Nearly-tight VC-dimension and Pseudodimension Bounds for Piecewise Linear Neural Networks[*]

Peter L. Bartlett   BARTLETT@CS.BERKELEY.EDU
Department of Statistics and Computer Science Division
University of California
Berkeley, CA 94720-3860, USA

Nick Harvey   NICKHAR@CS.UBC.CA
Christopher Liaw   CVLIAW@CS.UBC.CA
Abbas Mehrabian[†]   ABBAS.MEHRABIAN@GMAIL.COM
Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4, Canada

Editor: Nicolas Vayatis

Abstract

We prove new upper and lower bounds on the VC-dimension of deep neural networks with the ReLU activation function. These bounds are tight for almost the entire range of parameters. Letting W be the number of weights and L be the number of layers, we prove that the VC-dimension is $O(WL\log(W))$, and provide examples with VC-dimension $\Omega(WL\log(W/L))$. This improves both the previously known upper bounds and lower bounds. In terms of the number U of non-linear units, we prove a tight bound $\Theta(WU)$ on the VC-dimension. All of these bounds generalize to arbitrary piecewise linear activation functions, and also hold for the pseudodimensions of these function classes.

Combined with previous results, this gives an intriguing range of dependencies of the VC-dimension on depth for networks with different non-linearities: there is no dependence for piecewise-constant, linear dependence for piecewise-linear, and no more than quadratic dependence for general piecewise-polynomial.

Keywords: VC-dimension, pseudodimension, neural networks, ReLU activation function, statistical learning theory

1. Introduction

Deep neural networks underlie many of the recent breakthroughs in applied machine learning, particularly in image and speech recognition. These successes motivate a renewed study of these networks' theoretical properties.

Classification is one of the learning tasks in which deep neural networks have been particularly successful, e.g., for image recognition. A natural foundational question that arises is: what are the generalization guarantees of these networks in a statistical learning framework? An established way to address this question is by considering the VC-dimension, which characterizes uniform convergence of misclassification frequencies to probabilities (see Vapnik and Chervonenkis, 1971), and asymptotically determines the sample complexity of PAC learning with such classifiers (see Blumer, Ehrenfeucht, Haussler, and Warmuth, 1989).

Definition 1 (growth function, VC-dimension, shattering) *Let H denote a class of functions from $\mathcal{X}$ to $\{0, 1\}$ (the hypotheses, or the classification rules). For any non-negative integer m, we define the* growth function *of H as*

$$\Pi_H(m) := \max_{x_1, \ldots, x_m \in \mathcal{X}} |\{(h(x_1), \ldots, h(x_m)) : h \in H\}|\,.$$

If $|\{(h(x_1), \ldots, h(x_m)) : h \in H\}| = 2^m$, we say H shatters the set $\{x_1, \ldots, x_m\}$. The Vapnik-Chervonenkis dimension of H, denoted $\mathrm{VCdim}(H)$, is the size of the largest shattered set, i.e. the largest m such that $\Pi_H(m) = 2^m$. If there is no largest m, we define $\mathrm{VCdim}(H) = \infty$.

For a class of real-valued functions, such as those generated by neural networks, a natural measure of complexity that implies similar uniform convergence properties is the pseudodimension, introduced by Pollard (1990) (see, e.g., Anthony and Bartlett (1999, Theorem 19.2)).

Definition 2 (pseudodimension) *Let $\mathcal{F}$ be a class of functions from $\mathcal{X}$ to $\Re$. The pseudodimension of $\mathcal{F}$, written $\mathrm{Pdim}(\mathcal{F})$, is the largest integer m for which there exists $(x_1, \ldots, x_m, y_1, \ldots, y_m) \in \mathcal{X}^m \times \Re^m$ such that for any $(b_1, \ldots, b_m) \in \{0, 1\}^m$ there exists $f \in \mathcal{F}$ such that*

$$\forall i : f(x_i) > y_i \Longleftrightarrow b_i = 1$$

For a class $\mathcal{F}$ of real-valued functions, we define $\mathrm{VCdim}(\mathcal{F}) := \mathrm{VCdim}(\mathrm{sgn}(\mathcal{F}))$, where

$$\mathrm{sgn}(\mathcal{F}) := \{\mathrm{sgn}(f) : f \in \mathcal{F}\}$$

and $\mathrm{sgn}(x) = \mathbf{1}[x > 0]$. For any class $\mathcal{F}$, clearly $\mathrm{VCdim}(\mathcal{F}) \leq \mathrm{Pdim}(\mathcal{F})$. If $\mathcal{F}$ is the class of functions generated by a neural network $\mathcal{N}$ with a fixed architecture and fixed activation functions (see Section 1.3 for definitions), then it is not hard to see that indeed $\mathrm{Pdim}(\mathcal{F}) \leq \mathrm{VCdim}(\mathcal{F}')$, where $\mathcal{F}'$ is the class of functions generated by a certain neural network with one more parameter and one more layer than $\mathcal{N}$ (see Anthony and Bartlett (1999, Theorem 14.1) for a proof). Therefore, all the results of this paper automatically apply to the pseudodimensions of neural networks as well, after appropriate adjustments.

The main contribution of this paper is to prove nearly-tight bounds on the VC-dimension of deep neural networks in which the non-linear activation function is a piecewise linear function with a constant number of pieces. For simplicity we will henceforth refer to such networks as "piecewise linear networks". The activation function that is the most commonly used in practice is the *rectified linear unit*, also known as *ReLU* (see LeCun, Bengio, and

Hinton, 2015; Goodfellow, Bengio, and Courville, 2016). The ReLU function is defined as $\sigma(x) = \max\{0, x\}$, so it is clearly piecewise linear.

It is particularly interesting to understand how the VC-dimension is affected by the various attributes of the network: the number W of parameters (i.e., weights and biases), the number U of non-linear units (i.e., nodes), and the number L of layers. Among all networks with the same size (number of weights), is it true that those with more layers have larger VC-dimension?

Such a statement is indeed true, and previously known; however, a tight characterization of how depth affects VC-dimension was unknown prior to this work.

1.1. Our results

Our first main result is a new VC-dimension lower bound that holds even for the restricted family of ReLU networks.

Theorem 3 (Main lower bound) *There exists a universal constant C such that the following holds. Given any W, L with $W > CL > C^2$, there exists a ReLU network with $\leq L$ layers and $\leq W$ parameters with VC-dimension $\geq WL \log(W/L)/C$.*

Remark 4 *Our construction can be augmented slightly to give a neural network with linear threshold and identity activation functions with the same guarantees.*

Remark 5 *Our goal in this paper is to give asymptotic lower and upper bounds. However, one may wonder what is the smallest depth for which we can give a positive lower bound for the VC-dimension. Our Theorem 11 gives a positive lower bound as soon as the number of layers is at least 8 (which corresponds to $k = 1$ in that theorem).*

The proof appears in Section 2. Prior to our work, the best known lower bounds were $\Omega(WL)$ (see Bartlett, Maiorov, and Meir, 1998, Theorem 2) and $\Omega(W \log W)$ (see Maass, 1994, Theorem 1). We strictly improve both bounds to $\Omega(WL \log(W/L))$.

Our proof of Theorem 3 uses the "bit extraction" technique, which was also used by Bartlett et al. (1998) to give an $\Omega(WL)$ lower bound. We refine this technique to gain the additional logarithmic factor that appears in Theorem 3.

Unfortunately there is a barrier to refining this technique any further. Our next theorem shows the hardness of computing the mod function, implying that the bit extraction technique cannot yield a stronger lower bound than Theorem 3. Further discussion of this connection may be found in Remark 14.

Theorem 6 *Assume there exists a piecewise linear network with W parameters and L layers that computes a function $f : \Re \to \Re$, with the property that $|f(x) - (x \bmod 2)| < 1/2$ for all $x \in \{0, 1, \ldots, 2^m - 1\}$. Then we have $m = O(L \log(W/L))$.*

The proof of this theorem appears in Section 3. One interesting aspect of the proof is that it does not use Warren's lemma (Warren, 1968), which is a mainstay of VC-dimension upper bounds (see Goldberg and Jerrum, 1995; Bartlett et al., 1998; Anthony and Bartlett, 1999).

Our next main result is an upper bound on the VC-dimension of neural networks with piecewise polynomial activation functions.

Theorem 7 (Main upper bound) *Consider a neural network architecture with W parameters and U computation units arranged in L layers, so that each unit has connections only from units in earlier layers. Let k_i denote the number of units at the ith layer. Suppose that all non-output units have piecewise-polynomial activation functions with $p + 1$ pieces and degree no more than d, and the output unit has the identity function as its activation function.*

If $d = 0$, let W_i denote the number of parameters (weights and biases) at the inputs to units in layer i; if $d > 0$, let W_i denote the total number of parameters (weights and biases) at the inputs to units in all the layers **up to layer** *i (i.e., in layers $1, 2, \ldots, i$). Define the effective depth as*

$$\bar{L} := \frac{1}{W} \sum_{i=1}^{L} W_i,$$

and let

$$R := \sum_{i=1}^{L} k_i(1 + (i-1)d^{i-1}) \leq U + U(L-1)d^{L-1}. \tag{1}$$

For the class $\mathcal{F}$ of all (real-valued) functions computed by this network and $m \geq \bar{L}W$, we have

$$\Pi_{\mathrm{sgn}(\mathcal{F})}(m) \leq \prod_{i=1}^{L} 2 \left(\frac{2emk_ip(1 + (i-1)d^{i-1})}{W_i} \right)^{W_i},$$

and if $U > 2$ then

$$\mathrm{VCdim}(\mathcal{F}) \leq L + \bar{L}W \log_2(4epR \log_2(2epR)) = O(\bar{L}W \log(pU) + \bar{L}LW \log d).$$

In particular, if $d = 0$, then

$$\mathrm{VCdim}(\mathcal{F}) \leq L + W \log_2(4epU \log_2(2epU)) = O(W \log(pU));$$

and if $d = 1$, then

$$\mathrm{VCdim}(\mathcal{F}) \leq L + \bar{L}W \log_2 \left(4ep \sum_{i=1}^{L} ik_i \log_2 \left(\sum_{i=1}^{L} 2epik_i \right) \right) = O(\bar{L}W \log(pU)).$$

Remark 8 *The average depth $\bar{L}$ is always between 1 and L, and captures how the parameters are distributed in the network: it is close to 1 if they are concentrated near the output (or if the activation functions are piecewise-constant), while it is of order L if the parameters are concentrated near the input, or are spread out throughout the network. Hence, this suggests that edges and vertices closer to the input have a larger effect in increasing the VC-dimension, a phenomenon not observed before; and indeed our lower bound construction in Theorem 3 (as well as the lower bound construction from Bartlett et al. (1998)) considers a network with most of the parameters near the input.*

Remark 9 *If we have $\sum_{i=1}^{L} k_i \leq \sum_{i=1}^{L} W_i$, which holds for most network architectures used in practice, the upper bound on the growth function can be simplified into*

$$\Pi_{\mathrm{sgn}(\mathcal{F})}(m) \leq \left(4emp(1 + (L-1)d^{L-1}) \right)^{\sum W_i}.$$

The proofs of Theorem 7 and Remark 9 appear in Section 4. Prior to our work, the best known upper bounds were $O(W^2)$ (see Goldberg and Jerrum, 1995, Section 3.1) and $O(WL \log W + WL^2)$ (see Bartlett et al., 1998, Theorem 1), both of which hold for piecewise polynomial activation functions with a bounded number of pieces (for the remainder of this section, assume that $p = O(1)$ throughout); we strictly improve the second bound to $O(WL \log W)$ for the special case of piecewise linear functions ($d = 1$). Recall that ReLU is an example of a piecewise linear activation function. For the case $d = 0$, an $O(W \log U)$ bound for the VC-dimension was already proved using different techniques by Cover (1968) and by Baum and Haussler (1989, Corollary 2). Our Theorem 7 implies all of these upper bounds (except the $O(W^2)$ upper bound of Goldberg and Jerrum) using a unified technique, and gives a slightly more refined picture of the dependence of the VC-dimension on the distribution of parameters in a deep network.

To compare our upper and lower bounds, let $d(W, L)$ denote the largest VC-dimension of a piecewise linear network with W parameters and L layers. Theorems 3 and 7 imply there exist constants c, C such that

$$c \cdot WL \log(W/L) \;\leq\; d(W, L) \;\leq\; C \cdot WL \log W \,. \tag{2}$$

For neural networks arising in practice it would certainly be the case that L is significantly smaller than $W^{0.99}$, in which case our results determine the asymptotic bound $d(W, L) = \Theta(WL \log W)$. On the other hand, in the regime $L = \Theta(W)$, which is merely of theoretical interest, we also now have a tight bound $d(W, L) = \Theta(WL)$, obtained by combining Theorem 3 with results of Goldberg and Jerrum (1995). There is now only a very narrow regime, say $W^{0.99} \ll L \ll W$, in which the bounds of (2) are not asymptotically tight, and they differ only in the logarithmic factor.

Our final result is an upper bound for VC-dimension in terms of W and U (the number of non-linear units, or nodes). This bound is tight in the case $d = 1$ and $p = 2$, as discussed in Remark 12.

Theorem 10 *Consider a neural network with W parameters and U units with activation functions that are piecewise polynomials with at most p pieces and of degree at most d. Let $\mathcal{F}$ be the set of (real-valued) functions computed by this network. Then $\mathrm{VCDim}(\mathrm{sgn}(\mathcal{F})) = O(WU \log((d+1)p))$.*

The proof of this result appears in Section 5. The best known upper bound before our work was $O(W^2)$, implicitly proven for bounded d and p by Goldberg and Jerrum (1995, Section 3.1). Our theorem improves this to the tight result $O(WU)$.

We can summarize the tightest known results on the VC-dimension of neural networks with piecewise polynomial activation functions as follows: for classes $\mathcal{F}$ of functions computed by the class of networks with L layers, W parameters, and U units with the following non-linearities, we have the following bounds on VC-dimension:

Piecewise constant. $\mathrm{VCdim}(\mathcal{F}) = \Theta(W \log W)$ (Cover (1968) and Baum and Haussler (1989) showed the upper bound and Maass (1994) showed the lower bound).

Piecewise linear. $c \cdot WL \log(W/L) \;\leq\; \mathrm{VCdim}(\mathcal{F}) \;\leq\; C \cdot WL \log W$ (this paper).

Piecewise polynomial. $\text{VCdim}(\mathcal{F}) = O(WL^2 + WL \log W)$ (Bartlett et al., 1998), and $\text{VCdim}(\mathcal{F}) = O(WU)$ (this paper), and $\text{VCdim}(\mathcal{F}) = \Omega(WL \log(W/L))$ (this paper). We note that closing the gap here requires fundamentally new techniques (see Remark 14).

1.2. Related work

For a general overview on theoretical properties of neural networks, we refer the reader to the monograph by Anthony and Bartlett (1999). There are various methods, other than the VC-dimension, for bounding the generalization error of neural networks; see Jakubovitz, Giryes, and Rodrigues (2018) for a recent survey.

In the reset of this section, we review some previous work studying the impact of depth on the representational power of neural networks. It has long been known that two-layer networks with a variety of activation functions can approximate arbitrary continuous functions on compact sets (Hornik, 1991). Sontag (1992) showed that three-layer networks of linear threshold units can approximate inverses of continuous functions, whereas two-layer networks cannot. There are several recent papers that aim to understand which functions can be expressed using a neural network of a given depth and size. There are technical similarities between our work and these. Two striking papers considered the problem of approximating a deep neural network with a shallower network. Telgarsky (2016) shows that there is a ReLU network with L layers and $U = \Theta(L)$ units such that any network approximating it with only $O(L^{1/3})$ layers must have $\Omega(2^{L^{1/3}})$ units; this phenomenon holds even for real-valued functions. Eldan and Shamir (2016) show an analogous result for a high-dimensional 3-layer network that cannot be approximated by a 2-layer network except with an exponential blow-up in the number of nodes.

Very recently, several authors have shown that deep neural networks are capable of approximating broad classes of functions. Safran and Shamir (2017) show that a sufficiently non-linear C^2 function on $[0, 1]^d$ can be approximated with ϵ error in L_2 by a ReLU network with $O(\text{polylog}(1/\epsilon))$ layers and weights, but any such approximation with $O(1)$ layers requires $\Omega(1/\epsilon)$ weights. Yarotsky (2017) shows that any C^n-function on $[0, 1]^d$ can be approximated with ϵ error in L_∞ by a ReLU network with $O(\log(1/\epsilon))$ layers and $O((\frac{1}{\epsilon})^{d/n} \log(1/\epsilon))$ weights. Liang and Srikant (2017) show that a sufficiently smooth univariate function can be approximated with ϵ error in L_∞ by a network with ReLU and threshold gates with $\Theta(\log(1/\epsilon))$ layers and $O(\text{polylog}(1/\epsilon))$ weights, but that $\Omega(\text{poly}(1/\epsilon))$ weights would be required if there were only $o(\log(1/\epsilon))$ layers; they also prove analogous results for multivariate functions. Lastly, Cohen, Sharir, and Shashua (2016) draw a connection to tensor factorizations to show that, for a certain family of arithmetic circuits (in particular, without ReLU non-linearities), the set of functions computable by a shallow network have measure zero among those computable by a deep networks. For more recent results on representation power of neural networks, see Grohs, Perekrestenko, Elbrächter, and Bölcskei (2019) and the references therein.

1.3. Notation

A neural network is defined by an *activation function* $\psi : \Re \to \Re$, a directed acyclic graph, and a set of parameters: a weight for each edge of the graph, and a bias for each node of

the graph. Let W denote the number of parameters (weights and biases) of the network, U denote the number of computation units (nodes), and L denote the length of the longest path in the graph. We will say that the neural network has L layers.

Layer 0 consists of nodes with in-degree 0. We call these nodes input nodes and they simply output the real value given by the corresponding input to the network. We assume that the graph has a single sink node; this is the unique node in layer L, which we call the output layer. This output node can have predecessors in any layer $\ell < L$. For $1 \le \ell < L$, a node is in layer ℓ if it has a predecessor in layer $\ell - 1$ and no predecessor in any layer $\ell' \ge \ell$. (Note that for example there could be an edge connecting a node in layer 1 with a node in layer 3.) In the jargon of neural networks, layers 1 through $L - 1$ are called *hidden* layers.

The computation of a neural network proceeds as follows. For $i = 1, \ldots, L$, the input into a computation unit u at layer i is $w^\top x + b$, where x is the (real) vector corresponding to the outputs of the computational units with a directed edge to u, w is the corresponding vector of edge weights, and b is the bias parameter associated with u. For layers $1, \ldots, L-1$, the output of u is $\psi(w^\top x + b)$. For the output layer, we replace ψ with the identity, so the output is simply $w^\top x + b$. Since we consider VC-dimension, we will always take the sign of the output of the network, to make the output lie in $\{0, 1\}$ for binary classification.

A piecewise polynomial function with p pieces is a function f for which there exists a partition of $\Re$ into disjoint intervals (pieces) $I_1, \ldots, I_p$ and corresponding polynomials $f_1, \ldots, f_p$ such that if $x \in I_i$ then $f(x) = f_i(x)$. A piecewise linear function is a piecewise polynomial function in which each f_i is linear. The most common activation function used in practice is the rectified linear unit (ReLU) where $I_1 = (-\infty, 0]$, $I_2 = (0, \infty)$ and $f_1(x) = 0, f_2(x) = x$. We denote this function by $\sigma(x) := \max\{0, x\}$. The set $\{1, 2, \ldots, n\}$ is denoted $[n]$.

2. Proof of Theorem 3

The proof of our main lower bound uses the "bit extraction" technique from Bartlett et al. (1998), who proved an $\Omega(WL)$ lower bound. We refine this technique in a key way — we partition the input bits into blocks and extract multiple bits at a time instead of a single bit at a time. This yields a more efficient bit extraction network, and hence a stronger VC-dimension lower bound.

We show the following result, which immediately implies Theorem 3.

Theorem 11 *Let r, m, n be positive integers, and let $k = \lceil m/r \rceil$. There exists a ReLU network with $3 + 5k$ layers, $2 + n + 4m + k((11 + r)2^r + 2r + 2)$ parameters, $m + n$ input nodes and $m + 2 + k(5 \times 2^r + r + 1)$ computational nodes with VC-dimension $\ge mn$.*

Remark 12 *Choosing $r = 1$ gives a network with $W = O(m + n)$, $U = O(m)$ and VC-dimension $\Omega(mn) = \Omega(WU)$. This implies that the upper bound $O(WU)$ given in Theorem 10 is tight.*

To prove Theorem 3, assume W, L, and W/L are sufficiently large, and set $r = \log_2(W/L)/2$, $m = rL/8$, and $n = W - 5m2^r$ in Theorem 11. The rest of this section is devoted to proving Theorem 11.

Let $S_n \subseteq \Re^n$ denote the standard basis. We shatter the set $S_n \times S_m$. Given an arbitrary function $f \colon S_n \times S_m \to \{0,1\}$, we build a ReLU neural network that takes as input $(x_1, x_2) \in S_n \times S_m$ and outputs $f(x_1, x_2)$. Define n numbers $a_1, a_2, \ldots, a_n \in \{\frac{0}{2^m}, \frac{1}{2^m}, \ldots, \frac{2^m-1}{2^m}\}$ so that the ith digit of the binary representation of a_j equals $f(e_j, e_i)$. These numbers will be used as the parameters of the network, as described below.

Given input $(x_1, x_2) \in S_n \times S_m$, assume that $x_1 = e_j$ and $x_2 = e_i$. The network must output the ith bit of a_j. This "bit extraction approach" was used in Bartlett et al. (1998, Theorem 2) to give an $\Omega(WL)$ lower bound for the VC-dimension. We use a similar approach but we introduce a novel idea: we split the bit extraction into blocks and extract r bits at a time instead of a single bit at a time. This allows us to prove a lower bound of $\Omega(WL\log(W/L))$. One can ask, naturally, whether this approach can be pushed further. Our Theorem 6 implies that the bit extraction approach cannot give a lower bound better than $\Omega(WL\log(W/L))$ (see Remark 14).

The first layer of the network "selects" a_j, and the remaining layers "extract" the ith bit of a_j. In the first layer we have a single computational unit that calculates

$$a_j = (a_1, \ldots, a_n)^\top x_1 = \sigma\left((a_1, \ldots, a_n)^\top x_1\right).$$

This part uses 1 layer, 1 computation unit, and $1+n$ parameters.

The rest of the network extracts all bits of a_j and outputs the ith bit. The extraction is done in k steps, where in each step we extract the r most significant bits and zero them out. We will use the following building block for extracting r bits.

Lemma 13 *Suppose positive integers r and m are given. There exists a ReLU network with 5 layers, $5 \times 2^r + r + 1$ units and $11 \times 2^r + r2^r + 2r + 2$ parameters that given the real number $b = 0.b_1b_2\ldots b_m$ (in binary representation) as input, outputs the $(r+1)$-dimensional vector $(b_1, b_2, \ldots, b_r, 0.b_{r+1}b_{r+2}\ldots b_m)$.*

Figure 1 shows a schematic of the ReLU network in the above lemma.
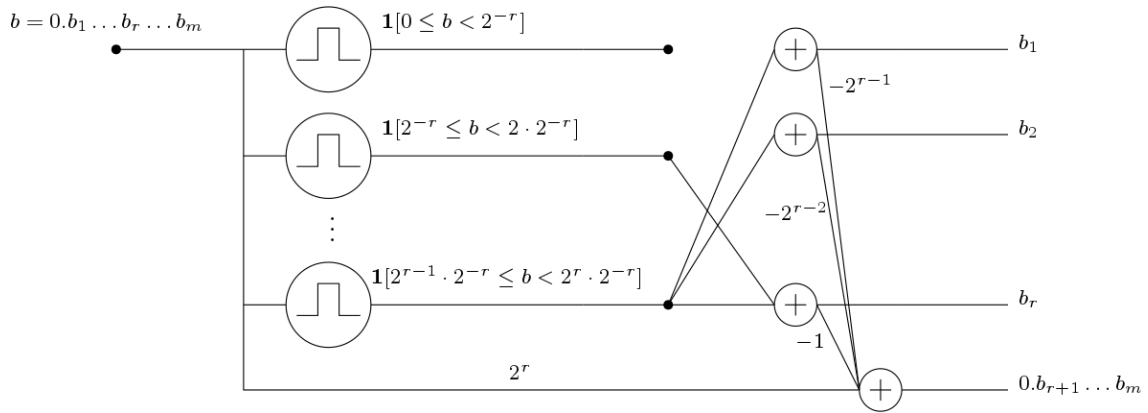

Figure 1: The ReLU network used to extract the most significant r bits of a number. Unlabeled edges indicate a weight of 1 and missing edges indicate a weight of 0.

8

Proof Partition $[0,1)$ into 2^r even subintervals. Observe that the values of $b_1, \ldots, b_r$ are determined by knowing which such subinterval b lies in. We first show how to design a two-layer ReLU network that computes the indicator function for an interval to any arbitrary precision. Using 2^r of these networks in parallel allows us to determine which subinterval b lies in and hence, determine the bits $b_1, \ldots, b_r$.

For any $a \le b$ and $\varepsilon > 0$, observe that the function $f(x) := \sigma(1 - \sigma(a/\varepsilon - x/\varepsilon)) + \sigma(1 - \sigma(x/\varepsilon - b/\varepsilon)) - 1$ has the property that, $f(x) = 1$ for $x \in [a, b]$, and $f(x) = 0$ for $x \notin (a - \varepsilon, b + \varepsilon)$, and $f(x) \in [0, 1]$ for all x. Thus we can use f to approximate the indicator function for $[a, b]$, to any desired precision. Moreover, this function can be computed with 3 layers, 5 units, and 11 parameters as follows. First, computing $\sigma(a/\varepsilon - x/\varepsilon)$ can be done with 1 unit, 1 layer, and 2 parameters. Computing $\sigma(1 - \sigma(a/\varepsilon - x/\varepsilon))$ can be done with 1 additional unit, 1 additional layer, and 2 additional parameters. Similarly, $\sigma(1 - \sigma(x/\varepsilon - b/\varepsilon))$ can be computed with 2 units, the same 2 layers, and 4 parameters. Computing the sum can be done with 1 additional layer, 1 additional unit, and 3 additional parameters. In total, computing f can be done with 3 layers, 5 units, and 11 parameters. We will choose $\varepsilon = 2^{-m-2}$ because we are working with m-digit numbers.

Thus, the values $b_1, \ldots, b_r$ can be generated by adding the corresponding indicator variables. (For instance, $b_1 = \sum_{k=2^{r-1}}^{2^r - 1} \mathbf{1}[b \in [k \cdot 2^{-r}, (k+1) \cdot 2^{-r})]$.) Finally, the remainder $0.b_{r+1}b_{r+2} \ldots b_m$ can be computed as $0.b_{r+1}b_{r+2} \ldots b_m = 2^r b - \sum_{k=1}^{r} 2^{r-k} b_k$.

Now we count the number of layers and parameters: we use 2^r small networks that work in parallel for producing the indicators, each has 3 layers, 5 units and 11 parameters. To produce $b_1, \ldots, b_r$ we need an additional layer, $r \times (2^r + 1)$ additional parameters, and r additional units. For producing the remainder we need 1 more layer, 1 more unit, and $r + 2$ more parameters. ∎

We use $\lceil m/r \rceil$ of these blocks to extract the bits of a_j, denoted by $a_{j,1}, \ldots, a_{j,m}$. Extracting $a_{j,i}$ is now easy, noting that if $x, y \in \{0, 1\}$ then $x \wedge y = \sigma(x + y - 1)$. So, since $x_2 = e_i$, we have

$$a_{j,i} = \sum_{t=1}^{m} x_{2,t} \wedge a_{j,t} = \sum_{t=1}^{m} \sigma(x_{2,t} + a_{j,t} - 1) = \sigma\left(\sum_{t=1}^{m} \sigma(x_{2,t} + a_{j,t} - 1) \right).$$

This calculation needs 2 layers, $1 + m$ units, and $1 + 4m$ parameters.

Remark 14 *Theorem 6 implies an inherent barrier to proving lower bounds using the "bit extraction" approach from Bartlett et al. (1998). Recall that this technique uses n binary numbers with m bits to encode a function $f: S_n \times S_m \to \{0, 1\}$ to show an $\Omega(mn)$ lower bound for VC-dimension, where S_k denotes the set of standard basis vectors in $\Re^k$. The network begins by selecting one of the n binary numbers, and then extracting a particular bit of that number. Bartlett et al. (1998) shows that it is possible to take $m = \Omega(L)$ and $n = \Omega(W)$, thus proving a lower bound of $\Omega(WL)$ for the VC-dimension. In Theorem 3 we showed we can increase m to $\Omega(L \log(W/L))$, improving the lower bound to $\Omega(WL \log(W/L))$. Theorem 6 implies that to extract just the least significant bit, one is forced to have $m = O(L \log(W/L))$; on the other hand, we always have $n \le W$. Hence there is no way to improve the VC-dimension lower bound by more than a constant via the bit extraction*

technique. In particular, for general piecewise polynomial networks, closing the gap between the $O(WL^2 + WL \log W)$ of Bartlett et al. (1998) and $\Omega(WL \log W/L)$ of this paper will require a different technique.

3. Proof of Theorem 6

For a piecewise polynomial function $\Re \rightarrow \Re$, *breakpoints* are the boundaries between the pieces. So if a function has p pieces, it has $p - 1$ breakpoints.

Lemma 15 *Let $f_1, \ldots, f_k : \Re \rightarrow \Re$ be piecewise polynomials of degree D, and suppose the union of their breakpoints has size B. Let $\psi : \Re \rightarrow \Re$ be a piecewise polynomial of degree d with b breakpoints. Let $w_1, \ldots, w_k \in \Re$ be arbitrary. The function $g(x) := \psi(\sum_i w_i f_i(x))$ is piecewise polynomial of degree Dd with at most $(B+1)(2+bD) - 1$ breakpoints.*

Proof Without loss of generality, assume that $w_1 = \cdots = w_k = 1$. The function $\sum_i f_i$ has $B + 1$ pieces. Consider one such interval $\mathcal{I}$. We will prove that it will create at most $2 + bD$ pieces in g. In fact, if $\sum_i f_i$ is constant on $\mathcal{I}$, g will have 1 piece on $\mathcal{I}$. Otherwise, for any point y, the equation $\sum_i f_i(x) = y$ has at most D solutions on $\mathcal{I}$. Let $y_1, \ldots, y_b$ be the breakpoints of ψ. Suppose we move along the curve $(x, \sum_i f_i(x))$ on $\mathcal{I}$. Whenever we hit a point (t, y_i) for some t, one new piece is created in g. So at most bD new pieces are created. In addition, we may have two pieces for the beginning and ending of $\mathcal{I}$. This gives a total of $2 + bD$ pieces per interval, as required. Finally, note that the number of breakpoints is one fewer than the number of pieces. ∎

Theorem 6 follows immediately from the following theorem.

Theorem 16 *Assume there exists a neural network with W parameters and L layers that computes a function $f : \Re \rightarrow \Re$, with the property that $|f(x) - (x \bmod 2)| < 1/2$ for all $x \in \{0, 1, \ldots, 2^m - 1\}$. Also suppose the activation functions are piecewise polynomials of degree at most $d \geq 1$, and have at most $p \geq 1$ pieces. Then we have*

$$m \leq L \log_2(13pd^{(L+1)/2} \cdot W/L).$$

In the special case of piecewise linear functions, this gives $m = O(L \log(W/L))$.

Proof For a node v of the network, let $\gamma(v)$ count the number of directed paths from the input node to v. Applying Lemma 15 iteratively gives that for a node v at layer $i \geq 1$, the number of breakpoints is bounded by $(6p)^i d^{i(i-1)/2} \gamma(v) - 1$. Let o denote the output node. Hence, o has at most $(6p)^L d^{L(L-1)/2} \gamma(o)$ pieces. The output of node o is piecewise polynomial of degree at most d^L. On the other hand, as we increase x from 0 to $2^m - 1$, the function $x \bmod 2$ flips $2^m - 1$ many times, which implies the output of o becomes equal to $1/2$ at least $2^m - 1$ times, thus we get

$$(6p)^L d^{L(L-1)/2} \gamma(o) \times d^L \geq 2^m - 1. \tag{3}$$

Let us now relate $\gamma(o)$ with W and L. Suppose that, for $i \in [L]$, there are W_i edges between layer i and previous layers. By the AM-GM inequality,

$$\gamma(o) \leq \prod_i (1 + W_i) \leq \left(\sum_i \frac{1 + W_i}{L} \right)^L \leq (2W/L)^L. \tag{4}$$

Combining Eqs. (3) and (4) gives the theorem. ∎

Telgarsky (2016) showed how to construct a function f which satisfies $f(x) = (x \bmod 2)$ for $x \in \{0, 1, \ldots, 2^m - 1\}$ using a neural network with $O(m)$ layers and $O(m)$ parameters. By choosing $m = k^3$, Telgarsky showed that any function g computable by a neural network with $\Theta(k)$ layers and $O(2^k)$ nodes must necessarily have $\|f - g\|_1 > c$ for some constant $c > 0$.

Our theorem above implies a qualitatively similar statement. In particular, if we choose $m = k^{1+\varepsilon}$, then for any function g computable by a neural network with $\Theta(k)$ layers and $O(2^{k^\varepsilon})$ parameters, there must exist $x \in \{0, 1, \ldots, 2^m - 1\}$ such that $|f(x) - g(x)| > 1/2$.

4. Proof of Theorem 7

The proof of this theorem is very similar to the proof of the upper bound for piecewise polynomial networks from Bartlett et al. (1998, Theorem 1) but optimized in a few places. The main technical tool in the proof is a bound on the growth function of a polynomially parametrized function class, due to Goldberg and Jerrum (1995). It uses an argument involving counting the number of connected components of semi-algebraic sets. The form stated here is Bartlett et al. (1998, Lemma 1), which is a slight improvement of a result of Warren (1968) (the proof can be found in Anthony and Bartlett (1999, Theorem 8.3)).

Lemma 17 *Suppose $W \leq M$ and let $P_1, \ldots, P_M$ be polynomials of degree at most D in W variables. Define*

$$K := \left| \left\{ (\mathrm{sgn}(P_1(a)), \ldots, \mathrm{sgn}(P_M(a))) : a \in \Re^W \right\} \right|,$$

i.e., K is the number of possible sign vectors attained by the polynomials. Then we have $K \leq 2(2eMD/W)^W$.

Proof [of Theorem 7]. For input $x \in \mathcal{X}$ and parameter vector $a \in \Re^W$, let $f(x, a)$ denote the output of the network. Then $\mathcal{F}$ is simply the class of functions $\{x \mapsto f(x, a) : a \in \Re^W\}$.

Fix $x_1, x_2, \ldots, x_m$ in $\mathcal{X}$. We view the parameters of the network, denoted a, as a collection of W real variables. We wish to bound

$$K := \left| \left\{ (\mathrm{sgn}(f(x_1, a)), \ldots, \mathrm{sgn}(f(x_m, a))) : a \in \Re^W \right\} \right|.$$

In other words, K is the number of sign patterns that the neural network can output for the sequence of inputs $(x_1, \ldots, x_m)$. We will prove geometric upper bounds for K, which will imply upper bounds for $\Pi_{\mathrm{sgn}(\mathcal{F})}(m)$.

For any partition $\mathcal{S} = \{P_1, P_2, \ldots, P_N\}$ of the parameter domain $\Re^W$, clearly we have

$$K \leq \sum_{i=1}^{N} |\{(\mathrm{sgn}(f(x_1, a)), \ldots, \mathrm{sgn}(f(x_m, a))) : a \in P_i\}|. \qquad (5)$$

We choose the partition in such a way that within each region P_i, the functions $f(x_j, \cdot)$ are all fixed polynomials of bounded degree, so that each term in this sum can be bounded via Lemma 17.

The partition is constructed iteratively layer by layer, through a sequence $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{L-1}$ of successive refinements, with the following properties:

1. We have $|\mathcal{S}_0| = 1$ and, for each $n \in [L-1]$,

$$\frac{|\mathcal{S}_n|}{|\mathcal{S}_{n-1}|} \leq 2 \left(\frac{2emk_np(1 + (n-1)d^{n-1})}{W_n} \right)^{W_n}. \qquad (6)$$

2. For each $n \in \{1, \ldots, L\}$, each element S of $\mathcal{S}_{n-1}$, each $j \in [m]$, and each unit u in the nth layer, when a varies in S, the net input to u is a fixed polynomial function in W_n variables of a, of total degree no more than $1 + (n-1)d^{n-1}$ (this polynomial may depend on S, j and u).

We may define $\mathcal{S}_0 = \{\Re^W\}$, which satisfies property 2 above, since the input to any node in layer 1 is of the form $w^T x_j + b$, which is an affine function of w, b.

Now suppose that $\mathcal{S}_0, \ldots, \mathcal{S}_{n-1}$ have been defined, and we want to define $\mathcal{S}_n$. For any $h \in [k_n], j \in [m]$, and $S \in \mathcal{S}_{n-1}$, let $p_{h,x_j,S}(a)$ denote the function describing the net input of the hth unit in the nth layer, in response to x_j, when $a \in S$. By the induction hypothesis this is a polynomial with total degree no more than $1 + (n-1)d^{n-1}$, and depends on at most W_n many variables.

Let $\{t_1, \ldots, t_p\}$ denote the set of breakpoints of the activation function. For any fixed $S \in \mathcal{S}_{n-1}$, by Lemma 17, the collection of polynomials

$$\{p_{h,x_j,S}(a) - t_i : h \in [k_n], j \in [m], i \in [p]\}$$

attains at most

$$\Pi := 2(2e(k_nmp)(1 + (n-1)d^{n-1})/W_n)^{W_n}$$

distinct sign patterns when $a \in \Re^W$. Thus, one can partition $\Re^W$ into this many regions, such that all these polynomials have the same signs within each region. We intersect all these regions with S to obtain a partition of S into at most Π subregions. Performing this for all $S \in \mathcal{S}_{n-1}$ gives our desired partition $\mathcal{S}_n$. Thus, the required property 1 (inequality (6)) is clearly satisfied.

Fix some $S' \in \mathcal{S}_n$. Notice that, when a varies in S', all the polynomials

$$\{p_{h,x_j,S}(a) - t_i : h \in [k_n], j \in [m], i \in [p]\}$$

have the same sign, hence the *input* of each nth layer unit lies between two breakpoints of the activation function, hence the *output* of each nth layer unit in response to an x_j

is a fixed polynomial in W_n variables of degree no more than $d(1 + (n-1)d^{n-1}) \le nd^n$. This implies that the *input* of every $(n+1)$th layer unit in response to an x_j is a fixed polynomial function of W_{n+1} variables of degree no more than $1 + nd^n$. (When $d = 0$, this affine function depends only on the W_{n+1} parameters in layer $n+1$; for $d > 0$, it is a polynomial function of all parameters up to layer $n+1$.)

Proceeding in this way we obtain a partition $\mathcal{S}_{L-1}$ of $\Re^W$ such that for each $S \in \mathcal{S}_{L-1}$, the network output in response to any x_j is a fixed polynomial of a of degree no more than $1 + (L-1)d^{L-1}$ (recall that the last node just outputs its input), and hence by Lemma 17 again,

$$|\{(\operatorname{sgn}(f(x_1, a)), \ldots, \operatorname{sgn}(f(x_m, a))) : a \in S\}| \le 2 \left(\frac{2em(1 + (L-1)d^{L-1})}{W_L} \right)^{W_L}.$$

On the other hand, applying (6) iteratively gives

$$|\mathcal{S}_{L-1}| \le \prod_{i=1}^{L-1} 2 \left(\frac{2emk_i p(1 + (i-1)d^{i-1})}{W_i} \right)^{W_i},$$

and thus using (5), and since the points $x_1, \ldots, x_m$ were chosen arbitrarily, we obtain

$$\Pi_{\operatorname{sgn}(\mathcal{F})}(m) \le \prod_{i=1}^{L} 2 \left(\frac{2emk_i p(1 + (i-1)d^{i-1})}{W_i} \right)^{W_i}$$

$$\le 2^L \left(\frac{2emp \sum k_i(1 + (i-1)d^{i-1})}{\sum W_i} \right)^{\sum W_i} \qquad \text{(weighted AM-GM)}$$

$$= 2^L \left(\frac{2empR}{\sum W_i} \right)^{\sum W_i} \qquad \text{(definition of } R \text{ in (1))} \qquad (7)$$

For the bound on the VC-dimension, from the definition of VC-dimension we find

$$2^{\operatorname{VCdim}(\mathcal{F})} = \Pi_{\operatorname{sgn}(\mathcal{F})}(\operatorname{VCdim}(\mathcal{F})) \le 2^L \left(\frac{2epR \cdot \operatorname{VCdim}(\mathcal{F})}{\sum W_i} \right)^{\sum W_i}$$

Notice that $U > 2$ implies $2eR \ge 16$, hence Lemma 18 below gives

$$\operatorname{VCdim}(\mathcal{F}) \le L + \left(\sum W_i \right) \log_2(4epR \log_2(2epR)) = O(\bar{L}W \log(pU) + \bar{L}LW \log d),$$

completing the proof. ∎

Lemma 18 *Suppose that $2^m \le 2^t(mr/w)^w$ for some $r \ge 16$ and $m \ge w \ge t \ge 0$. Then, $m \le t + w \log_2(2r \log_2 r)$.*

Proof We would like to show that $2^x > 2^t(xr/w)^w$ for all $x > t + w \log_2(2r \log_2 r) =: m$. Let $f(x) := x - t - w \log_2(xr/w)$. To show that $f(x) > 0$ for all $x > m$, we need only show that $f(m) \ge 0$ and $f'(x) > 0$ for all $x \ge m$. First, $f(m) \ge 0$ if and only if

$$w \log_2(2r \log_2 r) - w \log_2(mr/w) \ge 0,$$

13

if and only if

$$(2r \log_2 r) - (mr/w) \geq 0,$$

if and only if

$$2 \log_2 r - (t + w \log_2(2r \log_2 r))/w \geq 0,$$

if and only if

$$2 \log_2 r - t/w - \log_2(2r \log_2 r) \geq 0,$$

if and only if

$$r^2/2 \log_2 r \geq 2^{t/w},$$

which holds since $r \geq 16$ and $t/w \leq 1$. Finally, for $x \geq m$, we have $f'(x) \geq 0$ if and only if

$$1 - w/(x \ln(2)) \geq 0$$

if and only if

$$x \geq w/\ln 2,$$

which holds since $r \geq 16$ implies $x \geq m \geq w \log_2(2r \log_2 r) > w/\ln 2$. ∎

Proof [of Remark 9]. Note that $\sum k_i \leq \sum W_i$ implies $L \leq \sum W_i$, hence from (7) we have

$$\Pi_{\mathrm{sgn}(\mathcal{F})}(m) \leq 2^L \left(\frac{2empR}{\sum W_i} \right)^{\sum W_i}$$

$$\leq \left(\frac{4emp(1 + (L-1)d^{L-1}) \sum k_i}{\sum W_i} \right)^{\sum W_i} \qquad (L \leq \sum W_i)$$

$$\leq \left(4emp(1 + (L-1)d^{L-1}) \right)^{\sum W_i} \qquad \left(\sum k_i \leq \sum W_i \right).$$

∎

5. Proof of Theorem 10

The idea of the proof is that the sign of the output of a neural network can be expressed as a Boolean formula where each predicate is a polynomial inequality. For example, consider the following toy network, where the activation function of the hidden units is a ReLU.
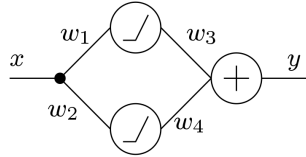
The sign of the output of the network is $\mathrm{sgn}(y) = \mathrm{sgn}(w_3\sigma(w_1x) + w_4\sigma(w_2x))$. Define the following Boolean predicates: $p_1 = (w_1x > 0)$, $p_2 = (w_2x > 0)$, $q_1 = (w_3w_1x > 0)$, $q_2 = (w_4w_2x > 0)$, and $q_3 = (w_3w_1x + w_4w_2x > 0)$. Then, we can write

$$\mathrm{sgn}(y) = (\neg p_1 \wedge \neg p_2 \wedge 0) \vee (p_1 \wedge \neg p_2 \wedge q_1) \vee (\neg p_1 \wedge p_2 \wedge q_2) \vee (p_1 \wedge p_2 \wedge q_3).$$

A theorem of Goldberg and Jerrum states that any class of functions that can be expressed using a relatively small number of distinct polynomial inequalities has small VC-dimension.

Theorem 19 (Theorem 2.2 of Goldberg and Jerrum (1995)) *Let k, n be positive integers and $f \colon \Re^n \times \Re^k \to \{0, 1\}$ be a function that can be expressed as a Boolean formula containing s distinct atomic predicates where each atomic predicate is a polynomial inequality or equality in $k + n$ variables of degree at most d. Let $\mathcal{F} = \{f(\cdot, w) : w \in \Re^k\}$. Then $\text{VCDim}(\mathcal{F}) \leq 2k \log_2(8eds)$.*

Proof [of Theorem 10]. Consider a neural network with W weights and U computation units, and assume that the activation function ψ is piecewise polynomial of degree at most d with p pieces. To apply Theorem 19, we will express the sign of the output of the network as a Boolean function consisting of less than $2(1 + p)^U$ atomic predicates, each being a polynomial inequality of degree at most $\max\{U + 1, 2d^U\}$.

Since the neural network graph is acyclic, it can be topologically sorted. For $i \in [U]$, let u_i denote the ith computation unit in the topological ordering. The input to each computation unit u lies in one of the p pieces of ψ. For $i \in [U]$ and $j \in [p]$, we say "u_i is in state j" if the input to u_i lies in the jth piece.

For u_1 and any j, the predicate "u_1 is in state j" is a single atomic predicate which is the quadratic inequality indicating whether its input lies in the corresponding interval. So, the state of u_1 can be expressed as a function of p atomic predicates. Conditioned on u_1 being in a certain state, the state of u_2 can be determined using p atomic predicates, which are polynomial inequalities of degree at most $2d + 1$. Consequently, the state of u_2 can be determined using $p + p^2$ atomic predicates, each of which is a polynomial of degree at most $2d + 1$. Continuing similarly, we obtain that for each i, the state of u_i can be determined using $p(1 + p)^{i-1}$ atomic predicates, each of which is a polynomial of degree at most $d^{i-1} + \sum_{j=0}^{i-1} d^j$. Consequently, the state of all nodes can be determined using less than $(1 + p)^U$ atomic predicates, each of which is a polynomial of degree at most $d^{U-1} + \sum_{j=0}^{U-1} d^j \leq \max\{U + 1, 2d^U\}$ (the output unit is linear). Conditioned on all nodes being in certain states, the sign of the output can be determined using one more atomic predicate, which is a polynomial inequality of degree at most $\max\{U + 1, 2d^U\}$.

In total, we have less than $2(1 + p)^U$ atomic polynomial-inequality predicates and each polynomial has degree at most $\max\{U + 1, 2d^U\}$. Thus, by Theorem 19, we get an upper bound of $2W \log_2(16e \cdot \max\{U + 1, 2d^U\} \cdot (1 + p)^U) = O(WU \log((1 + d)p))$ for the VC-dimension. ∎

Acknowledgments

for Mathematical and Statistical Frontiers (ACEMS). Part of this work was done while Peter Bartlett and Abbas Mehrabian were visiting the Simons Institute for the Theory of Computing at UC Berkeley.

References

Martin Anthony and Peter Bartlett. *Neural network learning: theoretical foundations*. Cambridge University Press, 1999.

Peter Bartlett, Vitaly Maiorov, and Ron Meir. Almost linear VC-dimension bounds for piecewise polynomial networks. *Neural Computation*, 10(8):2159–2173, Nov 1998.

Eric B. Baum and David Haussler. What size net gives valid generalization? *Neural Computation*, 1(1):151–160, 1989.

Anselm Blumer, A. Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 36(4):929–965, October 1989. ISSN 0004-5411. doi: 10.1145/76359.76371. URL http://doi.acm.org/10.1145/76359.76371. (Conference version in STOC'86).

Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 698–728, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR. URL http://proceedings.mlr.press/v49/cohen16.html.

Thomas M. Cover. Capacity problems for linear machines. In L. Kanal, editor, *Pattern Recognition*, pages 283–289. Thompson Book Co., 1968.

Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 907–940, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR. URL http://proceedings.mlr.press/v49/eldan16.html.

Paul W. Goldberg and Mark R. Jerrum. Bounding the Vapnik-Chervonenkis dimension of concept classes parameterized by real numbers. *Machine Learning*, 18(2):131–148, 1995. (Conference version in COLT'93).

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Philipp Grohs, Dmytro Perekrestenko, Dennis Elbrächter, and Helmut Bölcskei. Deep neural network approximation theory. *arXiv preprint arXiv:1901.02220*, 2019.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

Daniel Jakubovitz, Raja Giryes, and Miguel RD Rodrigues. Generalization error in deep learning. *arXiv preprint arXiv:1808.01174*, 2018.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Shyu Liang and R. Srikant. Why deep neural networks for function approximation?, 2017. arXiv:1610.04161.

Wolfgang Maass. Neural nets with superlinear VC-dimension. *Neural Computation*, 6(5): 877–884, Sept 1994.

David Pollard. *Empirical Processes: Theory and Applications*, volume 2. Institute of Mathematical Statistics, 1990.

Itay Safran and Ohad Shamir. Depth-width tradeoffs in approximating natural functions with neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2979–2987, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL `http://proceedings.mlr.press/v70/safran17a.html`.

Eduardo D. Sontag. Feedback stabilization using two-hidden-layer nets. *IEEE Transactions on Neural Networks*, 3(6):981–990, 1992.

Matus Telgarsky. benefits of depth in neural networks. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 1517–1539, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR. URL `http://proceedings.mlr.press/v49/telgarsky16.html`.

V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971. doi: 10.1137/1116025.

Hugh E. Warren. Lower bounds for approximation by nonlinear manifolds. *Transactions of the American Mathematical Society*, 133(1):167–178, 1968.

Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017.