

Depth Separations in Neural Networks: Separating the Dimension from the Accuracy

Itay Safran

Ben-Gurion University of the Negev

SAFRANI@BGU.AC.IL

Daniel Reichman

Worcester Polytechnic Institute

DREICHMAN@WPI.EDU

Paul Valiant

Purdue University

PVALIANT@PURDUE.EDU

Editors: Nika Haghtalab and Ankur Moitra

Abstract

We prove an exponential separation between depth 2 and depth 3 neural networks, when approximating a $\mathcal{O}(1)$ -Lipschitz target function to constant accuracy, with respect to a distribution with support in the unit ball, under the mild assumption that the weights of the depth 2 network are exponentially bounded. This resolves an open problem posed in [Safran et al. \(2019\)](#), and proves that the curse of dimensionality manifests itself in depth 2 approximation, even in cases where the target function can be represented efficiently using a depth 3 network. Previously, lower bounds that were used to separate depth 2 from depth 3 networks required that at least one of the Lipschitz constant, target accuracy or (some measure of) the size of the domain of approximation scale *polynomially* with the input dimension, whereas in our result these parameters are fixed to be *constants* independent of the input dimension: our parameters are simultaneously optimal. Our lower bound holds for a wide variety of activation functions, and is based on a novel application of a worst- to average-case random self-reducibility argument, allowing us to leverage depth 2 threshold circuits lower bounds in a new domain.

Keywords: Neural networks, Depth separations, Lower bounds, Function approximation.

1. Introduction

There is significant empirical evidence suggesting that depth plays a crucial role in the practical success of deep learning ([LeCun et al., 2015](#); [He et al., 2016](#)). From a purely theoretical perspective, while depth 2 neural networks are known to be universal approximators for continuous functions over compact domains ([Cybenko, 1989](#); [Hornik et al., 1989](#); [Leshno et al., 1993](#)), it is now well-established that greater depth may be necessary to obtain a compact representation of certain target functions ([Eldan and Shamir, 2016](#); [Telgarsky, 2016](#); [Poggio et al., 2017](#); [Daniely, 2017](#); [Yarotsky, 2017](#); [Liang and Srikant, 2016](#); [Safran and Shamir, 2017](#); [Safran et al., 2019](#); [Venturi et al., 2022](#); [Safran and Lee, 2022](#); [Nichani et al., 2023](#); [Safran et al., 2024](#)). Perhaps the most stark such examples are when separating depth 2 from depth 3 – following the seminal work of [Eldan and Shamir \(2016\)](#), many works have demonstrated *depth separations* where depth can be exponentially more beneficial than width, even when increased by just one: There exists a function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ that can be approximated to accuracy $\varepsilon > 0$ using a network of depth 3 and width $\text{poly}(d, 1/\varepsilon)$, yet any depth 2 network which approximates f to the same accuracy would require exponentially many more neurons (in d or $1/\varepsilon$).

Despite the growing number of situations where such separations can be shown, known results often involve contrived settings with complicated distributions and oscillatory target functions. On the other hand, functions arising in machine learning settings are often smooth, with bounded fluctuations. Moreover, it was recently shown that in some examples constructed to show depth separations with neural networks, the same property which prevents approximation using the shallow architecture, may also prove detrimental for *learning* the target function, even when using a network which is sufficiently deep to express the target function efficiently (Malach and Shalev-Shwartz, 2019; Malach et al., 2021). In light of this, several recent works have shifted their focus to devising depth separation results for functions that are arguably more natural (Safran and Shamir, 2017; Safran et al., 2019; Safran and Lee, 2022; Ren et al., 2023; Nichani et al., 2023; Safran et al., 2024). The rationale is that such results could be better aligned with learning problems arising in applications, rather than contrived examples that may not be representative of broader classes of functions.

In Safran et al. (2019), the authors observe: given a width lower bound for a depth two neural network approximating a target function f , that we can trade off the Lipschitz constant of f and the target accuracy parameters by rescaling f by a multiplicative factor. Moreover, one can also strengthen the accuracy lower bound, at the cost of dilating the domain of approximation, by using a change of variables (see (Safran et al., 2019, Theorem 9) for a precise statement). This observation suggests, that if either one of these parameters scales with the input dimension d (as is the case in all known separation results), then it is not possible to pinpoint whether the true cause of the difficulty in the approximation stems from the input dimension itself, or from the remaining parameters that were forced to scale with it. In other words, it is not clear if known approximation lower bounds for depth 2 networks are a manifestation of the curse of dimensionality, or if the difficulty lies somewhere else. To study the root cause for the inability of polynomially sized depth 2 networks to approximate certain functions, the authors pose the following question:

Can we show a superpolynomial depth 2 vs. depth 3 neural network separation result in terms of the dimension d , for approximating $\mathcal{O}(1)$ -Lipschitz functions up to constant accuracy ε , on a domain of bounded radius (all independent of d)? (Safran et al., 2019)

Their main result, is that in contrast to what previous separation results suggested, if one considers *radial* target functions that are commonly used to show such depth separations, then an exponential dependence on d is not possible. This is demonstrated by providing a general approximation result wherein width $\text{poly}(d)$ suffices.

Motivated by the above question, in this work, we prove a separation result between depth 2 and depth 3, where the target function is $\mathcal{O}(1)$ -Lipschitz, the domain of approximation is contained inside the unit ball, and the separation is exponential even if the target accuracy is an absolute constant. (See Table 1 for a comparison.) Stated somewhat informally, the following is our main result in this paper:

Theorem 1 (Informal version of Thm. 2) *There exists a sequence of distributions $\{\mathcal{D}_{4d}\}_{d=1}^{\infty}$ supported in the $4d$ -dimensional Euclidean ball, and a sequence of $\mathcal{O}(1)$ -Lipschitz functions $f_d : \mathbb{R}^{4d} \rightarrow \mathbb{R}$, such that, using any $\mathcal{O}(1)$ -Lipschitz or threshold activation function σ for the neural networks, we have the following:*

- f_d is approximable to arbitrary accuracy ε (in the L_{∞} sense) by a depth 3 neural network \mathcal{N}'_d with size and weights bounded by $\text{poly}(d, 1/\varepsilon)$.

Table 1: Several known lower bounds for approximation using depth 2 neural networks, where the target function is scaled to be $O(1)$ -Lipschitz, and for the sake of comparison, the distribution is scaled to either contain the unit ball (for compactly supported distributions), or normalized to have most of its probability mass contained within the unit ball (non-compactly supported distributions). An asterisk * denotes that the result in [Daniely \(2017\)](#) is not for a radial function, but can be reduced to one taking the form $\mathbf{x} \mapsto \frac{1}{2\pi d^3} \sin(2\pi d^3 \|\mathbf{x}\|^2)$, which oscillates in any direction from the origin (see ([Safran et al., 2019](#); [Vardi and Shamir, 2020](#))). In contrast, our result is non-oscillatory in all such directions, but it can be seen rather as an embedding of a Boolean function into the unit ball. [Safran et al. \(2019\)](#) use a reduction to the main result of [Eldan and Shamir \(2016\)](#) to show a lower bound for a non-oscillatory function, but the cost of removing this oscillation is that the accuracy gets worse by a factor of d^{-2} . To the best of our knowledge, our lower bound, which is highlighted in bold, is the only constant accuracy lower bound in this setting. Since in many machine learning applications the input dimension is quite large, we have that lower bounds that scale as $1/\text{poly}(d)$ are too permissive, in contrast to our constant lower bound.

Reference	Distribution	Function	Accuracy
Eldan and Shamir (2016)	Radial, heavy-tailed	Radial, oscillatory	$\Omega(d^{-5})$
Daniely (2017)	Radial*, compactly supported	Radial*, oscillatory	$\Omega(d^{-3})$
Safran et al. (2019)	Radial, heavy-tailed,	Radial, non-oscillatory	$\Omega(d^{-7})$
Venturi et al. (2022)	Product, heavy-tailed	Product, oscillatory	$\Omega(d^{-5})$
This paper	Compactly supported	Non-oscillatory	$\Omega(1)$

- f_d is inapproximable to accuracy better than 0.05 in the L_2 sense by any depth 2 neural network \mathcal{N}_d , unless \mathcal{N}_d has either width or weights exponential in the dimension d .

This result resolves the main open question of [Safran et al. \(2019\)](#), showing that depth 3 neural networks can be essentially exponentially more efficient than depth 2 neural networks, even for the mildest and most representative setting: where the target function f is $O(1)$ -Lipschitz, on a domain of radius 1, and for any commonly used activation function.

Our proof technique is quite different than those commonly used in the literature. While it is common to use some functional analysis tool such as Fourier spectrum analysis ([Eldan and Shamir, 2016](#)) or spherical harmonics ([Daniely, 2017](#)), our lower bound relies on a reduction to threshold circuits (with Boolean inputs), where an exact computation lower bound for the IP mod 2 function is used [Hajnal et al. \(1993\)](#). The key ingredient in our proof is a reduction from a worst-case complexity, where we are able to construct a network which is capable of classifying *all* the inputs correctly; to an average-case complexity, where intuitively, only a constant portion of the inputs are classified correctly. This construction utilizes the randomization of the input in a manner which preserves its output value, yet induces sufficient randomness so as to result in a concentration of measure which provides a worst-case guarantee. This is achieved by a careful analysis of properties of the probability distribution induced by our randomized self-reduction, and much of the proof of the lower bound is dedicated to this part.

We stress that lower bounds for approximate computation of continuous functions with real inputs, using depth 2 neural networks, do not follow in any straightforward way from lower bounds for depth 2 threshold circuits that *exactly* compute a Boolean function over the Boolean hypercube. Indeed it is explicitly stated in [Eldan and Shamir \(2016\)](#) that “lower bounds for Boolean circuits do not readily translate to neural networks of the type used in practice, which are real-valued and express continuous functions”. Furthermore, our paper focuses on domains of constant diameter

such as the unit ball, and this restriction rules out various techniques/constructions that are available for domains of non-constant radius, including the unit hypercube $[0, 1]^d$.

We point out that our separation holds for a wide family of activation functions that includes commonly used functions like the ReLU, threshold and sigmoidal activations. Similarly to other lower bounds in the literature that are shown with respect to compactly supported distributions (see related work section below), our result assumes a mild exponential upper bound on the magnitude of the weights of the approximating network. In contrast, lower bounds that do not impose any restrictions on the magnitude of the weights, rely on the technique in [Eldan and Shamir \(2016\)](#), and use heavy-tailed data distributions. To the best of our knowledge, it remains a major open problem to show a superpolynomial separation result between depth 2 and depth 3, with respect to a distribution with bounded support, and when allowing unbounded weights.

The remainder of this paper is structured as follows: After presenting our contributions in this paper in more detail below, we discuss related work in the literature. In [Sec. 2](#) we present the notation used throughout this paper, followed by a formal construction of our separation setting, the set of assumptions used in our results, and the main result in this paper. [Sec. 3](#) details our depth 2 lower bound, as well as provides a sketch of the proof. [Sec. 4](#) details our depth 3 positive approximation result, as well as provides a concrete example of the construction for the ReLU activation function.

Our contributions

- We prove that under mild assumptions on the activation function σ ([Assumption 1](#)), a depth 2 neural network with σ activations cannot approximate a certain $\mathcal{O}(1)$ -Lipschitz function f_d well (see [Eq. \(1\)](#)). It is shown that there exists a simple distribution supported on the Euclidean unit ball, such that f_d cannot be approximated to better than constant accuracy with respect to this distribution, unless the width of the network or the coefficient sizes scale as $\exp(\Omega(d))$ ([Thm. 3](#)).
- As an intermediate step in the derivation of our previous result, using an embedding technique, we show a general reduction from Boolean functions that are hard to approximate on the Boolean hypercube, to a function which is hard to approximate on the unit ball ([Thm. 10](#)). This result relies on a Rademacher complexity argument to show an approximation lower bound, and may be of independent interest.
- We show that under related mild assumptions on the activation function σ ([Assumption 2](#)), a depth 3 neural network with σ activations can approximate the same function f_d to arbitrary accuracy in an L_∞ sense, using width that scales polynomially with the target accuracy and d ([Thm. 4](#)).
- Combining our lower and upper bounds, we obtain our main result ([Thm. 2](#)), which demonstrates the manifestation of the curse of dimensionality in depth 2 approximation, even if the target function is easy to approximate using depth 3. This is in contrast with previously known results that only imply an exponential lower bound in a high-accuracy regime where the target accuracy scales as $1/\text{poly}(d)$.

Related work

Separating depth 2 from depth 3, continuous, L_2 lower bound setting. For concreteness, let us focus here on the setting of separating depth 2 from depth 3 in a continuous, L_2 lower bound setting. The seminal work of [Eldan and Shamir \(2016\)](#) provided the first exponential L_2 lower bound for depth 2 neural networks. The authors construct an approximately radial, oscillatory target function, and use a Fourier spectrum analysis argument to prove their unconstrained lower bound (imposing no bounds on the magnitude of the weights). As an artifact of this technique, a superposition argument is required to guarantee that the target function cannot be approximated with fewer than exponentially many neurons. This, however, also forces the Lipschitz constant and the number of oscillations to scale as $\text{poly}(d)$. [Venturi et al. \(2022\)](#) adapt the technique of Fourier spectrum analysis to a setting with product target functions and distributions, rather than radial ones. Due to this technique, both works ([Eldan and Shamir, 2016](#); [Venturi et al., 2022](#)) require the target accuracy to scale as $\Omega(d^{-5})$ for the curse of dimensionality to manifest. [Safran et al. \(2024\)](#) also use Fourier spectrum analysis to show a depth separation result between depth 2 and depth 3, for approximating the non-oscillatory 1-Lipschitz maximum function on the domain $[0, 1]^d$, with respect to the uniform distribution. However, their separation is only polynomial (showing that the size of a depth 2 network approximating the target function is polynomially larger than a depth 3 network approximating the function), and for accuracy which scales as $1/\text{poly}(d)$ rather than a constant.

[Daniely \(2017\)](#) used a simple and elegant harmonic analysis technique, to show a depth 2 vs. depth 3 separation, assuming the weights of the depth 2 network are at most exponential in the input dimension, for oscillatory target functions, over a product distribution on two spheres (that can be reduced to a radial distribution—see ([Safran et al., 2019](#); [Vardi and Shamir, 2020](#)) for more details). With some careful analysis, it can be shown that with this technique, the approximation error lower bound on a $\mathcal{O}(1)$ -Lipschitz function in a domain of constant radius is $\Omega(d^{-3})$. The works ([Safran and Lee, 2022](#); [Nichani et al., 2023](#)) prove depth 2 approximation lower bounds for non-oscillatory target functions, by using the main result in [Daniely \(2017\)](#), and decoupling the dependence of the input dimension from the accuracy in the lower bound. However, both results require the accuracy to scale as d^{-3} for the curse of dimensionality to manifest, a property inherited by the proof technique being used.

[Safran and Shamir \(2017\)](#) and [Safran et al. \(2019\)](#) use reductions to the main result of [Eldan and Shamir \(2016\)](#) to derive exponential depth 2 lower bounds for non-oscillatory functions. [Safran and Shamir \(2017\)](#) show this for non-Lipschitz ellipsoid indicator functions, and [Safran et al. \(2019\)](#) show this for a 1-Lipschitz radial function. Like the previous results using Fourier spectrum analysis, these results require accuracy scaling as $\Omega(d^{-5})$ to have exponential dependence on the input dimension, but additionally, simplifying the function to be non-oscillatory results in an additional d^{-2} factor, for an overall accuracy lower bound of $\Omega(d^{-7})$. As pointed out above, rescaling these results lets us interpret this as a constant approximation accuracy result, but for functions that are d^7 -Lipschitz. In contrast, our exponential lower bound is simultaneously for a non-oscillatory $\mathcal{O}(1)$ -Lipschitz function, for constant approximation accuracy.

Connection between neural networks and threshold circuits. The connection between approximation lower bounds when using neural networks and threshold circuits has been studied in multiple works recently. [Martens et al. \(2013\)](#) study lower bounds for depth 2 neural networks with non-threshold activation functions, using known threshold circuit lower bounds on the inner prod-

uct function. Their work differs from ours in a few ways: (i) theirs is a much more direct application of circuit lower bounds to bound *exact* computation of Boolean functions by neural networks, while we show a random self-reduction to bound the hardness of *average-case* approximation; and (ii) we focus on approximation on continuous domains rather than the discrete uniform distribution over the Boolean hypercube; (iii) their setting of “restricted Boltzmann machines” is different from our neural network setting. Mukherjee and Basu (2017) derive sub-linear size lower bounds for neural networks by showing reductions to known threshold circuit lower bounds. Vardi and Shamir (2020) show barriers for achieving depth separations in neural networks by using reductions to open problems in threshold circuits, and applying known “natural proof barrier” results, showing that such separations would solve long-standing open problem that are widely believed to be very difficult. Since their results only apply to networks of depth 4 or more, they do not apply in the setting investigated in this paper, which to the best of our knowledge does not solve any open problem in circuit complexity. Vardi et al. (2021) use communication complexity to derive size lower bounds for ReLU networks which compute $IP \bmod 2$. While this lower bound applies regardless of the depth of the circuit (similarly to the linear lower bound for $IP \bmod 2$ for threshold circuits), it is for a network size which is linear (up to logarithmic factors) in the input dimension, whereas we use a similar distribution, but in order to show an exponential separation between depth 2 and depth 3.

Progress on the open question posed in Safran et al. (2019). To the best of our knowledge, the only works that made progress with the open question posed in Safran et al. (2019), are (Safran et al., 2019; Hsu et al., 2021). Safran et al. (2019) observe that the target accuracy, Lipschitz parameter of the target function, and the radius of the domain of approximation, can all be traded off with polynomial factors. As one of their main contributions, the authors show that when all three parameters are held constant, then an exponential separation between depth 2 and depth 3 is not possible for radial functions, by proving a positive L_∞ approximation result in this setting where width polynomial in d suffices. This indicates that in order to resolve the question, one must consider non-radial target functions. Hsu et al. (2021) consider the question of how many randomly initialized ReLU neurons are required for the approximation of arbitrary functions with a constant Lipschitz parameter, with respect to the *uniform* distribution over $[0, 1]^d$, and with high probability. Their main positive approximation result is that perhaps surprisingly, if the target accuracy is fixed, then a depth 2, width $\text{poly}(d)$ random ReLU network will approximate any $\mathcal{O}(1)$ -Lipschitz function with high probability (hence – there exists a deterministic network with this approximation). However, since this result is with respect to a uniform L_2 approximation rather than an L_∞ approximation, it does not rule out results like ours that are lower bounds over *nonuniform* distributions. Hsu et al. (2021) does strongly suggest, however, that results like ours can only be possible for distributions that are far in some sense from the uniform distribution. Indeed, our lower bound is for a distribution with support in the unit ball that has most of its probability mass concentrated close to a discrete set of exponentially many points.

2. Setting and Main Result

In this section, we formally define our setting and present the notation and terminology used throughout the paper, before turning to present our assumptions and main result.

2.1. Preliminaries and notation

Notation and terminology. We let $[n]$ be shorthand for the set $\{1, \dots, n\}$. We denote vectors using bold-faced letters (e.g. \mathbf{x}) and matrices or random variables using upper-case letters (e.g. X). Given a vector $\mathbf{x} = (x_1, \dots, x_d) \in \mathbb{R}^d$, we define $\text{round}(\mathbf{x}) = (\text{round}(x_1), \dots, \text{round}(x_d))$, where $\text{round}(x)$ rounds x to the nearest integer. We let $\mathcal{U}(A)$ denote the uniform distribution on a set $A \subseteq \mathbb{R}^d$. We define the Boolean functions $\text{AND} : \{0, 1\}^2 \rightarrow \{0, 1\}$, $\text{AND}(x, y) = x \cdot y$; and $\text{IP}_d : \{0, 1\}^{2d} \rightarrow \{0, 1\}$, $\text{IP}_d(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle \bmod 2$. For some vector $\mathbf{x}_0 \in \mathbb{R}^d$ and real $r > 0$, we define the d -dimensional ball with radius r centered at \mathbf{x}_0 as $B_r^d(\mathbf{x}_0) := \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x} - \mathbf{x}_0\| \leq r\}$. For two sets $A, B \subseteq \mathbb{R}^d$, their *Minkowski sum* is defined as $A + B := \{\mathbf{a} + \mathbf{b} : \mathbf{a} \in A, \mathbf{b} \in B\}$.

Neural networks and threshold circuits. We consider fully connected, feed-forward neural networks, computing functions from \mathbb{R}^d to \mathbb{R} . A σ -network consists of layers of neurons. In every layer except for the output neuron, an affine function of the inputs is computed, followed by a computation of the non-linear activation function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$. The single output neuron simply computes an affine transformation of its inputs. Each layer with a non-linear activation is called a *hidden layer*, and the *depth* of a network is defined as the number of hidden layers plus one. The *width* of a network is defined as the number of neurons in the largest hidden layer, and the *size* of the network is the total number of neurons across all layers. Analogously, we define a *threshold network* as a σ -network which employs the threshold activation function; namely, where $\sigma(x) = 1$ for all $x \geq 0.5$, and $\sigma(x) = 0$ otherwise. We make the distinction between a threshold network and a *threshold circuit*, where in a threshold circuit, the output neuron also has a non-linear activation rather than a linear activation as is the case for threshold networks.

2.2. Formal construction

We begin with defining the distribution used to show our separation result. In short, this distribution is the uniform distribution over a Cartesian product of two sets. The first is a set which is sufficiently spread out so as to guarantee a constant distance between different components of the distribution (which implies that our function is $\mathcal{O}(1)$ -Lipschitz). The second can be seen as a continuous embedding of the $\text{IP} \bmod 2$ function inside the Euclidean unit ball. More formally, let $\mathbf{z}_1, \dots, \mathbf{z}_{2^{2d}} \in B_{0.8}^{2d}(\mathbf{0})$ be a set of points where each pair is at a distance of at least 0.4 apart (see the full proof for details about its existence), and let $\mathbf{x}_1, \dots, \mathbf{x}_{2^{2d}}$ be an enumeration of the $2d$ -dimensional Boolean hypercube. We define the sets

$$\mathcal{S}_{4d} := \left\{ \left(\mathbf{z}_i, \frac{1}{4\sqrt{d}} \mathbf{x}_i \right) \right\}_{i=1}^{2^{2d}} \quad \text{and} \quad \mathcal{C}_{4d} := \left[0, \frac{1}{12\sqrt{d}} \right]^{4d}.$$

Using these sets, we can define the support of our distribution as $\mathcal{A}_{4d} := \mathcal{S}_{4d} + \mathcal{C}_{4d}$, and our distribution to be the uniform distribution over the set \mathcal{A}_{4d} ; namely, the distribution $\mathcal{U}(\mathcal{A}_{4d})$. It is interesting to note that the last $2d$ coordinates of this distribution can also be seen as a scaled interpolation between the two distributions $\mathcal{U}([0, 1]^d)$ and $\mathcal{U}(\{0, 1\}^d)$, where as discussed in the related work subsection, the former is too spread to show constant accuracy lower bounds for $\mathcal{O}(1)$ -Lipschitz functions, and the latter is a discrete rather than a continuous distribution, which is the setting where neural networks are typically being used. Given an input $(\mathbf{z}, \mathbf{x}, \mathbf{y}) \in \mathbb{R}^{2d} \times \mathbb{R}^d \times \mathbb{R}^d$, we define our hard to approximate function $f_d : B_1^{4d}(\mathbf{0}) \rightarrow \mathbb{R}$ as

$$f_d(\mathbf{z}, \mathbf{x}, \mathbf{y}) := \text{IP}_d \left(\text{round} \left(3\sqrt{d}\mathbf{x} \right), \text{round} \left(3\sqrt{d}\mathbf{y} \right) \right). \quad (1)$$

In words, our function ignores the first $2d$ coordinates, scales the remaining $2d$ coordinates to the unit hypercube, rounds them to the nearest integer, and computes the IP mod 2 function on the resulting input. It is easy to verify that $\mathcal{A}_{4d} \subseteq B_1^{4d}(\mathbf{0})$ and that f_d is constant on each connected component of \mathcal{A}_{4d} . Since every pair of such components is at a constant distance apart, we also have that f_d is $\mathcal{O}(1)$ -Lipschitz. (See Appendix A.1 for further detail.) We note that Kirschbraun's theorem implies that f_d can be easily extended to the entire Euclidean ball, with the same $\mathcal{O}(1)$ Lipschitz constant.

2.3. Assumptions

Before we can present our main result in this paper, we will first formally state and discuss our assumptions. We begin with formally stating our assumption on the family of activation functions for which our lower bound holds.

Assumption 1 (Lower bound) *The activation function σ is either the threshold activation or a $\mathcal{O}(1)$ -Lipschitz function.*

Our above assumption is very mild, and it is satisfied by many commonly used activations such as ReLU, threshold and sigmoidal activations. We remark that our result holds in fact for an even wider class of activation functions, and that for the sake of simplicity, we chose to present it here using a function class that is more intuitive to comprehend, yet still encompasses essentially all activation functions that are interesting from a practical perspective. The interested reader is referred to Assumption 3 in the appendix for the most general form of our lower bound assumption.

Having discussed our lower bound assumption, we now move on to state and discuss our upper bound assumption.

Assumption 2 (Upper bound) *There exists a constant c_σ which depends solely on σ such that the following holds: For all $R > 0$ and any L -Lipschitz function $f : [-R, R] \rightarrow \mathbb{R}$, and for any δ , there exist scalars $a, \{\alpha_i, \beta_i, \gamma_i\}_{i=1}^w$, where $w, |a|, |\alpha_i|, |\beta_i|, |\gamma_i| \leq c_\sigma \frac{RL}{\delta}$ for all $i \in [w]$, such that the function*

$$h(x) = a + \sum_{i=1}^w \alpha_i \cdot \sigma(\beta_i x - \gamma_i)$$

satisfies

$$\sup_{x \in [-R, R]} |f(x) - h(x)| \leq \delta.$$

The above is a slight modification of Assumption 2 in Eldan and Shamir (2016), and is satisfied by many standard activation functions that are used in the literature, which in particular include threshold, ReLU and sigmoidal activations (see Lemma 8 in the appendix which implies that the threshold activation satisfies this property, and see Appendix A in Eldan and Shamir (2016) for a proof for the ReLU activation). We point out that we also require an additional mild requirement in our assumption, that the weights of the approximating network h are bounded in magnitude. This is in order to control the magnitude of the weights in our approximation of f_d , and get a valid separation that requires weights of polynomial magnitude, which stands in contrast to our lower bounds, where polynomially bounded weights imply that width exponential in d is necessary.

2.4. Main result

We are ready to present our main theorem in this paper:

Theorem 2 *Consider the sequence of distributions $\{\mathcal{U}(\mathcal{A}_{4d})\}_{d=1}^\infty$, and the sequence of $\mathcal{O}(1)$ -Lipschitz functions $f_d : \mathbb{R}^{4d} \rightarrow \mathbb{R}$ defined in Eq. (1). Then for all $C > 0$ and sufficiently large d , we have for any activation function σ that satisfies both Assumption 1 and Assumption 2, that the following hold.*

- For any depth 2 σ -network $\mathcal{N}_d : \mathbb{R}^d \rightarrow \mathbb{R}$, with weights bounded in magnitude by C , we have

$$\mathbb{E}_{\mathbf{x} \sim \mathcal{U}(\mathcal{A}_{4d})} \left[(f_d(\mathbf{x}) - \mathcal{N}_d(\mathbf{x}))^2 \right] > \frac{1}{400},$$

unless \mathcal{N}_d has width at least $\Omega \left(\frac{\exp(\Omega(d))}{\text{poly}(C)} \right)$.

- For all $\varepsilon > 0$, there exists a depth 3, width $\text{poly}(d, 1/\varepsilon)$, σ -network \mathcal{N}'_d , such that

$$\sup_{\mathbf{x} \in \mathcal{A}_{4d}} |f_d(\mathbf{x}) - \mathcal{N}'_d(\mathbf{x})| \leq \varepsilon,$$

where the asymptotic notation hides constants that depend solely on σ .

The above theorem is an immediate consequence of our lower bound (Thm. 3) and upper bound (Thm. 4), which will be presented in detail in the following sections.

3. Lower Bound

In this section, we present our lower bound for the approximation of the function f_d . Thereafter, we provide a proof sketch which conveys the main technical ideas behind the result. We begin with formally stating our lower bound as follows.

Theorem 3 *Suppose that σ satisfies Assumption 1, and let $\mathcal{N} : \mathbb{R}^{2d} \rightarrow \mathbb{R}$ be a depth 2 σ -network with weights bounded by C that satisfies*

$$\mathbb{E}_{\mathbf{x} \sim \mathcal{U}(\mathcal{A}_{4d})} \left[(\mathcal{N}(\mathbf{x}) - f_d(\mathbf{x}))^2 \right] \leq \frac{1}{400}.$$

Then, \mathcal{N} has width

$$\Omega \left(\frac{\exp(\Omega(d))}{\text{poly}(C)} \right),$$

where the asymptotic notation hides constants that depend solely on σ .

The proof of the above theorem relies on a worst- to average-case reduction in a neural network setting, followed by a reduction to threshold circuits. Given a network which approximates f_d well, we can use it to construct a network which achieves similar accuracy, but with margins that can be made arbitrarily more uniform due to a concentration of measure argument. The crux of our proof is identifying a construction that re-randomizes the input sufficiently well, effectively obtaining a strong enough concentration of measure, while also doing so in a manner which maintains the

output value of the function. Thereafter, a very careful technical analysis is required to establish that the accuracy lost due to this re-randomization process is at most a constant. We refer the reader to Subsection 3.1 for a more detailed proof sketch of the theorem, and to Appendix A.1 for the full proof.

We point out that the constant $\frac{1}{400}$ is arbitrary, and our proof technique is capable of improving this to be arbitrarily close to the constant $\frac{1}{16}$ (at the cost of increasing the constants hidden in the asymptotic notation). Since a trivial approximation of a single constant neuron (which returns the value 0.5) yields accuracy $\frac{1}{4}$, this indicates that our analysis is very tight, and that adding even exponentially many neurons does not improve upon the trivial approximation by much.

We remark that our weight boundedness assumption is mild, since our lower bound remains exponential in d , as long as C is in itself not exponential in d . In such a case, where the weights required for expressing f_d must have exponential magnitude, it is known that stable gradient descent must run for exponentially many iterations in order for the weights to reach such a magnitude (see [Safran and Lee \(2022\)](#) for a more formal result of this kind). This suggests that even if a network of size polynomial in d can approximate f_d well, then in practice, learning such a representation using standard techniques is not tractable, and it is therefore of lesser interest from a practical perspective.

3.1. Techniques, and proof sketch of Thm. 3

In this subsection, we detail the key ideas behind the proof of our lower bound. The reader is referred to Appendix A.1 for the full proof.

3.1.1. STEP 1: FROM A CONTINUOUS DISTRIBUTION ON THE UNIT BALL TO A DISCRETE DISTRIBUTION ON THE BOOLEAN HYPERCUBE

We begin with assuming that we have a depth 2, σ -network \mathcal{N} , which approximates f_d to accuracy $\frac{1}{400}$, where σ satisfies Assumption 1, and our goal is to lower bound its width. By our assumption, this network provides a good approximation with respect to the continuous distribution $\mathcal{U}(\mathcal{A}_{4d})$ whose support is contained inside the unit ball. Since our aim here is to eventually use lower bounds from threshold circuits to get a lower bound on the width of \mathcal{N} , we first reduce this lower bound over the continuous distribution to the discrete distribution $\mathcal{U}(\mathcal{S}_{4d})$. This is done by breaking the distribution to a sum of distributions on the sets \mathcal{S}_{4d} and \mathcal{C}_{4d} , and using the probabilistic method and Markov's inequality on the randomness induced by the latter continuous component. This implies the existence of a depth 2 neural network $\tilde{\mathcal{N}}$, which has width similar to \mathcal{N} , but can approximate f_d on the discrete set \mathcal{S}_{4d} to an average square loss of $\frac{1}{399}$.

Next, we wish to show an approximation lower bound reduction from Boolean functions on the Boolean hypercube to the unit ball. Our reduction technique, which may be of independent interest, relies on embedding a scaled copy of the Boolean hypercube inside the unit ball. Since fitting a hypercube inside the unit ball requires scaling by a factor of $\Theta(\sqrt{d})$, this forces the Lipschitz constant to scale with d . To maintain its independence from the input dimension, we first observe that while the unit ball has much smaller volume than the unit hypercube, their packing numbers scale similarly when considering coverings using balls of small (but constant) radii. This is useful, since our distribution has its mass centered at a discrete set of exponentially many points. By embedding the Boolean hypercube into the Cartesian product of a scaled Boolean hypercube and a randomly-chosen and well-spread set which achieves an optimal packing, we are able to maintain Lipschitzness. Lastly, to show that the hardness of approximation property persists, we use a

Rademacher complexity argument. Essentially, the hardness in the scaled Boolean hypercube component stems by our assumption that the function is hard to approximate, and the hardness in the packing component stems from its random nature which guarantees that any correlation between the point and its target value is negligible. With this reduction result (see Thm. 10 in the appendix), we establish that the existence of $\tilde{\mathcal{N}}$ implies the existence of a σ -network $\mathcal{N}' : \mathbb{R}^{2d} \rightarrow \mathbb{R}$ which computes IP_d to an average squared error of $\frac{1}{398}$ on the Boolean hypercube.

3.1.2. STEP 2: FROM WORST- TO AVERAGE-CASE USING RANDOMIZATION

In the previous step, we constructed a depth 2, σ -network \mathcal{N}' , which approximates IP_d uniformly over $\{0, 1\}^{2d}$, to an average squared error of $\frac{1}{398}$. Intuitively, this means that \mathcal{N}' computes IP_d well in an average-case sense, since by Markov's inequality, for at least a constant fraction of the inputs $\mathbf{x}, \mathbf{y} \in \{0, 1\}^d$, we have that $\text{round}(\mathcal{N}'(\mathbf{x}, \mathbf{y})) = \text{IP}_d(\mathbf{x}, \mathbf{y})$. Our aim is now to use \mathcal{N}' to construct a depth 2, σ -network \mathcal{N}'' , such that $\text{round}(\mathcal{N}''(\mathbf{x}, \mathbf{y})) = \text{IP}_d(\mathbf{x}, \mathbf{y})$ holds *for all inputs* $\mathbf{x}, \mathbf{y} \in \{0, 1\}^d$. To this end, we use a randomization scheme on the input, where we map it to a higher dimensional space, and use a higher dimensional architecture of \mathcal{N}' . The purpose of this scheme is to alter the input in a manner which induces as much randomness as possible, but while also keeping its output unchanged. This is achieved by identifying the following three different alterations:

- Using the identity

$$\text{IP}_d(\mathbf{x}, \mathbf{y}) = \text{IP}_d(\mathbf{x} + \mathbf{x}', \mathbf{y} + \mathbf{y}') + \text{IP}_d(\mathbf{x}', \mathbf{y}') + \text{IP}_d(\mathbf{x} + \mathbf{x}', \mathbf{y}') + \text{IP}_d(\mathbf{x}', \mathbf{y} + \mathbf{y}') \pmod{2},$$

which holds for all $\mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}' \in \{0, 1\}^d$, we can generate uniformly random binary vectors $\mathbf{x}', \mathbf{y}' \in \{0, 1\}^d$, and replace \mathbf{x} and \mathbf{y} with the vectors $(\mathbf{x} + \mathbf{x}', \mathbf{x}', \mathbf{x} + \mathbf{x}', \mathbf{x}')$ and $(\mathbf{y} + \mathbf{y}', \mathbf{y}', \mathbf{y} + \mathbf{y}', \mathbf{y}')$, respectively, while keeping the IP mod 2 output unchanged. This additional randomness is useful for handling cases where the inputs have a lot of structure (e.g., when \mathbf{x}, \mathbf{y} are the all-zero or all-one vectors).

- We can pad our modified inputs with $\mathcal{O}(d)$ many uniformly generated bits, such that pairs where $x_i = y_i = 1$ are conditioned to be an even number. Due to this conditioning, we have that the inner product mod 2 value is unchanged. This padding greatly increases the randomness of the input.
- Lastly, since addition mod 2 is commutative, we can sample a random permutation, and apply it to our modified input, once again altering our input while keeping its IP mod 2 value unchanged. This allows us to add valuable randomness to our input, since permuted vectors with an almost equal number of pairs of the form $x_i = y_i = 0$; $x_i = 0, y_i = 1$; $x_i = 1, y_i = 0$; $x_i = y_i = 1$ are much closer to uniformly sampled vectors.

Combining the above into a single randomization process, we have a new, higher dimensional input (but with at most a linear blow-up), which has the same IP mod 2 value, yet whose probability distribution is close (in a certain sense) to a uniformly random input from the higher dimensional space. The bulk of the technical analysis in our proof consists of proving that this process results in a distribution close enough to uniform, so as to incur at most a constant additional loss in our approximation. This requires a very careful and tight analysis of the distribution.

After constructing the architecture which performs the above randomization process, we repeat the process polynomially many times, we concatenate the obtained hidden layers which perform this computation, and we use the output neuron to average their outputs. This results in a concentration of measure, which makes the approximation error much more uniformly spread over the Boolean hypercube. By a union bound and the probabilistic method, we can find realizations of our random construction that achieve this, and modify our network to incorporate these realizations without changing the architecture of the network \mathcal{N}'' . This can be done since the operations of inverting a bit of the input or permuting the inputs are linear operations that can be absorbed in the weights of the hidden layer.

3.1.3. STEP 3: CONSTRUCTING A THRESHOLD CIRCUIT COMPUTING IP_d

Following the previous two steps, we now have a depth 2, σ -network \mathcal{N}'' , such that $\text{round}(\mathcal{N}''(\mathbf{x}, \mathbf{y})) = \text{IP}_d(\mathbf{x}, \mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \{0, 1\}^d$. The final step in our proof is to use this architecture to construct a depth 2 threshold circuit with the same property. To this end, we first need to approximate σ to arbitrary accuracy using a depth 2 threshold network on a compact domain. This can be done by constructing a piecewise linear approximation as follows: Beginning with the leftmost point in the domain of approximation, we choose a constant function which coalesces with σ at this point, and we extend it until its distance from σ deviates from our target accuracy, in which case we make a ‘jump’ to a different constant value, continuing in this manner until the approximation is complete. Since Assumption 1 guarantees that σ is ‘well-behaved’ in the sense that it cannot change its output too wildly, we have that we cannot perform too many such jumps; namely, σ can be approximated by a piecewise linear function with not too many discontinuities. Replacing each σ with a moderately sized depth 2 threshold network, we obtain a moderately sized depth 2 threshold network \mathcal{N}''' that satisfies $\text{round}(\mathcal{N}'''(\mathbf{x}, \mathbf{y})) = \text{IP}_d(\mathbf{x}, \mathbf{y})$. Lastly, we turn this threshold network into a threshold circuit by adding a threshold activation on the output neuron. It is a classic result in circuit complexity that IP_d cannot be computed by a depth 2 threshold circuit with bounded weights, unless its width is exponential in d (Hajnal et al., 1993). This implies a lower bound on the width of \mathcal{N} , from which the theorem follows.

4. Upper Bound

Having presented our lower bound, in this section, we turn to complement it with a relatively straightforward upper bound for depth 3 networks. We also provide a concrete example in the case where σ is the ReLU activation function, in which our required width and magnitude of the weights provides a stronger result than the bounds guaranteed in our theorem.

We now formally state our upper bound result below.

Theorem 4 *Let $\varepsilon > 0$, and suppose that σ satisfies Assumption 2. Then, there exists a depth 3, width $\mathcal{O}\left(\frac{d^2}{\varepsilon}\right)$ σ -network \mathcal{N} , with weights bounded in magnitude by $\mathcal{O}\left(\frac{d}{\varepsilon^2}\right)$, such that*

$$\sup_{\mathbf{x} \in \mathcal{A}_{4d}} |\mathcal{N}(\mathbf{x}) - f_d(\mathbf{x})| \leq \varepsilon.$$

The proof of the theorem, which appears in Appendix A.2, utilizes the fact that f_d can be approximated efficiently by composing two different simple functions, which focus on the last $2d$

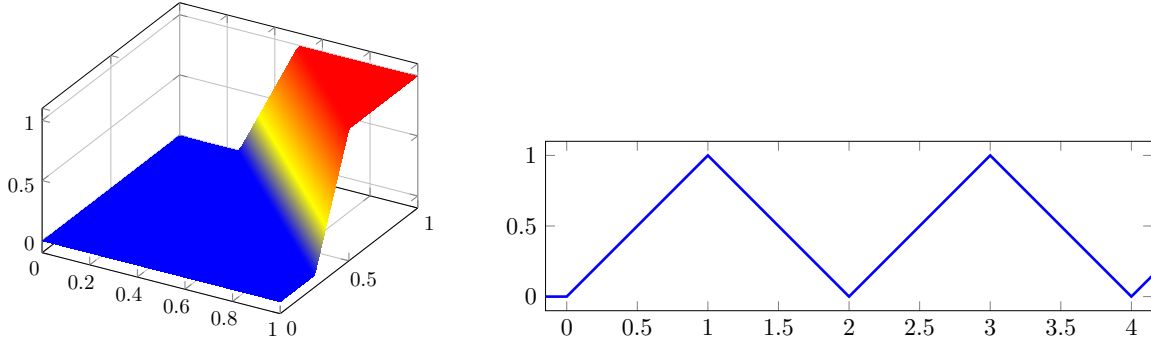


Figure 1: Computing f_d using a depth 3 ReLU network. The figure to the left plots the function $(x, y) \mapsto [4x + 4y - 5]_+ - [4x + 4y - 6]_+$, which equals $\text{AND}(\text{round}(x), \text{round}(y))$ for all $x, y \in [0, 0.25] \cup [0.75, 1]$. The figure to the right plots the function $x \mapsto t_d(x)$, defined in Example 1. When composing the latter with a scaled sum of the former, iterating over all pairs of coordinates x_i, y_i , we obtain a function that coalesces with f_d on \mathcal{A}_{4d} . Best viewed in color.

coordinates of the input. Since a depth 3 network has two hidden layers with non-linear σ activations, we are able to use each hidden layer to compute each function, and obtain the desired approximation.

We remark that we provide our positive approximation result in terms of the L_∞ norm rather than the L_2 norm with respect to the uniform distribution supported on \mathcal{A}_{4d} as our lower bound does. Since L_∞ approximation is more stringent than L_2 , this provides a more general result which implies an L_2 approximation of $f_d(\cdot, \cdot)$ with respect to *any* distribution supported on \mathcal{A}_{4d} .

To give a more concrete example of an approximation obtained by our theorem, we present the following specific construction when σ is the ReLU activation.

Example 1 Let $[x]_+ := \max\{0, x\}$ denote the ReLU activation function, and define

$$t_d(x) := [z]_+ + \sum_{i=1}^d 2 \cdot (-1)^i [z - i]_+.$$

Then, for an input $\mathbf{x}' := (\mathbf{z}, \mathbf{x}, \mathbf{y}) \in \mathbb{R}^{2d} \times \mathbb{R}^d \times \mathbb{R}^d$, the network given by

$$\mathcal{N}(\mathbf{z}, \mathbf{x}, \mathbf{y}) := t_d \left(\sum_{i=1}^d 3\sqrt{d} [4x_i + 4y_i - 5]_+ - 3\sqrt{d} [4x_i + 4y_i - 6]_+ \right)$$

satisfies

$$\sup_{\mathbf{x}' \in \mathcal{A}_d} |\mathcal{N}(\mathbf{x}') - f_d(\mathbf{x}')| = 0.$$

It is straightforward to verify that \mathcal{N} is a depth 3, width $2d$ ReLU network, with weights bounded in magnitude by $\mathcal{O}(d)$. Moreover, a simple computation shows that \mathcal{N} coalesces with f_d on \mathcal{A}_{4d} . Lastly, this approximation is also sparse, in the sense that it requires only $\mathcal{O}(d)$ neuron connections (see Fig. 1 for a visualization of the functions computed in each layer).

Following the works (Malach and Shalev-Shwartz, 2019; Malach et al., 2021), which show that some functions that were used to prove approximation lower bounds for neural networks cannot

be learned efficiently using standard methods, [Safran and Lee \(2022\)](#) have shown an optimization-based separation result where the deeper architecture can provably learn the efficient representation from finite data, using standard techniques such as gradient descent. It is interesting to note that [Example 1](#) provides a simple, linear in size and sparse approximation of f_d . This simplicity is much desired, since it may suggest that similarly to [Safran and Lee \(2022\)](#), learning this representation from finite data using a standard learning algorithm is tractable, and despite the simplicity of this ReLU approximation, our lower bound for this function provides a separation which is stronger than previously known results. We leave the study of proving such a stronger optimization-based separation result as an intriguing future work direction.

Acknowledgements.

Paul Valiant is partially supported by NSF award CCF-2127806. We thank Srikanth Srinivasan for useful discussions.

References

- Stéphane Boucheron, Olivier Bousquet, and Gábor Lugosi. Theory of classification: A survey of some recent advances. *ESAIM: probability and statistics*, 9:323–375, 2005.
- George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.
- Amit Daniely. Depth separation for neural networks. In *Conference on Learning Theory*, pages 690–696, 2017.
- Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Conference on learning theory*, pages 907–940. PMLR, 2016.
- Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. In *Conference On Learning Theory*, pages 297–299. PMLR, 2018.
- András Hajnal, Wolfgang Maass, Pavel Pudlák, Mario Szegedy, and György Turán. Threshold circuits of bounded depth. *Journal of Computer and System Sciences*, 46(2):129–154, 1993.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.
- Daniel Hsu, Clayton H Sanford, Rocco Servedio, and Emmanouil Vasileios Vlatakis-Gkaragkounis. On the approximation power of two-layer networks of random relus. In *Conference on Learning Theory*, pages 2423–2461. PMLR, 2021.
- Matthew Jenssen, Felix Joos, and Will Perkins. On kissing numbers and spherical codes in high dimensions. *Advances in Mathematics*, 335:307–321, 2018.

- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
- Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.
- Shiyu Liang and Rayadurgam Srikant. Why deep neural networks for function approximation? *arXiv preprint arXiv:1610.04161*, 2016.
- Eran Malach and Shai Shalev-Shwartz. Is deeper better only when shallow is good? *Advances in Neural Information Processing Systems*, 32, 2019.
- Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. The connection between approximation, depth separation and learnability in neural networks. In *Conference on Learning Theory*, pages 3265–3295. PMLR, 2021.
- James Martens, Arkadev Chattopadhyay, Toni Pitassi, and Richard Zemel. On the representational efficiency of restricted boltzmann machines. *Advances in Neural Information Processing Systems*, 26, 2013.
- Anirbit Mukherjee and Amitabh Basu. Lower bounds over boolean inputs for deep neural networks with relu gates. *arXiv preprint arXiv:1711.03073*, 2017.
- Eshaan Nichani, Alex Damian, and Jason D Lee. Provable guarantees for nonlinear feature learning in three-layer neural networks. *arXiv preprint arXiv:2305.06986*, 2023.
- Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5):503–519, 2017.
- Yunwei Ren, Mo Zhou, and Rong Ge. Depth separation with multilayer mean-field networks. *arXiv preprint arXiv:2304.01063*, 2023.
- Itay Safran and Jason Lee. Optimization-based separations for neural networks. In *Conference on Learning Theory*, pages 3–64. PMLR, 2022.
- Itay Safran and Ohad Shamir. Depth-width tradeoffs in approximating natural functions with neural networks. In *International conference on machine learning*, pages 2979–2987. PMLR, 2017.
- Itay Safran, Ronen Eldan, and Ohad Shamir. Depth separations in neural networks: what is actually being separated? In *Conference on Learning Theory*, pages 2664–2666. PMLR, 2019.
- Itay Safran, Daniel Reichman, and Paul Valiant. How many neurons does it take to approximate the maximum? In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3156–3183. SIAM, 2024.
- Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Matus Telgarsky. Benefits of depth in neural networks. In *Conference on learning theory*, pages 1517–1539. PMLR, 2016.

Gal Vardi and Ohad Shamir. Neural networks with small weights and depth-separation barriers. *arXiv preprint arXiv:2006.00625*, 2020.

Gal Vardi, Daniel Reichman, Toniann Pitassi, and Ohad Shamir. Size and depth separation in approximating benign functions with neural networks. In *Conference on Learning Theory*, pages 4195–4223. PMLR, 2021.

Luca Venturi, Samy Jelassi, Tristan Ozuch, and Joan Bruna. Depth separation beyond radial functions. *The Journal of Machine Learning Research*, 23(1):5309–5364, 2022.

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

Appendix A. Proofs

A.1. Proof of Thm. 3

Before we begin, we first state a relaxation of Assumption 1 which we will use in our proof. Albeit slightly technical, it greatly extends the types of activation functions for which our lower bound holds. The following definitions are required for understanding the assumption statement: Given a function $f : \mathbb{R} \rightarrow \mathbb{R}$, we define its *total variation* on the interval $[a, b] \subseteq \mathbb{R}$ as $V_a^b(f) := \sup_P \sum_{i=1}^{n_P} |f(x_i) - f(x_{i-1})|$, where the supremum is taken over all the possible partitions of the interval $[a, b]$. For a function class \mathcal{F} and sample $\mathbf{x}_1, \dots, \mathbf{x}_n$, the (empirical) Rademacher complexity of \mathcal{F} is given by $\mathcal{R}_n(\mathcal{F}) := \mathbb{E} [\sup_{f \in \mathcal{F}} |\frac{1}{n} \sum_{i=1}^n \xi_i f(\mathbf{x}_i)|]$, where the expectation is taken over ξ_1, \dots, ξ_n which are i.i.d. Rademacher random variables.

Assumption 3 (Lower bound – generalized) *The activation function σ is (Lebesgue) measurable and satisfies*

$$|\sigma(x)| \leq C_\sigma (1 + |x|^{\alpha_\sigma}),$$

and

$$V_a^b(\sigma) \leq C_\sigma (1 + (|a| + |b|)^{\alpha_\sigma}),$$

for all $x \in \mathbb{R}$ and real $a < b$; and the function class $\mathcal{N}_{w,C,d}$ of d -dimensional σ -networks of depth 2, width at most w , and weights bounded by C , satisfies

$$\mathcal{R}_n(\mathcal{N}_{w,C,d}) \leq \frac{(w \cdot C \cdot d)^{\alpha_\sigma}}{\sqrt{n}},$$

for any dataset of size n contained inside $[0, 1]^d$, and for some constants C_σ, α_σ , which depend solely on σ .

Our above assumption is very mild, and as previously discussed, it is satisfied by many commonly used activations such as ReLU, threshold and sigmoidal activations as a particular case. The fact that these activation functions satisfy the first two requirements is straightforward, and an appropriate Rademacher bound for thresholds follows from [Shalev-Shwartz and Ben-David \(2014\)](#),

Thm. 20.6) and [Boucheron et al. \(2005, Eq. \(6\)\)](#), and for $\mathcal{O}(1)$ -Lipschitz activation functions from [Golowich et al. \(2018\)](#). The boundedness of $|\sigma(x)|$ is a standard assumption when proving approximation lower bounds, and it is also used in [Eldan and Shamir \(2016\)](#) for example. Since our proof is based on a reduction to threshold circuits, our technique fails if we consider certain activation functions that are highly oscillatory, since there are such pathological activations that can be used to compute any Boolean function $f : \{0, 1\}^d \rightarrow \{0, 1\}$, even with just a single neuron. In light of this, we also make an assumption that the total variation of the activation function is polynomially bounded on a compact domain. Lastly, our assumption on the boundedness of the Rademacher complexity merely requires that the depth 2 networks which employ σ activations are statistically learnable. Such an assumption is again very mild, and holds for essentially any activation function which is used in practice.

Before we can prove the theorem, we will first state and prove several auxiliary lemmas that will be used later on. In what follows, we use the notation $\|\mathcal{D}\|_2$ to denote the L_2 norm of a discrete distribution \mathcal{D} . Namely, for a random variable $X \sim \mathcal{D}$ sampled from a sample space \mathcal{X} , we have

$$\|\mathcal{D}\|_2 := \sqrt{\sum_{x \in \mathcal{X}} \left(\mathbb{P}_{x \sim \mathcal{D}}[X = x] \right)^2}.$$

The following lemma provides an upper bound on certain subsets of the L_2 norm of the distribution we analyze in our reduction scheme.

Lemma 5 *For d, D positive integers divisible by 4, and given $d_1, d_2, d_3, d_4 \geq 0$ with $d_1 + d_2 + d_3 + d_4 = d$ then we have:*

$$\sum_{(D_1, D_2, D_3, D_4) : \sum_i D_i = D} \frac{\binom{D}{D_1, D_2, D_3, D_4}^2}{\binom{D+d}{D_1+d_1, D_2+d_2, D_3+d_3, D_4+d_4}} \leq e^{\frac{4}{D} \sum_{i=1}^4 (d_i - \frac{d}{4})^2} \cdot \left(1 + \frac{d}{D}\right)^{3/2} \cdot 4^{D-d} \quad (2)$$

Proof We start by comparing $\frac{(D_i+d_i)!}{D_i!}$ to $\frac{(D/4+d_i)!}{D/4!}$. For $D_i \geq D/4$ the ratio of these two expressions equals $\prod_{j=D/4+1}^{D_i} \frac{j+d_i}{j} \leq \prod_{j=D/4+1}^{D_i} \frac{D/4+d_i}{D/4} = (1 + \frac{4d_i}{D})^{D_i-D/4}$. On the other hand, for $D_i < D/4$ that ratio of our two expressions equals $\prod_{j=D_i+1}^{D/4} \frac{j}{j+d_i} \leq \prod_{j=D_i+1}^{D/4} \frac{D/4}{D/4+d_i} = (1 + \frac{4d_i}{D})^{D_i-D/4}$. Thus in all cases, $\frac{(D_i+d_i)!}{D_i!} \leq (1 + \frac{4d_i}{D})^{D_i-D/4} \frac{(D/4+d_i)!}{D/4!}$. We apply this inequality to bound the terms on the left hand side of Eq. (2):

$$\begin{aligned} \frac{\binom{D}{D_1, D_2, D_3, D_4}}{\binom{D+d}{D_1+d_1, D_2+d_2, D_3+d_3, D_4+d_4}} &= \frac{D!}{(D+d)!} \frac{(D_1+d_1)!}{D_1!} \frac{(D_2+d_2)!}{D_2!} \frac{(D_3+d_3)!}{D_3!} \frac{(D_4+d_4)!}{D_4!} \\ &\leq \frac{\binom{D}{D/4, D/4, D/4, D/4}}{\binom{D+d}{D/4+d_1, D/4+d_2, D/4+d_3, D/4+d_4}} \prod_{i=1}^4 \left(1 + \frac{4d_i}{D}\right)^{D_i-D/4} \end{aligned} \quad (3)$$

We will evaluate the sum in Equation 2 and using the identity

$$(p_1 + p_2 + p_3 + p_4)^D = \sum_{(D_1, D_2, D_3, D_4) : \sum_i D_i = D} \binom{D}{D_1, D_2, D_3, D_4} p_1^{D_1} p_2^{D_2} p_3^{D_3} p_4^{D_4}$$

along with the above equation we can write,

$$\begin{aligned}
 & \sum_{(D_1, D_2, D_3, D_4): \sum_i D_i = D} \frac{\binom{D}{D_1, D_2, D_3, D_4}^2}{\binom{D+d}{D_1+d_1, D_2+d_2, D_3+d_3, D_4+d_4}} \\
 &= \sum_{(D_1, D_2, D_3, D_4): \sum_i D_i = D} \binom{D}{D_1, D_2, D_3, D_4} \frac{\binom{D}{D_1, D_2, D_3, D_4}}{\binom{D+d}{D_1+d_1, D_2+d_2, D_3+d_3, D_4+d_4}} \\
 &\leq \frac{\binom{D}{D/4, D/4, D/4, D/4}}{\binom{D+d}{D/4+d_1, D/4+d_2, D/4+d_3, D/4+d_4}} \left(\prod_{i=1}^4 \left(1 + \frac{4d_i}{D}\right)^{-D/4} \right) \sum_{(D_1, D_2, D_3, D_4): \sum_i D_i = D} \binom{D}{D_1, D_2, D_3, D_4} \prod_{i=1}^4 \left(1 + \frac{4d_i}{D}\right)^{D_i} \\
 &= \frac{\binom{D}{D/4, D/4, D/4, D/4}}{\binom{D+d}{D/4+d_1, D/4+d_2, D/4+d_3, D/4+d_4}} \left(\prod_{i=1}^4 \left(1 + \frac{4d_i}{D}\right)^{-D/4} \right) \left(4 + \frac{4 \sum_i d_i}{D}\right)^D
 \end{aligned}$$

Since the second derivative of the function $\log(x)$ is ≥ -1 for inputs $x \geq 1$, we use the Lagrange remainder form of the Taylor expansion around any $\ell \geq 1$ to lower bound the logarithm function for all $x \geq 1$ as $\log(x) \geq \log(\ell) + \frac{1}{\ell}(x - \ell) - \frac{1}{2}(x - \ell)^2$. We use this to bound the last two terms above, bounding $\log(x)$ for $x = 1 + \frac{4d_i}{D}$, centering our approximation at $\ell = 1 + \frac{d}{D}$, and using the fact that $\sum_i d_i = d$:

$$\begin{aligned}
 \left(\prod_{i=1}^4 \left(1 + \frac{4d_i}{D}\right)^{-D/4} \right) \left(4 + \frac{4 \sum_i d_i}{D}\right)^D &\leq \left(\prod_{i=1}^4 \left(e^{\log(1 + \frac{d}{D}) + \frac{4d_i - d}{D(1 + \frac{d}{D})} - \frac{1}{2}(\frac{4d_i - d}{D})^2} \right)^{-D/4} \right) \left(4 + \frac{4 \sum_i d_i}{D}\right)^D \\
 &= 4^D e^{\sum_{i=1}^4 2 \frac{(d_i - \frac{d}{4})^2}{D}}
 \end{aligned}$$

We then turn to the first term, which we reexpress as

$$\frac{\binom{D}{D/4, D/4, D/4, D/4}}{\binom{D+d}{D/4+d_1, D/4+d_2, D/4+d_3, D/4+d_4}} = \frac{D!}{(D+d)!} \frac{(D/4+d_1)!}{(D/4)!} \frac{(D/4+d_2)!}{(D/4)!} \frac{(D/4+d_3)!}{(D/4)!} \frac{(D/4+d_4)!}{(D/4)!}$$

We bound $(D/4 + d_i)!$ by reexpressing it via the Gamma function as $\Gamma(1 + D/4 + d_i)$. We then use the Lagrange remainder form of the Taylor expansion of the function $f(x) := \log \Gamma(x)$ around $\ell = 1 + \frac{D}{4} + \frac{d}{4}$ to conclude that there exists y between ℓ and x such that

$$f(x) = f(\ell) + (x - \ell)f'(\ell) + \frac{1}{2}f''(y)(x - \ell)^2$$

The second derivative of the $\log \Gamma$ function at x is bounded by $\frac{x+1}{x^2} \leq \frac{x+1}{x^2-1} = \frac{1}{x-1}$. Thus, letting $x_i = 1 + D/4 + d_i$, we have, that there exist y_i between x_i and ℓ for which

$$\begin{aligned}
 \prod_{i=1}^4 (D/4 + d_i)! &\leq \prod_{i=1}^4 e^{f(\ell) + (x_i - \ell)f'(\ell) + \frac{1}{2}f''(y_i)(x_i - \ell)^2} \\
 &= (D/4 + d/4)!^4 e^{\frac{1}{2} \sum_{i=1}^4 f''(y_i)(d_i - \frac{d}{4})^2} \\
 &\leq (D/4 + d/4)!^4 e^{\frac{2}{D} \sum_{i=1}^4 (d_i - \frac{d}{4})^2}
 \end{aligned}$$

where the equality makes use of the fact that the middle (linear in x_i) term vanishes since $\sum_{i=1}^4 (x_i - \ell) = 0$; the last inequality makes use of the fact that both $x_i, \ell \geq 1 + D/4$, and thus, since y_i is between these, we have $y_i \geq 1 + D/4$, and hence our bound on the second derivative of $\log \Gamma(y)$ is at most $\frac{4}{D}$.

Thus, overall, we have shown

$$\begin{aligned} \sum_{(D_1, D_2, D_3, D_4): \sum_i D_i = D} \frac{\binom{D}{D_1, D_2, D_3, D_4}^2}{\binom{D+d}{D_1+d_1, D_2+d_2, D_3+d_3, D_4+d_4}} &\leq 4^D e^{\sum_{i=1}^4 2 \frac{(d_i - \frac{d}{4})^2}{D}} \frac{D!(D/4 + d/4)!^4}{(D+d)!(D/4)!^4} e^{\frac{2}{D} \sum_{i=1}^4 (d_i - \frac{d}{4})^2} \\ &= 4^D \frac{D!(D/4 + d/4)!^4}{(D+d)!(D/4)!^4} e^{\frac{4}{D} \sum_{i=1}^4 (d_i - \frac{d}{4})^2} \end{aligned}$$

Finally, we bound the ratio of factorials as follows. Define the function f on integer input $f(j) := 4^{-4j} \binom{4j}{j, j, j, j} = 4^{-4j} \frac{(4j)!}{j!^4}$. We will show that for any integers $k \geq j$ we have $\frac{f(j)}{f(k)} \leq (j/k)^{3/2}$. We have $\frac{f(j)}{f(j+1)} = \frac{(4j+4)^3}{(4j+1)(4j+2)(4j+3)}$. Expressing each term in the denominator as a weighted average of $4j$ and $4j+4$ and applying the (weighted) AM-GM inequality to each term on the denominator separately, we thus lower bound the denominator by $(4j)^{3/2}(4j+4)^{3/2}$ and thus upper bound $\frac{f(j)}{f(j+1)} \leq \frac{(4j+4)^{3/2}}{(4j)^{3/2}} = (\frac{j+1}{j})^{3/2}$. Multiplying this bound for all numbers between j and k yields $\frac{f(j)}{f(k)} \leq (j/k)^{3/2}$ as claimed.

Thus $\frac{D!(D/4+d/4)!^4}{(D+d)!(D/4)!^4} \leq (\frac{D+d}{D})^{3/2}$. And we have derived Eq. (2) as claimed. \blacksquare

The following lemma bounds the moment generating function of a certain distribution, which arises in our analysis of the reduction scheme.

Lemma 6 *For any dimension d and any vectors $\mathbf{x}, \mathbf{y} \in \{0, 1\}^d$, consider the process of picking random vectors $\mathbf{x}', \mathbf{y}' \leftarrow \{0, 1\}^d$ and outputting*

$$\begin{aligned} A &:= (\mathbf{x} + \mathbf{x}', \mathbf{x}', \mathbf{x} + \mathbf{x}', \mathbf{x}'), \\ B &:= (\mathbf{y} + \mathbf{y}', \mathbf{y}', \mathbf{y} + \mathbf{y}', \mathbf{y}'), \end{aligned}$$

where all additions are mod 2. Among these $4d$ entries, let d_1 count the number of indices $j \in \{1, \dots, 4d\}$ where $A_j = B_j = 0$; let d_2 count the entries where $A_j = 0, B_j = 1$; let d_3 count the entries where $A_j = 1, B_j = 0$; and let d_4 count the entries where $A_j = B_j = 1$. Then

$$\mathbb{E}_{d_1, d_2, d_3, d_4} \left[e^{s \sum_{i=1}^4 (d_i - d)^2} \right] \leq \left(\frac{1}{1 - 24ds} \right)^2$$

Proof Our overall analysis technique will be to first compute the moment generating function (MGF) of the distribution of (d_1, d_2, d_3, d_4) , centered at its mean, (d, d, d, d) . Explicitly, since we have a 4-dimensional distribution, the MGF has 4 parameters:

$$M(t_1, t_2, t_3, t_4) := \mathbb{E}_{d_1, d_2, d_3, d_4} \left[e^{\sum_{i=1}^4 t_i (d_i - d)} \right]$$

Crucially, the MGF of the sum of independent random vectors multiplies; thus we will compute the MGF for the contribution to (d_1, d_2, d_3, d_4) from each location A_j, B_j separately.

After we have computed the MGF, we will use this to compute the quantity in the lemma statement, $\mathbb{E}_{d_1, d_2, d_3, d_4} [e^{s \sum_{i=1}^4 (d_i - d)^2}]$. Since the MGF of the Gaussian $\mathcal{N}(0, 2s)$ equals e^{st^2} , the MGF for the corresponding 4-dimensional Gaussian, after a slight change of variables, yields the relation that, for any d_1, d_2, d_3, d_4 :

$$\mathbb{E}_{(t_1, t_2, t_3, t_4) \leftarrow \mathcal{N}(0, 2s)} [e^{\sum_{i=1}^4 t_i (d_i - d)}] = e^{s \sum_{i=1}^4 (d_i - d)^2}$$

and thus that

$$\mathbb{E}_{(t_1, t_2, t_3, t_4) \leftarrow \mathcal{N}(0, 2s)} [M(t_1, t_2, t_3, t_4)] = \mathbb{E}_{d_1, d_2, d_3, d_4} [e^{s \sum_{i=1}^4 (d_i - d)^2}] \quad (4)$$

Consider, for some index j , the 4 cases for the pair x_j, y_j , and the 4-tuple of pairs of entries $(x_j + x'_j, y_j + y'_j), (x'_j, y'_j), (x_j + x'_j, y'_j), (x'_j, y_j + y'_j)$ they induce in A, B .

If $x_j = 0, y_j = 0$, then the 4-tuple of pairs $(x_j + x'_j, y_j + y'_j), (x'_j, y'_j), (x_j + x'_j, y'_j), (x'_j, y_j + y'_j)$ produced in the vectors A, B contains simply 4 copies of the randomly chosen pair of bits (x'_j, y'_j) . Thus the contribution to the counts d_1, d_2, d_3, d_4 will be uniformly randomly chosen among the 4-tuples $(4, 0, 0, 0), (0, 4, 0, 0), (0, 0, 4, 0), (0, 0, 0, 4)$. The moment generating function of this uniform distribution over 4 possibilities is just the sum of the 4 terms in the definition of the MGF; as usual, we center the MGF of this portion of the distribution at its mean, which here is $(1, 1, 1, 1)$:

$$\begin{aligned} M_j(t_1, t_2, t_3, t_4) &= \frac{1}{4} e^{(4-1)t_1 + (0-1)t_2 + (0-1)t_3 + (0-1)t_4} + \frac{1}{4} e^{(0-1)t_1 + (4-1)t_2 + (0-1)t_3 + (0-1)t_4} \\ &\quad + \frac{1}{4} e^{(0-1)t_1 + (0-1)t_2 + (4-1)t_3 + (0-1)t_4} + \frac{1}{4} e^{(0-1)t_1 + (0-1)t_2 + (0-1)t_3 + (4-1)t_4} \end{aligned}$$

We can loosely bound this by $e^{6 \sum_{i=1}^4 t_i^2}$ as follows. Each of the 4 exponential terms is the exponential of the inner product of (t_1, t_2, t_3, t_4) with a vector v of length $\sqrt{12}$. Thus, for any unit 4-vector u , we have that the exponential $e^{\langle (t_1, t_2, t_3, t_4), v \rangle}$ has second derivative in direction u that is at most 12 times the value of the exponential $e^{\langle (t_1, t_2, t_3, t_4), v \rangle}$. Since this property—of having directional second derivative in direction u that is at most 12 times the function value—is preserved under scaling a function by a positive constant, and is preserved by function addition, we conclude that this property applies to the entire function $M_j(t_1, t_2, t_3, t_4)$. Namely, for any vector u , the second derivative of $M_j(t_1, t_2, t_3, t_4)$ in the direction u is at most $12M_j(t_1, t_2, t_3, t_4)$. Since $M_j(t_1, t_2, t_3, t_4)$ has value 1 at the origin, and 0 gradient, we can bound its value at any multiple xu by solving the differential equation $f(0) = 1, f'(0) = 0, f''(x) \leq 12f(x)$, to yield $f(x) \leq \cosh(\sqrt{12}x) \leq e^{(12/2)x^2}$. Since this bounds $M(t_1, t_2, t_3, t_4)$ when $(t_1, t_2, t_3, t_4) = xu$ for any unit vector u , we conclude that $M(t_1, t_2, t_3, t_4) \leq e^{6 \sum_{i=1}^4 t_i^2}$.

Moving on to the next case, if $x_j = 0, y_j = 1$ case, then the 4-tuple of pairs $(x_j + x'_j, y_j + y'_j), (x'_j, y'_j), (x_j + x'_j, y'_j), (x'_j, y_j + y'_j)$ contains 2 copies of (x'_j, y'_j) and 2 copies of $(x'_j, 1 + y'_j)$. Thus the contribution to the counts d_1, d_2, d_3, d_4 will be uniformly chosen among the 4-tuples $(2, 2, 0, 0), (0, 0, 2, 2)$, where there are only 2 possibilities in this case. The moment generating function of this uniform distribution over 2 possibilities is just the sum of the 2 terms in the definition of the MGF, which as usual we center at the mean, $(1, 1, 1, 1)$:

$$M_j(t_1, t_2, t_3, t_4) = \frac{1}{2} e^{(2-1)t_1 + (2-1)t_2 + (0-1)t_3 + (0-1)t_4} + \frac{1}{2} e^{(0-1)t_1 + (0-1)t_2 + (2-1)t_3 + (2-1)t_4}$$

This equals $\cosh(\langle (t_1, t_2, t_3, t_4), (1, 1, -1, -1) \rangle) \leq e^{\langle (t_1, t_2, t_3, t_4), (1, 1, -1, -1) \rangle^2 / 2} \leq e^{(\|(1, 1, -1, -1)\|_2^2 / 2) \sum_{i=1}^4 t_i^2}$, namely $e^{2 \sum_{i=1}^4 t_i^2}$.

The next case, if $x_j = 1, y_j = 0$, is analogous to the previous case. Here the 4-tuple of pairs $(x_j + x'_j, y_j + y'_j), (x'_j, y'_j), (x_j + x'_j, y'_j), (x'_j, y_j + y'_j)$ contains 2 copies of (x'_j, y'_j) and 2 copies of $(1 + x'_j, y'_j)$. Thus the contribution to the counts d_1, d_2, d_3, d_4 will be uniformly chosen among the 4-tuples $(2, 0, 2, 0), (0, 2, 0, 2)$. The moment generating function of this uniform distribution over 2 possibilities is just the sum of the 2 terms in the definition of the MGF, which as usual we center at the mean, $(1, 1, 1, 1)$:

$$M_j(t_1, t_2, t_3, t_4) = \frac{1}{2} e^{(2-1)t_1 + (0-1)t_2 + (2-1)t_3 + (0-1)t_4} + \frac{1}{2} e^{(0-1)t_1 + (2-1)t_2 + (0-1)t_3 + (2-1)t_4}$$

Analogously to the previous case, this is at most $e^{2 \sum_{i=1}^4 t_i^2}$.

Finally, in the case $x_j = 1, y_j = 1$, then the 4-tuple of pairs $(x_j + x'_j, y_j + y'_j), (x'_j, y'_j), (x_j + x'_j, y'_j), (x'_j, y_j + y'_j)$ equals $(1 + x'_j, 1 + y'_j), (x'_j, y'_j), (1 + x'_j, y'_j), (x'_j, 1 + y'_j)$ and thus always contributes $(1, 1, 1, 1)$ to the 4 counts. Thus the moment generating function centered at $(1, 1, 1, 1)$ is, trivially, $M_j(t_1, t_2, t_3, t_4) = 1$.

Putting together the pieces: since the moment generating function is simply the product of its contribution from each index j from 1 to d , and in all cases we bounded this contribution by $e^{6 \sum_{i=1}^4 t_i^2}$, we have that the overall MGF of all d indices is bounded as

$$M(t_1, t_2, t_3, t_4) \leq e^{6d \sum_{i=1}^4 t_i^2}$$

We now bound the desired quantity in this lemma via Eq. (4). We thus have

$$\begin{aligned} \mathbb{E}_{(t_1, t_2, t_3, t_4) \leftarrow \mathcal{N}(0, 2s)} [M(t_1, t_2, t_3, t_4)] &\leq \mathbb{E}_{(t_1, t_2, t_3, t_4) \leftarrow \mathcal{N}(0, 2s)} [e^{6d \sum_{i=1}^4 t_i^2}] \\ &= \int_{\mathbb{R}^4} \frac{1}{\sqrt{2\pi \cdot 2s}^4} e^{-\frac{1}{4s} \sum_{i=1}^4 t_i^2} e^{6d \sum_{i=1}^4 t_i^2} dt_1 dt_2 dt_3 dt_4 \\ &= \sqrt{\frac{1}{\frac{1}{4s} - 6d}}^4 = \left(\frac{1}{1 - 24ds} \right)^2 \end{aligned}$$

as desired. ■

The following proposition establishes an L_2 upper bound on the distribution induced by the randomization process in our reduction.

Proposition 7 *For any dimension d and any vectors $\mathbf{x}, \mathbf{y} \in \{0, 1\}^d$, let $D = 100d$. Pick random vectors $\mathbf{x}', \mathbf{y}' \leftarrow \{0, 1\}^d$; pick random vectors $\mathbf{x}'', \mathbf{y}'' \leftarrow \{0, 1\}^D$ such that there are an even number of indices $j \in \{1, \dots, D\}$ where $\mathbf{x}''_j = \mathbf{y}''_j = 1$; and pick a random permutation τ of $4d + D$ elements. The claim is that the pair of length $4d + D$ vectors*

$$\begin{aligned} X &:= \tau(\mathbf{x} + \mathbf{x}', \mathbf{x}', \mathbf{x} + \mathbf{x}', \mathbf{x}', \mathbf{x}''), \\ Y &:= \tau(\mathbf{y} + \mathbf{y}', \mathbf{y}', \mathbf{y} + \mathbf{y}', \mathbf{y}', \mathbf{y}''), \end{aligned}$$

considered as a distribution \mathcal{D} over domain $\{0, 1\}^{2(4d+D)}$, has L_2 norm at most $8 \cdot 2^{-(4d+D)}$, which is 8 times the L_2 norm of the uniform distribution over this domain.

Proof Among the $4d$ entries of the pair $A = (\mathbf{x} + \mathbf{x}', \mathbf{x}', \mathbf{x} + \mathbf{x}', \mathbf{x}')$, $B = (\mathbf{y} + \mathbf{y}', \mathbf{y}', \mathbf{y} + \mathbf{y}', \mathbf{y}')$ let d_1 count the number of indices where $A_j = B_j = 0$; let d_2 count the entries where $A_j = 0, B_j = 1$; let d_3 count the entries where $A_j = 1, B_j = 0$; and let d_4 count the entries where $A_j = B_j = 1$. Thus $d_1 + d_2 + d_3 + d_4 = 4d$.

Since $\mathcal{D} = \sum_{d_1, d_2, d_3, d_4} \Pr_{\mathcal{D}}[d_1, d_2, d_3, d_4] \mathcal{D}_{|d_1, d_2, d_3, d_4}$, by the triangle inequality we have

$$\|\mathcal{D}\|_2 \leq \sum_{d_1, d_2, d_3, d_4} \Pr_{\mathcal{D}}[d_1, d_2, d_3, d_4] \|\mathcal{D}_{|d_1, d_2, d_3, d_4}\|_2 \quad (5)$$

For the random vectors $\mathbf{x}'', \mathbf{y}''$, let D_1, D_2, D_3, D_4 count the number of indices $j \in \{1, \dots, D\}$ respectively where $x''_j = 0, y''_j = 0$; where $x''_j = 0, y''_j = 1$; where $x''_j = 1, y''_j = 0$; and where $x''_j = 1, y''_j = 1$. Thus $D_1 + D_2 + D_3 + D_4 = D$.

Thus there will be $D_1 + d_1$ total indices where $X_j = 0, Y_j = 0$, etc. Given these total counts $D_1 + d_1, D_2 + d_2, D_3 + d_3, D_4 + d_4$, the total number of rearrangements of these columns is the multinomial $\binom{D+4d}{D_1+d_1, D_2+d_2, D_3+d_3, D_4+d_4}$; and the random permutation τ will choose a uniformly random one of these arrangements.

The probability of the random vectors $\mathbf{x}'', \mathbf{y}''$ having counts exactly D_1, D_2, D_3, D_4 would be exactly $4^{-D} \binom{D}{D_1, D_2, D_3, D_4}$ if we were not conditioning on D_4 being even. The probability of D_4 being even is $\geq \frac{1}{2}$, since the difference between the number of instantiations where D_4 is even versus D_4 is odd can be exactly computed as $\sum_{(D_1, D_2, D_3, D_4): \sum_i D_i = D} \binom{D}{D_1, D_2, D_3, D_4} (-1)^{D_4} = (1 + 1 + 1 - 1)^D \geq 0$. Thus, by the law of conditional probability, we have

$$\begin{aligned} 4^{-D} \binom{D}{D_1, D_2, D_3, D_4} &= \mathbb{P}[D_1, D_2, D_3, D_4] \\ &= \mathbb{P}[D_1, D_2, D_3, D_4 | D_4 \text{ is even}] \mathbb{P}[D_4 \text{ is even}] + \mathbb{P}[D_1, D_2, D_3, D_4 | D_4 \text{ is odd}] \mathbb{P}[D_4 \text{ is odd}] \\ &\geq \frac{1}{2} \mathbb{P}[D_1, D_2, D_3, D_4 | D_4 \text{ is even}], \end{aligned}$$

therefore the probability of D_1, D_2, D_3, D_4 with even D_4 is $\leq 2 \cdot 4^{-D} \binom{D}{D_1, D_2, D_3, D_4}$.

Thus, overall, the random process $\mathcal{D}_{|d_1, d_2, d_3, d_4}$ can be described as picking D_1, D_2, D_3, D_4 with probability $\leq 2 \cdot 4^{-D} \binom{D}{D_1, D_2, D_3, D_4}$, and then uniformly splitting this probability among the $\binom{D+4d}{D_1+d_1, D_2+d_2, D_3+d_3, D_4+d_4}$ possible rearrangements of these columns. We point out that, for a distribution where, for different indices j , we have p_j probability mass uniformly divided among n_j elements, its squared L_2 norm is $\sum_j n_j (\frac{p_j}{n_j})^2 = \sum_j \frac{p_j^2}{n_j}$. Thus bound the L_2 norm of our conditional distribution as

$$\|\mathcal{D}_{|d_1, d_2, d_3, d_4}\|_2 \leq \sqrt{\sum_{(D_1, D_2, D_3, D_4): \sum_i D_i = D} \frac{\left(2 \cdot 4^{-D} \binom{D}{D_1, D_2, D_3, D_4}\right)^2}{\binom{D+4d}{D_1+d_1, D_2+d_2, D_3+d_3, D_4+d_4}}}$$

This expression is exactly $2 \cdot 4^{-D}$ times the square root of the expression bounded in Lemma 5, if in Lemma 5 we reparameterize d as $4d$. Namely,

$$\|\mathcal{D}_{|d_1, d_2, d_3, d_4}\|_2 \leq 2 \cdot e^{\frac{2}{D} \sum_{i=1}^4 (d_i - d)^2} \cdot \left(1 + \frac{4d}{D}\right)^{3/4} \cdot 2^{-(4d+D)}$$

Combining this bound with Eq. (5) (the triangle inequality), we have

$$\|\mathcal{D}\|_2 \leq \mathbb{E}_{d_1, d_2, d_3, d_4} [\|\mathcal{D}_{|d_1, d_2, d_3, d_4}\|_2] \leq 2 \left(1 + \frac{4d}{D}\right)^{3/4} \cdot 2^{-(4d+D)} \mathbb{E}_{d_1, d_2, d_3, d_4} [e^{\frac{2}{D} \sum_{i=1}^4 (d_i - d)^2}]$$

The right hand side is exactly the form of Lemma 6, applied with $s = \frac{2}{D}$, thus yielding our overall bound of

$$\|\mathcal{D}\|_2 \leq 2 \left(1 + \frac{4d}{D}\right)^{3/4} \cdot 2^{-(4d+D)} \left(\frac{1}{1 - 48\frac{d}{D}}\right)^2$$

For $D \geq 100d$, we see that $\|\mathcal{D}\|_2 \leq 8 \cdot 2^{-(4d+D)}$, as desired. \blacksquare

The following lemma guarantees that activation functions which satisfy Assumption 3, can be simulated to arbitrary accuracy, using depth 2 threshold networks of polynomial width.

Lemma 8 *Let σ be an activation function that satisfies Assumption 3. Then for all $\delta > 0$, there exists a depth 2 threshold network \mathcal{N} of width at most $\frac{\text{poly}(R)}{\delta}$ and weights of magnitude at most $\frac{\text{poly}(R)}{\delta}$, such that*

$$\sup_{x \in [-R, R]} |\mathcal{N}(x) - \sigma(x)| \leq \delta.$$

Proof We will first construct a piecewise constant function that approximates σ to an L_∞ distance of δ , and then we will compute this piecewise constant function precisely using a depth 2 threshold network.

Let $x_1 := -R$ and $y_1 := \sigma(x_1)$. We define the set

$$A_1 := \{a \in [-R, R] : \forall x \leq a, |\sigma(x) - y_1| \leq \delta\}.$$

Note that $A_1 \neq \emptyset$ since $x_1 \in A_1$, and define $x'_1 = x_1$ and $x_2 := \sup A_1$. We now split our analysis into three cases, depending on the continuity properties of σ at x_2 .

1. Suppose that σ is continuous at x_2 from the right. By the definition of A_1 we have that $x_1 < x_2$, since if $x_1 = x_2$ by contradiction, due to right continuity, we can find an interval $[x_1, b)$ for $b > x_1$ close enough to x_1 , such that $|\sigma(x) - y_1| \leq \delta$ for all $x \in [x_1, b)$, which contradict the definition of A_1 . We define $y_2 := \sigma(x_2)$, and we have by the definition of A_1 that $\sup_{x \in [x_1, x_2)} |\sigma(x) - y_1| \leq \delta$. Moreover, if σ is continuous at x_2 from both sides then this implies that $|y_2 - y_1| = \delta$, and if it is only continuous from the right this implies that $|y_2 - y_1| \geq \delta$. This can be seen to hold true since if we had $|y_2 - y_1| < \delta$ by contradiction, then from right continuity there exists an interval $[x_2, b)$ for some $b > x_2$ such that $|\sigma(x) - y_1| < \delta$ for all $x \in [x_2, b)$, which contradicts the definition of A_1 . We can now define the set

$$A_2 := \{a \in [x_2, R] : \forall x \in [x_2, a], |\sigma(x) - y_2| \leq \delta\},$$

and note that $A_2 \neq \emptyset$ since $x_2 \in A_2$. Define $x'_2 = x_2$, we can conclude that in this case, we have an interval $[x_1, x_2)$ such that $|\sigma(x) - y_1| \leq \delta$ for all $x \in [x_1, x_2)$, that $|\sigma(x_2) - y_2| = 0 \leq \delta$, and that $|\sigma(x'_1) - \sigma(x'_2)| \geq \delta$.

2. Suppose that σ is not continuous at x_2 from the right, but it is either continuous at x_2 from the left or $x_1 = x_2$, which in both cases implies that $|\sigma(x_2) - y_1| \leq \delta$. It must also hold that $|\sigma(x) - y_1| \leq \delta$ for all $x \in [x_1, x_2]$. Define $y_2 := \lim_{x \rightarrow x_2+} \sigma(x)$, note that by the definition of A_1 and y_2 , we have similarly to the previous case that $|y_2 - y_1| \geq \delta$, and let x'_2 be close enough to x_2 from the right so that from right continuity we get $|y_2 - \sigma(x'_2)| \leq 0.25\delta$, and thus $|\sigma(x'_1) - \sigma(x'_2)| \geq 0.75\delta$. We can now define the set

$$A_2 := \{a \in (x_2, R] : \forall x \in (x_2, a], |\sigma(x) - y_2| \leq \delta\},$$

which is not empty since $(x_2, x'_2] \subseteq A_2$.

3. Suppose that $x_1 < x_2$ and that σ has a discontinuity at x_2 from both sides, and note that together with the previous two items, this covers all possibilities. Then, since σ is of bounded variation, we have that both limits at the two sides exist and are finite. Next, if $x_2 \in A_1$, then this implies that $|\sigma(x_2) - y_1| \leq \delta$. We can now define $y_2 := \lim_{x \rightarrow x_2+} \sigma(x)$ and proceed in the same manner as in Item 2 to define A_2 and x'_2 . Otherwise, we have that $x_2 \notin A_1$, which implies that $|\sigma(x_2) - y_1| > \delta$. We define $y_2 := \sigma(x_2)$ which trivially implies $|\sigma(x_2) - y_2| = 0 \leq \delta$, and proceed to define x'_2 and A_2 as in Item 1.

Define $x_3 := \sup A_2$, we can continue in this manner and define sequences of target values y_1, y_2, \dots and points $x_1, x_2, \dots, x'_1, x'_2, \dots$ such that for all i , we have

- $|\sigma(x) - y_i| \leq \delta$ for all $x \in (x_i, x_{i+1})$.
- $|\sigma(x_i) - y_{i-1}| \leq \delta$ or $|\sigma(x_i) - y_i| \leq \delta$.
- $|\sigma(x_i) - \sigma(x_{i+1})| \geq \delta$ and $|\sigma(x_i) - \sigma(x'_i)| \leq 0.25\delta$, which imply $|\sigma(x'_i) - \sigma(x'_{i+1})| \geq 0.5\delta$.

We now bound the length n of the sequences required to get $R \in A_n$; namely, the number of piecewise constant segments required to approximate σ to accuracy δ uniformly on $[-R, R]$. We have by Assumption 3 that σ has total variation at most $C_\sigma(1 + 2R)^{\alpha_\sigma}$. On the other hand, we have from the above properties that the partition x'_1, x'_2, \dots, x'_n of $[-R, R]$ has total variation at least

$$\sum_{i=1}^{n-1} |\sigma(x'_{i+1}) - \sigma(x'_i)| \geq 0.5(n-1)\delta.$$

Combining these two inequalities, we obtain $0.5(n-1)\delta \leq C_\sigma(1 + 2R)^{\alpha_\sigma}$, implying $n = \frac{\text{poly}(R)}{\delta}$.

It now only remains to approximate a piecewise linear function, with $\frac{\text{poly}(R)}{\delta}$ constant segments, using a threshold network with a similar number of neurons. Define $y_0 = 0$, it is easy to verify that this approximation is given by the expression

$$\sum_{i=1}^n \xi_i (y_i - y_{i-1}) \sigma_{\text{thresh}}(\xi_i(x - x_i) - 0.5) - 0.5(\xi_i - 1)(y_i - y_{i-1}),$$

where $\xi_i \in \{-1, 1\}$ is chosen according to the discontinuity of our piecewise constant function at the point x_i . Namely, by setting $w_i := \xi_i$, $b_i := -\xi_i x_i - 0.5$, $v_i := \xi_i(y_i - y_{i-1})$ and $b_0 := -0.5 \sum_{i=1}^n (\xi_i - 1)(y_i - y_{i-1})$, we obtain a width n threshold network

$$\mathcal{N}(x) := \sum_{i=1}^n v_i \sigma_{\text{thresh}}(w_i x + b_i) + b_0,$$

satisfying

$$\sup_{x \in [-R, R]} |\sigma(x) - \mathcal{N}(x)| \leq \delta.$$

Lastly, by Assumption 3, we have that $|y_i| \leq \text{poly}(R)$ for all $i \in [n]$, implying that \mathcal{N} has weights of magnitude at most $\frac{\text{poly}(R)}{\delta}$. \blacksquare

The following proposition guarantees that functions on the Boolean hypercube, computed by depth 2 networks, which employ activation functions that satisfy Assumption 3, can be simulated to arbitrary accuracy using depth 2 threshold networks of width polynomial in the size of the network.

Proposition 9 *Let $\delta > 0$, suppose that σ satisfies Assumption 3, and let $f : \{0, 1\}^d \rightarrow \mathbb{R}$ be a function computed by a depth 2, width m σ -network, with weights bounded by C . Then there exists a depth 2 threshold network \mathcal{N} of width $\frac{m^2 \text{poly}(d, C)}{\delta}$ and weights bounded in magnitude by $\frac{\text{poly}(d, C)}{\delta}$, such that*

$$\max_{\mathbf{x} \in \{0, 1\}^d} |f(\mathbf{x}) - \mathcal{N}(\mathbf{x})| \leq \delta.$$

Proof Let

$$f(\mathbf{x}) := \sum_{i=1}^m v_i \sigma(\langle \mathbf{w}_i, \mathbf{x} \rangle + b_i) + b_0$$

be the function computed by the network f . We first observe that by our weight boundedness assumption, we have for all $i \in [m]$ that $|\langle \mathbf{w}_i, \mathbf{x} \rangle + b_i| \leq \|\mathbf{w}_i\| \|\mathbf{x}\| + C \leq (d+1)C$. We now use Lemma 8 to approximate each σ -neuron in f to accuracy $\frac{\delta}{mC}$, and obtain m depth 2, width $\frac{m \text{poly}(d, C)}{\delta}$ threshold networks $\mathcal{N}_1, \mathcal{N}_2, \dots, \mathcal{N}_m$, such that

$$|\mathcal{N}_i(\mathbf{x}) - \sigma(\langle \mathbf{w}_i, \mathbf{x} \rangle + b_i)| \leq \frac{\delta}{mC}$$

for all $i \in [m]$. Define the network \mathcal{N} given by

$$\mathcal{N}(\mathbf{x}) := \sum_{i=1}^m v_i \mathcal{N}_i(\mathbf{x}) + b_0,$$

fix any $\mathbf{x} \in \{0, 1\}^d$ and compute

$$|f(\mathbf{x}) - \mathcal{N}(\mathbf{x})| \leq \sum_{i=1}^m v_i |\sigma(\langle \mathbf{w}_i, \mathbf{x} \rangle + b_i) - \mathcal{N}_i(\mathbf{x})| \leq \sum_{i=1}^m C \frac{\delta}{mC} = \delta.$$

Lastly, since \mathcal{N} is a linear combination of depth 2 neural networks, each of which is of width at most $\frac{m \text{poly}(d, C)}{\delta}$ and weights of magnitude $\frac{\text{poly}(d, C)}{\delta}$, we have that \mathcal{N} is in itself a depth 2, width $\frac{m^2 \text{poly}(d, C)}{\delta}$ threshold network, with weights bounded by $\frac{\text{poly}(d, C)}{\delta}$ as required. \blacksquare

The following theorem, which may be of independent interest, establishes a reduction for hard-to-approximate functions on the Boolean hypercube to the unit ball.

Theorem 10 Suppose that σ satisfies Assumption 3, let $\beta := \max\{1, \alpha_\sigma\}$, let $\mathbf{x}_1, \dots, \mathbf{x}_{2^{2d}} \in \{0, 1\}^{2d}$ be an enumeration of the Boolean hypercube, and let $\mathcal{F}_{d,C,\beta}$ denote the function class of depth 2 σ -networks $\mathbb{R}^d \rightarrow \mathbb{R}$ with weights bounded by $C \leq 2^{\frac{d}{80\beta+80}}$ and with width at most $2^{\frac{d}{80\beta}}$. Further assume that $g : \{0, 1\}^{2d} \rightarrow \{0, 1\}$ is a Boolean function satisfying

$$\inf_{\mathcal{N} \in \mathcal{F}_{2d,3dC,\beta}} \mathbb{E}_{\mathbf{x} \sim \mathcal{U}(\{0,1\}^{2d})} \left[(\mathcal{N}(\mathbf{x}) - g(\mathbf{x}))^2 \right] > c + \varepsilon,$$

for some constant $c > 0$ and $\varepsilon > 0$. Then, there exists a set $\mathbf{z}_1, \dots, \mathbf{z}_{2^{2d}} \in \mathbb{R}^{2d}$ such that the set $\mathcal{S}_d := \left(\bigcup_{i=1}^{2^{2d}} \left\{ \left(\mathbf{z}_i, \frac{1}{4\sqrt{d}} \mathbf{x}_i \right) \right\} \right)$ satisfies $\mathcal{S}_d \subseteq B_{0.9}(\mathbf{0})$, every pair of points in \mathcal{S}_d is at distance at least 0.4 apart, and it holds that

$$\inf_{\mathcal{N} \in \mathcal{F}_{4d,C,\beta}} \mathbb{E}_{\left(\mathbf{z}, \frac{1}{4\sqrt{d}} \mathbf{x} \right) \sim \mathcal{U}(\mathcal{S}_d)} \left[\left(\mathcal{N} \left(\mathbf{z}, \frac{1}{4\sqrt{d}} \mathbf{x} \right) - g(\mathbf{x}) \right)^2 \right] > c.$$

Proof Let $\mathbf{z}_1, \dots, \mathbf{z}_{2^{2d}} \in B_{0.8}^{2d}(\mathbf{0})$ be a set of points such that $\min_{i \neq j} \|\mathbf{z}_i - \mathbf{z}_j\|_2 > 0.4$. To show that such a set exists, we use Jenssen et al. (2018, Eq. (2)) which guarantees that a set $\mathbf{z}'_1, \mathbf{z}'_2, \dots$ such that $\|\mathbf{z}'_i\| = 1$ for all i , which also satisfies $\langle \mathbf{z}'_i, \mathbf{z}'_j \rangle \leq \cos(\theta)$ for all $i \neq j$ and some θ , can be of size at least

$$(1 + o(1)) \sqrt{2\pi d} \cdot \frac{\cos \theta}{\sin^{d-1}(\theta)}.$$

Plugging $\theta = \pi/6$ in the above, it is at least 2^d for all $d \geq 1$. Moreover, we have

$$\|\mathbf{z}'_i - \mathbf{z}'_j\|^2 = \|\mathbf{z}'_i\|^2 + \|\mathbf{z}'_j\|^2 - 2\langle \mathbf{z}'_i, \mathbf{z}'_j \rangle \geq 2 - 2\cos(\pi/6) \geq 0.25,$$

where taking the square root of the above inequality and scaling all \mathbf{z}'_i in $2d$ dimensions by a factor of 0.8 gives the set $\mathbf{z}_1, \dots, \mathbf{z}_{2^{2d}}$.

Next, we have that $\|(\mathbf{z}_i, 0.08\mathbf{x}_j)\| \leq \sqrt{0.8^2 + 2 \cdot 0.08^2} \leq 0.9$ for all i and j , and that the minimal distance between any two points $(\mathbf{z}_{i_1}, 0.08\mathbf{x}_{j_1})$ and $(\mathbf{z}_{i_2}, 0.08\mathbf{x}_{j_2})$ for $i_1 \neq i_2$ is at least 0.4.

For a neural network \mathcal{N} , define the loss function $\ell_{\mathcal{N}}(\mathbf{z}, 0.08\mathbf{x}) := (\mathcal{N}(\mathbf{z}, 0.08\mathbf{x}) - g(\mathbf{x}))^2$. We now bound the loss over all instances in our domain and neural networks in our function class as follows. Starting with the output of a single neuron, we have by Assumption 3 that for any weight vector \mathbf{w} , bias term b , and input $\mathbf{x} \in \mathbb{R}^{4d}$ such that $\max_i |x_i| \leq 1$, that the following upper bound holds

$$|\sigma(\langle \mathbf{w}, \mathbf{x} \rangle + b)| \leq C_\sigma (1 + \|\mathbf{w}\|_\infty \cdot \|\mathbf{x}\|_1 + C)^{\alpha_\sigma} = \mathcal{O}(C^{\alpha_\sigma} d^{\alpha_\sigma}).$$

So the output of any function in $\mathcal{F}_{4d,C,\beta}$ is at most $\mathcal{O}(wC^{\alpha_\sigma+1}d^{\alpha_\sigma})$, and therefore

$$\ell_{\mathcal{N}}(\mathbf{z}, 0.08\mathbf{x}) = \mathcal{O}(w^2 C^{2\alpha_\sigma+2} d^{2\alpha_\sigma}) \quad (6)$$

for all $\mathcal{N} \in \mathcal{F}_{4d,C,\beta}$, $\mathbf{z} \in B_{0.8}^{2d}$ and $\mathbf{x} \in \{0, 1\}^{2d}$.

We define a sample size $n = 2^{2d/3}$, and consider the process of drawing a set of size n comprised of instances of the form $(\mathbf{z}_i, \mathbf{x}_j)$, where $i, j \sim \mathcal{U}([2^{2d}])$ are independent. Denote

$\mathcal{N}_S := \operatorname{argmin}_{\mathcal{N} \in \mathcal{F}_{4d,C,\beta}} \mathcal{L}_S(\mathcal{N})$ and $\mathcal{N}^* := \operatorname{argmin}_{\mathcal{N} \in \mathcal{F}_{4d,C,\beta}} \mathcal{L}_{\mathcal{D}}(\mathcal{N})$, where \mathcal{L}_S and $\mathcal{L}_{\mathcal{D}}$ are the empirical loss and risk, respectively, over the distribution \mathcal{D} defined by our process. We now have that

$$\mathcal{L}_{\mathcal{D}}(\mathcal{N}^*) - \mathcal{L}_S(\mathcal{N}_S) = \mathcal{L}_{\mathcal{D}}(\mathcal{N}^*) - \mathcal{L}_{\mathcal{D}}(\mathcal{N}_S) + \mathcal{L}_{\mathcal{D}}(\mathcal{N}_S) - \mathcal{L}_S(\mathcal{N}_S) \leq \mathcal{L}_{\mathcal{D}}(\mathcal{N}_S) - \mathcal{L}_S(\mathcal{N}_S),$$

where the inequality follows from the definition of \mathcal{N}^* . By Eq. (6) and a standard Rademacher complexity argument, we have with probability at least $1 - \delta$ that the above is at most

$$2\mathcal{R}_n(\mathcal{F}_{4d,C,\beta}) + 4\mathcal{O}(w^2 C^{2\alpha_\sigma+2} d^{2\alpha_\sigma}) \sqrt{\frac{2 \ln(2/\delta)}{n}} \quad (7)$$

(Shalev-Shwartz and Ben-David, 2014, Thm. 26.5). Recall that for all $\mathcal{N} \in \mathcal{F}_{4d,C,\beta}$ we have $w \leq 2^{\frac{d}{20\beta}} \leq 2^{\frac{d}{20}}$ and $C \leq 2^{\frac{d}{20\beta+20}} \leq 2^{\frac{d}{20\alpha_\sigma+20}}$, we can thus upper bound the second summand as follows

$$\mathcal{O}(w^2 C^{2\alpha_\sigma+2} d^{2\alpha_\sigma}) \sqrt{\frac{2 \ln(2/\delta)}{n}} = \mathcal{O}\left(2^{0.1d} \cdot 2^{0.1d} d^{2\alpha_\sigma}\right) \sqrt{\frac{\ln(2/\delta)}{2^{2d/3}}} = \mathcal{O}(\exp(-\Omega(d))) \sqrt{\ln(2/\delta)}. \quad (8)$$

Using Assumption 3 and the facts that $w \leq 2^{\frac{d}{20\beta}} \leq 2^{\frac{d}{20\alpha_\sigma}}$ and $C \leq 2^{\frac{d}{20\beta+20}} \leq 2^{\frac{d}{20\alpha_\sigma+20}} \leq 2^{\frac{d}{20\alpha_\sigma}}$, the Rademacher complexity term can also be upper bounded,

$$\mathcal{R}_n(\mathcal{F}_{4d,C,\beta}) \leq \frac{(w \cdot C \cdot 4d)^{\alpha_\sigma}}{\sqrt{n}} \leq 2^{-d/3} \cdot 2^{0.1d} (4d)^{\alpha_\sigma} = \mathcal{O}(\exp(-\Omega(d))).$$

Plugging the above and Eq. (8) in Eq. (7), rearranging, and using the fact that $\delta \leq 1$, we arrive at

$$\mathcal{L}_{\mathcal{D}}(\mathcal{N}^*) \leq \mathcal{L}_S(\mathcal{N}_S) + \mathcal{O}(\exp(-\Omega(d))) \sqrt{\ln(2/\delta)}.$$

Conditioning the above bound on the event where the sample S consists of distinct choices of i and j , which happens with probability at least $1 - \Omega(2^{-2d/3}) \geq 0.5$, we have that the following bound holds for such distinct choices with probability at least $1 - \delta$

$$\mathcal{L}_{\mathcal{D}}(\mathcal{N}^*) \leq \mathcal{L}_S(\mathcal{N}_S) + \mathcal{O}(\exp(-\Omega(d))) \sqrt{\ln(4/\delta)}.$$

We now describe a different process for sampling sets S that is equivalent to the above conditional distribution. Sample a random permutation of the numbers $[2^{2d}]$, and sample a random permutation of the 2^{2d} points in the set $\{0, 1\}^{2d}$. Next, pair up integers and points; and then let P_j be the j^{th} such point. We now produce $2^{4d/3}$ different samples S_j of size $2^{2d/3}$ by defining the k^{th} sample to consist of points $P_{(k-1)2^{2d/3} : k2^{2d/3}-1}$. For any given hypothesis function \mathcal{N} , the average error over points in P equals the average error over these $2^{4d/3}$ sample sets of the average error in the sample; and thus the minimum error over any $\mathcal{N} \in \mathcal{F}_{4d,C,\beta}$ for P is at least the average over these $2^{4d/3}$ sample sets of the minimum error over that sample set. Denoting the resulting concatenated set of $2^{4d/3}$ samples by \mathcal{S}_d , and its empirical minimizer by $\mathcal{N}_{\mathcal{S}_d}$, we have that

$$\frac{1}{2^{4d/3}} \sum_{j=1}^{2^{4d/3}} \mathcal{L}_{S_j}(\mathcal{N}_{S_j}) \leq \mathcal{L}_{\mathcal{S}_d}(\mathcal{N}_{\mathcal{S}_d}).$$

Thus, using a union bound to bound the probability that any of these $2^{4d/3}$ sample sets is in the “bad” fraction of sample sets where i and j are not distinct, we have with probability at least $1 - \delta$ that

$$\min_{\mathcal{N} \in \mathcal{F}_{4d,C,\beta}} \mathbb{E}_{i,j \sim \mathcal{U}([2^{2d}])} \left[(\mathcal{N}(\mathbf{z}_i, \mathbf{x}_j) - g(\mathbf{x}_j))^2 \right] = \mathcal{L}_{\mathcal{D}}(\mathcal{N}^*) \leq \mathcal{L}_{\mathcal{S}_d}(\mathcal{N}_{\mathcal{S}_d}) + \mathcal{O}(\exp(-\Omega(d))) \sqrt{\ln \left(\frac{2^{4d/3+2}}{\delta} \right)}. \quad (9)$$

Focusing on the left hand side first, we observe that since i and j are independent, there must exist some $i' \in [2^{2d}]$ for which the expectation term is smaller than the above, namely

$$\min_{\mathcal{N} \in \mathcal{F}_{4d,C,\beta}} \mathbb{E}_{j \sim \mathcal{U}([2^{2d}])} \left[(\mathcal{N}(\mathbf{z}_{i'}, \mathbf{x}_j) - g(\mathbf{x}_j))^2 \right] \leq \min_{\mathcal{N} \in \mathcal{F}_{4d,C,\beta}} \mathbb{E}_{i,j \sim \mathcal{U}([2^{2d}])} \left[(\mathcal{N}(\mathbf{z}_i, \mathbf{x}_j) - g(\mathbf{x}_j))^2 \right].$$

However, since $\mathbf{z}_{i'}$ is fixed, for any $\mathcal{N} \in \mathcal{F}_{4d,C,\beta}$, we can construct a network $\mathcal{N}' \in \mathcal{F}_{2d,3dC,\beta}$ such that $\mathcal{N}(\mathbf{z}_{i'}, \mathbf{x}_j) = \mathcal{N}'(\mathbf{x}_j)$. This holds true since the first hidden layer of \mathcal{N}' can absorb the linear combination obtained by multiplying $\mathbf{z}_{i'}$ with the corresponding weight vectors into the bias terms, which will increase their magnitudes by at most $2dC$ for a total less than $3dC$. Thus, we have

$$\min_{\mathcal{N} \in \mathcal{F}_{2d,3dC,\beta}} \mathbb{E}_{j \sim \mathcal{U}([2^{2d}])} \left[(\mathcal{N}(\mathbf{x}_j) - g(\mathbf{x}_j))^2 \right] \leq \min_{\mathcal{N} \in \mathcal{F}_{4d,C,\beta}} \mathbb{E}_{j \sim \mathcal{U}([2^{2d}])} \left[(\mathcal{N}(\mathbf{z}_{i'}, \mathbf{x}_j) - g(\mathbf{x}_j))^2 \right].$$

By our theorem assumption, the above left hand side is at least $c + \varepsilon$, so by plugging this back in Eq. (9) we arrive at

$$c + \varepsilon < \mathcal{L}_{\mathcal{S}_d}(\mathcal{N}_{\mathcal{S}_d}) + \mathcal{O}(\exp(-\Omega(d))) \sqrt{\ln \left(\frac{2^{4d/3+2}}{\delta} \right)}.$$

Finally, taking $\delta = 0.5$ for example,¹ we have by the probabilistic method that there exists some realization of the matchings between \mathbf{z}_i ’s and \mathbf{x}_j ’s, and hence a set \mathcal{S}_d , such that for sufficiently large d , it holds that

$$c + \varepsilon < \mathcal{L}_{\mathcal{S}_d}(\mathcal{N}_{\mathcal{S}_d}) + \varepsilon,$$

from which the theorem follows. ■

With the above auxiliary results at hand, we are finally ready to prove the theorem.

Proof [Proof of Thm. 3] First, for any natural d and real $C > 0$, define $w(d, C)$ to be the minimal width required for a depth 2 σ -network with weights bounded in magnitude by C to approximate f_d to accuracy at most $\frac{1}{400}$. Our goal is therefore to derive a lower bound on $w(d, C)$.

Let $D := 100d$, and let $\mathcal{S}_{4D+16d} \subseteq B_{0.9}^{4D+16d}(\mathbf{0})$ be some set to be defined later, with the property that any two points in this set are at a distance of at least 0.4 apart (essentially, this will be the set whose existence is established by Thm. 10). Recall the shorthand $\mathcal{C}_{4d} := \left[0, \frac{1}{12\sqrt{d}}\right]^{4d}$, we define

$$\mathcal{A}_{4D+16d} := \mathcal{S}_{4D+16d} + \mathcal{C}_{4D+16d}.$$

1. We remark that this argument will work even for δ which decays doubly exponentially to zero with d since we have a logarithmic term multiplying an exponentially decaying term, which indicates that all matchings between \mathbf{z}_i ’s and \mathbf{x}_j ’s, except for a minuscule fraction, satisfy our requirements.

Observe that due to $\|\mathbf{z}\| \leq 0.9$ for all $\mathbf{z} \in \mathcal{S}_{4D+16d}$ and $\|\mathbf{c}\| \leq 1/36$ for all $\mathbf{c} \in \mathcal{C}_{4D+16d}$, it holds that $\|\mathbf{x}\| \leq 1$ for all $\mathbf{x} \in \mathcal{A}_{4D+16d}$. Moreover, the minimal distance between any two points in \mathcal{A}_{4D+16d} is at least the minimal distance between two points in \mathcal{S}_{4D+16d} minus the diameter of \mathcal{C}_{4D+16d} , i.e. $1/36$. Thus, the minimal distance between any two points in \mathcal{A}_{4D+16d} is constant.

We first assume that we are given a depth 2, width $w(D + 4d, C)$ σ -network \mathcal{N} , with weights bounded by C , that satisfies

$$\mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim \mathcal{U}(\mathcal{A}_{4D+16d})} \left[(\mathcal{N}(\mathbf{x}, \mathbf{y}) - f_{D+4d}(\mathbf{x}, \mathbf{y}))^2 \right] \leq \frac{1}{400},$$

and we will show that this implies the existence of a σ -network of a similar size and with a similar magnitude of the weights, which gives a similar accuracy when approximating f_{D+4d} uniformly over the discrete domain \mathcal{S}_{4D+16d} .

Using the law of total expectation, we can break the above expectation into two iterated expectations as follows

$$\mathbb{E}_{\mathbf{c} \sim \mathcal{U}(\mathcal{C}_{4D+16d})} \left[\mathbb{E}_{\mathbf{z} \sim \mathcal{U}(\mathcal{S}_{4D+16d})} \left[(\mathcal{N}(\mathbf{z} + \mathbf{c}) - f_{D+4d}(\mathbf{z} + \mathbf{c}))^2 \right] \right] \leq \frac{1}{400}.$$

For $\mathbf{c} \sim \mathcal{U}(\mathcal{C}_{4D+16d})$, we can define the non-negative random variable

$$X_{\mathbf{c}} := \mathbb{E}_{\mathbf{z} \sim \mathcal{U}(\mathcal{S}_{4D+16d})} \left[(\mathcal{N}(\mathbf{z} + \mathbf{c}) - f_{D+4d}(\mathbf{z} + \mathbf{c}))^2 \right].$$

This allows us to rewrite the former inequality more compactly as

$$\mathbb{E}_{\mathbf{c} \sim \mathcal{U}(\mathcal{C}_{4D+16d})} [X_{\mathbf{c}}] \leq \frac{1}{400}. \quad (10)$$

Using Markov's inequality on $X_{\mathbf{c}}$, we have

$$\mathbb{P}_{\mathbf{c} \sim \mathcal{U}(\mathcal{C}_{4D+16d})} \left[X_{\mathbf{c}} < \frac{400}{399} \mathbb{E}_{\mathbf{c} \sim \mathcal{U}(\mathcal{C}_{4D+16d})} [X_{\mathbf{c}}] \right] \geq 1 - \frac{399}{400} > 0.$$

Namely, by virtue of Eq. (10) there must exist some $\mathbf{c} \in \mathcal{C}_{4D+16d}$ such that

$$\mathbb{E}_{\mathbf{z} \sim \mathcal{U}(\mathcal{S}_{4D+16d})} \left[(\mathcal{N}(\mathbf{z} + \mathbf{c}) - f_{D+4d}(\mathbf{z} + \mathbf{c}))^2 \right] = X_{\mathbf{c}} < \frac{400}{399} \mathbb{E}_{\mathbf{c} \sim \mathcal{U}(\mathcal{C}_{4D+16d})} [X_{\mathbf{c}}] \leq \frac{1}{399}. \quad (11)$$

Next, we have by the definition of f that $f_{D+4d}(\mathbf{z} + \mathbf{c}) = f_{D+4d}(\mathbf{z})$ for all $\mathbf{c} \in \mathcal{C}_{4D+16d}$ and all $\mathbf{z} \in \mathcal{S}_{4D+16d}$. Moreover, there exists a σ -network $\tilde{\mathcal{N}}$ of the same depth and width as \mathcal{N} , such that $\mathcal{N}(\mathbf{z} + \mathbf{c}) = \tilde{\mathcal{N}}(\mathbf{z})$ for all $\mathbf{c} \in \mathcal{C}_{4D+16d}$ and all $\mathbf{z} \in \mathcal{S}_{4D+16d}$. This holds true since shifting the input by a constant vector \mathbf{c} merely shifts the biases in the first hidden layer by the same vector. Since this operation is linear, it can be absorbed in the hidden layer without changing the architecture of $\tilde{\mathcal{N}}$, and where the magnitude of the weights is now at most $4dC$.

The above observations, the definition of $X_{\mathbf{c}}$, and Eq. (11), imply the existence of some $\mathbf{c} \in \mathcal{C}_{4D+16d}$ that satisfies

$$\mathbb{E}_{\mathbf{x}, \mathbf{y} \sim \mathcal{U}(\mathcal{S}_{4D+16d})} \left[\left(\tilde{\mathcal{N}}(\mathbf{z}) - f_{D+4d}(\mathbf{z}) \right)^2 \right] = \mathbb{E}_{\mathbf{z} \sim \mathcal{U}(\mathcal{S}_{4D+16d})} \left[(\mathcal{N}(\mathbf{z} + \mathbf{c}) - f_{D+4d}(\mathbf{z} + \mathbf{c}))^2 \right] \leq \frac{1}{399}.$$

Next, letting $\beta = \{1, \alpha_\sigma\}$ where α_σ is the constant guaranteed by Assumption 3, we assume that the magnitude of the weights of $\tilde{\mathcal{N}}$, $4dC$, is at most $2^{\frac{d}{80\beta+80}}$. This is justified, since otherwise we would have that $C = \Omega(\exp(\Omega(d)))$, in which case the lower bound we are proving becomes trivial for sufficiently small constants hidden in the asymptotic notation. Similarly, we assume that the width of $\tilde{\mathcal{N}}$, $w(D + 4d, 4dC)$, is at most $2^{\frac{d}{80\beta}}$, since otherwise the lower bound we are proving already holds. For the $\text{IP}_d(\cdot, \cdot)$ function, $c := \frac{1}{399}$ and $\varepsilon := \frac{1}{398} - c$, we have by (the contrapositive of) Thm. 10 that

$$\inf_{\mathcal{N} \in \mathcal{F}_{2d, 3dC, \beta}} \mathbb{E}_{\mathbf{x}, \mathbf{y} \sim \mathcal{U}(\{0,1\}^d)} \left[(\mathcal{N}(\mathbf{x}, \mathbf{y}) - \text{IP}_d(\mathbf{x}, \mathbf{y}))^2 \right] \leq c + \varepsilon = \frac{1}{398}.$$

Namely, there exists some $\mathcal{N}' : \mathbb{R}^{2d} \rightarrow \mathbb{R}$ such that

$$\mathbb{E}_{\mathbf{x}, \mathbf{y} \sim \mathcal{U}(\{0,1\}^d)} \left[(\mathcal{N}'(\mathbf{x}, \mathbf{y}) - \text{IP}_d(\mathbf{x}, \mathbf{y}))^2 \right] \leq \frac{1}{398}. \quad (12)$$

Let $n \in \mathbb{N}$ to be determined later, we now construct a neural network $\mathcal{N}'' : \mathbb{R}^{D+4d} \rightarrow \mathbb{R}$ that will achieve a small margin on all the inputs $\mathbf{x}, \mathbf{y} \in \{0,1\}^d$ simultaneously. The network will consist of n blocks, where each of which has the architecture of \mathcal{N}' , and is re-randomized using the following process for every $j \in [n]$:

- We sample two binary vectors $\mathbf{x}'_j, \mathbf{y}'_j \in \{0,1\}^d$ uniformly at random.
- We sample two binary vectors $\mathbf{x}''_j, \mathbf{y}''_j \in \{0,1\}^D$ uniformly at random, until the number of pairs $(x_{j,i}, y_{j,i})$ that are both one is an even number for every $j \in [n]$.
- We concatenate our sample into two binary vectors $(\mathbf{x} + \mathbf{x}', \mathbf{x}' + \mathbf{x}', \mathbf{x}' + \mathbf{x}'')$ and $(\mathbf{y} + \mathbf{y}', \mathbf{y}' + \mathbf{y}', \mathbf{y}' + \mathbf{y}'')$.
- We sample a permutation $\tau_j : \mathbb{R}^{D+4d} \rightarrow \mathbb{R}$ uniformly at random, and use this permutation to permute the previous two binary vectors, denoting the results as $\hat{\mathbf{x}}_j$ and $\hat{\mathbf{y}}_j$, respectively.
- We modify each \mathcal{N}'_j to simulate their computation on the inputs $\hat{\mathbf{x}}_j, \hat{\mathbf{y}}_j$. For the coordinates receiving $\mathbf{x} + \mathbf{x}'$ or $\mathbf{y} + \mathbf{y}'$ as input, this can be done by keeping the weights in the hidden layer unchanged in the case where their corresponding coordinate in \mathbf{x}' was drawn as 0, and by composing the weights with the transformation $x_i \mapsto 1 - x_i$ whenever $x'_i = 1$. Since this is a linear transformation of the input, it can be absorbed into the weights of the first hidden layer without changing the architecture of \mathcal{N}'_j . Lastly, multiplying the weights of the hidden layer with the permutation matrix which corresponds to the sampled permutation τ_j , which is also a linear transformation, also allows us to keep the architecture of \mathcal{N}'_j unchanged.

We now turn to bound the approximation error in absolute value of the network \mathcal{N}'' when approximating IP_{D+4d} . Fix some $\mathbf{x}, \mathbf{y} \in \{0,1\}^d$, and let \mathcal{D} denote the distribution over the set $\{0,1\}^{2(D+4d)}$ which is induced by the randomness in picking $\mathbf{x}'_j, \mathbf{y}'_j \in \{0,1\}^d$, $\mathbf{x}''_j, \mathbf{y}''_j \in \{0,1\}^D$ and a uniformly chosen permutation τ_j of $[D + 4d]$, as described in the above random construction

of \mathcal{N}'_j . Cauchy-Schwarz, combined with Eq. (12) and Proposition 7's bound on $\|\mathcal{D}\|_2$ yields

$$\begin{aligned} \mathbb{E}_{(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \sim \mathcal{D}} [|\mathcal{N}'(\hat{\mathbf{x}}, \hat{\mathbf{y}}) - \text{IP}_{D+4d}(\hat{\mathbf{x}}, \hat{\mathbf{y}})|] &= \sum_{\hat{\mathbf{x}}, \hat{\mathbf{y}} \in \{0,1\}^{D+4d}} \frac{\mathbb{P}[X = \hat{\mathbf{x}}, Y = \hat{\mathbf{y}}]}{\mathcal{D}} \cdot |\mathcal{N}'(\hat{\mathbf{x}}, \hat{\mathbf{y}}) - \text{IP}_{D+4d}(\hat{\mathbf{x}}, \hat{\mathbf{y}})| \\ &\leq \|\mathcal{D}\|_2 \sqrt{\sum_{\hat{\mathbf{x}}, \hat{\mathbf{y}} \in \{0,1\}^{D+4d}} (\mathcal{N}'(\hat{\mathbf{x}}, \hat{\mathbf{y}}) - \text{IP}_{D+4d}(\hat{\mathbf{x}}, \hat{\mathbf{y}}))^2} \\ &\leq 8\sqrt{\frac{1}{398}} < 0.41. \end{aligned} \quad (13)$$

Next, we formally define \mathcal{N}'' as the network

$$\mathcal{N}''(\mathbf{x}, \mathbf{y}) := \frac{1}{n} \sum_{j=1}^n \mathcal{N}'_j(\hat{\mathbf{x}}_j, \hat{\mathbf{y}}_j).$$

Note that since \mathcal{N}'' is a linear combination of depth 2 networks, it is in itself a depth 2 network. Moreover, we remark that this random process always preserves the value of the inner product. To see this, fix any $\mathbf{x}', \mathbf{y}', \mathbf{x}'', \mathbf{y}'', \tau$ chosen according to the above random process. Then by taking equalities that are mod 2, we have

$$\begin{aligned} \text{IP}_{D+4d}(\tau(\hat{\mathbf{x}}), \tau(\hat{\mathbf{y}})) &= \text{IP}_{D+4d}(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \\ &= \text{IP}_d(\mathbf{x} + \mathbf{x}', \mathbf{y} + \mathbf{y}') + \text{IP}_d(\mathbf{x}', \mathbf{y}') + \text{IP}_d(\mathbf{x} + \mathbf{x}', \mathbf{y}') + \text{IP}_d(\mathbf{x}', \mathbf{y} + \mathbf{y}') + \text{IP}_D(\mathbf{x}'', \mathbf{y}'') \\ &= \text{IP}_d(\mathbf{x} + \mathbf{x}', \mathbf{y} + \mathbf{y}') + \text{IP}_d(\mathbf{x}', \mathbf{y}') + \text{IP}_d(\mathbf{x} + \mathbf{x}', \mathbf{y}') + \text{IP}_d(\mathbf{x}', \mathbf{y} + \mathbf{y}') \\ &= \text{IP}_d(\mathbf{x}, \mathbf{y} + \mathbf{y}') + \text{IP}_d(\mathbf{x}, \mathbf{y}') = \text{IP}_d(\mathbf{x}, \mathbf{y}). \end{aligned}$$

where the first equality follows from the fact that permutations preserve sums, the second equality is by the definition of our construction of $\hat{\mathbf{x}}, \hat{\mathbf{y}}$, the third equality is due to \mathbf{x}'' and \mathbf{y}'' always having an even number of pairs $x_i = 1, y_i = 1$ which implies $\text{IP}_D(\mathbf{x}'', \mathbf{y}'') = 0$, and the last two equalities follow from basic properties of the inner product mod 2. Thus, we have

$$\text{IP}_d(\mathbf{x}, \mathbf{y}) = \text{IP}_{D+4d}(\hat{\mathbf{x}}_j, \hat{\mathbf{y}}_j)$$

for all $\mathbf{x}, \mathbf{y} \in \{0, 1\}^d$ and all $j \in [n]$. Having defined \mathcal{N}'' , we now bound the approximation error in absolute value which it achieves on an arbitrary input $\mathbf{x}, \mathbf{y} \in \{0, 1\}^d$, over the randomness induced by the previously described process. Compute

$$\begin{aligned} |\mathcal{N}''(\mathbf{x}, \mathbf{y}) - \text{IP}_d(\mathbf{x}, \mathbf{y})| &= \left| \frac{1}{n} \sum_{j=1}^n \mathcal{N}'_j(\hat{\mathbf{x}}_j, \hat{\mathbf{y}}_j) - \mathbb{E}[\mathcal{N}'_1(\hat{\mathbf{x}}_1, \hat{\mathbf{y}}_1)] + \mathbb{E}[\mathcal{N}'_1(\hat{\mathbf{x}}_1, \hat{\mathbf{y}}_1)] - \text{IP}_{D+4d}(\hat{\mathbf{x}}_1, \hat{\mathbf{y}}_1) \right| \\ &\leq \left| \frac{1}{n} \sum_{j=1}^n \mathcal{N}'_j(\hat{\mathbf{x}}_j, \hat{\mathbf{y}}_j) - \mathbb{E}[\mathcal{N}'_1(\hat{\mathbf{x}}_1, \hat{\mathbf{y}}_1)] \right| + \mathbb{E}[|\mathcal{N}'_1(\hat{\mathbf{x}}_1, \hat{\mathbf{y}}_1) - \text{IP}_{D+4d}(\hat{\mathbf{x}}_1, \hat{\mathbf{y}}_1)|], \end{aligned}$$

where the expectation is taken over the randomness in sampling $(\hat{\mathbf{x}}_j, \hat{\mathbf{y}}_j)$ from \mathcal{D} . Using Eq. (13), we can upper bound the above by

$$\left| \frac{1}{n} \sum_{j=1}^n \mathcal{N}'_j(\hat{\mathbf{x}}_j, \hat{\mathbf{y}}_j) - \mathbb{E}[\mathcal{N}'_1(\hat{\mathbf{x}}_1, \hat{\mathbf{y}}_1)] \right| + 0.41. \quad (14)$$

Next, we will use Hoeffding's inequality to upper bound the absolute value term above, but first we will derive an upper bound on the magnitude of the output of the network \mathcal{N}'_j . We have that each neuron in \mathcal{N}'_j has weights of magnitude at most $2C$, therefore, from Assumption 3 we get that the output of each hidden neuron is upper bounded by $\text{poly}(d, C)$ over the domain $\{0, 1\}^{D+4d}$. This implies that the output of the output neuron is at most $B := \text{poly}(d, C)w(D + 4d, C)$, since the width of \mathcal{N}'_j is $w(D + 4d, C)$ and its output neuron has weights bounded by C . We now use this bound, and the fact that $\mathcal{N}'_j(\hat{\mathbf{x}}_j, \hat{\mathbf{y}}_j)$, $j = 1, \dots, n$ are independent with respect to the randomness in sampling $\mathbf{x}'_j, \mathbf{x}''_j, \mathbf{y}'_j, \mathbf{y}''_j, \tau_j$, to invoke Hoeffding's inequality, yielding

$$\mathbb{P} \left[\left| \frac{1}{n} \sum_{j=1}^n \mathcal{N}'_j(\hat{\mathbf{x}}_j, \hat{\mathbf{y}}_j) - \mathbb{E} [\mathcal{N}'_1(\hat{\mathbf{x}}_1, \hat{\mathbf{y}}_1)] \right| > 0.04 \right] \leq 2 \exp \left(-0.5n \frac{0.04^2}{B^2} \right).$$

Setting $n = 2500B^2d = \text{poly}(d, C, w(D + 4d, C))$, we can upper bound the above by $2 \exp(-2d)$, which is strictly less than 2^{-2d} for all $d \geq 2$. Taking a union bound over all 2^{2d} possibilities for the inputs $(\mathbf{x}, \mathbf{y}) \in \{0, 1\}^{2d}$, we have by substituting our Hoeffding bound in Eq. (14), that with positive probability, \mathcal{N}'' satisfies $|\mathcal{N}''(\mathbf{x}, \mathbf{y}) - \text{IP}_d(\mathbf{x}, \mathbf{y})| \leq 0.41 + 0.04 = 0.45$ for all inputs $(\mathbf{x}, \mathbf{y}) \in \{0, 1\}^{2d}$. By the probabilistic method, this implies the existence of particular realizations of the random variables

$$\mathbf{x}'_1, \dots, \mathbf{x}'_n, \mathbf{x}''_1, \dots, \mathbf{x}''_n, \mathbf{y}'_1, \dots, \mathbf{y}'_n, \mathbf{y}''_1, \dots, \mathbf{y}''_n, \tau'_1, \dots, \tau'_n$$

with this property. Define the neural network \mathcal{N}''' as the network obtained from substituting these variables with the above realizations, and note that doing so merely decreases the input dimension of the network, while keeping the computation in the hidden layer linear, which thus does not change the architecture of \mathcal{N}''' and maintains its width.

To conclude the derivation so far, we have shown the existence of a depth 2 σ -network \mathcal{N}''' , which has width $\text{poly}(d, C, w(D + 4d, C))$, and satisfies

$$\max_{(\mathbf{x}, \mathbf{y}) \in \{0, 1\}^{2d}} |\mathcal{N}'''(\mathbf{x}, \mathbf{y}) - \text{IP}_d(\mathbf{x}, \mathbf{y})| < 0.45.$$

We now use Proposition 9 with $\delta = 0.04$ to obtain a threshold network $\bar{\mathcal{N}}$ from \mathcal{N}''' , having width $\text{poly}(d, C, w(D + 4d, C))$ and weights of magnitude at most $\text{poly}(d, C)$, such that for all $\mathbf{x}, \mathbf{y} \in \{0, 1\}^d$

$$|\bar{\mathcal{N}}(\mathbf{x}, \mathbf{y}) - \text{IP}_d(\mathbf{x}, \mathbf{y})| \leq |\bar{\mathcal{N}}(\mathbf{x}, \mathbf{y}) - \mathcal{N}'''(\mathbf{x}, \mathbf{y})| + |\mathcal{N}'''(\mathbf{x}, \mathbf{y}) - \text{IP}_d(\mathbf{x}, \mathbf{y})| \leq 0.45 + 0.04 = 0.49.$$

Constructing a threshold circuit from $\bar{\mathcal{N}}$, which employs a threshold activation on its output neuron, we obtain a threshold circuit which computes $\text{IP}_d(\cdot, \cdot)$. We lower bound the width of the circuit by using the following fact adapted from Hajnal et al. (1993) which appears in Martens et al. (2013), and is stated here in a slightly modified manner for the sake of completeness.

Fact 1 *For a depth 2 threshold network \mathcal{N} of width m and weights bounded in magnitude by C , that satisfies*

$$\max_{(\mathbf{x}, \mathbf{y}) \in \{0, 1\}^{2d}} |\mathcal{N}(\mathbf{x}, \mathbf{y}) - \text{IP}_d(\mathbf{x}, \mathbf{y})| \leq 0.5 - \delta$$

for some $\delta \in (0, 0.5)$, we have

$$m \geq \Omega \left(\frac{\delta 2^{d/3}}{C} \right).$$

Substituting $\delta = 0.01$, the above fact and our expression for the width of $\tilde{\mathcal{N}}$ imply the inequality

$$\text{poly}(d, C, w(D + 4d, C)) \geq \Omega\left(\frac{2^{d/3}}{\text{poly}(d, C)}\right).$$

Simplifying the above by using more asymptotic notation and absorbing terms that are polynomial in d into the exponent, we have

$$w(D + 4d, C) \geq \Omega\left(\frac{2^{\Omega(d)}}{\text{poly}(C)}\right).$$

Recall that $D = 100d$. Letting $d' := 104d$ which implies $d = \Omega(d')$ and performing a change of variables, the theorem follows. \blacksquare

A.2. Proof of Thm. 4

Proof Since the first $2d$ coordinates of our input hold no useful information for computing f_d , we simply ignore them. Next, by performing a simple change of variables (see [Safran et al. \(2019, Thm. 9\)](#)), we can assume for convenience that the remaining $2d$ coordinates are scaled by a factor of $3\sqrt{d}$ (which will scale the magnitude of the weights), so that the domain of approximation is $[0, 1]^{2d}$.

Let $\varepsilon > 0$. First, if $\varepsilon > \frac{1}{2}$, then we have

$$\sup_{\mathbf{x}, \mathbf{y} \in \mathcal{A}_d} \left| \frac{1}{2} - f_d(\mathbf{x}, \mathbf{y}) \right| = \frac{1}{2}.$$

Namely, a network which computes a constant function satisfies our requirements. We can thus assume from now on that $\varepsilon \leq \frac{1}{2}$.

Next, we define a few auxiliary functions that will be used in the construction of our approximation. Let

$$g_1(z) := \begin{cases} 0, & z \in (-\infty, 5], \\ z - 5, & z \in (5, 6), \\ 1, & z \in [6, \infty), \end{cases}$$

and Let

$$g_2(z) := \begin{cases} z \bmod 1, & \lfloor z \rfloor \text{ even}, \\ 1 - (z \bmod 1), & \lfloor z \rfloor \text{ odd}. \end{cases}$$

Let $\delta_1 := \frac{\varepsilon}{2d}$, we have from Assumption 2 and the fact that $g_1(\cdot)$ is 1-Lipschitz, that there exists some depth 2 σ -network h_1 , of width $\mathcal{O}\left(\frac{d}{\varepsilon}\right)$ and weights bounded by $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$, which satisfies

$$|g_1(z) - h_1(z)| \leq \frac{\varepsilon}{2d}, \quad \forall z \in [0, 8]. \quad (15)$$

Likewise, there exists some depth 2 σ -network h_2 , of width $\mathcal{O}\left(\frac{d}{\varepsilon}\right)$ and weights bounded by $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$, which satisfies

$$|g_2(z) - h_2(z)| \leq \frac{\varepsilon}{2}, \quad \forall z \in [-2d - 1, 2d + 1]. \quad (16)$$

Now, it is easy to verify that

$$g_1(4x + 4y) = \text{AND}(\text{round}(x), \text{round}(y)) \quad \forall x, y \in \mathcal{A}_1,$$

implying

$$\sum_{i=1}^d g_1(4x_i + 4y_i) = \langle \text{round}(\mathbf{x}), \text{round}(\mathbf{y}) \rangle \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{A}_d,$$

and thus

$$g_2\left(\sum_{i=1}^d g_1(4x_i + 4y_i)\right) = \text{IP}_d(\text{round}(\mathbf{x}), \text{round}(\mathbf{y})) = f_d(\mathbf{x}, \mathbf{y}) \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{A}_d.$$

We now define the network \mathcal{N} which approximates $f_d(\mathbf{x}, \mathbf{y})$ well on the set \mathcal{A}_d . For all $\mathbf{x}, \mathbf{y} \in \mathcal{A}_d$, define

$$\mathcal{N}(\mathbf{x}, \mathbf{y}) := h_2\left(\sum_{i=1}^d h_1(4x_i + 4y_i)\right).$$

We construct a depth 3 neural network that computes the above function as follows:

- The first hidden layer will consist of d copies $h_{1,i}$, $i = 1, \dots, d$, of the depth 2, width $\mathcal{O}\left(\frac{d}{\varepsilon}\right)$ network h_1 . Each $h_{1,i}$ will receive (x_i, y_i) as input; thus, the weight assigned to the coordinates x_i, y_i is set to 4, and the weight assigned to the remaining coordinates is set to zero. Note that the width of the first layer is therefore $\mathcal{O}\left(\frac{d^2}{\varepsilon}\right)$, and the magnitude of its weights is bounded by $\mathcal{O}\left(\frac{\sqrt{d}}{\varepsilon}\right)$.
- The second hidden layer will consist of a single copy of the network computing the function h_2 . This implies that the width of this layer is $\mathcal{O}\left(\frac{d}{\varepsilon}\right)$. Each neuron in this layer will assign the same set of weights for each incoming output from the output neurons of the components $h_{1,i}$, $i = 1, \dots, d$. Since this sum is a linear operation performed on these output neurons, we can effectively absorb the output neurons into the hidden neurons in the second layer, and thus avoid adding a third hidden layer. Note that by absorbing weights of magnitude at most $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ in a layer with weights bounded by $\mathcal{O}\left(\frac{d}{\varepsilon}\right)$, we get that the weights in the second hidden layer are of magnitude at most $\mathcal{O}\left(\frac{d}{\varepsilon^2}\right)$.

Concluding our construction of \mathcal{N} , we have that its width is $\mathcal{O}\left(\frac{d^2}{\varepsilon}\right)$, due to the first hidden layer; and its magnitude of the weights is $\mathcal{O}\left(\frac{d}{\varepsilon^2}\right)$, due to the second hidden layer.

Denote $\mathcal{B}_d := ([0, 0.25] \cup [0.75, 1])^d$. It is therefore only left, given arbitrary $\mathbf{x}, \mathbf{y} \in \mathcal{B}_d$, to upper bound the expression

$$\begin{aligned} |\mathcal{N}(\mathbf{x}, \mathbf{y}) - f_d(\mathbf{x}, \mathbf{y})| &= \left| h_2\left(\sum_{i=1}^d h_1(4x_i + 4y_i)\right) - g_2\left(\sum_{i=1}^d g_1(4x_i + 4y_i)\right) \right| \\ &\leq \left| h_2\left(\sum_{i=1}^d h_1(4x_i + 4y_i)\right) - g_2\left(\sum_{i=1}^d h_1(4x_i + 4y_i)\right) \right| \\ &\quad + \left| g_2\left(\sum_{i=1}^d h_1(4x_i + 4y_i)\right) - g_2\left(\sum_{i=1}^d g_1(4x_i + 4y_i)\right) \right|. \quad (17) \end{aligned}$$

We begin with upper bounding the first absolute value term. By virtue of Eq. (15), we have

$$\left| \sum_{i=1}^d h_1(4x_i + 4y_i) - \sum_{i=1}^d g_1(4x_i + 4y_i) \right| \leq \sum_{i=1}^d |h_1(4x_i + 4y_i) - g_1(4x_i + 4y_i)| \leq \frac{\varepsilon}{2}, \quad (18)$$

implying

$$\left| \sum_{i=1}^d h_1(4x_i + 4y_i) \right| \leq \left| \sum_{i=1}^d g_1(4x_i + 4y_i) \right| + \frac{\varepsilon}{2} \leq \sum_{i=1}^d |g_1(4x_i + 4y_i)| + \frac{1}{4} \leq d + \frac{1}{4},$$

for all $\mathbf{x}, \mathbf{y} \in \mathcal{B}_d$. The above and Eq. (16) imply

$$\left| h_2 \left(\sum_{i=1}^d h_1(4x_i + 4y_i) \right) - g_2 \left(\sum_{i=1}^d h_1(4x_i + 4y_i) \right) \right| \leq \frac{\varepsilon}{2}. \quad (19)$$

Moving on to bound the second absolute value term in Eq. (17), we have by the fact that $g_2(\cdot)$ is 1-Lipschitz that

$$\left| g_2 \left(\sum_{i=1}^d h_1(4x_i + 4y_i) \right) - g_2 \left(\sum_{i=1}^d g_1(4x_i + 4y_i) \right) \right| \leq \left| \sum_{i=1}^d h_1(4x_i + 4y_i) - \sum_{i=1}^d g_1(4x_i + 4y_i) \right| \leq \frac{\varepsilon}{2},$$

where in the second inequality we used Eq. (18). Plugging the above and Eq. (19) back in Eq. (17), we get

$$|\mathcal{N}(\mathbf{x}, \mathbf{y}) - f_d(\mathbf{x}, \mathbf{y})| \leq \varepsilon,$$

for all $\mathbf{x}, \mathbf{y} \in \mathcal{B}_d$, as desired. ■