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

Itay Safran[1], Daniel Reichman[2], and Paul Valiant[1]

[1]Purdue University
[2]Worcester Polytechnic Institute

Abstract

We prove an exponential separation between depth 2 and depth 3 neural networks, when approximating an $\mathcal{O}(1)$-Lipschitz target function to constant accuracy, with respect to a distribution with support in $[0,1]^d$, assuming exponentially bounded weights. This addresses an open problem posed in Safran et al. [20], and proves that the curse of dimensionality manifests in depth 2 approximation, even in cases where the target function can be represented efficiently using depth 3. Previously, lower bounds that were used to separate depth 2 from depth 3 required that at least one of the Lipschitz parameter, target accuracy or (some measure of) the size of the domain of approximation scale polynomially with the input dimension, whereas we fix the former two and restrict our domain to the unit hypercube. Our lower bound holds for a wide variety of activation functions, and is based on a novel application of an average- to worst-case random self-reducibility argument, to reduce the problem to threshold circuits lower bounds.

1 Introduction

There is significant empirical evidence suggesting that depth plays a crucial role in the practical success of deep learning [8, 5]. From a purely theoretical perspective, while depth 2 neural networks are known to be universal approximators for continuous functions over compact domains [1, 6, 9], it is now well-established that depth may be necessary to obtain a compact representation of certain target functions [3, 22, 16, 2, 26, 10, 19, 20, 25, 18, 15, 21]. Perhaps the most stark such examples are when separating depth 2 from depth 3 – following the seminal work of Eldan and Shamir [3], 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 $f : \mathbb{R}^d \to \mathbb{R}$ that can be approximated to accuracy $\varepsilon > 0$ using a network of depth 3 and width $\mathrm{poly}(d, 1/\varepsilon)$, 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 settings 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 optimization, even when using a network which is sufficiently deep to express the target function efficiently [11, 12]. In light of this, several recent works have shifted their focus to devising depth separation results for functions that are arguably more natural [19, 20, 18, 17, 15, 21]. 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. [20], the authors observe, given a depth 2 lower bound for neural network approximation, that the Lipschitz constant of the target function and the target accuracy parameters, can be traded off by rescaling the function 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 [20, Theorem 9] for a precise statement). This important 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 are a manifestation of the curse of dimensionality, or if the difficulty lies somewhere else. To study the root cause for the hardness of approximation, 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)?* [20]

Their main result, is that perhaps surprisingly, 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 $\mathrm{poly}(d)$ 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 hypercube, and the separation is exponential even if the target accuracy is an absolute constant. We point out that for technical reasons, our result does not quite resolve the above question, since the unit hypercube has radius which scales as $\sqrt{d}$. However, our result does provide a separation where the effects of the parameters other than the input dimension are minimal compared to other results in the literature (see Table 1 for a comparison). Moreover, despite not being bounded in a ball of constant radius, the unit hypercube is a natural domain to study approximation problems. This is supported by the fact that its d-dimensional Lebesgue measure is constant (namely, the volume of the domain is independent of d), whereas the unit ball needs to be rescaled by a factor of $\Theta(\sqrt{d})$ to have a constant volume. So at least in some sense, our domain of approximation is more independent of the input dimension than a domain of constant radius.

Our proof technique is quite different than those commonly used in the literature. While it is common to use some technical analysis tool such as Fourier spectrum analysis [3] or spherical harmonics [2], our lower bound relies on a reduction to threshold circuits, where an exact computation lower bound for the IP mod 2 function is used. The key ingredient in our proof is a reduction from an average-case complexity, where intuitively, only a constant portion of the inputs are classified correctly; to a worst-case complexity, where we are able to construct a network which is capable of classifying *all* the inputs 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 point out that our separation holds for a wide family of activation functions, which also 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, our result assumes a mild exponential upper bound on the magnitude of the weights of the approximating network. In contrast,

Table 1: Several known lower bounds for approximation using depth 2 neural networks, where the target function is scaled to be $\mathcal{O}(1)$-Lipschitz, and for the sake of comparison, the distribution is scaled to either contain the unit hypercube (for compactly supported distributions), or normalized to have most of its probability mass contained within the unit hypercube (non-compactly supported distributions). An asterisk * denotes that the result in Daniely [2] is not for a radial function, but can be reduced to one taking the form $\mathbf{x} \mapsto \frac{1}{2\pi d^{2.5}} \sin(2\pi d^{2.5} \|\mathbf{x}\|^2)$, which oscillates in any direction from the origin (see [20, 23]). In contrast, our result is non-oscillatory in all such directions, but it can be seen rather as an extension of a Boolean function to the unit hypercube. Safran et al. [20] use a reduction to the main result of Eldan and Shamir [3] 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 $1/\operatorname{poly}(d)$ are too permissive, in contrast to our constant lower bound.

Reference	Distribution	Function	Accuracy
Eldan and Shamir [3]	Radial, heavy-tailed	Radial, oscillatory	$\Omega\left(d^{-4}\right)$
Daniely [2]	Radial*, compactly supported	Radial*, oscillatory	$\Omega\left(d^{-2}\right)$
Safran et al. [20]	Radial, heavy-tailed,	Radial, non-oscillatory	$\Omega\left(d^{-6}\right)$
Venturi et al. [25]	Product, heavy-tailed	Product, oscillatory	$\Omega\left(d^{-4}\right)$
This paper	**Product, compactly supported**	**Non-oscillatory**	$\boldsymbol{\Omega\left(1\right)}$

lower bounds that do not impose any restrictions on the magnitude of the weights, rely on the technique in Eldan and Shamir [3], 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 well-approximate a certain $\mathcal{O}(1)$-Lipschitz function f_d (see Eq. (1)). It is shown that there exists a simple distribution with support in $[0,1]^{2d}$, such that f_d cannot be approximated to better than constant accuracy with respect to this distribution, unless the width of the network scales as $\Omega\left(\frac{\exp(\Omega(d))}{\operatorname{poly}(C)}\right)$ (Thm. 3.1).

- We prove that under different, 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. 4.1).

- Combining our lower and upper bounds, we obtain our main result (Thm. 2.1), 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 $1/\operatorname{poly}(d)$.

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 [3] 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 $\operatorname{poly}(d)$. Venturi et al. [25] 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 [3, 25] require the target accuracy to scale as $\Omega\left(d^{-4}\right)$ for the curse of dimensionality to manifest. Safran et al. [21] 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 in magnitude, and for accuracy which scales as $1/\operatorname{poly}(d)$ rather than a constant.

Daniely [2] used a simple and elegant harmonic analysis technique, to show a depth 2 vs. depth 3 separation, using exponentially bounded weights, for oscillatory target functions, over a product distribution on two spheres (that can be reduced to a radial distribution – see [20, 23] for more details). With some careful analysis, it can be shown that with this technique, the approximation error required for the curse of dimensionality to manifest is $\Omega\left(d^{-2}\right)$. [18, 15] prove depth 2 approximation lower bounds for non-oscillatory target functions, by using the main result in Daniely [2], 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^{-2} for the curse of dimensionality to manifest, a property inherited by the proof technique being used.

Safran and Shamir [19], Safran et al. [20] use reductions to the main result of Eldan and Shamir [3] to derive exponential depth 2 lower bounds for non-oscillatory functions. Safran and Shamir [19] show this for non-Lipschitz ellipsoid indicator functions, and Safran et al. [20] show this for a 1-Lipschitz radial function. Like the previous results using Fourier spectrum analysis, these results require accuracy scaling with $\Omega\left(d^{-4}\right)$ 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 $\Omega\left(d^{-6}\right)$. In contrast, our lower bound is also for a non-oscillatory function, but it is already exponential for constant 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. [13] study lower bounds for depth 2 neural networks with non-threshold activation functions, by constructing a threshold network which computes the same real-valued function, and applying known threshold circuit lower bounds. Their work differs from our in a few ways: (i) While we use a similar reduction technique, we only approximate non-threshold activations using thresholds rather than compute the exact same real function; (ii) we focus on approximation on continuous domains rather than the discrete uniform distribution over the Boolean hypercube; and most importantly, (iii) we consider a weaker notion

of *average-case* approximation, whereas their work deals with *exact* computation of Boolean functions. Mukherjee and Basu [14] derive sub-linear size lower bounds for neural networks by showing reductions to known threshold circuit lower bounds. Vardi and Shamir [23] 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. [24] use communication complexity to derive size lower bounds for ReLU networks which compute IP mod 2. While this lower bound applies to any depth, it is for a network size which is sublinear 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. [20]. To the best of our knowledge, the only works that made progress with the open question posed in Safran et al. [20], are [20, 7]. Safran et al. [20] 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. [7] 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 $\mathrm{poly}(d)$ random ReLU network will approximate any $\mathcal{O}(1)$-Lipschitz function with high probability (hence – there exists a 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 imply that lower bounds exponential in d are not possible for distributions on $[0,1]^d$ that are different from the uniform distribution. Indeed, our lower bound is for a distribution with support in $[0,1]^{2d}$, that has more probability mass concentrated closer to the Boolean hypercube.

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 $\{1,\ldots,n\}$. 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 $\mathbf{x} = (x_1,\ldots,x_d) \in \mathbb{R}^d$, we define $\mathrm{round}(\mathbf{x}) = (\mathrm{round}(x_1),\ldots,\mathrm{round}(x_d))$, where $\mathrm{round}(x)$ 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 $\mathrm{AND} : \{0,1\}^2 \to \{0,1\}$, $\mathrm{AND}(x,y) = x \cdot y$; and $\mathrm{IP}_d : \{0,1\}^{2d} \to \{0,1\}$, $\mathrm{IP}_d(\mathbf{x},\mathbf{y}) = \langle \mathbf{x},\mathbf{y} \rangle \mod 2$. Given a function $f : \mathbb{R} \to \mathbb{R}$, we define its *total variation* on the interval $[a,b] \subseteq \mathbb{R}$ as $V_a^b(f) \coloneqq \sup_P \sum_{i=1}^{n_P} |f(x_i) - f(x_{i-1})|$, where the supremum is taken over all the possible partitions of the interval $[a,b]$.

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 $\sigma : \mathbb{R} \to \mathbb{R}$. 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. Let

$$\mathcal{A}_d \coloneqq ([0, 0.25] \cup [0.75, 1])^d$$

be the support of our distribution. We define our distribution to be the uniform distribution over the set $\mathcal{A}_{2d}$; namely, the distribution $\mathcal{U}(\mathcal{A}_{2d})$. It is interesting to note that this distribution can also be seen as an 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 – the setting where neural networks are typically being used. We define our hard to approximate function $f_d : [0,1]^{2d} \to \mathbb{R}$ as

$$f_d(\mathbf{x}, \mathbf{y}) \coloneqq \mathrm{IP}_d(\mathrm{round}(\mathbf{x}), \mathrm{round}(\mathbf{y})) \cdot \min_{i \in [d]} \{4|x_i - 0.5|, 4|y_i - 0.5|, 1\}, \tag{1}$$

where the inputs satisfy $\mathbf{x}, \mathbf{y} \in [0,1]^d$. It is straightforward to verify that $f_d(\mathbf{x}, \mathbf{y}) = \mathrm{IP}_d(\mathrm{round}(\mathbf{x}), \mathrm{round}(\mathbf{y}))$ for all $\mathbf{x}, \mathbf{y} \in \mathcal{A}_d$. Moreover, it is easy to see that f_d is 4-Lipschitz on $[0,1]^{2d}$:[1] It equals 0 if there exists $i \in [d]$ such that $x_i = 0.5$ or $y_i = 0.5$, so it suffices to prove that it is 4-Lipschitz on each one of the 2^{2d} disjoint sub-hypercubes of side length 0.5 that are contained in $[0,1]^{2d}$. For each such sub-hypercube, if IP_d equals 0, then f_d equals 0 on the sub-hypercube, and it is clearly 4-Lipschitz. Otherwise, IP_d equals 1 on the sub-hypercube, and therefore for all $\mathbf{x}, \mathbf{y}$ in this sub-hypercube we have $f_d(\mathbf{x}, \mathbf{y}) = \min_{i \in [d]} \{4|x_i - 0.5|, 4|y_i - 0.5|, 1\} = 4 \min_{i \in [d]} \{|x_i - 0.5|, |y_i - 0.5|, 0.25\}$, which is 4-Lipschitz since the minimum function is 1-Lipschitz.

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.

[1] We remark that while we define f_d to be 4-Lipschitz on $[0,1]^{2d}$, this is in fact not necessary, since our distribution is only supported on $\mathcal{A}_{2d} \subset [0,1]^{2d}$, and therefore it does not matter how f_d behaves outside of this set. Nevertheless, we extend it to be Lipschitz on $[0,1]^{2d}$ to highlight its simple structure compared to some of the other functions that were used to separate depth 2 from depth 3.

Assumption 1 (Lower bound)**.** *The activation function σ is (Lebesgue) measurable and satisfies*

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

and

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

for all $x \in \mathbb{R}$, $a < b$ and for some constants C_σ, α_σ, which depend solely on σ.

The boundedness of $|\sigma(x)|$ is a standard assumption when proving approximation lower bounds, and it is also used in Eldan and Shamir [3] 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 $f : \{0,1\}^d \to \{0,1\}$, 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. Such an assumption is very mild, and holds for essentially any activation function which is used in practice.

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 $f : [-R, R] \to \mathbb{R}$, 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 [3], 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 A.4 in the appendix which implies that the threshold activation satisfies this property, and see Appendix A in Eldan and Shamir [3] 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 polynomials 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 2.1. *Consider the sequence of distributions $\{\mathcal{U}(\mathcal{A}_{2d})\}_{d=1}^\infty$, and the sequence of $\mathcal{O}(1)$-Lipschitz functions $f_d : \mathbb{R}^{2d} \to \mathbb{R}$ 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 $\mathcal{N}_d : \mathbb{R}^d \to \mathbb{R}$, with weights bounded in magnitude by C, we have*

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

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

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

$$\sup_{\mathbf{x},\mathbf{y}\in\mathcal{A}_d}\left|f_d(\mathbf{x},\mathbf{y}) - \mathcal{N}'_d(\mathbf{x},\mathbf{y})\right| \leq \varepsilon,$$

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

The above theorem is an immediate consequence of our lower bound (Thm. 3.1) and upper bound (Thm. 4.1), 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 3.1. *Suppose that σ satisfies Assumption 1, and let $\mathcal{N} : \mathbb{R}^{2d} \to \mathbb{R}$ be a depth 2 σ-network with weights bounded by C that satisfies*

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

Then, $\mathcal{N}$ has width

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

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

The proof of the above theorem relies on an average- to worst-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 [18] 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. 3.1

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 to a discrete distribution

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 $\mathcal{U}(\mathcal{A}_{2d})$. 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 aim to reduce this lower bound over the continuous distribution to the discrete distribution $\mathcal{U}(\{0,1\}^{2d})$. To this end, we use a similar argument to the one used in Vardi et al. [24, Prop. 6.1], who observe that the distribution $\mathcal{U}(\mathcal{A}_{2d})$ can be decomposed into the sum of the two distributions $\mathcal{U}([0,0.25]^{2d})$ and $\mathcal{U}(\{0,0.75\}^{2d})$. By using Markov's inequality on the randomness induced by the former continuous component, we have that with positive probability, we can find a depth 2 neural network $\mathcal{N}'$, which has width similar to $\mathcal{N}$; and a discrete sub-cube with side length 0.75, which is contained in $\mathcal{A}_{2d}$; such that $\mathcal{N}'$ approximates IP_d to an average square loss of $\frac{1}{399}$. By performing linear operations in the hidden layer of $\mathcal{N}'$, which do not affect its size, we are able to modify it without changing its architecture, making it approximate IP_d effectively with respect to the distribution $\mathcal{U}(\{0,1\}^{2d})$.

3.1.2 Step 2: From average- to worst-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 $\frac{1}{399}$. 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 $\mathbf{x},\mathbf{y} \in \mathcal{A}_d$, we have that $\mathrm{round}(\mathcal{N}'(\mathbf{x},\mathbf{y})) = \mathrm{IP}_d(\mathbf{x},\mathbf{y})$. Our aim is now to use $\mathcal{N}'$ to construct a depth 2, σ-network $\mathcal{N}''$, such that $\mathrm{round}(\mathcal{N}''(\mathbf{x},\mathbf{y})) = \mathrm{IP}_d(\mathbf{x},\mathbf{y})$ holds *for all inputs* $\mathbf{x},\mathbf{y} \in \mathcal{A}_d$. 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

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

  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 $\mathrm{round}(\mathcal{N}''(\mathbf{x}, \mathbf{y})) = \mathrm{IP}_d(\mathbf{x}, \mathbf{y})$ 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 each such jump increases the variation of the constructed approximation, and since σ is of bounded variation due to Assumption 1, 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 $\mathrm{round}(\mathcal{N}'''(\mathbf{x}, \mathbf{y})) = \mathrm{IP}_d(\mathbf{x}, \mathbf{y})$. 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 [4]. 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 an 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 4.1. *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*

$$\sup_{(\mathbf{x},\mathbf{y})\in\mathcal{A}_{2d}} |\mathcal{N}(\mathbf{x},\mathbf{y}) - f_d(\mathbf{x},\mathbf{y})| \leq \varepsilon.$$

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. 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 $\mathcal{A}_{2d}$ 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 $\mathcal{A}_{2d}$.

To give a more concrete example of an approximation obtained by our theorem, below we specify how this construction can be done when σ is the ReLU activation.

Example 4.2. *Let $[x]_+ \coloneqq \max\{0, x\}$ denote the ReLU activation function, and define*

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

Then, the network given by

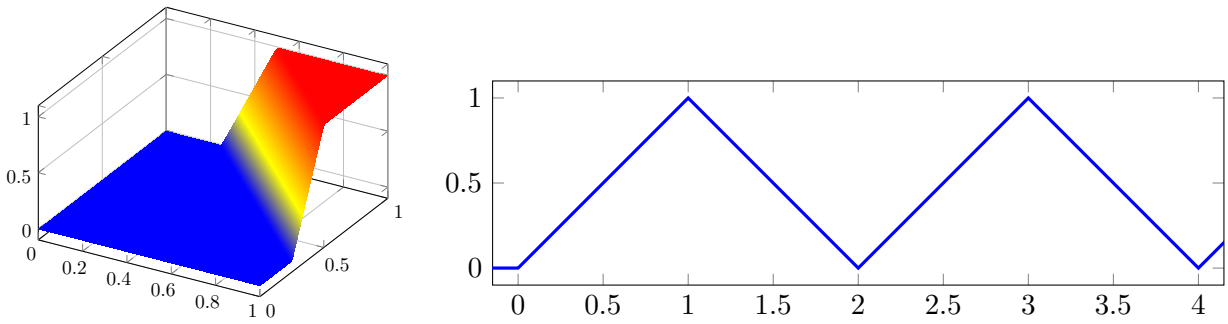
$$\mathcal{N}(\mathbf{x},\mathbf{y}) \coloneqq t_d\left(\sum_{i=1}^{d} [4x_i + 4y_i - 5]_+ - [4x_i + 4y_i - 6]_+\right)$$

satisfies

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

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 $\mathcal{A}_{2d}$. 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 [11, 12], 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 [18] 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 4.2 provides a simple, linear in size and sparse approximation of f_d. This simplicity



(a) (b)

Figure 1: Computing f_d using a depth 3 ReLU network. Subfigure 1a plots the function $(x, y) \mapsto [4x + 4y - 5]_+ - [4x + 4y - 6]_+$, which equals $\mathrm{AND}(\mathrm{round}(x), \mathrm{round}(y))$ for all $x, y \in \mathcal{A}_1$. Subfigure 1b plots the function $x \mapsto t_d(x)$, defined in Example 4.2. When composing the latter with a sum of the former, iterating over all pairs of coordinates x_i, y_i, we obtain a function that coalesces with f_d on $\mathcal{A}_{2d}$. Best viewed in color.

is much desired, since it may suggest that similarly to Safran and Lee [18], 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

[1] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[2] Amit Daniely. Depth separation for neural networks. In *Conference on Learning Theory*, pages 690–696, 2017.

[3] Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Conference on learning theory*, pages 907–940. PMLR, 2016.

[4] 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.

[5] 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.

[6] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[7] 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.

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

[9] 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.

[10] Shiyu Liang and Rayadurgam Srikant. Why deep neural networks for function approximation? *arXiv preprint arXiv:1610.04161*, 2016.

[11] Eran Malach and Shai Shalev-Shwartz. Is deeper better only when shallow is good? *Advances in Neural Information Processing Systems*, 32, 2019.

[12] 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.

[13] James Martens, Arkadev Chattopadhya, Toni Pitassi, and Richard Zemel. On the representational efficiency of restricted boltzmann machines. *Advances in Neural Information Processing Systems*, 26, 2013.

[14] Anirbit Mukherjee and Amitabh Basu. Lower bounds over boolean inputs for deep neural networks with relu gates. *arXiv preprint arXiv:1711.03073*, 2017.

[15] 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.

[16] 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.

[17] Yunwei Ren, Mo Zhou, and Rong Ge. Depth separation with multilayer mean-field networks. *arXiv preprint arXiv:2304.01063*, 2023.

[18] Itay Safran and Jason Lee. Optimization-based separations for neural networks. In *Conference on Learning Theory*, pages 3–64. PMLR, 2022.

[19] 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.

[20] 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.

[21] 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.

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

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

[24] 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.

[25] 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.

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

A Proofs

A.1 Proof of Thm. 3.1

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

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

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 A.1. *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:*

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

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):

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

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,

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

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 $\sum_i d_i = d$:

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

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

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

Finally, we bound the ratio of factorials as follows. Define the function f on integer input $f(j) \coloneqq 4^{-4j}\binom{4j}{j,j,j,j} = 4^{-4j}\frac{(4j)!}{j!^4}$. 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 $\frac{D!(D/4+d/4)!^4}{(D+d)!(D/4)!^4} \leq \left(\frac{D+d}{D}\right)^{3/2}$. And we have derived Eq. (2) as claimed. □

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

Lemma A.2. *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*

$$A \coloneqq (\mathbf{x} + \mathbf{x}', \mathbf{x}', \mathbf{x} + \mathbf{x}', \mathbf{x}'),$$
$$B \coloneqq (\mathbf{y} + \mathbf{y}', \mathbf{y}', \mathbf{y}', \mathbf{y} + \mathbf{y}'),$$

where all additions are mod 2. Among these $4d$ entries, let d_1 count the number of indices $j \in \{1, \ldots, 4d\}$ 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

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

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:

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

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

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

and thus that

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

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 $(1,1,1,1)$:

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

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 $f(0) = 1, f'(0) = 0, f''(x) \le 12f(x)$, to yield $f(x) \le \cosh(\sqrt{12}x) \le e^{(12/2)x^2}$. 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 $M(t_1,t_2,t_3,t_4) \le e^{6\sum_{i=1}^4 t_i^2}$.

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 $\cosh(\langle(t_1,t_2,t_3,t_4),(1,1,-1,-1)\rangle) \le e^{\langle(t_1,t_2,t_3,t_4),(1,1,-1,-1)\rangle^2/2} \le e^{(||(1,1,-1,-1)||_2^2/2)\sum_{i=1}^4 t_i^2}$, 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 the 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

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

as desired. □

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

Proposition A.3. *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 $j \in \{1, \ldots, D\}$ where $\mathbf{x}''_j = y''_j = 1$; and pick a random permutation τ of $4d + D$ elements. The claim is that the pair of length $4d + D$ vectors*

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

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

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

For the random vectors $\mathbf{x}'', \mathbf{y}''$, let D_1, D_2, D_3, D_4 count the number of indices $j \in \{1, \ldots, D\}$ 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

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

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 A.1, if in Lemma A.1 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

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

The right hand side is exactly the form of Lemma A.2, 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. □

The following lemma guarantees that activation functions which satisfy Assumption 1, can be simulated to arbitrary accuracy, using depth 2 threshold networks of polynomial width.

Lemma A.4. *Let σ be an activation function that satisfies Assumption 1. Then for all $\delta > 0$, there exists a depth 2 threshold network $\mathcal{N}$ of width at most $\frac{\mathrm{poly}(R)}{\delta}$ and weights of magnitude at most $\frac{\mathrm{poly}(R)}{\delta}$, 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 $x_1 \coloneqq -R$ and $y_1 \coloneqq \sigma(x_1)$. We define the set

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

Note that $A_1 \neq \emptyset$ since $x_1 \in A_1$, and define $x_1' = x_1$ and $x_2 \coloneqq \sup A_1$. 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 $y_2 \coloneqq \sigma(x_2)$, 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
$$A_2 \coloneqq \{a \in [x_2, R] : \forall x \in [x_2, a],\ |\sigma(x) - y_2| \leq \delta\},$$
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 $y_2 \coloneqq \lim_{x \to x_{2+}} \sigma(x)$, 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
$$A_2 \coloneqq \{a \in (x_2, R] : \forall x \in (x_2, a],\ |\sigma(x) - y_2| \leq \delta\},$$
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 $y_2 \coloneqq \lim_{x \to x_{2+}} \sigma(x)$ 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 $y_2 \coloneqq \sigma(x_2)$ 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 $x_3 \coloneqq \sup A_2$, we can continue in this manner and define sequences of target values $y_1, y_2, \ldots$ and points $x_1, x_2, \ldots, x_1', x_2', \ldots$ 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 1 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 $x'_1, x'_2, \ldots, x'_n$ of $[-R, R]$ has total variation at least

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

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 $w_i \coloneqq \xi_i$, $b_i \coloneqq -\xi_i x_i - 0.5$, $v_i \coloneqq \xi_i(y_i - y_{i-1})$ and $b_0 \coloneqq -0.5\sum_{i=1}^{n}(\xi_i - 1)(y_i - y_{i-1})$, we obtain a width n threshold network

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

satisfying

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

Lastly, by Assumption 1, we have that $|y_i| \leq \text{poly}(R)$ for all $i \in [n]$, implying that $\mathcal{N}$ has weights of magnitude at most $\frac{\text{poly}(R)}{\delta}$. □

The following proposition guarantees that functions on the Boolean hypercube, computed by depth 2 networks, which employ activation functions that satisfy Assumption 1, can be simulated to arbitrary accuracy using depth 2 threshold networks of width polynomial in the size of the network.

Proposition A.5. *Let $\delta > 0$, suppose that σ satisfies Assumption 1, and let $f : \{0,1\}^d \to \mathbb{R}$ 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

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

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 A.4 to approximate each σ-neuron in f to accuracy $\frac{\delta}{mC}$, and obtain m depth 2, width $\frac{m\,\mathrm{poly}(d,C)}{\delta}$ threshold networks $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_m$, such that

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

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

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

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

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

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

With the above lemmas and propositions at hand, we are finally ready to prove the theorem.

Proof of Thm. 3.1. 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 $D \coloneqq 100d$, 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

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

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 IP_{D+4d} uniformly over the Boolean hypercube.

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

$$\mathop{\mathbb{E}}_{\mathbf{c}\sim\mathcal{U}\left([0,0.25]^{2D+8d}\right)}\left[\mathop{\mathbb{E}}_{\mathbf{z}\sim\mathcal{U}(\{0,0.75\}^{2D+8d})}\left[\left(\mathcal{N}(\mathbf{z}+\mathbf{c}) - f_{D+4d}(\mathbf{z}+\mathbf{c})\right)^2\right]\right] \leq \frac{1}{400}.$$

For $\mathbf{c} \sim \mathcal{U}\left([0,0.25]^{2D+8d}\right)$, we can define the non-negative random variable

$$X_{\mathbf{c}} \coloneqq \mathop{\mathbb{E}}_{\mathbf{z}\sim\mathcal{U}(\{0,0.75\}^{2D+8d})}\left[\left(\mathcal{N}(\mathbf{z}+\mathbf{c}) - f_{D+4d}(\mathbf{z}+\mathbf{c})\right)^2\right].$$

This allows us to rewrite the former inequality more compactly as

$$\mathop{\mathbb{E}}_{\mathbf{c}\sim\mathcal{U}\left([0,0.25]^{2D+8d}\right)}[X_{\mathbf{c}}] \leq \frac{1}{400}. \tag{6}$$

Using Markov's inequality on $X_{\mathbf{c}}$, and by virtue of Eq. (6), we have

$$\mathop{\mathbb{P}}_{\mathbf{c}\sim\mathcal{U}\left([0,0.25]^{2D+8d}\right)}\left[X_{\mathbf{c}} < \frac{400}{399}\mathop{\mathbb{E}}_{\mathbf{c}\sim\mathcal{U}\left([0,0.25]^{2D+8d}\right)}[X_{\mathbf{c}}] \leq \frac{1}{399}\right] \geq 1 - \frac{399}{400} > 0.$$

Namely, there must exist some $\mathbf{c} \in [0, 0.25]^{2D+8d}$ such that

$$\mathop{\mathbb{E}}_{\mathbf{z}\sim\mathcal{U}(\{0,0.75\}^{2D+8d})}\left[\left(\mathcal{N}(\mathbf{z}+\mathbf{c}) - f_{D+4d}(\mathbf{z}+\mathbf{c})\right)^2\right] = X_{\mathbf{c}} \leq \frac{1}{399}. \tag{7}$$

Next, we have by the definition of f that $f_{D+4d}(\mathbf{z}+\mathbf{c}) = f_{D+4d}(\mathbf{z}) = f_{D+4d}\left(\frac{4}{3}\mathbf{z}\right) = \mathrm{IP}_{D+4d}\left(\frac{4}{3}\mathbf{z}\right)$ for all $\mathbf{c} \in [0, 0.25]^{2D+8d}$ and all $\mathbf{z} \in \{0, 0.75\}^{2D+8d}$. Moreover, there exists a σ-network $\mathcal{N}'$ of the same depth and width as $\mathcal{N}$, such that $\mathcal{N}(\mathbf{z}+\mathbf{c}) = \mathcal{N}'\left(\frac{4}{3}\mathbf{z}\right)$ for all $\mathbf{c} \in [0, 0.25]^{2D+8d}$ and all $\mathbf{z} \in \{0, 0.75\}^{2D+8d}$. 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, and scaling the inputs by a multiplicative constant $\frac{4}{3}$ is equivalent to multiplying the weights of the hidden neurons by 0.75. Since both operations are linear, they can be absorbed in the hidden layer without changing the architecture of $\mathcal{N}'$, and where the magnitude of the weights is now at most $2C$.

The above observations, the definition of $X_{\mathbf{c}}$, and Eq. (7), imply the existence of some $\mathbf{c} \in [0, 0.25]^{2D+8d}$ that satisfies

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y}\sim\mathcal{U}(\{0,1\}^{D+4d})}\left[\left(\mathcal{N}'(\mathbf{x},\mathbf{y}) - \mathrm{IP}_{D+4d}(\mathbf{x},\mathbf{y})\right)^2\right] &= \mathop{\mathbb{E}}_{\mathbf{z}\sim\mathcal{U}(\{0,0.75\}^{2D+8d})}\left[\left(\mathcal{N}'\left(\frac{4}{3}\mathbf{z}\right) - \mathrm{IP}_{D+4d}\left(\frac{4}{3}\mathbf{z}\right)\right)^2\right] \\
&= \mathop{\mathbb{E}}_{\mathbf{z}\sim\mathcal{U}(\{0,0.75\}^{2D+8d})}\left[\left(\mathcal{N}(\mathbf{z}+\mathbf{c}) - f_{D+4d}(\mathbf{z}+\mathbf{c})\right)^2\right] \\
&\leq \frac{1}{399}.
\end{aligned} \tag{8}$$

Let $n \in \mathbb{N}$ to be determined later, we now construct a neural network $\mathcal{N}'' : \mathbb{R}^{D+4d} \to \mathbb{R}$ 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 $(\mathbf{x}+\mathbf{x}', \mathbf{x}', \mathbf{x}+\mathbf{x}', \mathbf{x}', \mathbf{x}'')$ and $(\mathbf{y}+\mathbf{y}', \mathbf{y}', \mathbf{y}', \mathbf{y}+\mathbf{y}', \mathbf{y}'')$.
- We sample a permutation $\tau_j : \mathbb{R}^{D+4d} \to \mathbb{R}$ 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. (8) and Proposition A.3's bound on $\|\mathcal{D}\|_2$ yields

$$\begin{aligned}
\mathbb{E}_{(\hat{\mathbf{x}},\hat{\mathbf{y}})\sim\mathcal{D}}\left[\left|\mathcal{N}'(\hat{\mathbf{x}},\hat{\mathbf{y}}) - \mathrm{IP}_{D+4d}(\hat{\mathbf{x}},\hat{\mathbf{y}})\right|\right] &= \sum_{\hat{\mathbf{x}},\hat{\mathbf{y}}\in\{0,1\}^{D+4d}} \mathop{\mathbb{P}}_{\mathcal{D}}[X = \hat{\mathbf{x}}, Y = \hat{\mathbf{y}}] \cdot \left|\mathcal{N}'(\hat{\mathbf{x}},\hat{\mathbf{y}}) - \mathrm{IP}_{D+4d}(\hat{\mathbf{x}},\hat{\mathbf{y}})\right| \\
&\leq ||\mathcal{D}||_2 \sqrt{\sum_{\hat{\mathbf{x}},\hat{\mathbf{y}}\in\{0,1\}^{D+4d}} \left(\mathcal{N}'(\hat{\mathbf{x}},\hat{\mathbf{y}}) - \mathrm{IP}_{D+4d}(\hat{\mathbf{x}},\hat{\mathbf{y}})\right)^2} \\
&\leq 8\sqrt{\frac{1}{399}} < 0.41. \qquad (9)
\end{aligned}$$

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

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

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

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

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 $\mathrm{IP}_D(\mathbf{x}'', \mathbf{y}'') = 0$, and the last two equalities follow from basic properties of the inner product mod 2. Thus, we have

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

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

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

where the expectation is taken over the randomness in sampling $(\hat{\mathbf{x}}_j,\hat{\mathbf{y}}_j)$ from $\mathcal{D}$. Using Eq. (9), we can upper bound the above by

$$\left|\frac{1}{n}\sum_{j=1}^{n}\mathcal{N}_j'(\hat{\mathbf{x}}_j,\hat{\mathbf{y}}_j) - \mathbb{E}\left[\mathcal{N}_1'(\hat{\mathbf{x}}_1,\hat{\mathbf{y}}_1)\right]\right| + 0.41. \tag{10}$$

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 1 we get that the output of each hidden neuron is upper bounded by $\mathrm{poly}(d,C)$ over the domain $\{0,1\}^{D+4d}$. This implies that the output of the output neuron is at most $B \coloneqq \mathrm{poly}(d,C)w(D+4d,C)$, 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 $\mathcal{N}_j'(\hat{\mathbf{x}}_j,\hat{\mathbf{y}}_j)$, $j=1,\ldots,n$ 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

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

Setting $n = 2500B^2d = \mathrm{poly}(d,C,w(D+4d,C))$, 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. (10), that with positive probability, $\mathcal{N}''$ satisfies $|\mathcal{N}''(\mathbf{x},\mathbf{y}) - \mathrm{IP}_d(\mathbf{x},\mathbf{y})| \leq 0.41 + 0.04 = 0.45$ 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

$$\mathbf{x}_1',\ldots,\mathbf{x}_n',\mathbf{x}_1'',\ldots,\mathbf{x}_n'',\mathbf{y}_1',\ldots,\mathbf{y}_n',\mathbf{y}_1'',\ldots,\mathbf{y}_n'',\tau_1',\ldots,\tau_n'$$

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 $\mathrm{poly}(d,C,w(D+4d,C))$, and satisfies

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

We now use Proposition A.5 with $\delta = 0.04$ to obtain a threshold network $\bar{\mathcal{N}}$ from $\mathcal{N}'''$, having width $\mathrm{poly}(d,C,w(D+4d,C))$ and weights of magnitude at most $\mathrm{poly}(d,C)$, such that for all $\mathbf{x},\mathbf{y} \in \{0,1\}^d$

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

Constructing a threshold circuit from $\bar{\mathcal{N}}$, which employs a threshold activation on its output neuron, we obtain a threshold circuit which computes $\mathrm{IP}_d(\cdot,\cdot)$. We lower bound the width of the circuit by using the following fact adapted from Hajnal et al. [4] which appears in Martens et al. [13], and is stated here in a slightly modified manner for the sake of completeness.

Fact A.6. *For a depth 2 threshold network $\mathcal{N}$ of width m and weights bounded in magnitude by C, that satisfies*

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

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 $\bar{\mathcal{N}}$ imply the inequality

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

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

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

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

A.2 Proof of Thm. 4.1

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

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

and Let

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

Let $\delta_1 \coloneqq \frac{\varepsilon}{2d}$, 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

$$|g_1(z) - h_1(z)| \leq \frac{\varepsilon}{2d}, \quad \forall z \in [0, 8]. \tag{11}$$

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

$$|g_2(z) - h_2(z)| \leq \frac{\varepsilon}{2}, \quad \forall z \in [-2d-1, 2d+1]. \tag{12}$$

Now, it is easy to verify that

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

implying

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

and thus

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

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

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

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

- The first hidden layer will consist of d copies $h_{1,i}$, $i = 1, \ldots, d$, 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 $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$.
- 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 $h_{1,i}$, $i = 1, \ldots, d$. 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.

It is therefore only left, given arbitrary $\mathbf{x}, \mathbf{y} \in \mathcal{A}_d$, to upper bound the expression

$$\begin{aligned}
|\mathcal{N}(\mathbf{x}, \mathbf{y}) - f_d(\mathbf{x}, \mathbf{y})| &= \left| h_2\left(\sum_{i=1}^{d} h_1(4x_i + 4y_i)\right) - g_2\left(\sum_{i=1}^{d} g_1(4x_i + 4y_i)\right) \right| \\
&\leq \left| h_2\left(\sum_{i=1}^{d} h_1(4x_i + 4y_i)\right) - g_2\left(\sum_{i=1}^{d} h_1(4x_i + 4y_i)\right) \right| \\
&\quad + \left| g_2\left(\sum_{i=1}^{d} h_1(4x_i + 4y_i)\right) - g_2\left(\sum_{i=1}^{d} g_1(4x_i + 4y_i)\right) \right|.
\end{aligned} \tag{13}$$

We begin with upper bounding the first absolute value term. By virtue of Eq. (11), we have

$$\left|\sum_{i=1}^{d} h_1(4x_i + 4y_i) - \sum_{i=1}^{d} g_1(4x_i + 4y_i)\right| \leq \sum_{i=1}^{d} |h_1(4x_i + 4y_i) - g_1(4x_i + 4y_i)| \leq \frac{\varepsilon}{2}, \tag{14}$$

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 $\mathbf{x}, \mathbf{y} \in \mathcal{A}$. The above and Eq. (12) imply

$$\left|h_2\left(\sum_{i=1}^{d} h_1(4x_i + 4y_i)\right) - g_2\left(\sum_{i=1}^{d} h_1(4x_i + 4y_i)\right)\right| \leq \frac{\varepsilon}{2}. \tag{15}$$

Moving on to bound the second absolute value term in Eq. (13), 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. (14). Plugging the above and Eq. (15) back in Eq. (13), we get

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

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