generalization bound on classical max-pooling neurons: rescaling is only used in the proof to reduce to another representation of the parameters that realize the same function and that is more handy to work with.

Algorithm 1 Normalization of parameters for norm $q \in (0, \infty]$

```

1: Consider a topological sorting  $v_1, \dots, v_k$  of the neurons
2: for  $v = v_1, \dots, v_k$  do
3:   if  $v \notin N_{\text{in}} \cup N_{\text{out}}$  then
4:      $\lambda_v \leftarrow \left\| \begin{pmatrix} \boldsymbol{\theta}^{\rightarrow v} \\ b_v \end{pmatrix} \right\|_q$ 
5:     if  $\lambda_v = 0$  then
6:        $\boldsymbol{\theta}^{v \rightarrow} \leftarrow 0$ 
7:     else
8:        $\begin{pmatrix} \boldsymbol{\theta}^{\rightarrow v} \\ b_v \end{pmatrix} \leftarrow \frac{1}{\lambda_v} \begin{pmatrix} \boldsymbol{\theta}^{\rightarrow v} \\ b_v \end{pmatrix}$  ▷ normalize incoming weights and bias
9:        $\boldsymbol{\theta}^{v \rightarrow} \leftarrow \lambda_v \times \boldsymbol{\theta}^{v \rightarrow}$  ▷ rescale outgoing weights to preserve the function  $R_\theta$ 

```

Lemma B.1. Consider $q \in (0, \infty]$. If $\boldsymbol{\theta}$ is the output of Algorithm 1 for the input $\tilde{\boldsymbol{\theta}}$ then $\boldsymbol{\theta}$ is q -normalized (Definition B.1), $R_\boldsymbol{\theta} = R_{\tilde{\boldsymbol{\theta}}}$ and $\Phi(\boldsymbol{\theta}) = \Phi(\tilde{\boldsymbol{\theta}})$.

Proof. **Step 1.** We first prove that for every $v \in N \setminus (N_{\text{in}} \cup N_{\text{out}})$:

$$\left\| \begin{pmatrix} \boldsymbol{\theta}^{\rightarrow v} \\ b_v \end{pmatrix} \right\|_q \in \{0, 1\}. \quad (10)$$

Consider $v \notin N_{\text{in}} \cup N_{\text{out}}$. Right after v has been processed by the for loop of Algorithm 1 (line 2), it is clear that Equation (10) holds true for v . It remains to see that b_v and $\boldsymbol{\theta}^{\rightarrow v}$ are not modified until the end of Algorithm 1. A bias can only be modified when this is the turn of the corresponding neuron, so b_v is untouched after the iteration corresponding to v . And $\boldsymbol{\theta}^{\rightarrow v}$ can only be modified when treating antecedents of v , but since antecedents must be before v in any topological sorting, they cannot be processed after v . This proves that $\left\| \begin{pmatrix} \boldsymbol{\theta}^{\rightarrow v} \\ b_v \end{pmatrix} \right\|_q \in \{0, 1\}$.

Step 2. We now prove that if $\left\| \begin{pmatrix} \boldsymbol{\theta}^{\rightarrow v} \\ b_v \end{pmatrix} \right\|_q = 0$ then it also holds $\boldsymbol{\theta}^{v \rightarrow} = 0$ (*outgoing* edges of v).

Indeed, $\left\| \begin{pmatrix} \boldsymbol{\theta}^{\rightarrow v} \\ b_v \end{pmatrix} \right\|_q = 0$ implies $\lambda_v = 0$ in Algorithm 1, which implies that $\boldsymbol{\theta}^{v \rightarrow} = 0$ right after rescaling (line 6) at the end of the iteration corresponding to v . Because the next iterations can only involve multiplication of the coordinates of $\boldsymbol{\theta}^{v \rightarrow}$ by scalars, $\boldsymbol{\theta}^{v \rightarrow} = 0$ is also satisfied at the end of the algorithm. This proves the claim.

Step 3. In order to prove that $\boldsymbol{\theta}$ is q -normalized, it only remains to establish that $\|\Phi^{\rightarrow v}(\boldsymbol{\theta})\|_q = \left\| \begin{pmatrix} \boldsymbol{\theta}^{\rightarrow v} \\ b_v \end{pmatrix} \right\|_q$. We prove this by induction on a topological sorting of the neurons.

A useful fact for the induction is that for every $v \notin N_{\text{in}}$:

$$\Phi^{\rightarrow v}(\boldsymbol{\theta}) = \begin{pmatrix} (\Phi^{\rightarrow u}(\boldsymbol{\theta})\boldsymbol{\theta}^{u \rightarrow v})_{u \in \text{ant}(v)} \\ b_v \end{pmatrix} \quad (11)$$

where we recall that $\Phi^{\rightarrow u}(\cdot) = 1$ for input neurons u . Equation (11) holds because $\Phi^{\rightarrow v}$ is the path-lifting of $G^{\rightarrow v}$ (see Definition A.2), and the only paths in $G^{\rightarrow v}$ are $p = v$, and the paths going through antecedents of v (v has antecedents since it is not an input neuron).

Let's now start the induction. By definition, the first neuron $v \notin N_{\text{in}}$ in a topological sorting has only input neurons as antecedents. Therefore, $\Phi^{\rightarrow u}(\theta) = 1$ for every $u \in \text{ant}(v)$. Using Equation (11), we get

$$\|\Phi^{\rightarrow v}(\theta)\|_q^q = |b_v|^q + \sum_{u \in \text{ant}(v)} |\theta^{u \rightarrow v}|^q = \left\| \begin{pmatrix} \theta^{\rightarrow v} \\ b_v \end{pmatrix} \right\|_q^q$$

This shows the claim for v .

Assume the result to be true for $v \notin N_{\text{in}}$ and all the neurons before v in the considered topological order (in particular, for every $u \in \text{ant}(u)$). By Equation (11), we have

$$\|\Phi^{\rightarrow v}(\theta)\|_q^q = |b_v|^q + \sum_{u \in \text{ant}(v)} \|\Phi^{\rightarrow u}(\theta)\|_q^q |\theta^{u \rightarrow v}|^q.$$

The induction hypothesis guarantees that $c_u := \|\Phi^{\rightarrow u}(\theta)\|_q = \left\| \begin{pmatrix} \theta^{\rightarrow u} \\ b_u \end{pmatrix} \right\|_q$ for every $u \in \text{ant}(v)$. By Equation (10), we also have $c_u \in \{0, 1\}$ for every $u \in \text{ant}(v)$. When $c_u = 1$, it clearly holds $\|\Phi^{\rightarrow u}(\theta)\|_q^q |\theta^{u \rightarrow v}|^q = |\theta^{u \rightarrow v}|^q$. Otherwise, $c_u = 0$, hence $\theta^{u \rightarrow v} = 0$ as proved above, and we also obtain $\|\Phi^{\rightarrow u}(\theta)\|_q^q |\theta^{u \rightarrow v}|^q = |\theta^{u \rightarrow v}|^q$. We deduce that

$$\|\Phi^{\rightarrow v}(\theta)\|_q^q = |b_v|^q + \sum_{u \in \text{ant}(v)} |\theta^{u \rightarrow v}|^q = \left\| \begin{pmatrix} \theta^{\rightarrow v} \\ b_v \end{pmatrix} \right\|_q^q.$$

This concludes the induction.

Step 4. Each of the activation functions $\rho \in \{\text{id}, \text{ReLU}, k\text{-pool}\}$ is positive homogeneous, so classical arguments for layered fully-connected networks (see e.g. (Stock & Gribonval, 2023, Lemma 1 and Theorem 1)) are easily adapted to show that R_θ and $\Phi(\theta)$ are both unchanged at each iteration of Algorithm 1 where $\lambda_v \neq 0$. Inspecting the iterations involving a neuron such that $\lambda_v = 0$ reveals that: (i) $\Phi_p(\theta) = 0$ is unchanged for all paths p going through this neuron; (ii) $\Phi_p(\theta)$ is also unchanged for other paths since they only involve entries of θ that are kept identical. As a result, $\Phi(\theta)$ is also preserved at each such iteration. Finally, for each of the considered activations, $\lambda_v = 0$ implies that $v(\theta, x) = 0$ for any x (Equation (1)), so setting the outgoing weights $\theta^{v \rightarrow} = 0$ does not change R_θ . This completes the proof. \square

C PATH-NORMS AND PRODUCTS OF OPERATOR NORMS

For $0 < q \leq \infty$, recall that the mixed (quasi-)norm $\|M\|_{q, \infty}$ of a matrix M is:

$$\|M\|_{q, \infty} = \max_{\text{row of } M} \|\text{row}\|_q.$$

For $q = 1$, this is the operator norm of M induced by the L^∞ -norm on the input and output spaces. As a consequence, this is the Lipschitz constant of $x \mapsto Mx + b$ with respect to these norms, and by composition, it is well-known (see, e.g., Combettes & Pesquet (2020), where tighter bounds are also provided) that it can be used to derive a bound on the Lipschitz constant of layered fully-connected ReLU networks.

Lemma C.1 (see, e.g., Combettes & Pesquet (2020)). *For a layered fully-connected ReLU network with biases of the form*

$$R_\theta(x) = M_L \text{ReLU}(M_{L-1} \dots \text{ReLU}(M_1 x + b_1) + b_{L-1}) + b_L,$$

it holds for every x, x' and for $q = 1$:

$$\|R_\theta(x) - R_\theta(x')\|_\infty \leq \left(\prod_{\ell=1}^L \|M_\ell\|_{q, \infty} \right) \|x - x'\|_\infty.$$

Proof. As this is well-known we only reproduce the proof for $L = 2$, and the general case easily follows by an induction: (highlighting in orange x and x'):

$$\begin{aligned}
\|R_{\theta}(x) - R_{\theta}(x')\|_{\infty} &= \|M_2 \text{ReLU}(M_1 x + b_1) + b_2 - M_2 \text{ReLU}(M_1 x' + b_1) + b_2\|_{\infty} \\
&= \|M_2 (\text{ReLU}(M_1 x + b_1) - \text{ReLU}(M_1 x' + b_1))\|_{\infty} \\
&\leq \|M_2\|_{1,\infty} \|\text{ReLU}(M_1 x + b_1) - \text{ReLU}(M_1 x' + b_1)\|_{\infty} \\
&\leq \|M_2\|_{1,\infty} \|M_1 x + b_1 - M_1 x' + b_1\|_{\infty} && \text{ReLU is 1-Lipschitz} \\
&\leq \|M_2\|_{1,\infty} \|M_1 (x - x')\|_{\infty} \\
&\leq \|M_2\|_{1,\infty} \|M_1\|_{1,\infty} \|x - x'\|_{\infty}.
\end{aligned}$$

□

Another bound on the Lipschitz constant with respect to the L^{∞} -norm on both the input and the output is the mixed path-norm $\|\Phi(\theta)\|_{1,\infty}$ (Lemma A.2), which is even valid for an arbitrary DAG ReLU network. Is there any relation between the mixed path-norm $\|\Phi(\theta)\|_{1,\infty}$ and the product of operator norm $\prod_{\ell=1}^L \|M_{\ell}\|_{1,\infty}$ for layered fully-connected networks (LFCNs)?

Comparing the Lipschitz constants for a scalar-valued LFCN. It is known that for a *scalar-valued* (i.e., with $d_{\text{out}} = 1$) LFCN without biases, corresponding to $R_{\theta}(x) = M_L \text{ReLU}(M_{L-1} \dots \text{ReLU}(M_1 x))$, the bound $\|\Phi(\theta)\|_q \leq \prod_{\ell=1}^L \|M_{\ell}\|_{q,\infty}$ holds (Neyshabur et al., 2015, Theorem 5) for every $0 < q \leq \infty$, with equality if the parameters have been rescaled properly. In the specific case of a scalar-valued DAG ReLU network, it holds $\|\Phi(\theta)\|_q = \|\Phi(\theta)\|_{q,\infty}$ (Definition A.4). Thus, the result from Neyshabur et al. (2015) shows that the mixed path-norm $\|\Phi(\theta)\|_{1,\infty}$ is a tighter Lipschitz bound than the product of operator norms $\prod_{\ell=1}^L \|M_{\ell}\|_{1,\infty}$ in the specific case of *scalar-valued* LFCN.

We now extend the comparison made in Neyshabur et al. (2015) from a *scalar-valued LFCN* without biases to an *arbitrary DAG ReLU network*. We will again derive that the mixed path-norm is a lower bound on (an extension of) the product of layers' norm for a general *DAG ReLU network*. Compared to Neyshabur et al. (2015), this requires both our extension of the notion of path-norm to mixed path-norms, and extending the notion of product of layers' norm to deal with a DAG.

From now on, we always consider $q \in (0, \infty)$ and $r \in (0, \infty]$ if not specified otherwise.

From scalar-valued to vector-valued networks: from standard to mixed L^q -norms. Lemma A.2 shows that going from a *scalar-valued* network (LFCN or, more generally a DAG) to a *vector-valued* network requires going from standard L^q -norms $\|\Phi(\theta)\|_q$, as considered in Neyshabur et al. (2015), to mixed L^q -norms $\|\Phi(\theta)\|_{q,r}$, which is, to the best of our knowledge, first introduced in Definition A.4.

From LFCN to DAG: extending the product of layers' norm. There is no notion of a "layer" M_{ℓ} in a general DAG, hence the product $\prod_{\ell=1}^L \|M_{\ell}\|_{q,\infty}$ makes no sense anymore in this general context. We now introduce a quantity that generalizes this product for a general DAG.

Definition C.1. Consider $q \in (0, \infty)$. For practical purposes, we denote $\gamma_v := b_v$ when $v \in N \setminus N_{\text{in}}$, and $\gamma_v := 1$ if $v \in N_{\text{in}}$. For every path $p \in \mathcal{P}$, consider

$$\Pi(\theta, p, q) := \left(\sum_{\ell=0}^{\text{length}(p)} |\gamma_{p_{\ell}}|^q \prod_{k=\ell+1}^{\text{length}(p)} \|\theta^{\rightarrow p_k}\|_q^q \right)^{1/q}, \quad (12)$$

with the convention that an empty product is equal to one. For $q \in (0, \infty)$ and $r \in (0, \infty]$, define:

$$\Pi_{q,r}(\theta) := \left\| \left(\max_{p \in \mathcal{P}^{\rightarrow v}} \Pi(\theta, p, q) \right)_{v \in N_{\text{out}}} \right\|_r. \quad (13)$$

Let us check that $\Pi_{q,\infty}$ generalizes the product of layers' norm to an arbitrary DAG.

Lemma C.2. Consider $q \in (0, \infty)$. In a LFCN $R_\theta(x) = M_L \text{ReLU}(M_{L-1} \dots \text{ReLU}(M_1 x))$ with L layers and without biases, it holds:

$$\Pi_{q,\infty}(\theta) = \prod_{\ell=1}^L \|M_\ell\|_{q,\infty}.$$

Proof. In a LFCN, all the neurons of two consecutive layers are connected, so $\prod_{\ell=1}^L \|M_\ell\|_{q,\infty}$ is also the maximum over all paths starting from an input neuron (and ending at an output neuron, by definition of \mathcal{P}) of the product of L^q -norms:

$$\prod_{\ell=1}^L \|M_\ell\|_{q,\infty} = \max_{p \in \mathcal{P}, p_0 \in N_{\text{in}}} \prod_{\ell=1}^{\text{length}(p)} \|\theta^{\rightarrow p_\ell}\|_q.$$

Since there are no biases (corresponding to $\gamma_u = b_u = 0$ for every $u \notin N_{\text{in}}$) we have

$$\Pi(\theta, p, q) = \begin{cases} \prod_{\ell=1}^{\text{length}(p)} \|\theta^{\rightarrow p_\ell}\|_q & \text{if } p_0 \in N_{\text{in}}, \\ 0 & \text{otherwise.} \end{cases}$$

This shows the claim:

$$\begin{aligned} \Pi_{q,\infty}(\theta) &= \max_{v \in N_{\text{out}}} \max_{p \in \mathcal{P}^{\rightarrow v}} \Pi(\theta, p, q) = \max_{p \in \mathcal{P}} \Pi(\theta, p, q) \\ &= \max_{p \in \mathcal{P}, p_0 \in N_{\text{in}}} \prod_{\ell=1}^{\text{length}(p)} \|\theta^{\rightarrow p_\ell}\|_q = \prod_{\ell=1}^{\text{length}(p)} \|M_\ell\|_{q,\infty}. \quad \square \end{aligned}$$

The next theorem proves that the mixed path-norm is a lower bound on this extended product of layers' norm. In particular, this shows that the Lipschitz bound given by $\|\Phi(\theta)\|_{1,\infty}$ (which is computable in a single forward-pass by Theorem A.1) is always at least as good as the one given by $\prod_{\ell=1}^L \|M_\ell\|_{1,\infty}$ for a LFCN without biases. This extends as claimed the result from Neysshabur et al. (2015) to the vector-valued case.

Theorem C.1. Consider $q \in (0, \infty)$ and $r \in (0, \infty]$. For every DAG ReLU network (Definition 2.2) and every parameters θ :

$$\|\Phi(\theta)\|_{q,r} \leq \Pi_{q,r}(\theta).$$

If θ is q -normalized (Definition B.1) then:

$$\|\Phi(\theta)\|_{q,r} = \Pi_{q,r}(\theta) = \left\| \left(\left\| \begin{pmatrix} \theta^{\rightarrow v} \\ b_v \end{pmatrix} \right\|_q \right)_{v \in N_{\text{out}}} \right\|_r.$$

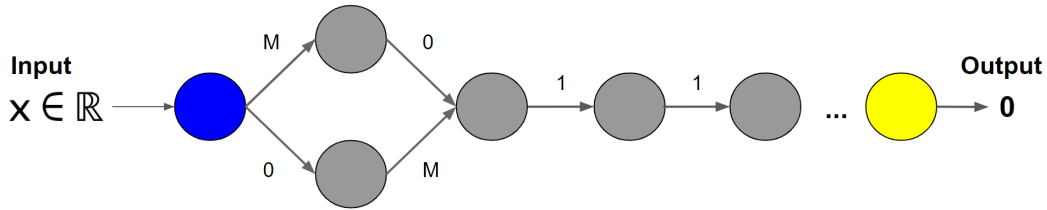


Figure 2: A network which path-norm is zero while the product of operator norms scales as M^2 .

Proof of Theorem C.1. First, when θ is q -normalized, by Definition B.1 and Equation (8):

$$\|\Phi(\theta)\|_{q,r} = \left\| \left(\left\| \begin{pmatrix} \theta^{\rightarrow v} \\ b_v \end{pmatrix} \right\|_q \right)_{v \in N_{\text{out}}} \right\|_r.$$

Let us prove by induction on a topological sorting of the neurons that for every $v \in N$:

$$\|\theta^{\rightarrow v}\|_q \leq \max_{p \in \mathcal{P}^{\rightarrow v}} \Pi(\theta, p, q) \quad (14)$$

with equality if θ is q -normalized. As a direct consequence of Equations (8) and (13), this will prove that $\|\Phi(\theta)\|_{q,r} \leq \Pi_{q,r}(\theta)$, with equality when θ is q -normalized, yielding all the claimed results.

We start with $v \in N_{\text{in}}$ since input neurons are the first to appear in a topological sorting. In this case, the only path in $\mathcal{P}^{\rightarrow v}$ is $p = v$, for which it holds $\Pi(\theta, p, q) = |\gamma_{p_0}| = |\gamma_v| = 1$ by Definition C.1. Moreover, we also have $\Phi^{\rightarrow v}(\theta) = 1$ (Definition A.3). This proves Equation (14) and the case of equality for input neurons even if θ is not normalized.

Now, consider $v \notin N_{\text{in}}$ and assume Equation (14), and the case of equality for q -normalized parameters, to be true for every antecedent of v . In this case, we have $\Phi^{\rightarrow v}(\theta) = \left(\begin{array}{c} (\Phi^{\rightarrow u}(\theta)\theta^{u \rightarrow v})_{u \in \text{ant}(v)} \\ b_v \end{array} \right)$ so it holds

$$\|\Phi^{\rightarrow v}(\theta)\|_q^q = |b_v|^q + \sum_{u \in \text{ant}(v)} \|\Phi^{\rightarrow u}(\theta)\|_q^q |\theta^{u \rightarrow v}|^q \leq |b_v|^q + \|\theta^{\rightarrow v}\|_q^q \max_{u \in \text{ant}(v)} \|\Phi^{\rightarrow u}(\theta)\|_q^q.$$

Using the induction hypothesis on every $u \in \text{ant}(v)$ yields:

$$\|\Phi^{\rightarrow v}(\theta)\|_q^q \leq |b_v|^q + \|\theta^{\rightarrow v}\|_q^q \max_{u \in \text{ant}(v)} \max_{p \in \mathcal{P}^{\rightarrow u}} (\Pi(\theta, p, q))^q$$

Consider $u \in \text{ant}(v)$ and $p \in \mathcal{P}^{\rightarrow u}$, and denote by $\tilde{p} = p \rightarrow v$ the path p concatenated with the edge $u \rightarrow v$. By Definition C.1, we have (highlighting in orange the important changes)

$$\begin{aligned} & |b_v|^q + \|\theta^{\rightarrow v}\|_q^q (\Pi(\theta, p, q))^q \\ &= \underbrace{|b_v|^q}_{=|\gamma_v|^q \text{ since } v \notin N_{\text{in}}} + \|\theta^{\rightarrow v}\|_q^q \sum_{\ell=0}^{\text{length}(p)} |\gamma_{p_\ell}|^q \prod_{k=\ell+1}^{\text{length}(p)} \|\theta^{\rightarrow p_k}\|_q^q \\ &= |\gamma_{\tilde{p}_{\text{end}}}|^q + \|\theta^{\rightarrow \tilde{p}_{\text{end}}}\|_q^q \sum_{\ell=0}^{\text{length}(\tilde{p})-1} |\gamma_{\tilde{p}_\ell}|^q \prod_{k=\ell+1}^{\text{length}(\tilde{p})-1} \|\theta^{\rightarrow \tilde{p}_k}\|_q^q = (\Pi(\theta, \tilde{p}, q))^q \end{aligned}$$

We deduce that

$$\|\Phi^{\rightarrow v}(\theta)\|_q^q \leq \max_{u \in \text{ant}(v)} \max_{p \in \mathcal{P}^{\rightarrow u}} (\Pi(\theta, p \rightarrow u, q))^q = \max_{\tilde{p} \in \mathcal{P}^{\rightarrow v}} (\Pi(\theta, \tilde{p}, q))^q.$$

This proves Equation (14) for v . To conclude the induction, it remains to treat the equality case for v assuming that θ is q -normalized. Since $v \notin N_{\text{in}}$, $\text{ant}(v) \neq \emptyset$, and since each neuron $u \in \text{ant}(v)$ cannot be an output neuron, the fact that θ is q -normalized implies by Definition B.1 that $\|\Phi^{\rightarrow u}(\theta)\|_q \in \{0, 1\}$, with $\theta^{u \rightarrow} = 0$ as soon as $\|\Phi^{\rightarrow u}(\theta)\|_q = 0$. This implies

$$\|\Phi^{\rightarrow u}(\theta)\|_q^q |\theta^{u \rightarrow v}|^q = |\theta^{u \rightarrow v}|^q$$

As a result

$$\sum_{u \in \text{ant}(v)} \|\Phi^{\rightarrow u}(\theta)\|_q^q |\theta^{u \rightarrow v}|^q = \sum_{u \in \text{ant}(v)} |\theta^{u \rightarrow v}|^q = \|\theta^{\rightarrow v}\|_q^q = \|\theta^{\rightarrow v}\|_q^q \max_{u \in \text{ant}(v)} \|\Phi^{\rightarrow u}(\theta)\|_q^q.$$

By the induction hypothesis, we also have

$$\max_{u \in \text{ant}(v)} \|\Phi^{\rightarrow u}(\theta)\|_q^q = \max_{u \in \text{ant}(v)} \max_{p \in \mathcal{P}^{\rightarrow u}} (\Pi(\theta, p, q))^q.$$

Putting everything together yields

$$\begin{aligned} \|\Phi^{\rightarrow v}(\theta)\|_q^q &= |b_v|^q + \sum_{u \in \text{ant}(v)} \|\Phi^{\rightarrow u}(\theta)\|_q^q |\theta^{u \rightarrow v}|^q \\ &= |b_v|^q + \|\theta^{\rightarrow v}\|_q^q \max_{u \in \text{ant}(v)} \max_{p \in \mathcal{P}^{\rightarrow u}} (\Pi(\theta, p, q))^q \\ &= \max_{p \in \mathcal{P}^{\rightarrow v}} \Pi(\theta, p, q)^q. \end{aligned}$$

This shows the case of equality and concludes the induction. \square

D RELEVANT (AND APPARENTLY NEW) CONTRACTION LEMMAS

The main result is Lemma D.1.

Lemma D.1. *Consider finite sets I, W, Z , and for each $z \in Z$, consider a set $T^z \subset (\mathbb{R}^W)^I$. We denote $t = (t_i)_{i \in I} \in T^z$ with $t_i = (t_{i,w})_{w \in W} \in \mathbb{R}^W$. Consider functions $f_{i,z} : \mathbb{R}^W \rightarrow \mathbb{R}$ and a finite family $\varepsilon = (\varepsilon_j)_{j \in J}$ of independent identically distributed Rademacher variables, with the index set J that will be clear from the context. Finally, consider a convex and non-decreasing function $g : \mathbb{R} \rightarrow \mathbb{R}$. Assume that at least one of the following setting holds.*

Setting 1: scalar input case. $|W| = 1$ and for every $i \in I$ and $z \in Z$, $f_{i,z}$ is 1-Lipschitz with $f_{i,z}(0) = 0$.

Setting 2: *-max-pooling case. For every $i \in I$ and $z \in Z$, there is $k_{i,z} \in \mathbb{N}_{>0}$ such that for every $t \in T^z$, $f_{i,z}(t) = t_{(k_{i,z})}$ is the $k_{i,z}$ -th largest coordinate of t .

Then we have:

$$\mathbb{E} \max_{z \in Z} \sup_{t \in T^z} g \left(\sum_{i \in I} \varepsilon_{i,z} f_{i,z}(t_i) \right) \leq \mathbb{E} \max_{z \in Z} \sup_{t \in T^z} g \left(\sum_{i \in I, w \in W} \varepsilon_{i,w,z} t_{i,w} \right). \quad (15)$$

The scalar input case is a simple application of Theorem 4.12 in [Ledoux & Talagrand \(1991\)](#). Indeed, for $z \in Z$ and $t \in T^z$, define $s(z, t) \in \mathbb{R}^{I \times Z}$ to be the matrix with coordinates $(i, v) \in I \times Z$ given by $[s(z, t)]_{i,v} = t_i$ if $v = z$, 0 otherwise. Define also $f_{i,v} = f_i$. Since $f_{i,v}(0) = 0$, it holds:

$$\sum_{i \in I} \varepsilon_{i,z} f_i(t_i) = \sum_{i \in I, v \in Z} \varepsilon_{i,v} f_{i,v}([s(z, t)]_{i,v}).$$

If $S := \{s(z, t), z \in Z, t \in T^z\}$ and $J := I \times Z$ then the result claimed in the scalar case reads

$$\mathbb{E} \sup_{s \in S} g \left(\sum_{j \in J} \varepsilon_j f_j(s_j) \right) \leq \mathbb{E} \sup_{s \in S} g \left(\sum_{j \in J} \varepsilon_j s_j \right).$$

The latter is true by Theorem 4.12 of [Ledoux & Talagrand \(1991\)](#). However, we present an additional proof below, which we employ to establish the new scenario involving *-max-pooling. This alternative proof closely follows the structure of the proof outlined in Theorem 4.12 of [Ledoux & Talagrand \(1991\)](#): the beginning of the proof is the same for the scalar case and the *-max-pooling case, and then the arguments become specific to each case.

Note that Theorem 4.12 of [Ledoux & Talagrand \(1991\)](#) does not apply for the *-max-pooling case because the t_i 's are now vectors. The most related result we could find is a vector-valued contraction inequality ([Maurer, 2016](#)) that is known in the specific case where $|Z| = 1$, g is the identity, and for arbitrary 1-Lipschitz functions $f_{i,z}$ such that $f_{i,z}(0) = 0$ (with a different proof, and with a factor $\sqrt{2}$ on the right-hand side). Here, the vector-valued case we are interested in is $f_{i,z} = k_{i,z}$ -pool and $g = \exp$, which is covered by Lemma D.1. We could not find it stated elsewhere.

In the proof of Lemma D.1, we reduce to the simpler case where $|Z| = 1$ and $|I| = 1$ that corresponds to the next lemma. Again, the scalar input case is given by [Ledoux & Talagrand \(1991, Equation \(4.20\)\)](#) while the *-max-pooling case is apparently new.

Lemma D.2. *Consider a finite set W , a set T of elements $t = (t_1, t_2) \in \mathbb{R}^W \times \mathbb{R}$ and a function $f : \mathbb{R}^W \rightarrow \mathbb{R}$. Consider also a convex non-decreasing function $F : \mathbb{R} \rightarrow \mathbb{R}$ and a family of iid Rademacher variables $(\varepsilon_j)_{j \in J}$ where J will be clear from the context. Assume that we are in one of the two following situations.*

Scalar input case. f is 1-Lipschitz, satisfies $f(0) = 0$ and has a scalar input ($|W| = 1$).

***-max-pooling case.** There is $k \in \mathbb{N}_{>0}$ such that f computes the k -th largest coordinate of its input.

Denoting $t_1 = (t_{1,w})_{w \in W}$, it holds:

$$\mathbb{E} \sup_{t \in T} F(\varepsilon_1 f(t_1) + t_2) \leq \mathbb{E} \sup_{t \in T} F \left(\sum_w \varepsilon_{1,w} t_{1,w} + t_2 \right).$$

The proof of Lemma D.2 is postponed. We now prove Lemma D.1.

Proof of Lemma D.1. First, because of the Lipschitz assumptions on the f_i 's and the convexity of g , everything is measurable and the expectations are well defined.

We prove the result by reducing to the simpler case of Lemma D.2. This is inspired by the reduction done in the proof of [Ledoux & Talagrand \(1991, Theorem 4.12\)](#) in the special case of scalar t_i 's ($|W| = 1$).

Reduce to the case $|Z| = 1$ by conditioning and iteration. For $z \in Z$, define

$$A_z := \sup_{t \in T^z} g \left(\sum_{i \in I} \varepsilon_{i,z} f_{i,z}(t_i) \right),$$

$$B_z := \sup_{t \in T^z} g \left(\sum_{i \in I, w \in W} \varepsilon_{i,w,z} t_{i,w} \right).$$

Lemma D.3 applies since these random variables are independent. Thus, it is enough to prove that for every $c \in [-\infty, \infty)$:

$$\mathbb{E} \max(A_z, c) \leq \mathbb{E} \max(B_z, c).$$

Define $F(x) = \max(g(x), c)$. This can be rewritten as (inverting the supremum and the maximum)

$$\mathbb{E} \sup_{t \in T^z} F \left(\sum_{i \in I} \varepsilon_{i,z} f_{i,z}(t_i) \right) \leq \mathbb{E} \sup_{t \in T^z} F \left(\sum_{i \in I, w \in W} \varepsilon_{i,w,z} t_{i,w} \right). \quad (16)$$

We just reduced to the case where there is a single z to consider, up to the price of replacing g by F . Since g and $x \mapsto \max(x, c)$ are non-decreasing and convex, so is F by composition. Alternatively, note that we could also have reduced to the case $|Z| = 1$ by defining $S := \{s(z, t), z \in Z, t \in T^z\}$ just as it is done right after the statement of Lemma D.1. In order to apply Lemma D.2, it remains to reduce to the case $|I| = 1$.

Reduce to the case $|I| = 1$ by conditioning and iteration. Lemma D.4 shows that in order to prove Equation (16), it is enough to prove that for every $i \in I$ and every subset $R \subset \mathbb{R}^W \times \mathbb{R}$, denoting $r = (r_1, r_2) \in \mathbb{R}^W \times \mathbb{R}$, it holds

$$\mathbb{E} \sup_{r \in R} F(\varepsilon_{i,z} f_{i,z}(r_1) + r_2) \leq \mathbb{E} \sup_{r \in R} F \left(\sum_{w \in W} \varepsilon_{i,w,z} r_{1,w} + r_2 \right).$$

We just reduced to the case $|I| = 1$ since one can now consider the indices i one by one. The latter inequality is now a direct consequence of Lemma D.2. This proves the result. \square

Lemma D.3. Consider a finite set Z and independent families of independent real random variables $(A_z)_{z \in Z}$ and $(B_z)_{z \in Z}$. If for every $z \in Z$ and every constant $c \in [-\infty, \infty)$, it holds $\mathbb{E} \max(A_z, c) \leq \mathbb{E} \max(B_z, c)$, then

$$\mathbb{E} \max_{z \in Z} A_z \leq \mathbb{E} \max_{z \in Z} B_z.$$

Proof of Lemma D.3. The proof is by conditioning and iteration. To prove the result, it is enough to prove that if

$$\mathbb{E} \max_{z \in Z} A_z \leq \mathbb{E} \max \left(\max_{z \in Z_1} A_z, \max_{z \in Z_2} B_z \right)$$

for some partition Z_1, Z_2 of Z , with Z_2 possibly empty for the initialization of the induction, then for every $z_0 \in Z_1$:

$$\mathbb{E} \max_{z \in Z} A_z \leq \mathbb{E} \max \left(\max_{z \in Z_1 \setminus \{z_0\}} A_z, \max_{z \in Z_2 \cup \{z_0\}} B_z \right),$$

with the convention that the maximum over an empty set is $-\infty$. Indeed, the claim would then come directly by induction on the size of Z_2 .

Now, consider an arbitrary partition Z_1, Z_2 of Z , with Z_2 possibly empty, and consider $z_0 \in Z_1$. It is then enough to prove that

$$\mathbb{E} \max \left(\max_{z \in Z_1} A_z, \max_{z \in Z_2} B_z \right) \leq \mathbb{E} \max \left(\max_{z \in Z_1 \setminus \{z_0\}} A_z, \max_{z \in Z_2 \cup \{z_0\}} B_z \right). \quad (17)$$

Define the random variable $C = \max(\max_{z \in Z_1 \setminus \{z_0\}} A_z, \max_{z \in Z_2} B_z)$ which may be equal to $-\infty$ when the maximum is over empty sets, and which is independent of A_{z_0} and B_{z_0} . It holds:

$$\max \left(\max_{z \in Z_1} A_z, \max_{z \in Z_2} B_z \right) = \max(A_{z_0}, C)$$

and

$$\max \left(\max_{z \in Z_1 \setminus \{z_0\}} A_z, \max_{z \in Z_2 \cup \{z_0\}} B_z \right) = \max(B_{z_0}, C).$$

Equation (17) is then equivalent to

$$\mathbb{E} \max(A_{z_0}, C) \leq \mathbb{E} \max(B_{z_0}, C)$$

with C independent of A_{z_0} and B_{z_0} . For a constant $c \in [-\infty, \infty)$, denote $A(c) = \mathbb{E} \max(A_{z_0}, c)$ and $B(c) = \mathbb{E} \max(B_{z_0}, c)$. We have:

$$\begin{aligned} \mathbb{E} \max(A_{z_0}, C) &= \mathbb{E} (\mathbb{E} (\max(A_{z_0}, C) | A_{z_0})) && \text{law of total expectation} \\ &= \mathbb{E} A(C) && \text{independence of } C \text{ and } A_{z_0}. \end{aligned}$$

and similarly $\mathbb{E} \max(B_{z_0}, C) = \mathbb{E} B(C)$. It is then enough to prove that $A(C) \leq B(C)$ almost surely. Since $C \in [-\infty, \infty)$, this is true by assumption. This proves the claims. \square

Lemma D.4. Consider finite sets I, W and independent families of independent real random variables $(\varepsilon_i)_{i \in I}$ and $(\varepsilon_{i,w})_{i \in I, w \in W}$. Consider functions $f_i : \mathbb{R}^W \rightarrow \mathbb{R}$ and $F : \mathbb{R} \rightarrow \mathbb{R}$ that are continuous. Assume that for every $i \in I$ and every subset $R \subset \mathbb{R}^W \times \mathbb{R}$, denoting $r = (r_1, r_2) \in R$ with $r_1 = (r_{1,w})_w \in \mathbb{R}^W$ and $r_2 \in \mathbb{R}$ the components of r , it holds

$$\mathbb{E} \sup_{r \in R} F(\varepsilon_i f_i(r_1) + r_2) \leq \mathbb{E} \sup_{r \in R} F\left(\sum_{w \in W} \varepsilon_{i,w} r_{1,w} + r_2\right).$$

Consider an arbitrary $T \subset (\mathbb{R}^W)^I$ and for $t = (t_i)_{i \in I} \in T$, denote $t_{i,w}$ the w -th coordinate of $t_i \in \mathbb{R}^W$. It holds:

$$\mathbb{E} \sup_{t \in T} F\left(\sum_{i \in I} \varepsilon_i f_i(t_i)\right) \leq \mathbb{E} \sup_{t \in T} F\left(\sum_{i \in I, w \in W} \varepsilon_{i,w} t_{i,w}\right).$$

Proof of Lemma D.4. The continuity assumption on F and the f_i 's is only used to make all the considered suprema measurable. The proof goes by conditioning and iteration. For any $J \subset I$, denote ε_J the family that contains both $(\varepsilon_j)_{j \in J}$ and $(\varepsilon_{j,w})_{j \in J, w \in W}$. Define

$$\begin{aligned} h_J(t, \varepsilon_J) &:= \sum_{j \in J} \varepsilon_j f_j(t_j), \\ H_J(t, \varepsilon_J) &:= \sum_{j \in J, w \in W} \varepsilon_{j,w} t_{j,w}, \end{aligned}$$

with the convention that an empty sum is zero. To make notations lighter, if $J = \{j\}$ then we may write h_j and H_j instead of h_J and H_J . We also omit to write the dependence on ε_J as soon as possible. What we want to prove is thus equivalent to

$$\mathbb{E} \sup_{t \in T} F(h_I(t)) \leq \mathbb{E} \sup_{t \in T} F(H_I(t)).$$

It is enough to prove that for every partition I_1, I_2 of I , with I_2 possibly empty, if

$$\mathbb{E} \sup_{t \in T} F(h_I(t)) \leq \mathbb{E} \sup_{t \in T} F(h_{I_1}(t) + H_{I_2}(t)),$$

then for every $j \in I_1$,

$$\mathbb{E} \sup_{t \in T} F(h_I(t)) \leq \mathbb{E} \sup_{t \in T} F(h_{I_1 \setminus \{j\}}(t) + H_{I_2 \cup \{j\}}(t)).$$

Indeed, the result would then come by induction on the size of I_2 . Fix an arbitrary partition I_1, I_2 of I with I_2 possibly empty, and $j \in I_1$. It is then enough to prove that

$$\mathbb{E} \sup_{t \in T} F(h_{I_1}(t) + H_{I_2}(t)) \leq \mathbb{E} \sup_{t \in T} F(h_{I_1 \setminus \{j\}}(t) + H_{I_2 \cup \{j\}}(t)). \quad (18)$$

Denote $\varepsilon_{-j} := \varepsilon_{I \setminus \{j\}}$ and $\varphi(t, \varepsilon_{-j}) := h_{I_1 \setminus \{j\}}(t, \varepsilon_{I \setminus \{j\}}) + H_{I_2}(t, \varepsilon_{I_2})$. It holds:

$$h_{I_1}(t) + H_{I_2}(t) = h_j(t, \varepsilon_j) + \varphi(t, \varepsilon_{-j})$$

and, writing $\varepsilon_{j,\cdot} = (\varepsilon_{j,w})_{w \in W}$:

$$h_{I_1 \setminus \{j\}}(t) + H_{I_2 \cup \{j\}}(t) = H_j(t, \varepsilon_{j,\cdot}) + \varphi(t, \varepsilon_{-j}).$$

Consider the measurable functions

$$g(\varepsilon_j, \varepsilon_{-j}) := \sup_{t \in T} F(h_j(t, \varepsilon_j) + \varphi(t, \varepsilon_{-j}))$$

and

$$G(\varepsilon_{j,\cdot}, \varepsilon_{-j}) := \sup_{t \in T} F(H_j(t, \varepsilon_{j,\cdot}) + \varphi(t, \varepsilon_{-j})).$$

Denote Δ the ambient space of ε_{-j} and consider a constant $\delta \in \Delta$. Define $\hat{g}(\delta) = \mathbb{E}g(\varepsilon_j, \delta)$ and $\hat{G}(\delta) = \mathbb{E}G(\varepsilon_{j,\cdot}, \delta)$. It holds

$$\begin{aligned} \mathbb{E} \sup_{t \in T} F(h_{I_1}(t) + H_{I_2}(t)) &= \mathbb{E}g(\varepsilon_j, \varepsilon_{-j}) && \text{by definition of } g \\ &= \mathbb{E}(\mathbb{E}(g(\varepsilon_j, \varepsilon_{-j}) | \varepsilon_{-j})) && \text{law of total expectation} \\ &= \mathbb{E}\hat{g}(\varepsilon_{-j}) && \text{independence of } \varepsilon_j \text{ and } \varepsilon_{-j} \end{aligned}$$

and similarly $\mathbb{E} \sup_{t \in T} F(h_{I_1 \setminus \{j\}}(t) + H_{I_2 \cup \{j\}}(t)) = \mathbb{E}\hat{G}(\varepsilon_{-j})$. Thus, Equation (18) is equivalent to $\mathbb{E}\hat{g}(\varepsilon_{-j}) \leq \mathbb{E}\hat{G}(\varepsilon_{-j})$. For every $\delta \in \Delta$, we can define $R(\delta) = \{(t_j, \varphi(t, \delta)) \in \mathbb{R}^W \times \mathbb{R}, t \in T\}$ and it holds

$$\hat{g}(\delta) = \mathbb{E} \sup_{r \in R} F(\varepsilon_j f_j(r_1) + r_2)$$

and

$$\hat{G}(\delta) = \mathbb{E} \sup_{r \in R} F\left(\sum_{w \in W} \varepsilon_{j,w} r_{1,w} + r_2\right).$$

Thus, $\hat{g}(\delta) \leq \hat{G}(\delta)$ for every $\delta \in \Delta$ by assumption. This shows the claim. \square

Proof of Lemma D.2. Recall that we want to prove

$$\mathbb{E} \sup_{t \in T} F(\varepsilon_1 f(t_1) + t_2) \leq \mathbb{E} \sup_{t \in T} F\left(\sum_{w \in W} \varepsilon_{1,w} t_{1,w} + t_2\right). \quad (19)$$

Scalar input case. In this case, $|W| = 1$ i.e. the inputs t_1 are scalar and the result is well-known, see [Ledoux & Talagrand \(1991, Equation \(4.20\)\)](#).

k -max-pooling case. In this case, f computes the k -th largest coordinate of its input. Computing explicitly the expectation where the only random thing is $\varepsilon_1 \in \{-1, 1\}$, the left-hand side of Equation (19) is equal to

$$\frac{1}{2} \sup_{t \in T} F(f(t_1) + t_2) + \frac{1}{2} \sup_{s \in T} F(-f(s_1) + s_2).$$

Consider $s, t \in T$. Recall that $s_1, t_1 \in \mathbb{R}^W$. Denote $s_{1,(k)}$ the k -th largest component of vector s_1 . The set $\{w \in W : s_{1,w} \leq s_{1,(k)}\}$ has at least $|W| - k + 1$ elements, and $\{w \in W : t_{1,(k)} \leq t_{1,w}\}$

has at least k elements, so their intersection is not empty. Consider *any*⁸ $w(s, t)$ in this intersection. We are now going to use that both $f(t_1) = t_{1,(k)} \leq t_{1,w(s,t)}$ and $-f(s_1) = -s_{1,(k)} \leq -s_{1,w(s,t)}$. Even if we are not going to use it, note that this implies $f(t) - f(s) \leq t_{1,w(s,t)} - s_{1,w(s,t)}$: we are exactly using an argument that establishes that f is 1-Lipschitz. Since $f(t_1) = t_{1,(k)} \leq t_{1,w(s,t)}$ and F is non-decreasing, it holds:

$$\begin{aligned} F(f(t_1) + t_2) &\leq F(t_{1,w(s,t)} + t_2) \\ &\stackrel{\varepsilon \text{ centered}}{=} F\left(t_{1,w(s,t)} + \mathbb{E}\left(\sum_{w \neq w(s,t)} \varepsilon_{1,w} t_{1,w}\right) + t_2\right) \\ &\leq \mathbb{E}F\left(t_{1,w(s,t)} + \sum_{w \neq w(s,t)} \varepsilon_{1,w} t_{1,w} + t_2\right). \end{aligned}$$

Moreover, $-f(s_1) = -s_{1,(k)} \leq -s_{1,w(s,t)}$ so that in a similar way:

$$\begin{aligned} F(-f(s_1) + s_2) &\leq F(-s_{1,w(s,t)} + s_2) \\ &\leq F\left(-s_{1,w(s,t)} + \mathbb{E}\left(\sum_{w \neq w(s,t)} \varepsilon_{1,w} s_{1,w}\right) + s_2\right) \\ &\leq \mathbb{E}F\left(-s_{1,w(s,t)} + \sum_{w \neq w(s,t)} \varepsilon_{1,w} s_{1,w} + s_2\right). \end{aligned}$$

At the end, we get

$$\begin{aligned} &\frac{1}{2}F(f(t_1) + t_2) + \frac{1}{2}F(-f(s_1) + s_2) \\ &\leq \frac{1}{2}\mathbb{E}F\left(t_{1,w(s,t)} + \sum_{w \neq w(s,t)} \varepsilon_{1,w} t_{1,w} + t_2\right) \\ &\quad + \frac{1}{2}\mathbb{E}F\left(-s_{1,w(s,t)} + \sum_{w \neq w(s,t)} \varepsilon_{1,w} s_{1,w} + s_2\right) \\ &\leq \frac{1}{2}\mathbb{E}\sup_{r \in T} F\left(r_{1,w(s,t)} + \sum_{w \neq w(s,t)} \varepsilon_{1,w} r_{1,w} + r_2\right) \\ &\quad + \frac{1}{2}\mathbb{E}\sup_{r \in T} F\left(-r_{1,w(s,t)} + \sum_{w \neq w(s,t)} \varepsilon_{1,w} r_{1,w} + r_2\right) \\ &= \mathbb{E}\sup_{r \in T} F\left(\varepsilon_{1,w(s,t)} r_{1,w(s,t)} + \sum_{w \neq w(s,t)} \varepsilon_{1,w} r_{1,w} + r_2\right) \\ &= \mathbb{E}\sup_{r \in T} F\left(\sum_w \varepsilon_{1,w} r_{1,w} + r_2\right). \end{aligned}$$

The latter is independent of s, t . Taking the supremum over all $s, t \in T$ yields Equation (19) and thus the claim. \square

E PEELING ARGUMENT

First, we state a simple lemma that will be used several times.

⁸The choice of a specific w has no importance, unlike when defining the activations of k -max-pooling neurons.

Lemma E.1. Consider a vector $\varepsilon \in \mathbb{R}^n$ with iid Rademacher coordinates, meaning that $\mathbb{P}(\varepsilon_i = 1) = \mathbb{P}(\varepsilon_i = -1) = 1/2$. Consider a function $g : \mathbb{R} \rightarrow \mathbb{R}_{\geq 0}$. Consider a set $X \subset \mathbb{R}^n$. It holds:

$$\mathbb{E}_{\varepsilon} \sup_{x \in X} g \left(\left| \sum_{i=1}^n \varepsilon_i x_i \right| \right) \leq 2 \mathbb{E}_{\varepsilon} \sup_{x \in X} g \left(\sum_{i=1}^n \varepsilon_i x_i \right).$$

Proof of Lemma E.1. Since $g \geq 0$, it holds $g(|x|) \leq g(x) + g(-x)$. Thus

$$\mathbb{E}_{\varepsilon} \sup_{x \in X} g \left(\left| \sum_{i=1}^n \varepsilon_i x_i \right| \right) \leq \mathbb{E}_{\varepsilon} \sup_{x \in X} g \left(\sum_{i=1}^n \varepsilon_i x_i \right) + \mathbb{E}_{\varepsilon} \sup_{x \in X} g \left(\sum_{i=1}^n (-\varepsilon_i) x_i \right).$$

Since ε is symmetric, that is $-\varepsilon$ has the same distribution as ε , we deduce that the latter is just $2\mathbb{E}_{\varepsilon} \sup_{x \in X} g \left(\sum_{i=1}^n \varepsilon_i x_i \right)$. This proves the claim. \square

Notations We now fix for all the next results of this section n vectors $x_1, \dots, x_n \in \mathbb{R}^{d_{\text{in}}}$, for some $d_{\text{in}} \in \mathbb{N}_{>0}$. We denote $x_{i,u}$ the coordinate u of x_i .

For any neural network architecture, recall that $v(\theta, x)$ is the output of neuron v for parameters θ and input x , and $\text{ant}^d(v)$ is the set of neurons u for which there exists a path from u to v of distance d . For a set of neurons V , denote $R_V(\theta, x) = (v(\theta, x))_{v \in V}$.

Introduction to peeling This section shows that some expected sum over output neurons v can be reduced to an expected maximum over $\text{ant}(v)$, and iteratively over an expected maximum over $\text{ant}^d(v)$ for increasing d 's. Eventually, the maximum is only over input neurons as soon as d is large enough. We start with the next lemma which is the initialization of the induction over d : it peels off the output neurons v to reduce to their antecedents $\text{ant}(v)$.

Lemma E.2. Consider a neural network architecture as in Definition 2.2 with $N_{\text{in}} \cap N_{\text{out}} = \emptyset$. Consider an associated set Θ of parameters θ such that $\sum_{v \in N_{\text{out}}} \|\theta^{\rightarrow v}\|_1 + |b_v| \leq r$. Consider a family of independent Rademacher variables $(\varepsilon_j)_{j \in J}$ with J that will be clear from the context. Consider a non-decreasing function $g : \mathbb{R} \rightarrow \mathbb{R}_{\geq 0}$. Consider a new neuron v_{bias} and set by convention $x_{v_{\text{bias}}} = 1$ for every input x . It holds

$$\begin{aligned} \mathbb{E}_{\varepsilon} g \left(\sup_{\theta \in \Theta} \sum_{\substack{i=1, \dots, n, \\ v \in N_{\text{out}}}} \varepsilon_{i,v} v(\theta, x_i) \right) \\ \leq \mathbb{E}_{\varepsilon} g \left(r \max_{v \in N_{\text{out}}} \max_{u \in (\text{ant}(v) \cap N_{\text{in}}) \cup \{v_{\text{bias}}\}} \left| \sum_{i=1}^n \varepsilon_{i,v} x_{i,u} \right| \right) \\ + \mathbb{E}_{\varepsilon} g \left(r \max_{v \in N_{\text{out}}} \max_{u \in \text{ant}(v) \setminus N_{\text{in}}} \sup_{\theta} \left| \sum_{i=1}^n \varepsilon_{i,v} u(\theta, x_i) \right| \right) \end{aligned}$$

where in the last term if $\text{ant}(v) \setminus N_{\text{in}} = \emptyset$, by convention $\max_{u \in \text{ant}(v) \setminus N_{\text{in}}} = -\infty$, and $g(-\infty) := 0$.

Proof of Lemma E.2. Recall that for a set of neurons V , we denote $R_V(\theta, x) = (v(\theta, x))_{v \in V}$. Recall that by Definition 2.2, output neurons v have $\rho_v = \text{id}$ so for every $v \in N_{\text{out}}$, $\theta \in \Theta$ and every input x :

$$v(\theta, x) = \left\langle \begin{pmatrix} \theta^{\rightarrow v} \\ b_v \end{pmatrix}, \begin{pmatrix} R_{\text{ant}(v)}(\theta, x) \\ 1 \end{pmatrix} \right\rangle.$$

Denote by v_{bias} a new neuron that computes the constant function equal to one ($v_{\text{bias}}(\boldsymbol{\theta}, x) = 1$), we get:

$$\begin{aligned}
& \mathbb{E}_{\boldsymbol{\varepsilon}} g \left(\sup_{\boldsymbol{\theta}} \sum_{\substack{i=1, \dots, n \\ v \in N_{\text{out}}}} \varepsilon_{i,v} v(\boldsymbol{\theta}, x_i) \right) \\
&= \mathbb{E}_{\boldsymbol{\varepsilon}} g \left(\sup_{\boldsymbol{\theta}} \sum_{v \in N_{\text{out}}} \left\langle \left(\begin{array}{c} \boldsymbol{\theta}^{\rightarrow v} \\ b_v \end{array} \right), \sum_{i=1}^n \varepsilon_{i,v} \left(\begin{array}{c} R_{\text{ant}(v)}(\boldsymbol{\theta}, x_i) \\ 1 \end{array} \right) \right\rangle \right) \\
&\stackrel{\text{H\"older}}{\leq} \mathbb{E}_{\boldsymbol{\varepsilon}} g \left(\sup_{\boldsymbol{\theta}} \underbrace{\left(\sum_{v \in N_{\text{out}}} \|\boldsymbol{\theta}^{\rightarrow v}\|_1 + |b_v| \right)}_{\leq r \text{ by assumption}} \max_{v \in N_{\text{out}}} \left(\left| \sum_{i=1}^n \varepsilon_{i,v} \right|, \max_{u \in \text{ant}(v)} \left| \sum_{i=1}^n \varepsilon_{i,v} u(\boldsymbol{\theta}, x_i) \right| \right) \right) \\
&\leq \mathbb{E}_{\boldsymbol{\varepsilon}} g \left(r \max_{v \in N_{\text{out}}} \max_{u \in \text{ant}(v) \cup \{v_{\text{bias}}\}} \sup_{\boldsymbol{\theta}} \left| \sum_{i=1}^n \varepsilon_{i,v} u(\boldsymbol{\theta}, x_i) \right| \right).
\end{aligned}$$

Everything is non-negative so the maximum over $u \in \text{ant}(v) \cup \{v_{\text{bias}}\}$ is smaller than the sum of the maxima over $u \in (\text{ant}(v) \cap N_{\text{in}}) \cup \{v_{\text{bias}}\}$ and $u \in \text{ant}(v) \setminus N_{\text{in}}$. Note that when u is an input neuron, it simply holds $u(\boldsymbol{\theta}, x_i) = x_{i,u}$. This proves the result. \square

We now show how to peel neurons to reduce the maximum over $\text{ant}^d(v)$ to $\text{ant}^{d+1}(v)$. Later, we will repeat that until the maximum is only on input neurons. Compared to the previous lemma, note the presence of an index $m = 1, \dots, M$ in the maxima. This is because after d steps of peeling (when the maximum over u has been reduced to $u \in \text{ant}^d(v)$), we will have $M = K^{d-1}$ where K is the kernel size. Indeed, the number of copies indexed by m gets multiplied by K after each peeling step.

Lemma E.3. Consider a neural network architecture with an associated set Θ of parameters $\boldsymbol{\theta}$ such that every neuron $v \notin N_{\text{out}} \cup N_{\text{in}}$ satisfies $\|\boldsymbol{\theta}^{\rightarrow v}\|_1 + |b_v| \leq 1$. Assume that $b_v = 0$ for every $v \in N_{*-\text{pool}}$. Consider a family of independent Rademacher variables $(\varepsilon_j)_{j \in J}$ with J that will be clear from the context. Consider arbitrary $M, d \in \mathbb{N}$ and a convex non-decreasing function $g : \mathbb{R} \rightarrow \mathbb{R}_{\geq 0}$. Take a symbol v_{bias} which does not correspond to a neuron ($v_{\text{bias}} \notin N$) and set by convention $x_{v_{\text{bias}}} = 1$ for every input x . Define $P := |\{k \in \mathbb{N}_{>0}, \exists u \in N_{k-\text{pool}}\}|$ as the number of different types of $*$ -max-pooling neurons in G , and $K := \max_{u \in N_{*-\text{pool}}} |\text{ant}(u)|$ the maximal kernel size of the network ($K := 1$ if $P = 0$). It holds:

$$\begin{aligned}
& \mathbb{E}_{\boldsymbol{\varepsilon}} g \left(\max_{\substack{v \in N_{\text{out}}, \\ m=1, \dots, M}} \max_{u \in \text{ant}^d(v) \setminus N_{\text{in}}} \sup_{\boldsymbol{\theta}} \left| \sum_{i=1}^n \varepsilon_{i,v,m} u(\boldsymbol{\theta}, x_i) \right| \right) \\
&\leq (3 + 2P) \mathbb{E}_{\boldsymbol{\varepsilon}} g \left(\max_{\substack{v \in N_{\text{out}}, \\ m=1, \dots, KM}} \max_{u \in (\text{ant}^{d+1}(v) \cap N_{\text{in}}) \cup \{v_{\text{bias}}\}} \left| \sum_{i=1}^n \varepsilon_{i,v,m} x_{i,u} \right| \right) \\
&\quad + (3 + 2P) \mathbb{E}_{\boldsymbol{\varepsilon}} g \left(\max_{\substack{v \in N_{\text{out}}, \\ m=1, \dots, KM}} \max_{u \in \text{ant}^{d+1}(v) \setminus N_{\text{in}}} \sup_{\boldsymbol{\theta}} \left| \sum_{i=1}^n \varepsilon_{i,v,m} u(\boldsymbol{\theta}, x_i) \right| \right)
\end{aligned}$$

with similar convention as in Lemma E.2 for empty maxima.

Proof. Step 1: split the neurons depending on their activation function. In the term that we want to bound from above, the neurons $u \in \text{ant}^d(v) \setminus N_{\text{in}}$ are not input neurons so they compute something of the form $\rho_u(\dots)$ where ρ_u is the activation associated with u , 1-Lipschitz, and satisfies $\rho_u(0) = 0$. The first step of the proof is to get rid of ρ_u using a contraction lemma similar to Theorem 4.12 in Ledoux & Talagrand (1991). However, here, the function ρ_u depends on the neuron u , what we are taking a maximum over so that classical contraction lemmas do not apply directly. To resolve

this first obstacle, we split the neurons according to their activation function. Below, we highlight in **orange** what is important and/or the changes from one line to another. Denote N_ρ the neurons that have ρ as their associated activation function, and the term with a maximum over all $u \in N_\rho$ is denoted:

$$e(\rho) := \mathbb{E}_\varepsilon g \left(\max_{m=1, \dots, M} \max_{v \in N_{\text{out}}, u \in (\text{ant}^d(v) \cap N_\rho) \setminus N_{\text{in}}} \sup_{\boldsymbol{\theta}} \left| \sum_{i=1}^n \varepsilon_{i,v,m} u(\boldsymbol{\theta}, x_i) \right| \right),$$

with the convention $e(\rho) = 0$ if N_ρ is empty. This yields a first bound

$$\mathbb{E}_\varepsilon g \left(\max_{m=1, \dots, M} \max_{v \in N_{\text{out}}, u \in \text{ant}^d(v) \setminus N_{\text{in}}} \sup_{\boldsymbol{\theta}} \left| \sum_{i=1}^n \varepsilon_{i,v,m} u(\boldsymbol{\theta}, x_i) \right| \right) \leq e(\text{ReLU}) + e(\text{id}) + \sum_k e(k\text{-pool})$$

where the sum of the right-hand side is on all the $k \in \mathbb{N}_{>0}$ such that there is at least one neuron in $N_{k\text{-pool}}$. Define $E(\rho)$ to be the same thing as $e(\rho)$ but without the absolute values:

$$E(\rho) := \mathbb{E}_\varepsilon g \left(\max_{m=1, \dots, M} \max_{v \in N_{\text{out}}, u \in (\text{ant}^d(v) \cap N_\rho) \setminus N_{\text{in}}} \sup_{\boldsymbol{\theta}} \sum_{i=1}^n \varepsilon_{i,v,m} u(\boldsymbol{\theta}, x_i) \right).$$

Lemma E.1 gets rid of the absolute values by paying a factor 2:

$$e(\rho) \leq 2E(\rho).$$

We now want to bound each $E(\rho)$.

Step 2: get rid of the *-max-pooling and ReLU activation functions. Since the maximal kernel size is K , any *-max-pooling neuron u must have at most K antecedents. When a $u \in N_{*\text{-pool}}$ has less than K antecedents, we artificially add neurons w to $\text{ant}(u)$ to make it of cardinal K , and we set by convention $\boldsymbol{\theta}^{w \rightarrow u} = 0$. We also fix an arbitrary order on the antecedents of u and write $\text{ant}(u)_w$ for the antecedent number w , with $R_{\text{ant}(u)_w}$ the function associated with this neuron. For a ReLU or *-max-pooling neuron u , define the pre-activation of u to be

$$\text{pre}_u(\boldsymbol{\theta}, x) := \begin{cases} \left\langle \left(\begin{array}{c} \boldsymbol{\theta}^{\rightarrow u} \\ b_u \end{array} \right), \left(\begin{array}{c} R_{\text{ant}(u)_1}(\boldsymbol{\theta}, x) \\ \vdots \\ R_{\text{ant}(u)_k}(\boldsymbol{\theta}, x) \end{array} \right) \right\rangle & \text{if } u \in N_{\text{ReLU}}, \\ (b_u + \boldsymbol{\theta}^{\text{ant}(u)_w \rightarrow u} R_{\text{ant}(u)_w}(\boldsymbol{\theta}, x))_{w=1, \dots, k} & \text{otherwise when } u \in N_{*\text{-pool}}. \end{cases}$$

where we recall that $b_u = 0$ for $u \in N_{*\text{-pool}}$ by assumption. We nevertheless keep b_u in the computation until the point where this assumption is apparently actually needed to continue. Note that the pre-activation has been defined to satisfy $u(\boldsymbol{\theta}, x) = \rho_u(\text{pre}_u(\boldsymbol{\theta}, x))$. When ρ is the ReLU or k -pool, we can thus rewrite $E(\rho)$ in terms of the pre-activations:

$$E(\rho) = \mathbb{E}_\varepsilon g \left(\max_{m=1, \dots, M} \max_{v \in N_{\text{out}}, u \in (\text{ant}^d(v) \cap N_\rho) \setminus N_{\text{in}}} \sup_{\boldsymbol{\theta}} \sum_{i=1}^n \varepsilon_{i,v,m} \rho(\text{pre}_u(\boldsymbol{\theta}, x_i)) \right).$$

Consider the finite set $Z = \{(v, m), v \in N_{\text{out}}, m = 1, \dots, M\}$ and for every $z = (v, m) \in Z$, define $T^z = \{(\text{pre}_u(\boldsymbol{\theta}, x_i))_{i=1, \dots, n} : u \in (\text{ant}^d(v) \cap N_\rho) \setminus N_{\text{in}}, \boldsymbol{\theta} \in \Theta\}$. An element of T^z will be denoted $t = (t_i)_{i=1}^n \in T^z \subset \mathbb{R}^n$ if $\rho = \text{ReLU}$, and $t = (t_i)_{i=1}^n \in T^z \subset (\mathbb{R}^k)^n$ with $t_i = (t_{i,w})_{w=1}^k \in \mathbb{R}^k$ if $u \in N_{*\text{-pool}}$. We can again rewrite $E(\rho)$ as

$$E(\rho) = \mathbb{E}_\varepsilon g \left(\max_{z \in Z} \sup_{t \in T^z} \sum_{i=1}^n \varepsilon_{i,z} \rho(t_i) \right).$$

We now want to get rid of the activation function ρ with a contraction lemma. There is a second difficulty that prevents us from directly applying classical contraction lemmas such as Theorem 4.12 of [Ledoux & Talagrand \(1991\)](#). It is the presence of a maximum over multiple copies indexed by $z \in Z$ of a supremum that depends on iid families $(\varepsilon_{i,z})_{i=1 \dots n}$. Indeed, Theorem 4.12 of [Ledoux & Talagrand \(1991\)](#) only deals with a single copy ($|Z| = 1$). This motivates the contraction lemma established for the occasion in Lemma D.1. Once the activation functions removed, we can conclude separately for $\rho = \text{ReLU}$, id and $\rho = k\text{-pool}$.

Step 3a: deal with $\rho = k\text{-pool}$ via rescaling. In the case $\rho = k\text{-pool}$, Lemma D.1 shows that

$$\begin{aligned} & \mathbb{E}_{\varepsilon} g \left(\max_{z \in Z} \sup_{t \in T^z} \sum_{i=1}^n \varepsilon_{i,z} k\text{-pool}(t_i) \right) \\ & \leq \mathbb{E}_{\varepsilon} g \left(\max_{z \in Z} \sup_{t \in T^z} \sum_{\substack{i=1, \dots, n, \\ w=1, \dots, K}} \varepsilon_{i,z,w} t_{i,w} \right). \end{aligned}$$

The right-hand side is equal to

$$\mathbb{E}_{\varepsilon} g \left(\max_{\substack{v \in N_{\text{out}}, \\ m=1, \dots, M}} \sup_{u \in (\text{ant}^d(v) \cap N_{* \text{-pool}}) \setminus N_{\text{in}},} \sum_{\substack{i=1, \dots, n, \\ w=1, \dots, K}} \varepsilon_{i,v,m,w} (b_u + \theta^{\text{ant}(u)_w \rightarrow u} R_{\text{ant}(u)_w}(\theta, x_i)) \right). \quad (20)$$

We now deal with this using the assumption on the norm of incoming weights. Recalling that $b_u = 0$ for every $u \in N_{* \text{-pool}}$:

$$\begin{aligned} & \sum_{\substack{i=1, \dots, n, \\ w=1, \dots, K}} \varepsilon_{i,v,m,w} \underbrace{(b_u)}_{=0} + \theta^{\text{ant}(u)_w \rightarrow u} R_{\text{ant}(u)_w}(\theta, x_i) \\ & = \sum_{w=1, \dots, K} \theta^{\text{ant}(u)_w \rightarrow u} \left(\sum_{i=1, \dots, n} \varepsilon_{i,v,m,w} R_{\text{ant}(u)_w}(\theta, x_i) \right) \\ & \leq \underbrace{\|\theta^{\rightarrow u}\|_1}_{\leq 1 \text{ by assumption}} \max_{w=1, \dots, K} \left| \sum_{i=1, \dots, n} \varepsilon_{i,v,m,w} R_{\text{ant}(u)_w}(\theta, x_i) \right| \\ & \leq \max_{\text{decoupling } w \text{ and } \text{ant}(u)_w} \max_{w \in \text{ant}(u)} \max_{w'=1, \dots, K} \left| \sum_{i=1}^n \varepsilon_{i,v,m,w'} w(\theta, x_i) \right|. \end{aligned}$$

Note for the curious reader that this inequality is the current obstacle when the biases are nonzero since we would end up with $K|b_u| + \|\theta^{\rightarrow u}\|_1$ and we could not anymore use that this is ≤ 1 .

We deduce that Equation (20) is bounded from above by

$$\mathbb{E}_{\varepsilon} g \left(\max_{\substack{v \in N_{\text{out}}, \\ m=1, \dots, M}} \sup_{u \in (\text{ant}^d(v) \cap N_{* \text{-pool}}) \setminus N_{\text{in}},} \max_{w \in \text{ant}(u)} \max_{w'=1, \dots, K} \left| \sum_{i=1}^n \varepsilon_{i,v,m,w'} w(\theta, x_i) \right| \right).$$

Instead of having $\varepsilon_{i,v,m,w'}$ with $m = 1, \dots, M$ and $w' = 1, \dots, K$, we re-index it as $\varepsilon_{i,v,m}$ with $m = 1, \dots, KM$. Note also that $u \in (\text{ant}^d(v) \cap N_{* \text{-pool}}) \setminus N_{\text{in}}$ and $w \in \text{ant}(u)$ implies $w \in \text{ant}^{d+1}(v)$, so considering a maximum over $w \in \text{ant}^{d+1}(v)$ can only yield something larger. Moreover, we can add a new neuron v_{bias} that computes the constant function equal to one ($v_{\text{bias}}(\theta, x) = 1$) and add v_{bias} to the maximum over w . Implementing all these changes, Equation (20) is bounded by

$$H := \mathbb{E}_{\varepsilon} g \left(\max_{\substack{v \in N_{\text{out}}, \\ m=1, \dots, KM}} \sup_{w \in \text{ant}^{d+1}(v) \cup \{v_{\text{bias}}\}} \sup_{\theta \in \Theta} \left| \sum_{i=1}^n \varepsilon_{i,v,m} w(\theta, x_i) \right| \right).$$

We now derive similar inequalities when $\rho = \text{id}$ and $\rho = \text{ReLU}$.

Step 3b: deal with $\rho = \text{id}$, ReLU via rescaling. In the case $\rho = \text{ReLU}$, Lemma D.1 shows that

$$\begin{aligned} & \mathbb{E}_\varepsilon g \left(\max_{z \in Z} \sup_{t \in T^z} \sum_{i=1}^n \varepsilon_{i,z} \text{ReLU}(t_i) \right) \\ & \leq \mathbb{E}_\varepsilon g \left(\max_{z \in Z} \sup_{t \in T^z} \sum_{i=1, \dots, n} \varepsilon_{i,z} t_i \right). \end{aligned}$$

The difference with the $*$ -max-pooling case is that each t_i is scalar so this does not introduce an additional index w to the Rademacher variables. The right-hand side can be rewritten as

$$\mathbb{E}_\varepsilon g \left(\max_{\substack{v \in N_{\text{out}}, \\ m=1, \dots, M}} \sup_{u \in (\text{ant}^d(v) \cap N_{\text{ReLU}}) \setminus N_{\text{in}}, \theta \in \Theta} \sum_{i=1, \dots, n} \varepsilon_{i,v,m} \left\langle \left(\begin{array}{c} \theta^{\rightarrow u} \\ b_u \end{array} \right), \left(\begin{array}{c} R_{\text{ant}(u)}(\theta, x_i) \\ 1 \end{array} \right) \right\rangle \right)$$

We can only increase the latter by considering a maximum over all $u \in \text{ant}^d(v)$, not only the ones in N_{ReLU} . We also add absolute values. This is then bounded by

$$F := \mathbb{E}_\varepsilon g \left(\max_{\substack{v \in N_{\text{out}}, \\ m=1, \dots, M}} \sup_{u \in \text{ant}^d(v) \setminus N_{\text{in}}, \theta \in \Theta} \left| \sum_{i=1, \dots, n} \varepsilon_{i,v,m} \left\langle \left(\begin{array}{c} \theta^{\rightarrow u} \\ b_u \end{array} \right), \left(\begin{array}{c} R_{\text{ant}(u)}(\theta, x_i) \\ 1 \end{array} \right) \right\rangle \right| \right). \quad (21)$$

This means that $E(\text{ReLU}) \leq F$. Let us also observe that $e(\text{id}) \leq F$. Indeed, recall that by definition

$$e(\text{id}) = \mathbb{E}_\varepsilon g \left(\max_{\substack{v \in N_{\text{out}}, \\ m=1, \dots, M}} \max_{u \in (\text{ant}^d(v) \cap N_{\text{id}}) \setminus N_{\text{in}}} \sup_{\theta} \left| \sum_{i=1}^n \varepsilon_{i,v,m} u(\theta, x_i) \right| \right).$$

We can only increase the latter by considering a maximum over all $u \in \text{ant}^d(v)$. Moreover, for an identity neuron u , it holds $u(\theta, x) = \left\langle \left(\begin{array}{c} \theta^{\rightarrow u} \\ b_u \end{array} \right), \left(\begin{array}{c} R_{\text{ant}(u)}(\theta, x) \\ 1 \end{array} \right) \right\rangle$. This shows that $e(\text{id}) \leq F$. It remains to bound F using that the assumption on the norm of the parameters. Introduce a new neuron v_{bias} that computes the constant function equal to one: $v_{\text{bias}}(\theta, x) = 1$. Note that

$$\begin{aligned} & \sum_{i=1, \dots, n} \varepsilon_{i,v,m} \left\langle \left(\begin{array}{c} \theta^{\rightarrow u} \\ b_u \end{array} \right), \left(\begin{array}{c} R_{\text{ant}(u)}(\theta, x_i) \\ 1 \end{array} \right) \right\rangle \\ & = \left\langle \left(\begin{array}{c} \theta^{\rightarrow u} \\ b_u \end{array} \right), \sum_{i=1, \dots, n} \varepsilon_{i,v,m} \left(\begin{array}{c} R_{\text{ant}(u)}(\theta, x_i) \\ 1 \end{array} \right) \right\rangle \\ & \stackrel{\text{Hölder}}{\leq} \underbrace{(\|\theta^{\rightarrow u}\|_1 + |b_u|)}_{\leq 1 \text{ by assumption}} \max_{w \in \text{ant}(u) \cup \{v_{\text{bias}}\}} \left| \sum_{i=1}^n \varepsilon_{i,v,m} w(\theta, x_i) \right|. \end{aligned}$$

This shows that

$$F \leq \mathbb{E}_\varepsilon g \left(\max_{\substack{v \in N_{\text{out}}, \\ m=1, \dots, M}} \sup_{u \in \text{ant}^d(v) \setminus N_{\text{in}}, \theta \in \Theta} \max_{w \in \text{ant}(u) \cup \{v_{\text{bias}}\}} \left| \sum_{i=1}^n \varepsilon_{i,v,m} w(\theta, x_i) \right| \right).$$

Obviously, introducing additional copies of ε to make the third index going from $m = 1$ to KM can only make it larger. Moreover, $u \in \text{ant}^d(v) \setminus N_{\text{in}}$ and $w \in \text{ant}(u)$ implies $w \in \text{ant}^{d+1}(v)$, so we can instead consider a maximum over $w \in \text{ant}^{d+1}(v)$. This gives the upper-bound

$$\begin{aligned} F & \leq \mathbb{E}_\varepsilon g \left(\max_{\substack{v \in N_{\text{out}}, \\ m=1, \dots, KM}} \max_{w \in \text{ant}^{d+1}(v) \cup \{v_{\text{bias}}\}} \sup_{\theta \in \Theta} \left| \sum_{i=1}^n \varepsilon_{i,v,m} w(\theta, x_i) \right| \right) \\ & = H. \end{aligned}$$

Step 4: putting everything together. At the end, recalling that there are at most P different $k \in \mathbb{N}_{>0}$ associated with an existing k -max-pooling neuron, we get the final bound

$$\begin{aligned}
& \mathbb{E}_{\varepsilon} g \left(\max_{\substack{v \in N_{\text{out}}, \\ m=1, \dots, M}} \max_{u \in \text{ant}^d(v) \setminus N_{\text{in}}} \sup_{\theta} \left| \sum_{i=1}^n \varepsilon_{i,v,m} u(\theta, x_i) \right| \right) \\
& \leq e(\text{id}) + e(\text{ReLU}) + \sum_k e(k\text{-pool}) \\
& \leq e(\text{id}) + 2E(\text{ReLU}) + 2 \sum_k E(k\text{-pool}) \\
& \leq F + 2F + 2 \sum_k E(k\text{-pool}) \\
& \leq H + 2H + 2 \sum_k H \\
& \leq H + 2H + 2PH = (3 + 2P)H.
\end{aligned}$$

The term $(3+2P)H$ can again be bounded by splitting the maximum over $w \in \text{ant}^{d+1}(v) \cup \{v_{\text{bias}}\}$ between the w 's that are input neurons, and those that are not, since everything is non-negative. This yields the claim. \square

Remark E.1 (Improved dependencies on the kernel size). *Note that in the proof of Lemma E.3, the multiplication of M by K can be avoided if there are no $*$ -max-pooling neurons in $\text{ant}^d(v)$. Because of skip connections, even if there is a single $*$ -max-pooling neuron in the architecture, it can be in $\text{ant}^d(v)$ for many d 's. A more advanced version of the argument is to peel only the ReLU and identity neurons, by leaving the $*$ -max-pooling neurons as they are, until we reach a set of $*$ -max-pooling neurons large enough that we decide to peel simultaneously. This would prevent the multiplication by K every time d is increased.*

We can now state the main peeling theorem, which directly result from Lemma E.2 and Lemma E.3 by induction on d . Note that these lemmas contain assumptions on the size of the incoming weights of the different neurons. These assumptions are met using Algorithm 1, which rescales the parameters without changing the associated function nor the path-norm.

Theorem E.1. *Consider a neural network architecture as in Definition 2.2 with $N_{\text{out}} \cap N_{\text{in}} = \emptyset$. Assume that $b_v = 0$ for every $v \in N_{*}\text{-pool}$. Define $P := |\{k \in \mathbb{N}_{>0}, \exists u \in N_{k\text{-pool}}\}|$ the number of different types of $*$ -max-pooling neurons in G , and $K := \max_{u \in N_{*}\text{-pool}}$ the maximum kernel size ($K := 1$ by convention if $P = 0$). Denote by v_{bias} a new input neuron and define $x_{v_{\text{bias}}} = 1$ for any input x . For any set of parameters Θ associated with the network, such that $\|\Phi(\theta)\|_1 \leq r$ for every $\theta \in \Theta$, it holds for every convex non-decreasing function $g : \mathbb{R} \rightarrow \mathbb{R}_{\geq 0}$*

$$\begin{aligned}
& \mathbb{E}_{\varepsilon} g \left(\sup_{\theta \in \Theta} \sum_{\substack{i=1, \dots, n, \\ v \in N_{\text{out}}}} \varepsilon_{i,v} v(\theta, x_i) \right) \\
& \leq \frac{(3 + 2P)^D}{2 + 2P} \mathbb{E}_{\varepsilon} g \left(r \max_{\substack{v \in N_{\text{out}}, \\ m=1, \dots, K^{D-1}}} \max_{u \in N_{\text{in}} \cup \{v_{\text{bias}}\}} \left| \sum_{i=1}^n \varepsilon_{i,v,m} x_{i,u} \right| \right).
\end{aligned}$$

Proof of Theorem E.1. Without loss of generality, we can replace Θ by its image under Algorithm 1 with $q = 1$, as Algorithm 1 does not change the associated function R_{θ} nor the path-norm $\|\Phi(\theta)\|_1$ (Lemma B.1) so that we still have $\|\Phi(\theta)\|_1 \leq r$ and the supremum over $\theta \in \Theta$ on the left-hand side can be taken over rescaled parameters. By Lemma B.1, the parameters are 1-normalized (Definition B.1), so we will be able to use Lemma E.2 and Lemma E.3. Indeed, 1-normalized parameters θ satisfy $\sum_{v \in N_{\text{out}}} \|\theta^{\rightarrow v}\|_1 + |b_v| = \|\Phi(\theta)\|_1 \leq r$ so Lemma E.2 applies. We also have $\|\theta^{\rightarrow v}\|_1 + |b_v| \leq 1$ for every $v \notin N_{\text{in}} \cup N_{\text{out}}$, by definition of 1-normalization, so Lemma E.3 also applies.

By induction on $d \geq 1$, we prove that (highlighting in **orange** what is important)

$$\begin{aligned} & \mathbb{E}_{\varepsilon} g \left(\sup_{\boldsymbol{\theta} \in \Theta} \sum_{\substack{i=1, \dots, n, \\ v \in N_{\text{out}}}} \varepsilon_{i,v} v(\boldsymbol{\theta}, x_i) \right) \\ & \leq \sum_{\ell=1}^d (3+2P)^{\ell-1} \mathbb{E}_{\varepsilon} g \left(r \max_{\substack{v \in N_{\text{out}}, \\ m=1, \dots, K^{\ell-1}}} \max_{u \in (\text{ant}^{\ell}(v) \cap N_{\text{in}}) \cup \{v_{\text{bias}}\}} \left| \sum_{i=1}^n \varepsilon_{i,v,m} \mathbf{x}_{i,u} \right| \right) \\ & \quad + (3+2P)^{d-1} \mathbb{E}_{\varepsilon} g \left(r \max_{\substack{v \in N_{\text{out}}, \\ m=1, \dots, K^{d-1}}} \max_{u \in \text{ant}^d(v) \setminus N_{\text{in}}} \sup_{\boldsymbol{\theta} \in \Theta} \left| \sum_{i=1}^n \varepsilon_{i,v,m} \mathbf{u}(\boldsymbol{\theta}, x_i) \right| \right), \end{aligned}$$

with the same convention as in Lemma E.2 for maxima over empty sets. This is true for $d = 1$ by Lemma E.2. The induction step is verified using Lemma E.3. This concludes the induction. Applying the result for $d = D$, and since $\text{ant}^D(v) \setminus N_{\text{in}} = \emptyset$, we get:

$$\begin{aligned} & \mathbb{E}_{\varepsilon} g \left(\sup_{\boldsymbol{\theta} \in \Theta} \sum_{\substack{i=1, \dots, n, \\ v \in N_{\text{out}}}} \varepsilon_{i,v} v(\boldsymbol{\theta}, x_i) \right) \\ & \leq \sum_{d=1}^D (3+2P)^{d-1} \mathbb{E}_{\varepsilon} g \left(r \max_{\substack{v \in N_{\text{out}}, \\ m=1, \dots, K^{d-1}}} \max_{u \in (\text{ant}^d(v) \cap N_{\text{in}}) \cup \{v_{\text{bias}}\}} \left| \sum_{i=1}^n \varepsilon_{i,v,m} x_{i,u} \right| \right). \end{aligned}$$

We can only increase the right-hand side by considering maximum over all $u \in N_{\text{in}} \cup \{v_{\text{bias}}\}$ and by adding independent copies indexed from $m = 1$ to $m = K^{D-1}$. Moreover, $\sum_{d=1}^D (3+2P)^{d-1} = ((3+2P)^D - 1)/(2+2P)$. This shows the final bound:

$$\begin{aligned} & \mathbb{E}_{\varepsilon} g \left(\sup_{\boldsymbol{\theta} \in \Theta} \sum_{\substack{i=1, \dots, n, \\ v \in N_{\text{out}}}} \varepsilon_{i,v} v(\boldsymbol{\theta}, x_i) \right) \\ & \leq \frac{(3+2P)^D}{2+2P} \mathbb{E}_{\varepsilon} g \left(r \max_{\substack{v \in N_{\text{out}}, \\ m=1, \dots, K^{D-1}}} \max_{u \in N_{\text{in}} \cup \{v_{\text{bias}}\}} \left| \sum_{i=1}^n \varepsilon_{i,v,m} x_{i,u} \right| \right). \end{aligned}$$

□

F DETAILS TO DERIVE THE GENERALIZATION BOUND (THEOREM 3.1)

Proof of Theorem 3.1. The proof is given directly in the general case with nonzero biases on every $v \notin N_{*-\text{pool}}$ and uses an additional input neuron v_{bias} . We highlight along the proof where improved results can be obtained assuming zero biases, yielding Theorem 3.1 as a consequence. Define the random matrices $E = (\varepsilon_{i,v})_{i,v} \in \mathbb{R}^{n \times d_{\text{out}}}$ and $R(\boldsymbol{\theta}, \mathbf{X}) = (v(\boldsymbol{\theta}, \mathbf{X}_i))_{i,v} \in \mathbb{R}^{n \times d_{\text{out}}}$ so that $\langle E, R(\boldsymbol{\theta}, \mathbf{X}) \rangle = \sum_{i,v} \varepsilon_{i,v} (R_{\boldsymbol{\theta}}(\mathbf{X}_i))_v$. It holds:

$$\begin{aligned} \mathbb{E}_{\mathbf{Z}} \ell\text{-generalization error of } \hat{\boldsymbol{\theta}}(\mathbf{Z}) & \leq \frac{2}{n} \mathbb{E}_{\mathbf{Z}, \varepsilon} \left(\sup_{\boldsymbol{\theta}} \sum_{i=1}^n \varepsilon_i \ell(R_{\boldsymbol{\theta}}(\mathbf{X}_i), \mathbf{Y}_i) \right) \\ & \leq \frac{2\sqrt{2}L}{n} \mathbb{E}_{\mathbf{Z}, \varepsilon} \left(\sup_{\boldsymbol{\theta}} \langle E, R(\boldsymbol{\theta}, \mathbf{X}) \rangle \right). \end{aligned}$$

The first inequality is the symmetrization property given by Shalev-Shwartz & Ben-David (2014, Theorem 26.3), and the second inequality is the vector-valued contraction property given by Maurer (2016). These are the relevant versions of very classical arguments that are widely used to

reduce the problem to the Rademacher complexity of the model (Bach, 2024, Propositions 4.2 and 4.3)(Wainwright, 2019, Equations (4.17) and (4.18))(Bartlett & Mendelson, 2002, Proof of Theorem 8)(Shalev-Shwartz & Ben-David, 2014, Theorem 26.3)(Ledoux & Talagrand, 1991, Equation (4.20)). In particular, this step has nothing specific with neural networks. Note that the assumption on the loss is used for the second inequality.

We now condition on $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$ and denote \mathbb{E}_ε the conditional expectation. For any random variable $\lambda(\mathbf{Z}) > 0$ measurable in \mathbf{Z} , it holds

$$\begin{aligned} \mathbb{E}_\varepsilon \left(\sup_{\boldsymbol{\theta}} \langle E, R(\boldsymbol{\theta}, \mathbf{X}) \rangle \right) &= \frac{1}{\lambda(\mathbf{Z})} \log \exp \left(\lambda(\mathbf{Z}) \mathbb{E}_\varepsilon \left(\sup_{\boldsymbol{\theta}} \langle E, R(\boldsymbol{\theta}, \mathbf{X}) \rangle \right) \right) \\ &\stackrel{\lambda \text{ measurable in } \mathbf{Z}}{=} \frac{1}{\lambda(\mathbf{Z})} \log \exp \left(\mathbb{E}_\varepsilon \left(\lambda(\mathbf{Z}) \sup_{\boldsymbol{\theta}} \langle E, R(\boldsymbol{\theta}, \mathbf{X}) \rangle \right) \right) \\ &\leq \frac{1}{\lambda(\mathbf{Z})} \log \mathbb{E}_\varepsilon \exp \left(\lambda(\mathbf{Z}) \sup_{\boldsymbol{\theta}} \langle E, R(\boldsymbol{\theta}, \mathbf{X}) \rangle \right). \end{aligned}$$

For $z = ((x_i, y_i))_{i=1}^n \in (\mathbb{R}^{d_{\text{in}}} \times \mathbb{R}^{d_{\text{out}}})^n$, denote

$$e(z) = \mathbb{E}_\varepsilon \exp \left(\lambda(z) \sup_{\boldsymbol{\theta}} \langle E, R(\boldsymbol{\theta}, x) \rangle \right).$$

Since \mathbf{Z} is independent of ε , it holds

$$\mathbb{E}_\varepsilon \exp \left(\lambda(\mathbf{Z}) \sup_{\boldsymbol{\theta}} \langle E, R(\boldsymbol{\theta}, \mathbf{X}) \rangle \right) = e(\mathbf{Z}).$$

Denote $r = \sup_{\boldsymbol{\theta} \in \Theta} \|\Phi(\boldsymbol{\theta})\|_1$. For z as above, simply denote $\lambda := \lambda(z)$. Since $N_{\text{in}} \cap N_{\text{out}} = \emptyset$ and the biases of $*$ -max-pooling neurons are null, the peeling argument given by Theorem E.1 for $g : t \in \mathbb{R} \mapsto \exp(\lambda t)$ guarantees:

$$e(z) \leq \frac{(3 + 2P)^D}{2 + 2P} \mathbb{E}_\varepsilon \exp \left(\lambda r \max_{m=1, \dots, K^{D-1}} \max_{v \in N_{\text{out}}, u \in N_{\text{in}} \cup \{v_{\text{bias}}\}} \left| \sum_{i=1}^n \varepsilon_{i,v,m} x_{i,u} \right| \right),$$

where $x_{i,u}$ is coordinate u of vector $x_i \in \mathbb{R}^{d_{\text{in}}}$, and where v_{bias} is an added neuron for which we set by convention $x_{v_{\text{bias}}} = 1$ for any input x . It is easy to check that the same bound holds true with a maximum only over $u \in N_{\text{in}}$ (not considering v_{bias} in the maximum) when all biases are constrained to be null. In such a setting, all the $\max_{u \in N_{\text{in}} \cup \{v_{\text{bias}}\}}$ below can be replaced by $\max_{u \in N_{\text{in}}}$. Denote

$$\sigma(x) := \max_{u \in N_{\text{in}} \cup \{v_{\text{bias}}\}} \left(\sum_{i=1}^n x_{i,u}^2 \right)^{1/2} \geq \sqrt{n}.$$

Using Lemma F.1, it holds

$$\mathbb{E}_\varepsilon \exp \left(\lambda r \max_{\substack{v \in N_{\text{out}}, \\ u \in N_{\text{in}} \cup \{v_{\text{bias}}\}, \\ m=1, \dots, K^{D-1}}} \left| \sum_{i=1}^n \varepsilon_{i,v,m} (\mathbf{X}_i)_u \right| \right) \leq 2K^{D-1} (d_{\text{in}} + 1) d_{\text{out}} \exp \left(\frac{(r\lambda(z)\sigma(x))^2}{2} \right).$$

When biases are constrained to be null, $d_{\text{in}} + 1$ is replaced by d_{in} . Putting everything together, we get:

$$\mathbb{E}_\varepsilon \left(\sup_{\boldsymbol{\theta}} \langle E, R(\boldsymbol{\theta}, \mathbf{X}) \rangle \right) = e(\mathbf{Z}) \leq \left(\frac{1}{\lambda} \log(C_1) + \lambda(\mathbf{Z}) C_2(\mathbf{X}) \right)$$

with

$$C_1 = 2K^{D-1} (d_{\text{in}} + 1) d_{\text{out}} \times \frac{(3 + 2P)^D}{2 + 2P} = \frac{3 + 2P}{1 + P} ((3 + 2P)K)^{D-1} (d_{\text{in}} + 1) d_{\text{out}}$$

(again with $d_{\text{in}} + 1$ replaced by d_{in} when all biases are null) and

$$C_2(\mathbf{X}) = \frac{1}{2}(r\sigma(\mathbf{X}))^2.$$

Choosing $\lambda(\mathbf{Z}) = \sqrt{\frac{\log(C_1)}{C_2(\mathbf{X})}}$ yields:

$$\begin{aligned} \mathbb{E}_{\boldsymbol{\varepsilon}} \left(\sup_{\boldsymbol{\theta}} \langle E, R(\boldsymbol{\theta}, \mathbf{X}) \rangle \right) &\leq 2\sqrt{\log(C_1)C_2(\mathbf{X})} \\ &\leq \underbrace{\sqrt{2}\sigma(\mathbf{X})r}_{=2\sqrt{C_2(\mathbf{X})}} \underbrace{\left(\log \left(\frac{3+2P}{1+P} (d_{\text{in}}+1)d_{\text{out}} \right) + D \log((3+2P)K) \right)^{1/2}}_{\geq \sqrt{\log(C_1)}} \end{aligned}$$

with $d_{\text{in}} + 1$ replaced by d_{in} when all biases are null. Taking the expectation on both sides over \mathbf{Z} , and multiplying this by $\frac{2\sqrt{2}L}{n}$ yields Theorem 3.1. \square

The next lemma is classical (Golowich et al., 2018, Section 7.1) and is here only for completeness.

Lemma F.1. For any $d, k \in \mathbb{N}_{>0}$ and $\lambda > 0$, it holds

$$\mathbb{E}_{\boldsymbol{\varepsilon}} \exp \left(\lambda \max_{\substack{m=1,\dots,k, \\ u=1,\dots,d}} \left| \sum_{i=1}^n \varepsilon_{i,m}(\mathbf{X}_i)_u \right| \right) \leq 2kd \max_{u=1,\dots,d} \exp \left(\frac{\lambda^2}{2} \sum_{i=1}^n (\mathbf{X}_i)_u^2 \right).$$

Proof. It holds

$$\mathbb{E}_{\boldsymbol{\varepsilon}} \exp \left(\lambda \max_{\substack{m=1,\dots,k, \\ u=1,\dots,d}} \left| \sum_{i=1}^n \varepsilon_{i,m}(\mathbf{X}_i)_u \right| \right) \leq \sum_{\substack{m=1,\dots,k, \\ u=1,\dots,d}} \mathbb{E}_{\boldsymbol{\varepsilon}} \exp \left(\lambda \left| \sum_{i=1}^n \varepsilon_{i,m}(\mathbf{X}_i)_u \right| \right).$$

For given u and m :

$$\begin{aligned} \mathbb{E}_{\boldsymbol{\varepsilon}} \exp \left(\lambda \left| \sum_{i=1}^n \varepsilon_{i,m}(\mathbf{X}_i)_u \right| \right) &\stackrel{\text{Lemma E.1}}{\leq} 2\mathbb{E}_{\boldsymbol{\varepsilon}} \exp \left(\lambda \sum_{i=1}^n \varepsilon_{i,m}(\mathbf{X}_i)_u \right) \\ &= 2 \prod_{i=1}^n \frac{\exp(\lambda(\mathbf{X}_i)_u) + \exp(-\lambda(\mathbf{X}_i)_u)}{2} \leq 2 \exp \left(\frac{\lambda^2}{2} \sum_{i=1}^n (\mathbf{X}_i)_u^2 \right) \end{aligned}$$

using $\exp(x) + \exp(-x) \leq 2\exp(x^2/2)$ in the last inequality. \square

G THE CROSS-ENTROPY LOSS IS LIPSCHITZ

Theorem 3.1 applies to the cross-entropy loss with $L = \sqrt{2}$. To see this, first recall that with C classes, the cross-entropy loss is defined as

$$\ell : (x, y) \in \mathbb{R}^C \times \{0, 1\}^C \mapsto - \sum_{c=1}^{d_{\text{out}}} y_c \log \left(\frac{\exp(x_c)}{\sum_d \exp(x_d)} \right).$$

Consider $y \in \{0, 1\}^C$ with exactly one nonzero coordinate and an exponent $p \in [1, \infty]$ with conjugate exponent p' ($1/p + 1/p' = 1$). For every $x, x' \in \mathbb{R}^C$:

$$\ell(x, y) - \ell(x', y) \leq 2^{1/p'} \|x - x'\|_p.$$

Consider a class $c \in \{1, \dots, C\}$ and take $y \in \{0, 1\}^C$ to be a one-hot encoding of c (meaning that $y_{c'} = \mathbb{1}_{c'=c}$). Consider an exponent $p \in [1, \infty]$ with conjugate exponent p' ($1/p + 1/p' = 1$).

1). The function $f : x \mapsto \ell(x, y) = -\sum_c y_c \log \left(\frac{\exp(x_c)}{\sum_{c'=1}^C \exp(x_{c'})} \right) = -\log \left(\frac{\exp(x_c)}{\sum_{c'=1}^C \exp(x_{c'})} \right)$ is continuously differentiable so that for every $x, x' \in \mathbb{R}^C$:

$$f(x) - f(x') = \int_0^1 \langle \nabla f(tx + (1-t)x'), x - x' \rangle dt \leq \sup_{t \in [0,1]} \|\nabla f(tx + (1-t)x')\|_p \|x - x'\|_{p'}.$$

In order to differentiate f , let's start to differentiate $g(x) = \frac{\exp(x_c)}{\sum_{c'=1}^C \exp(x_{c'})}$. Denote ∂_i the partial derivative with respect to coordinate i . For $i \neq c$:

$$\begin{aligned} \partial_c g(x) &= \frac{\exp(x_c) (\sum_{c'} \exp(x_{c'})) - \exp(x_c) (\exp(x_c))}{(\sum_{c'} \exp(x_{c'}))^2} \\ &= g(x) \frac{\sum_{c' \neq c} \exp(x_{c'})}{\sum_{c'} \exp(x_{c'})}. \\ \partial_i g(x) &= \frac{0 (\sum_{c'} \exp(x_{c'})) - \exp(x_c) (\exp(x_i))}{(\sum_{c'} \exp(x_{c'}))^2} \\ &= g(x) \frac{-\exp(x_i)}{\sum_{c'} \exp(x_{c'})}. \end{aligned}$$

Since $f(x) = (-\log \circ h)(x)$:

$$\begin{aligned} \partial_i f(x) &= -\frac{\partial_i g(x)}{g(x)} \\ &= \frac{1}{\sum_{c'=1}^C \exp(x_{c'})} \times \begin{cases} -\sum_{c' \neq c} e^{x_{c'}} & \text{if } i = c, \\ e^{x_i} & \text{otherwise.} \end{cases} \end{aligned}$$

Thus

$$\begin{aligned} \|\nabla f(x)\|_p^p &= \sum_{i=1}^C |\partial_i f(x)|^p \\ &= \frac{\left(\sum_{c' \neq c} \exp(x_{c'}) \right)^p + \sum_{c' \neq c} \exp(x_{c'})^p}{\left(\sum_{c'=1}^C \exp(x_{c'}) \right)^p} \\ &\leq 2 \frac{\left(\sum_{c' \neq c} \exp(x_{c'}) \right)^p}{\left(\sum_{c'=1}^C \exp(x_{c'}) \right)^p} \\ &\leq 2 \frac{\left(\sum_{c' \neq c} \exp(x_{c'}) \right)^p}{\left(\sum_{c' \neq c} \exp(x_{c'}) \right)^p} \\ &= 2. \end{aligned}$$

where we used in the first inequality that $\|v\|_p^p \leq \|v\|_1^p$ for any vector v . This shows that for every $x, x' \in \mathbb{R}^C$:

$$\ell(x, y) - \ell(x', y) \leq 2^{1/p} \|x - x'\|_{p'}.$$

H THE TOP-1 ACCURACY LOSS IS NOT LIPSCHITZ

Theorem 3.1 does not apply to the top-1 accuracy loss $\ell(\hat{y}, y) = \mathbb{1}_{\arg \max \hat{y} = \arg \max y}$ as Equation (3) cannot be satisfied by ℓ . Indeed, it is easy to construct situations where $\hat{y}_1 = R_\theta(x_1)$ is arbitrarily close to $\hat{y}_2 = R_\theta(x_2)$ with x_2 correctly classified, while x_1 is not (just take x_2 on the boundary decision of the network and x_1 on the wrong side of the boundary), so that the left-hand side is equal to 1 and the right-hand side is arbitrarily small. Thus, there is no finite $L > 0$ that could satisfy Equation (3).

I THE MARGIN-LOSS IS LIPSCHITZ

For $\hat{y} \in \mathbb{R}^{d_{\text{out}}}$ and a one-hot encoding $y \in \mathbb{R}^{d_{\text{out}}}$ of the class c of x (meaning that $y_{c'} = \mathbb{1}_{c'=c}$ for every c'), the margin $M(\hat{y}, y)$ is defined by

$$M(\hat{y}, y) := [\hat{y}]_c - \max_{c' \neq c} [\hat{y}]_{c'}.$$

For $\gamma > 0$, recall that the γ -margin-loss is defined by

$$\ell(\hat{y}, y) = \begin{cases} 0 & \text{if } \gamma < M(\hat{y}, y), \\ 1 - \frac{M(\hat{y}, y)}{\gamma} & \text{if } 0 \leq M(\hat{y}, y) \leq \gamma, \\ 1 & \text{if } M(\hat{y}, y) < 0. \end{cases} \quad (22)$$

For any class c and one-hot encoding y of c , it is known that $\hat{y} \in \mathbb{R}^{d_{\text{out}}} \mapsto M(\hat{y}, y)$ is 2-Lipschitz with respect to the L^2 -norm on \hat{y} (Bartlett et al., 2017, Lemma A.3). Moreover, the function

$$r \in \mathbb{R} \mapsto \begin{cases} 0 & \text{if } r < -\gamma, \\ 1 + \frac{r}{\gamma} & \text{if } -\gamma \leq r \leq 0, \\ 1 & \text{if } r > 0. \end{cases}$$

is $\frac{1}{\gamma}$ -Lipschitz. By composition, this shows that $\hat{y} \in \mathbb{R}^{d_{\text{out}}} \mapsto \ell_\gamma(\hat{y}, y)$ is $\frac{2}{\gamma}$ -Lipschitz with respect to the L^2 -norm.

Proof of Theorem 3.2. Since the labels \mathbf{Y} are one-hot encodings, we equivalently consider \mathbf{Y} either in $\mathbb{R}^{d_{\text{out}}}$ or in $\{1, \dots, d_{\text{out}}\}$. It holds (Bartlett et al., 2017, Lemma A.4)

$$\mathbb{P} \left(\arg \max_c [R_\theta(\mathbf{X})]_c \neq \mathbf{Y} \right) \leq \mathbb{E} (\ell_\gamma(R_\theta(\mathbf{X}), \mathbf{Y}))$$

for any $\gamma > 0$ and associated γ -margin-loss ℓ_γ . Thus, considering the generalization error for ℓ_γ :

$$\mathbb{P} \left(\arg \max_c [R_\theta(\mathbf{X})]_c \neq \mathbf{Y} \right) \leq \underbrace{\frac{1}{n} \sum_{i=1}^n \ell_\gamma \left(R_{\hat{\theta}(\mathbf{Z})}(\mathbf{X}_i), \mathbf{Y}_i \right)}_{= \text{training error of } \hat{\theta}(\mathbf{Z})} + \mathbb{E}_{\mathbf{Z}} \ell_\gamma\text{-generalization error of } \hat{\theta}(\mathbf{Z}).$$

By definition of ℓ_γ , the training error of $\hat{\theta}(\mathbf{Z})$ is at most $\frac{1}{n} \sum_{i=1}^n \mathbb{1}_{[R_{\hat{\theta}(\mathbf{Z})}(\mathbf{X}_i)]_{\mathbf{Y}_i} \leq \gamma + \max_{c \neq \mathbf{Y}_i} [R_{\hat{\theta}(\mathbf{Z})}(\mathbf{X}_i)]_c}$. Moreover, Theorem 3.1 can be used to bound the generalization error associated with ℓ_γ with $L = 2/\gamma$. This proves the claim. \square

J DETAILS ON THE EXPERIMENTS OF SECTION 4

Details for Table 2. All the experiments are done on ImageNet-1k using 99% of the 1,281,167 images of the training set for training, the other 1% is used for validation. Thus, $n = 1268355 = \lfloor 0.99 \times 1281167 \rfloor$ in our experiments, $d_{\text{in}} = 224 \times 224 \times 3 = 150528$, $d_{\text{out}} = 1000$. We also estimated $B = 2.640000104904175$ by taking the maximum of the L^∞ norms of the training images normalized for inference⁹. The PyTorch code for normalization at inference is standard:

```

1 inference_normalization = transforms.Compose([
2     transforms.Resize(256),
3     transforms.CenterCrop(224),
4     transforms.ToTensor(),
5     transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229,
6     0.224, 0.225]),
7 ])

```

⁹The constant σ in Theorem 3.1 corresponds to data \mathbf{Z}_i drawn from the distribution for which we want to evaluate the test error. This is then the data normalized for inference. Thus, the training loss appearing in Theorem 3.1 is also evaluated on the training data \mathbf{Z}_i normalized for inference. In the experiments, we ignore this fact and still evaluate the training loss on the data augmented for training. Moreover, note that it is not possible to recover the training images augmented for training from the images normalized for inference, because cropping is done at random for training. Thus, the real life estimator is not a function of the images \mathbf{Z}_i normalized for inference, and Theorem 3.1 does not apply stricto sensu. This fact is ignored here.

We consider ResNets. They have a single max-pooling layer of kernel size 3×3 so that $K = 9$. The depth is $D = 3 + \# \text{ basic blocks} \times \# \text{ conv per basic block}$, where 3 accounts for the conv1 layer, the average-pooling layer, the fc layer, and the rest accounts for all the convolutional layers in the basic blocks. Note that the average-pooling layer can be incorporated into the next fc layer as it only contains identity neurons, see the discussion after Theorem 3.1, so we can actually consider $D = 2 + \# \text{ basic blocks} \times \# \text{ conv per basic block}$. Table 4 details the relevant values related to basic blocks.

Table 4: Number of basic blocks, of convolutional layer per basic blocks and associated D for ResNets (He et al., 2016, Table 1).

ResNet	18	34	50	101	152
# basic blocks	8	16		33	50
# conv per basic block	2		3		
D	18	34	50	101	152

Pretrained ResNets. The PyTorch pretrained weights that have been selected are the ones with the best performance: `ResNetX_Weights.IMAGENET1K_V1` for ResNets 18 and 34, and `ResNetX_Weights.IMAGENET1K_V2` otherwise.

Choice of $\gamma > 0$ for Theorem 3.2. In Equation (4), note that there is a trade-off when choosing $\gamma > 0$. Indeed, the first term of the right-hand side is non-decreasing with γ while the second one is non-increasing. The first term is simply the proportion of datapoints that are not correctly classified with a margin at least equal to γ . Defining the margin of input i on parameters θ to be $R_{\theta}(\mathbf{X}_i)_{\mathbf{Y}_i} - \arg \max_{c \neq \mathbf{Y}_i} R_{\theta}(\mathbf{X}_i)_c$, this means that the first term is (approximately) equal to q if $\gamma = \gamma(q)$ is the q -quantile of the distribution of the margins over the training set.

Note that since the second term in Equation (4) is of order $1/\sqrt{n}$, it would be desirable to choose the $1/\sqrt{n}$ -quantile (up to a constant) for γ . However, this is not possible in practice as soon as the training top-1 accuracy is too large compared to $1/\sqrt{n}$ (eg. on ImageNet). Indeed, if the training top-1 error is equal to $e \in [0, 1]$, then at least a proportion e of the data margins should be negative¹⁰ so that any q -quantile with $q < e$ is negative and cannot be considered for Theorem 3.2

The distribution of the margins on the training set of ImageNet can be found in Figure 3. The maximum training margin is roughly of size 30, which is insufficient to compensate the size of the L^1 path-norm of pretrained ResNets reported in Table 3. For $\gamma > 30$, the first term of the right-hand side of Theorem 3.2 is greater than one, so that the bound is not informative. This shows that there is no possible choice for $\gamma > 0$ that makes the bound informative on these pretrained ResNets. Table 5 reports a quantile for these pretrained ResNets.

Table 5: The q -quantile $\gamma(q)$ for $q = \frac{1}{3}e + \frac{2}{3}$, with e being the top-1 error, on ImageNet, of pretrained ResNets available on PyTorch.

ResNet	18	34	50	101	152
$\gamma(q)$	5.0	5.6	4.2	5.6	5.8

Details for sparse networks. ResNet18 is trained on 99% of ImageNet with a single GPU using SGD for 90 epochs, learning rate 0.1, weight-decay 0.0001, batch size 1024, and a multi-step scheduler where the learning rate is divided by 10 at epochs 30, 60 and 80. The epoch out of the 90 ones with maximum validation top-1 accuracy is considered as the final epoch. Pruning is done iteratively accordingly to Frankle et al. (2021). We prune 20% of the remaining weights of each convolutional layer, and 10% of the final fully connected layer, at each pruning iteration, save the mask and rewind the weights to their values after the first 5 epochs of the dense network, and train for 85 remaining epochs, before pruning again etc. Results for a single run are shown in Figure 4.

Details for increasing the train size. Instead of training on 99% of ImageNet ($n = 1268355$), we trained a ResNet18 on $n/2^k$ samples drawn at random, for $1 \leq k \leq 5$. For each given k , the results

¹⁰A data margin is negative if and only if it is misclassified.

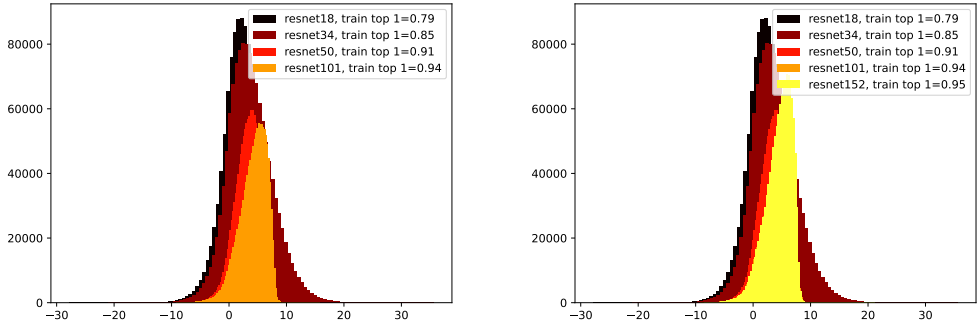


Figure 3: Distribution of the margins on the training set of ImageNet, with the pretrained ResNets available on PyTorch.

are averaged over 3 seeds. The hyperparameters are the same as for sparse networks (except that we do not perform any pruning here): 90 epochs etc. Results are in Figure 5.

K MAIN RELATED WORKS

Given the extent of the literature on generalization bounds, we apologize in advance for papers the reader may find missing below.

Previous definitions of the path-lifting and the path-activations In the work of Kawaguchi et al. (2017, Section 5.1) the path-lifting and the path-activations are evoked in the case of a ReLU DAG with max-pooling neurons *and no biases*, but with no explicit definitions. Definition A.3 gives a formal definition for these objects, and extend it to the case where there are biases, which requires extending the path-lifting to paths starting from neurons that are not input neurons. Moreover, Definition A.3 extends it to arbitrary k -max-pooling neurons (classical max-pooling neurons correspond to $k = 1$).

Note also that the formula Equation (5) is stated in the specific case of Kawaguchi et al. (2017, Section 5.1) (as an explicit sum rather than an inner product), without proof since the objects are not explicitly defined in Kawaguchi et al. (2017).

A formal definition of the path-lifting is given in the specific case of layered fully-connected ReLU neural networks with biases in the work of Stock & Gribonval (2023, Definition 6). Moreover, it is proved that Equation (5) holds *in this specific case* in Stock & Gribonval (2023, Corollary 3). Definition A.3 and Equation (5) generalize the latter to an arbitrary DAG with $*$ -max-pooling or identity neurons (allowing in particular for skip connections, max-pooling and average-pooling).

The rest of the works we are aware of only define and consider the norm of the path-lifting, but not the lifting itself. The most general setting being the one of Neyshabur et al. (2015) with a general DAG, *but without max or identity neurons, nor biases*. Not defining the path-lifting and the path-activations makes notations arguably heavier since Equation (5) is then always written with an explicit sum over all paths, with explicit product of weights along each path, and so on.

Previous generalization bounds based on path-norm See Table 1 for a comparison. Appendix L tackles some other bounds that do not appear in Table 1.

Empirical evaluation of path-norm The formula given in Theorem A.1 is the first one to fully encompass modern networks with biases, average/ $*$ -max-pooling, and skip connections, such as ResNets. An equivalent formula is stated for *layered fully-connected* ReLU networks *without biases* (and *no pooling/skip connections*) in Dziugaite et al. (2020, Appendix C.6.5) and Jiang et al. (2020, Equations (43) and (44)) but without proof (and only for L^2 -path-norm instead of general mixed $L^{q,r}$ path-norm). Actually, *this equivalent formula turns out to be false when there are $*$ -max-pooling neurons* as one must replace $*$ -max-pooling neurons with identity ones, see Theorem A.1. Care must

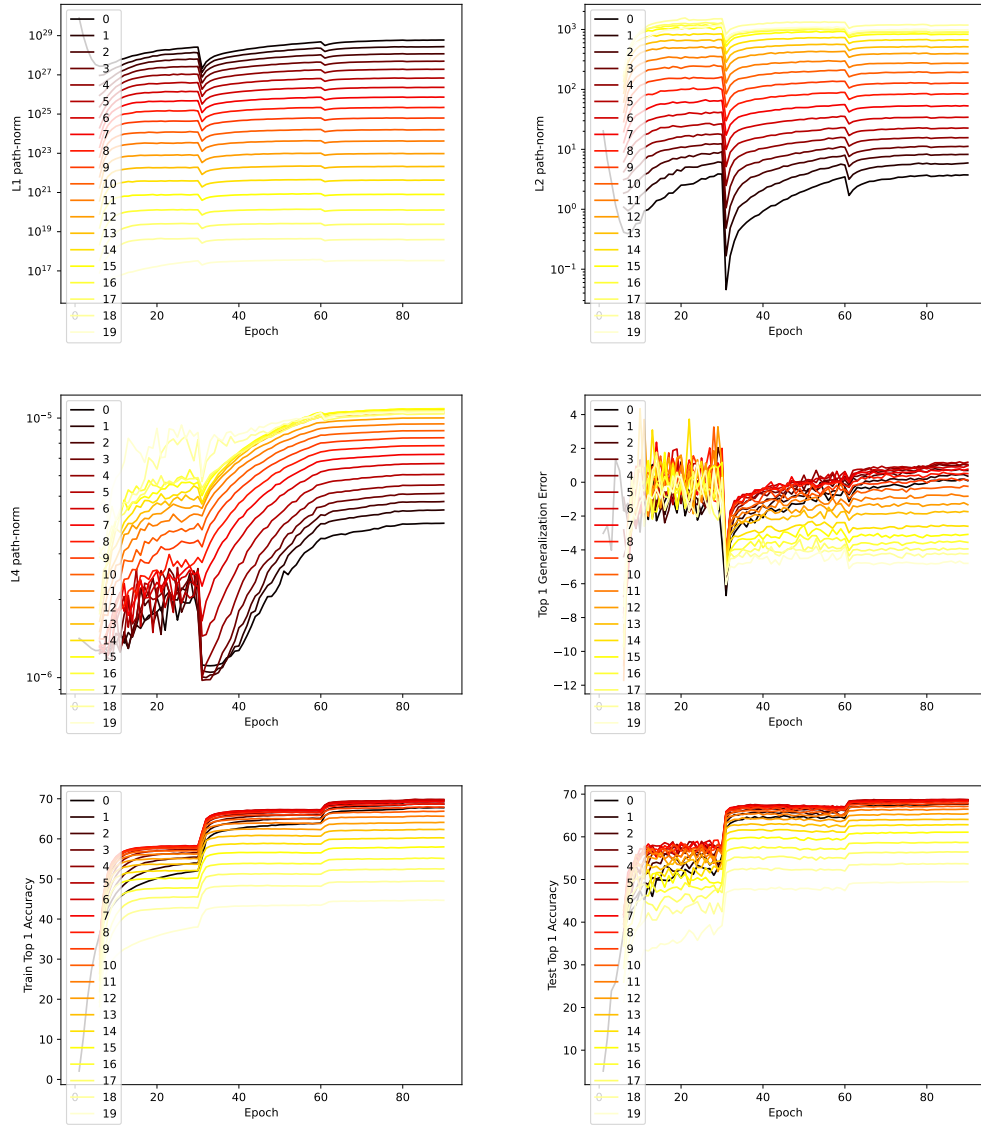


Figure 4: L^q path-norm ($q = 1, 2, 4$), test top-1 accuracy, training top-1 accuracy, and the top-1 generalization error (difference between test top-1 and train top-1) during the training of a ResNet18 on ImageNet. The pruning iteration is indicated in legend, with 0 corresponding to the dense network. The color also indicates the degree of sparsity: from dense (black) to extremely sparse (yellow).

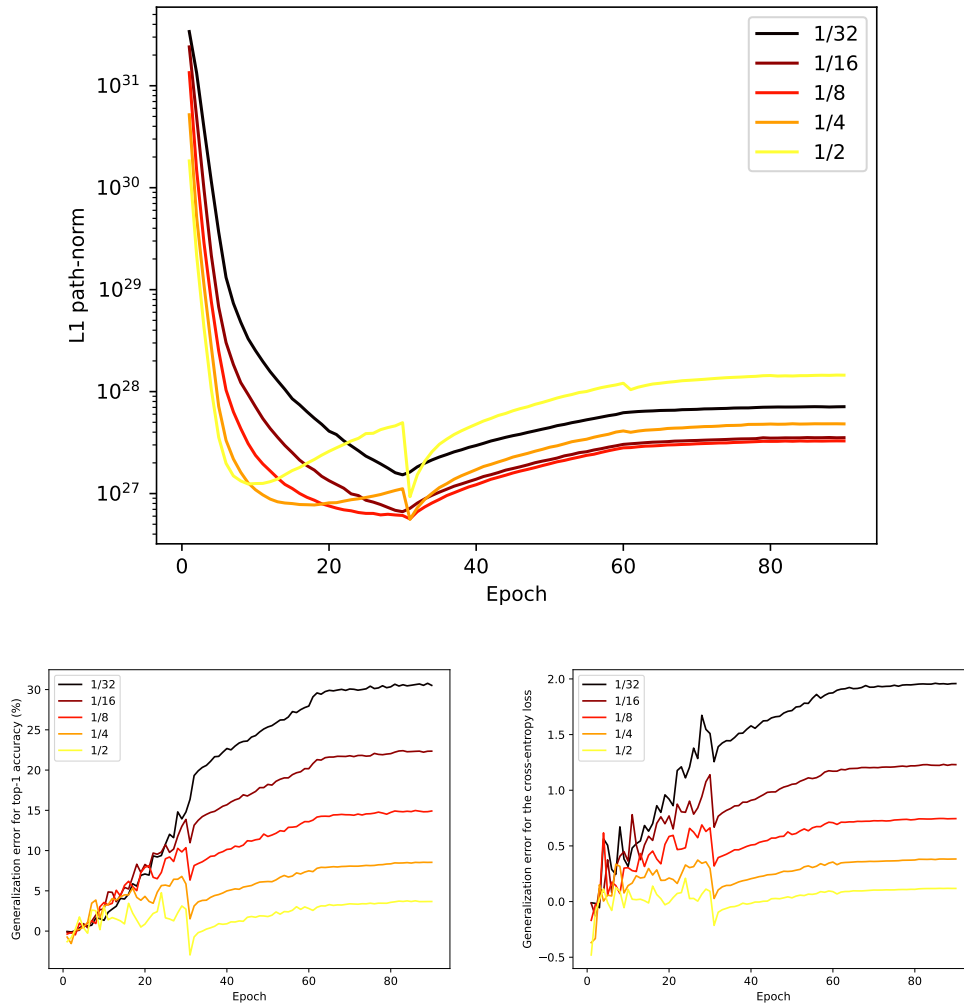


Figure 5: L^1 path-norm, and empirical generalization errors for both the top-1 accuracy and the cross-entropy during the training of a ResNet18 on a subset of the training images of ImageNet. The legend indicates the size of the subset considered, *e.g.* $1/m$ corresponds to $1/m$ of 99% of ImageNet, leaving the other 1% out for validation. The color also indicates the size of the subset: from small (black) to large (yellow).

also be taken with average-pooling neurons that must be rescaled by considering them as identity neurons.

We could not find reported numerical values of the path-norm except for toy examples (Dziugaite, 2018; Furusho, 2020; Zheng et al., 2019). Details are in Appendix L.

Appendix L also discusses 1) inherent limitations of Theorem 3.1 which are common to many generalization bounds, and 2) the applicability of Theorem 3.1 compared to PAC-Bayes bounds.

L MORE RELATED WORKS

More generalization bounds using path-norm In E et al. (2022) is established an additional bound to the one appearing in Table 1, for a class of functions and a complexity measure that are related to the *infinite-depth* limits of residual networks and the path-norm. However, it is unclear how this result implies anything for actual neural networks with the actual path-norm ¹¹.

The bound in Zheng et al. (2019) only holds for layered fully-connected ReLU neural networks (*no max or identity neurons*) with *no biases* and *it grows exponentially* with the depth of the network. It is not included in Table 1 because it requires an additional assumption: the coordinates of the path-lifting must not only be bounded from above, but also *from below*. The reason for this assumption is not discussed in Zheng et al. (2019), and it is unclear whether this is at all desirable, since a small path-norm can only be better for generalization in light of Theorem 3.1.

Theorem 4.3 in Golowich et al. (2018) is a bound that holds for layered fully-connected ReLU neural networks with *no biases (no max and no identity neurons)* and it *depends on the product of operator norms of the layers*. It has the merit of having no dependence on the size of the architecture (depth, width, number of neurons etc.). However, it requires an additional assumption: each layer must have an operator norm bounded from below, so that it only applies to a restricted set of layered fully-connected networks. Moreover, it is unclear whether such an assumption is desirable: there are networks with arbitrary small operator norms that realize the zero function, and the latter has a generalization error equal to zero.

Theorem 8 in Kawaguchi et al. (2017) gives a generalization bound for *scalar-valued* ($d_{\text{out}} = 1$) models with an output of the form $\langle \Phi(\theta), \mathbf{A}(\theta', x)x \rangle$ for some specific parameters θ, θ' that have no reason to be equal. This is orthogonal to the case of neural networks where one must have $\theta = \theta'$, and it is therefore not included in Table 1. Theorem 5 in Kawaguchi et al. (2017) can be seen as a possible first step to derive a bound based on path-norm in the specific case of the mean squared error loss. However, as discussed in more details below, Theorem 5 in Kawaguchi et al. (2017) is a rewriting of the generalization error with several terms that are as complex to bound as the original generalization error, resulting in a bound being as hard as the generalization error to evaluate/estimate.

More details about Theorem 5 in Kawaguchi et al. (2017) We start by re-deriving Theorem 5 in Kawaguchi et al. (2017). In the specific case of mean squared error, using that

$$\|R_{\theta}(x) - y\|_2^2 = \|R_{\theta}(x)\|_2^2 + \|y\|_2^2 - 2 \langle R_{\theta}(x), y \rangle,$$

¹¹E et al. (2022) starts from the fact that the infinite-depth limits of residual networks can be characterized with partial differential equations. Thus E et al. (2022) establishes a bound for functions characterized by similar, but different, partial differential equations, using what seems to be an analogue of path-norm for these new functions. However, even if the characterizations of these functions are closed, as it is said in E et al. (2022), "it is unclear how the two spaces are related".

it is possible to rewrite the generalization error as follows:

$$\begin{aligned} \text{generalization error of } \hat{\theta}(\mathbf{Z}) &= \mathbb{E} \left(\|R_{\hat{\theta}(\mathbf{Z})}(\tilde{\mathbf{X}}) - \tilde{\mathbf{Y}}\|_2^2 | \mathbf{Z} \right) - \frac{1}{n} \sum_{i=1}^n \|R_{\hat{\theta}(\mathbf{Z})}(\mathbf{X}_i) - \mathbf{Y}_i\|_2^2 \\ &= \mathbb{E} \left(\|R_{\hat{\theta}(\mathbf{Z})}(\tilde{\mathbf{X}})\|_2^2 | \mathbf{Z} \right) - \frac{1}{n} \sum_{i=1}^n \|R_{\hat{\theta}(\mathbf{Z})}(\mathbf{X}_i)\|_2^2 \\ &\quad + \mathbb{E} \left(\|\tilde{\mathbf{Y}}\|_2^2 | \mathbf{Z} \right) - \sum_{i=1}^n \|\mathbf{Y}_i\|_2^2 \\ &\quad - 2\mathbb{E} \left(\left\langle R_{\hat{\theta}(\mathbf{Z})}(\tilde{\mathbf{X}}), \tilde{\mathbf{Y}} \right\rangle | \mathbf{Z} \right) - \frac{1}{n} \sum_{i=1}^n \left\langle R_{\hat{\theta}(\mathbf{Z})}(\mathbf{X}_i), \mathbf{Y}_i \right\rangle. \end{aligned}$$

It is then possible to make the L^2 path-norm appear. For instance, for one-dimensional output networks, it can be proven (see Lemma A.1) that $R_{\theta}(x) = \langle \Phi(\theta), z(x, \theta) \rangle$ with $\Phi(\theta)$ the path-lifting of parameters θ , and z that typically depends on the path-activations and the input, so that the first term above can be rewritten

$$\Phi(\hat{\theta}(\mathbf{Z}))^T \left(\mathbb{E} \left(z(\tilde{\mathbf{X}}, \hat{\theta}(\mathbf{Z})) z(\tilde{\mathbf{X}}, \hat{\theta}(\mathbf{Z}))^T | \mathbf{Z} \right) - \frac{1}{n} \sum_{i=1}^n z(\mathbf{X}_i, \hat{\theta}(\mathbf{Z})) z(\mathbf{X}_i, \hat{\theta}(\mathbf{Z}))^T \right) \Phi(\hat{\theta}(\mathbf{Z})).$$

Let us call a "generalization error like quantity" any term of the form

$$\mathbb{E} \left(f_{\hat{\theta}(\mathbf{Z})}(\tilde{\mathbf{Z}}) | \mathbf{Z} \right) - \frac{1}{n} \sum_{i=1}^n f_{\hat{\theta}(\mathbf{Z})}(\mathbf{Z}_i),$$

that is, any term that can be represented as a difference between the estimator learned from training data \mathbf{Z} evaluated on test data $\tilde{\mathbf{Z}}$, and the evaluation on the training data. We see that the derivation above replaces the classical generalization error with two others quantities similar in definition to the generalization error. This derivation, which is specific to mean squared error, leads to Theorem 5 in Kawaguchi et al. (2017). Very importantly, note that this derivation trades a single quantity similar to generalization error for two new such quantities. There is no discussion in Kawaguchi et al. (2017) on how to bound these two new terms. Without any further new idea, there is no other way than the ones developed in the literature so far: reduce the problem to bounding a Rademacher complexity (as it is done in Theorem 3.1), or use the PAC-Bayes framework, and so on.

More on numerical evaluation of path-norm In Dziugaite (2018, Section 2.9.1) is reported numerical evaluations after 5 epochs of SGD on a one hidden layer network trained on a binary variant of MNIST. Furusho (2020, Figure 9 and Section 3.3.1) deals with 1d regression with 5 layers and 100 width. Experiments in Zheng et al. (2019) are on MNIST. Note that it is not clear whether the path-norm used in Zheng et al. (2019) corresponds to the one defined in Definition A.3. Indeed, the references given in Zheng et al. (2019) for the definition of the path-norm are both Neyshabur et al. (2015) and Neyshabur et al. (2017), but these two papers have two different definitions of the path-norm. The one in Neyshabur et al. (2015) corresponds to the norm of the path-lifting as defined in Definition A.3 (but in simpler settings: no pooling etc.), while the one in Neyshabur et al. (2017) corresponds to the latter divided by the margin of the estimator.

For completeness, let us also mention that it is reported in Dziugaite et al. (2020); Jiang et al. (2020) whether the path-norm correlates with the empirical generalization error or not, but there is no report of numerical values of the path-norm. In Neyshabur et al. (2017) are reported the quotient of path-norms with margins, but not the path-norms alone.

Inherent limitations of uniform convergence bounds Theorem 3.1 has some inherent limitations due to its nature. It is *data-dependent* as it depends on the input distribution. However, it does not depend on the label distribution, making it uninformative as soon as Θ is so much expressive that it can fit random labels. Networks that can fit random labels have already been found empirically (Zhang et al., 2021), and it is open whether this stays true with a constraint on the path-norm.

Theorem 3.1 is *based on a uniform convergence bound*¹² as any other bound also based on a control of a Rademacher complexity. Nagarajan & Kolter (2019) empirically argue that even the tightest

¹²A uniform convergence bound on a model class \mathcal{F} is a bound on $\mathbb{E}_{\mathbf{Z}} \sup_{f \in \mathcal{F}} \text{generalization error } f(\mathbf{Z})$. This worst-case type of bound can lead to potential limitations when \mathcal{F} is too expressive.

uniform convergence bound holding with high probability must be loose on some synthetic datasets. If this was confirmed theoretically, this would still allow uniform bounds to be tight when considering other datasets than the one in Nagarajan & Kolter (2019), such as real-world datasets, or when the estimator considered in Nagarajan & Kolter (2019) is not in Θ (for instance because of constraints on the slopes via the path-norm).

Finally, Theorem 3.1 can provide theoretical guarantees on the generalization error of the output of a learning algorithm, but only *a posteriori*, after training. In order to have a priori guarantees, one should have to derive a priori knowledge on the path-norm at the end of the learning algorithm.

Comparison to PAC-Bayes bounds Another interesting direction to understand generalization of neural networks is the PAC-Bayes framework (Guedj, 2019; Alquier, 2021). Unfortunately, PAC-Bayes bounds cannot be exactly computed on networks that are trained in a usual way. Indeed, these bounds typically involve a KL-divergence, or something related, for which there is no closed form except for very specific distributions (iid Gaussian/Cauchy weights...) that do not correspond to the distributions of the weights after a usual training^{13,14}. We are aware of two research directions that try to get over this issue. The first way is to change the learning algorithm by enforcing the weights to be iid normally distributed, and then optimize the parameters of these normal distributions, see for instance the pioneer work of Dziugaite & Roy (2017). The merit of this new learning algorithm is that it has explicitly been designed with the goal of having a small generalization error. Practical performance are worse than with usual training, but this leads to networks with an associated non-vacuous generalization bound. To the best of our knowledge, this is the only way to get a non-vacuous bound¹⁵, and unfortunately, this does not apply to usual training. The second way to get over the intractable evaluation of the KL-divergence is to 1) try to approximate the bound within reasonable time, and 2) try to quantify the error made with the approximation (Pérez & Louis, 2020). Unfortunately, to the best of our knowledge, approximation is often based on a distribution assumption of the weights that is not met in practice (*e.g.* iid Gaussian weights), approximation is costly, and the error is unclear when applied to networks trained usually. For instance, the bound in Pérez & Louis (2020, Section 5) 1) requires at least $O(n^2)$ operations to be evaluated, with n being the number of training examples, thus being prohibitive for large n (Pérez & Louis, 2020, Section 7), and 2) it is unclear what error is being made when using a Gaussian process as an approximation of the neural network learned by SGD.

¹³The randomness of the weights after training comes from the random initialization and the randomness in the algorithm (*e.g.* random batch in SGD).

¹⁴For instance, independence is not empirically observed, see Frankle et al. (2020, Section 5.2)

¹⁵Except, of course, for methods that are based on the evaluation of the performance on held-out data.