Deep Network Approximation for Smooth Functions

Jianfeng Lu * Zuowei Shen † Haizhao Yang ‡ Shijun Zhang §

Abstract

This paper establishes the optimal approximation error characterization of deep rectified linear unit (ReLU) networks for smooth functions in terms of both width and depth simultaneously. To that end, we first prove that multivariate polynomials can be approximated by deep ReLU networks of width $\mathcal{O}(N)$ and depth $\mathcal{O}(L)$ with an approximation error $\mathcal{O}(N^{-L})$. Through local Taylor expansions and their deep ReLU network approximations, we show that deep ReLU networks of width $\mathcal{O}(N\ln N)$ and depth $\mathcal{O}(L\ln L)$ can approximate $f\in C^s([0,1]^d)$ with a nearly optimal approximation error $\mathcal{O}(\|f\|_{C^s([0,1]^d)}N^{-2s/d}L^{-2s/d})$. Our estimate is non-asymptotic in the sense that it is valid for arbitrary width and depth specified by $N\in\mathbb{N}^+$ and $L\in\mathbb{N}^+$, respectively.

Key words. Deep ReLU Network, Smooth Function, Polynomial Approximation, Function Composition, Curse of Dimensionality.

1 Introduction

Deep neural networks have made significant impacts in many fields of computer science and engineering, especially for large-scale and high-dimensional learning problems. Well-designed neural network architectures, efficient training algorithms, and high-performance computing technologies have made neural-network-based methods very successful in tremendous real applications. Especially in supervised learning; e.g., image classification and objective detection, the great advantages of neural-network-based methods over traditional learning methods have been demonstrated. Understanding the approximation capacity of deep neural networks has become a key question for revealing the power of deep learning. A large number of experiments in real applications have shown the large capacity of deep network approximation from many empirical points of view, motivating much effort in establishing the theoretical foundation of deep network approximation. One of the fundamental problems is the characterization of the optimal approximation error of deep neural networks of arbitrary depth and width.

*Department of Mathematics, Department of Physics, and Department of Chemistry, Duke University (jianfeng@math.duke.edu).

†Department of Mathematics, National University of Singapore (matzuows@nus.edu.sg).

‡Department of Mathematics, Purdue University (haizhao@purdue.edu).

§Department of Mathematics, National University of Singapore (zhangshijun@u.nus.edu).

1.1 Main result

Previously, the quantitative characterization of the approximation power of deep feed-forward neural networks (FNNs) with rectified linear unit (ReLU) activation functions was provided in [41]. For ReLU FNNs with width $\mathcal{O}(N)$ and depth $\mathcal{O}(L)$, the deep network approximation of $f \in C([0,1]^d)$ admits an approximation error $\mathcal{O}\big(\omega_f(N^{-2/d}L^{-2/d})\big)$ in the L^p-norm for any $p \in [1,\infty]$, where $\omega_f(\cdot)$ is the modulus of continuity of f. In particular, for the class of Hölder continuous functions, the approximation error is nearly optimal.① The next question is whether the smoothness of functions can improve the approximation error. In this paper, we investigate the deep network approximation of smaller function space, such as the smooth function space $C^s([0,1]^d)$.

In Theorem 1.1 below, we prove by construction that ReLU FNNs with width $\mathcal{O}(N\ln N)$ and depth $\mathcal{O}(L\ln L)$ can approximate $f \in C^s([0,1]^d)$ with a nearly optimal approximation error $\mathcal{O}(\|f\|_{C^s([0,1]^d)}N^{-2s/d}L^{-2s/d})$, where the norm $\|\cdot\|_{C^s([0,1]^d)}$ is defined as

$$\|f\|_{C^s([0,1]^d)} \coloneqq \max\big\{\|\partial^{\boldsymbol{\alpha}} f\|_{L^\infty([0,1]^d)} : \|\boldsymbol{\alpha}\|_1 \le s,\ \boldsymbol{\alpha} \in \mathbb{N}^d\big\}, \quad \text{for any } f \in C^s([0,1]^d).$$

Theorem 1.1. *Given a smooth function $f \in C^s([0,1]^d)$ with $s \in \mathbb{N}^+$, for any $N, L \in \mathbb{N}^+$, there exists a function ϕ implemented by a ReLU FNN with width $C_1(N+2)\log_2(8N)$ and depth $C_2(L+2)\log_2(4L)+2d$ such that*

$$\|\phi - f\|_{L^\infty([0,1]^d)} \le C_3\|f\|_{C^s([0,1]^d)}N^{-2s/d}L^{-2s/d},$$

where $C_1 = 17s^{d+1}3^d d$, $C_2 = 18s^2$, and $C_3 = 85(s+1)^d 8^s$.

As we can see from Theorem 1.1, the smoothness improves the approximation error in N and L; e.g., $s \ge d$ implies $N^{-2s/d}L^{-2s/d} \le N^{-2}L^{-2}$. However, we would like to remark that the improved approximation error is at the price of a much larger prefactor larger than d^d if $s \ge d$. The proof of Theorem 1.1 will be presented in Section 2.2 and its tightness will be discussed in Section 2.3. In fact, the logarithmic terms in width and depth in Theorem 1.1 can be further reduced if the approximation error is weakened. Given any $\widetilde{N}, \widetilde{L} \in \mathbb{N}^+$ with

$$\widetilde{N} \ge C_1(1+2)\log_2(8) = 17s^{d+1}3^{d+2}d \quad \text{and} \quad \widetilde{L} \ge C_2(1+2)\log_2(4)+2d = 108s^2+2d,$$

there exist $N, L \in \mathbb{N}^+$ such that

$$C_1(N+2)\log_2(8N) \le \widetilde{N} < C_1\big((N+1)+2\big)\log_2\big(8(N+1)\big)$$

and

$$C_2(L+2)\log_2(4L)+2d \le \widetilde{L} < C_2\big((L+1)+2\big)\log_2\big(4(L+1)\big)+2d.$$

It follows that

$$N \ge \frac{N+3}{4} > \frac{\widetilde{N}}{4C_1\log_2(8N+8)} \ge \frac{\widetilde{N}}{4C_1\log_2(8\widetilde{N}+8)} = \frac{\widetilde{N}}{68s^{d+1}3^d d\log_2(8\widetilde{N}+8)}$$

and

$$L \ge \frac{L+3}{4} > \frac{\widetilde{L}-2d}{4C_2\log_2(4L+4)} \ge \frac{\widetilde{L}-2d}{4C_2\log_2(4\widetilde{L}+4)} = \frac{\widetilde{L}-2d}{72s^2\log_2(4\widetilde{L}+4)}.$$

Thus, we have an immediate corollary.

① "nearly optimal" up to a logarithmic factor.

Corollary 1.2. *Given a function $f \in C^s([0,1]^d)$ with $s \in \mathbb{N}^+$, for any $\widetilde{N}, \widetilde{L} \in \mathbb{N}^+$, there exists a function ϕ implemented by a ReLU FNN with width $\widetilde{N}$ and depth $\widetilde{L}$ such that*

$$\|\phi - f\|_{L^\infty([0,1]^d)} \le \widetilde{C}_1 \|f\|_{C^s([0,1]^d)} \Big(\tfrac{\widetilde{N}}{\widetilde{C}_2 \log_2(8\widetilde{N}+8)}\Big)^{-2s/d} \Big(\tfrac{\widetilde{L}-2d}{\widetilde{C}_3 \log_2(4\widetilde{L}+4)}\Big)^{-2s/d},$$

for any $\widetilde{N} \ge 17 s^{d+1} 3^{d+2} d$ and $\widetilde{L} \ge 108 s^2 + 2d$, where $\widetilde{C}_1 = 85(s+1)^d 8^s$, $\widetilde{C}_2 = 68 s^{d+1} 3^d d$, and $\widetilde{C}_3 = 72 s^2$.

Theorem 1.1 and Corollary 1.2 characterize the approximation error in terms of total number of neurons (with an arbitrary distribution in width and depth) and the smoothness of the target function to be approximated. The only result in this direction we are aware of in the literature is Theorem 4.1 of [46]. It shows that ReLU FNNs with width $2d+10$ and depth L achieve a nearly optimal error $\mathcal{O}((\frac{L}{\ln L})^{-2s/d})$ for sufficiently large L when approximating functions in the unit ball of $C^s([0,1]^d)$. This result is essentially a special case of Corollary 1.2 by setting $\widetilde{N} = \mathcal{O}(1)$ and $\widetilde{L}$ sufficiently large.

1.2 Contributions and related work

Our key contributions can be summarized as follows.

(i) **Upper bound**: We provide a **quantitative** and **non-asymptotic** approximation error $\mathcal{O}(\|f\|_{C^s([0,1]^d)} N^{-2s/d} L^{-2s/d})$ when the ReLU FNN has width $\mathcal{O}(N \ln N)$ and depth $\mathcal{O}(L \ln L)$ for functions in $C^s([0,1]^d)$ in Theorem 1.1. In real applications, the first question is to decide the network width and depth since they are two required hyper-parameters. The approximation error as a function of width and depth in this paper can directly answer this question, while the approximation results in terms of the total number of parameters in the literature cannot, because there are many architectures sharing the same number of parameters. Actually, an immediate corollary of our theorem as we shall discuss can also describe our theory in terms of the total number of parameters. Furthermore, our results contain approximation error estimates for both wide networks with fixed finite depth and deep networks with fixed finite width.

(ii) **Lower bound**: Through the Vapnik-Chervonenkis (VC) dimension upper bound of ReLU FNNs in [22], we prove a lower bound

$$C\big(N^2 L^2 (\ln N)^3 (\ln L)^3\big)^{-s/d}, \quad \text{for some positive constant } C,$$

for the approximation error of the functions in the unit ball of $C^s([0,1]^d)$ approximated by ReLU FNNs with width $\mathcal{O}(N \ln N)$ and depth $\mathcal{O}(L \ln L)$ in Section 2.3. Thus, the approximation error $\mathcal{O}(N^{-2s/d} L^{-2s/d})$ in Theorem 1.1 is nearly optimal for the unit ball of $C^s([0,1]^d)$.

(iii) **Approximation of polynomials**: It is proved by construction in Proposition 4.1 that ReLU FNNs with width $\mathcal{O}(N)$ and depth $\mathcal{O}(L)$ can approximate polynomials on $[0,1]^d$ with an approximation error $\mathcal{O}(N^{-L})$. This is a non-trivial extension of the result $\mathcal{O}(2^{-L})$ for polynomial approximation by fixed-width ReLU FNNs with depth L in [44].

(iv) **Uniform approximation**: The approximation error in this paper is measured in the $L^\infty([0,1]^d)$-norm as a result of Theorem 2.1. To achieve this, given a ReLU FNN approximating the target function f uniformly well on $[0,1]^d$ except for a small region, we develop a technique to construct a new ReLU FNN with a similar size to approximate f **uniformly** well on $[0,1]^d$ in Theorem 2.1. This technique can be applied to improve approximation errors from the L^p-norm to the L^∞-norm for other function spaces in general, e.g., the continuous function space in [41], which is of independent interest.

In particular, if we denote the best approximation error of functions in $C_u^s([0,1]^d)$ approximated by ReLU FNNs with width $\widetilde{N}$ and depth $\widetilde{L}$ as

$$\varepsilon_{s,d}(\widetilde{N},\widetilde{L}) \coloneqq \sup_{f\in C_u^s([0,1]^d)} \Big(\inf_{\phi\in\mathcal{NN}(\text{width}\le\widetilde{N};\,\text{depth}\le\widetilde{L})} \|\phi - f\|_{L^\infty([0,1]^d)} \Big), \quad \text{for any } \widetilde{N},\widetilde{L}\in\mathbb{N}^+,$$

where $C_u^s([0,1]^d)$ denotes the unit ball of $C^s([0,1]^d)$ defined by

$$C_u^s([0,1]^d) \coloneqq \Big\{ f\in C^s([0,1]^d) : \|\partial^{\boldsymbol{\alpha}} f\|_{L^\infty([0,1]^d)} \le 1, \text{ for all } \boldsymbol{\alpha}\in\mathbb{N}^d \text{ with } \|\boldsymbol{\alpha}\|_1 \le s \Big\}.$$

By combining the upper and lower bounds stated above, we have

$$\underbrace{C_1(s,d)\cdot\Big(\widetilde{N}^2\widetilde{L}^2\ln(\widetilde{N}\widetilde{L})\Big)^{-s/d}}_{\text{proved in Section 2.3}} \le \varepsilon_{s,d}(\widetilde{N},\widetilde{L}) \le \underbrace{C_2(s,d)\cdot\Big(\tfrac{\widetilde{N}^2\widetilde{L}^2}{(\ln\widetilde{N}\ln\widetilde{L})^2}\Big)^{-s/d}}_{\text{shown in Corollary 1.2}},$$

where $C_1(s,d)$ and $C_2(s,d)$ are two positive constants in s and d and $C_2(s,d)$ can be **explicitly** represented by s and d.

The expressiveness of deep neural networks has been studied extensively from many perspectives, e.g., in terms of combinatorics [34], topology [8], VC-dimension [7, 22, 39], fat-shattering dimension [2, 27], information theory [37], and classical approximation theory [4,5,9,12,14,15,20,21,24,29,32,35,42–45,47]. In the early works of approximation theory for neural networks, the universal approximation theorem [15, 23, 24] without approximation errors showed that, given any $\varepsilon > 0$, there exists a sufficiently large neural network approximating a target function in a certain function space within an error ε. For one-hidden-layer neural networks and functions with integral representations, Barron [5, 6] showed an asymptotic approximation error $\mathcal{O}(\frac{1}{\sqrt{N}})$ in the L^2-norm, leveraging an idea that is similar to Monte Carlo sampling for high-dimensional integrals. For very deep ReLU neural networks with width fixed as $\mathcal{O}(d)$ and depth $\mathcal{O}(L)$, Yarotsky [45, 46] showed that the nearly optimal approximation errors for Lipschitz continuous functions and functions in the unit ball of $C^s([0,1]^d)$ are $\mathcal{O}(L^{-2/d})$ and $\mathcal{O}((L/\ln L)^{-2s/d})$, respectively. Note that the results are asymptotic in the sense that L is required to be sufficiently large and the prefactors of these rates are unknown. To obtain a generic result that characterizes the approximation error for arbitrary width and depth with known prefactors to guide applications, the last three authors demonstrated in [41] that the nearly optimal approximation error for ReLU FNNs with width $\mathcal{O}(N)$ and depth $\mathcal{O}(L)$ to approximate Lipschitz continuous functions on $[0,1]^d$ is $\mathcal{O}(N^{-2/d}L^{-2/d})$. Such a nearly optimal error is further improved to an optimal one $\mathcal{O}\big((N^2L^2\ln N)^{-1/d}\big)$ in a more recent paper [42]. In this paper, we extend this generic framework to $C^s([0,1]^d)$ with a nearly optimal approximation error $\mathcal{O}(\|f\|_{C^s([0,1]^d)}N^{-2s/d}L^{-2s/d})$.

Most related works are summarized in Table 1 for the comparison of our contributions in this paper and the results in the literature.

Table 1: A summary of existing approximation errors of ReLU FNNs for $\mathrm{Lip}([0,1]^d)$ (the Lipschitz continuous function space) and $C_u^s([0,1]^d)$ (the unit ball of $C^s([0,1]^d)$).

paper	function class	width	depth	approximation error	$L^p([0,1]^d)$-norm	tightness	valid for
[44]	polynomial	$\mathcal{O}(1)$	$\mathcal{O}(L)$	$\mathcal{O}(2^{-L})$	$p=\infty$		any $L\in\mathbb{N}^+$
this paper	polynomial	$\mathcal{O}(N)$	$\mathcal{O}(L)$	$\mathcal{O}(N^{-L})$	$p=\infty$		any $N,L\in\mathbb{N}^+$
[40]	$\mathrm{Lip}([0,1]^d)$	$\mathcal{O}(N)$	3	$\mathcal{O}(N^{-2/d})$	$p\in[1,\infty)$	nearly tight in N	any $N\in\mathbb{N}^+$
[45]	$\mathrm{Lip}([0,1]^d)$	$2d+10$	$\mathcal{O}(L)$	$\mathcal{O}(L^{-2/d})$	$p=\infty$	nearly tight in L	large $L\in\mathbb{N}^+$
[41]	$\mathrm{Lip}([0,1]^d)$	$\mathcal{O}(N)$	$\mathcal{O}(L)$	$\mathcal{O}(N^{-2/d}L^{-2/d})$	$p\in[1,\infty]$	nearly tight in N and L	any $N,L\in\mathbb{N}^+$
[42]	$\mathrm{Lip}([0,1]^d)$	$\mathcal{O}(N)$	$\mathcal{O}(L)$	$\mathcal{O}\big((N^2L^2\ln N)^{-1/d}\big)$	$p\in[1,\infty]$	tight in N and L	any $N,L\in\mathbb{N}^+$
[46]	$C_u^s([0,1]^d)$	$2d+10$	$\mathcal{O}(L)$	$\mathcal{O}\big((L/\ln L)^{-2s/d}\big)$	$p=\infty$	neatly tight in L	large $L\in\mathbb{N}^+$
this paper	$C_u^s([0,1]^d)$	$\mathcal{O}(N\ln N)$	$\mathcal{O}(L\ln L)$	$\mathcal{O}(N^{-2s/d}L^{-2s/d})$	$p=\infty$	nearly tight in N and L	any $N,L\in\mathbb{N}^+$
this paper	$C_u^s([0,1]^d)$	$\mathcal{O}(N)$	$\mathcal{O}(L)$	$\mathcal{O}\big((N/\ln N)^{-2s/d}(L/\ln L)^{-2s/d}\big)$	$p=\infty$	nearly tight in N and L	any $N,L\in\mathbb{N}^+$

1.3 Discussion

We will discuss the comparison of our theory with existing works and the application scope in machine learning.

Approximation errors in $\mathcal{O}(N)$ and $\mathcal{O}(L)$ versus $\mathcal{O}(W)$

It is is fundamental and indispensable to characterize deep network approximation in terms of width $\mathcal{O}(N)$[②] and depth $\mathcal{O}(L)$ simultaneously in realistic applications, while the approximation in terms of the number of nonzero parameters W is probably only of interest in theory. First, networks used in practice are specified via width and depth and, therefore, Theorem 1.1 can provide an error bound for such networks. However, existing results in W cannot serve this purpose because they may be only valid for networks with other widths and depths. Theories in terms of W essentially have a single variable to control the network size in three types of structures: 1) a fixed width N and a varying depth L; 2) a fixed depth L and a varying width N; 3) both the width and depth are controlled by the target error ε (e.g., N is a polynomial of $\frac{1}{\varepsilon^d}$ and L is a polynomial of $\ln(\frac{1}{\varepsilon})$). Therefore, given a network with arbitrary width N and depth L, there might not be a known theory in terms of W to quantify the performance of this structure. Second, the error characterization in terms of N and L is more useful than that in terms of W, because most existing optimization and generalization analyses are based on N and L [1, 3, 10, 13, 17, 18, 25, 26], to the best of our knowledge. Approximation results in terms of N and L are more consistent with optimization and generalization analysis tools to obtain a full error analysis.

Most existing approximation theories for deep neural networks so far focus on the approximation error in the number of parameters W [4, 5, 9, 11, 12, 14, 15, 19–21, 24, 29–33, 35–38, 43–47]. Controlling two variables N and L in our theory is more challenging than controlling one variable W in the literature. The characterization of deep network approximation in terms of N and L can imply an approximation error in terms of W, while this may not be true the other way around, e.g., our theorems cannot be derived from results in [46]. Let us discuss the first type of structure mentioned in the previous paragraph, which includes the best-known result for a nearly optimal approximation error, $\mathcal{O}((W/\ln W)^{-2s/d})$, for functions in the unit ball of $C^s([0,1]^d)$ using ReLU FNNs with W parameters [46]. As an example to show how Theorem 1.1 in terms of N and

②For simplicity, we omit $\mathcal{O}(\cdot)$ in the following discussion.

L can be applied to show a similar result in terms of W. The main idea is to specify the value of N and L in Theorem 1.1 to show the desired corollary. For example, if we let $N = \mathcal{O}(1)$ in Theorem 1.1, then we have the following corollary, which is essentially equivalent to Theorem 4.1 of [46].

Corollary 1.3. *Given any function f in the unit ball of $C^s([0,1]^d)$ with $s \in \mathbb{N}^+$, there exists a function ϕ implemented by a ReLU FNN with W parameters such that*

$$\|\phi - f\|_{L^\infty([0,1]^d)} \le \mathcal{O}\big(\big(\tfrac{W}{\ln W}\big)^{-2s/d}\big), \quad \text{for large } W \in \mathbb{N}^+.$$

As we can see in this example, it is simple to derive Corollary 1.3 above and Theorem 4.1 of [46] using Theorem 1.1 in this paper. However, Theorem 1.1 cannot be derived from any existing result that characterizes approximation errors in terms of the number of parameters. Therefore, Theorem 1.1 goes beyond existing results on the approximation of deep neural networks.

Note that the logarithmic term in the approximation error is not significant in the case of $s > 1$ since it can be cancelled out in the sense that $\big(\frac{W}{\ln W}\big)^{-2s/d} \lesssim W^{-2\widetilde{s}/d}$ for any $\widetilde{s} \in (1, s)$. We remark that Theorem 3.3 of [46] provides a better approximation error by a logarithmic term: ReLU FNNs with W nonzero parameters can approximate a function f in the unit ball of $C^s([0,1]^d)$ within an error $\mathcal{O}(W^{-2s/d})$. However, the network architecture therein is relatively complex and s-dependent as stated by the authors of [46]. In fact, it contains many s-dependent blocks (sub-networks), making it difficult to implement if s is not known in applications. In contrast, our network architecture in Corollary 1.2 is simple and can be pre-specified once the width $\widetilde{N}$ and depth $\widetilde{L}$ therein are given.

Continuity of the weight selection

We would like to discuss the continuity of the weight selection as a map $\Sigma : F_{s,d} \to \mathbb{R}^W$, where $F_{s,d}$ denotes the unit ball of the d-dimensional Sobolev space with smoothness s. For a fixed network architecture with a fixed number of parameters W, let $g : \mathbb{R}^W \to C([0,1]^d)$ be the map of realizing a ReLU FNN from a given set of parameters in $\mathbb{R}^W$ to a function in $C([0,1]^d)$. Suppose that the map Σ is continuous such that $\|f - g(\Sigma(f))\|_{L^\infty([0,1]^d)} \le \varepsilon$ for all $f \in F_{s,d}$. Then $W \ge c\varepsilon^{-d/s}$ with some constant c depending only on s. This conclusion is given in Theorem 3 of [44], which is a corollary of Theorem 4.2 of [16] in a more general form. These theorems mean that the weight selection map Σ corresponding to our constructive proof in Theorem 1.1 in this paper is not continuous, since our error is better than $\mathcal{O}(W^{-s/d})$. Theorem 4.2 of [16] is essentially a min-max criterion to evaluate weight selection maps maintaining continuity: the approximation error obtained by minimizing over all continuous selections Σ and network realizations g and maximizing over all target functions is bounded below by $\mathcal{O}(W^{-s/d})$. In the worst scenario, a continuous weight selection cannot enjoy an approximation error beating $\mathcal{O}(W^{-s/d})$. However, Theorem 4.2 of [16] does not exclude the possibility that most functions of interest in practice may still enjoy a continuous weight selection with the approximation error in Theorem 1.1. It would be interesting in future work to investigate whether continuous weight selection is possible for many functions commonly encountered in real applications.

Application scope of our theory in machine learning

In deep learning, given a target function f, the final goal is to train a function $\phi(\boldsymbol{x};\boldsymbol{\theta})$ approximating f well, where $\phi(\boldsymbol{x};\boldsymbol{\theta})$ is a function in $\boldsymbol{x}\in\mathcal{X}$ realized by a network architecture parameterized with $\boldsymbol{\theta}\in\mathbb{R}^W$. To get the best solution, one needs to identify the expected risk minimizer

$$\boldsymbol{\theta}_{\mathcal{D}} \coloneqq \mathop{\arg\min}_{\boldsymbol{\theta}\in\mathbb{R}^W} R_{\mathcal{D}}(\boldsymbol{\theta}), \quad \text{where } R_{\mathcal{D}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim U(\mathcal{X})}\left[\ell\big(\phi(\boldsymbol{x};\boldsymbol{\theta}), f(\boldsymbol{x})\big)\right],$$

with a loss function usually taken as $\ell(y,y') = \frac{1}{2}|y-y'|^2$ and an unknown data distribution $U(\mathcal{X})$.

In practice, only data samples $\{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i=1}^n$ instead of f and $U(\mathcal{X})$ are available. Thus, the empirical risk minimizer $\boldsymbol{\theta}_{\mathcal{S}}$ is used to model/approximate the expected risk minimizer $\boldsymbol{\theta}_{\mathcal{D}}$, where

$$\boldsymbol{\theta}_{\mathcal{S}} \coloneqq \mathop{\arg\min}_{\boldsymbol{\theta}\in\mathbb{R}^W} R_{\mathcal{S}}(\boldsymbol{\theta}), \quad \text{where } R_{\mathcal{S}}(\boldsymbol{\theta}) \coloneqq \frac{1}{n}\sum_{i=1}^n \ell\big(\Phi(\boldsymbol{x}_i,\boldsymbol{\theta}), f(\boldsymbol{x}_i)\big). \tag{1.1}$$

In real applications, only a numerical solution (denoted as $\boldsymbol{\theta}_{\mathcal{N}}$) is achieved when a numerical optimization method is applied to solve (1.1). Hence, the actually learned function generated by the network is $\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{N}})$. Since $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}})$ is the expected inference error over all possible data samples, it can quantify how good $\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{N}})$ is. Note that

$$\begin{aligned} R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}}) &= \underbrace{[R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{N}})]}_{\text{GE}} + \underbrace{[R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{N}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{S}})]}_{\text{OE}} + \underbrace{[R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{S}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{D}})]}_{\le 0 \text{ by (1.1)}} + \underbrace{[R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{D}}) - R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})]}_{\text{GE}} + \underbrace{R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})}_{\text{AE}} \\ &\le \underbrace{R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})}_{\text{Approximation error (AE)}} + \underbrace{[R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{N}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{S}})]}_{\text{Optimization error (OE)}} + \underbrace{[R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{N}})] + [R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{D}}) - R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})]}_{\text{Generalization error (GE)}}. \end{aligned} \tag{1.2}$$

Constructive approximation provides an upper bound of $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$ in terms of the network size. For example, Theorem 1.1 and its corollaries provide an upper bound $\mathcal{O}(\|f\|_{C^s([0,1]^d)} N^{-2s/d} L^{-2s/d})$ of $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$ for $C^s([0,1]^d)$. The second term of (1.2) is bounded by the optimization error of the numerical algorithm applied to solve the empirical loss minimization problem in (1.1). The study of the bounds for the third and fourth terms is referred to as the generalization error analysis of neural networks.

One of the key targets in the area of deep learning is to develop algorithms to reduce $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}})$. Our analysis here provides an upper bound of the approximation error $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$ for smooth functions, which is crucial to control $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}})$. Instead of deriving an approximator to attain the error bound, deep learning algorithms aim to identify a solution $\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{N}})$ reducing the generalization and optimization errors in (1.2). Solutions minimizing both generalization and optimization errors will lead to a good solution only if we also have a good upper bound estimate of $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$ as shown in (1.2). Independent of whether our analysis here leads to a good approximator, which is an interesting topic to pursue, the theory here does provide a key ingredient in the error analysis of deep learning algorithms.

We would like to emphasize that the introduction of the ReLU activation function to image classification is one of the key techniques that boost the performance of deep learning [28] with surprising generalization, which is the main reason that we focus on ReLU FNNs in this paper.

Organization: The rest of the present paper is organized as follows. In Section 2, we prove Theorem 1.1 by combining two theorems (Theorems 2.1 and 2.2) that will be proved later. We will also discuss the optimality of Theorem 1.1 in Section 2. Next, Theorem 2.1 will be proved in Section 3 while Theorem 2.2 will be shown in Section 4. Several propositions supporting Theorem 2.2 will be presented in Section 5. Finally, Section 6 concludes this paper with a short discussion.

2 Approximation of smooth functions

In this section, we will prove the quantitative approximation error in Theorem 1.1 by construction and discuss its tightness. Notation throughout the proof will be summarized in Section 2.1. The proof of Theorem 1.1 is mainly based on Theorems 2.1 and 2.2, which will be proved in Sections 3 and 4, respectively. To show the tightness of Theorem 1.1, we will introduce the VC-dimension in Section 2.3.

2.1 Notation

Now let us summarize the main notation of this paper as follows.

- Let $\mathbb{R}$, $\mathbb{Q}$, and $\mathbb{Z}$ denote the set of real numbers, rational numbers, and integers, respectively.
- Let $\mathbb{N}$ and $\mathbb{N}^+$ denote the set of natural numbers and positive natural numbers, respectively. That is, $\mathbb{N}^+ = \{1,2,3,\cdots\}$ and $\mathbb{N} = \mathbb{N}^+ \bigcup \{0\}$.
- Vectors and matrices are denoted in a bold font. Standard vectorization is adopted in matrix and vector computation. For example, a scalar plus a vector means adding the scalar to each entry of the vector. Additionally, "[" and "]" are used to partition matrices (vectors) into blocks, e.g., $\boldsymbol{A} = \left[\begin{smallmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{smallmatrix}\right]$ and $\boldsymbol{v} = \begin{bmatrix} v_1 \\ \vdots \\ v_d \end{bmatrix} = [v_1,\cdots,v_d]^T \in \mathbb{R}^d$.
- Let $\mathbb{1}_S$ be the characteristic (indicator) function on a set S; i.e., $\mathbb{1}_S$ is equal to 1 on S and 0 outside S.
- Let $\mathcal{B}(\boldsymbol{x}, r) \subseteq \mathbb{R}^d$ be the closed ball with a center $\boldsymbol{x} \subseteq \mathbb{R}^d$ and a radius $r \geq 0$.
- Similar to "min" and "max", let $\mathrm{mid}(x_1,x_2,x_3)$ be the middle value of three inputs x_1, x_2, and x_3[③]. For example, $\mathrm{mid}(2,1,3) = 2$ and $\mathrm{mid}(3,2,3) = 3$.
- The set difference of two sets A and B is denoted by $A \backslash B \coloneqq \{x : x \in A,\ x \notin B\}$.
- For a real number $p \in [1,\infty)$, the p-norm of $\boldsymbol{x} = [x_1,x_2,\cdots,x_d]^T \in \mathbb{R}^d$ is defined by

$$\|\boldsymbol{x}\|_p \coloneqq \left(|x_1|^p + |x_2|^p + \cdots + |x_d|^p\right)^{1/p}.$$

③ "mid" can be defined via $\mathrm{mid}(x_1,x_2,x_3) = x_1 + x_2 + x_3 - \max(x_1,x_2,x_3) - \min(x_1,x_2,x_3)$, which can be implemented by a ReLU FNN.

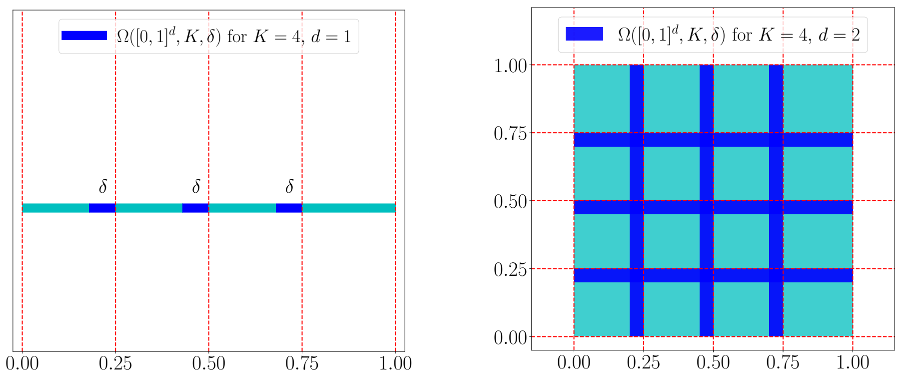
- For any $x \in \mathbb{R}$, let $\lfloor x \rfloor := \max\{n : n \le x,\ n \in \mathbb{Z}\}$ and $\lceil x \rceil := \min\{n : n \ge x,\ n \in \mathbb{Z}\}$.
- Assume $\boldsymbol{n} \in \mathbb{N}^d$; then $f(\boldsymbol{n}) = \mathcal{O}(g(\boldsymbol{n}))$ means that there exists positive C independent of $\boldsymbol{n}$, f, and g such that $f(\boldsymbol{n}) \le C g(\boldsymbol{n})$ when all entries of $\boldsymbol{n}$ go to $+\infty$.
- The modulus of continuity of a continuous function $f \in C([0,1]^d)$ is defined as

$$\omega_f(r) \coloneqq \sup\left\{|f(\boldsymbol{x}) - f(\boldsymbol{y})| : \|\boldsymbol{x} - \boldsymbol{y}\|_2 \le r,\ \boldsymbol{x}, \boldsymbol{y} \in [0,1]^d\right\}, \quad \text{for any } r \ge 0.$$

- A d-dimensional multi-index is a d-tuple $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_d]^T \in \mathbb{N}^d$. Several related notation are listed below.
  - $\|\boldsymbol{\alpha}\|_1 = |\alpha_1| + |\alpha_2| + \cdots + |\alpha_d|$;
  - $\boldsymbol{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d}$, where $\boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T$;
  - $\boldsymbol{\alpha}! = \alpha_1! \alpha_2! \cdots \alpha_d!$;
  - $\partial^{\boldsymbol{\alpha}} = \frac{\partial^{\alpha_1}}{\partial x_1^{\alpha_1}} \frac{\partial^{\alpha_2}}{\partial x_2^{\alpha_2}} \cdots \frac{\partial^{\alpha_d}}{\partial x_d^{\alpha_d}}$.
- For any closed cube $Q \subseteq \mathbb{R}^d$ and a real number $r > 0$, let rQ denote the closed cube which shares the same center of Q and whose sidelength is the product of r and the sidelength of Q.
- Given any $K \in N^+$ and $\delta \in (0, \frac{1}{K})$, define a trifling region $\Omega([0,1]^d, K, \delta)$ of $[0,1]^d$ as

$$\Omega([0,1]^d, K, \delta) \coloneqq \bigcup_{i=1}^{d} \left\{\boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T \in [0,1]^d : x_i \in \cup_{k=1}^{K-1} \left(\tfrac{k}{K} - \delta, \tfrac{k}{K}\right)\right\}. \tag{2.1}$$

  In particular, $\Omega([0,1]^d, K, \delta) = \varnothing$ if $K = 1$. See Figure 1 for two examples of the trifling region.



(a) (b)

Figure 1: Two examples of the trifling region. (a) $K = 4, d = 1$. (b) $K = 4, d = 2$.

- Given $E \subseteq \mathbb{R}^d$, let $C^s(E)$ denote the set containing all functions, all k-th order partial derivatives of which exist and are continuous on E for any $k \in \mathbb{N}$ with $0 \le k \le s$. In particular, $C^0(E)$, also denoted by $C(E)$, is the set of continuous functions on E. For the case $s=\infty$, $C^\infty(E)=\bigcap_{s=0}^\infty C^s(E)$. The C^s-norm is defined by

$$\|f\|_{C^s(E)} \coloneqq \max\left\{\|\partial^{\boldsymbol{\alpha}} f\|_{L^\infty(E)} : \boldsymbol{\alpha}\in\mathbb{N}^d \text{ with } \|\boldsymbol{\alpha}\|_1\le s\right\}.$$

Generally, E is assigned as $[0,1]^d$ in this paper. In particular, the closed unit ball of $C^s([0,1]^d)$ is denoted by

$$C^s_u([0,1]^d) \coloneqq \left\{f\in C^s([0,1]^d) : \|f\|_{C^s([0,1]^d)}\le 1\right\}.$$

- We use "$\mathcal{NN}$" as "functions implemented by ReLU FNNs" for short and use Python-type notation to specify a class of functions implemented by ReLU FNNs with several conditions. To be precise, we use $\mathcal{NN}(\mathrm{c}_1;\ \mathrm{c}_2;\ \cdots;\ \mathrm{c}_m)$ to denote the function set containing all functions implemented by ReLU FNN architectures satisfying m conditions given by $\{\mathrm{c}_i\}_{1\le i\le m}$, each of which may specify the number of inputs (#input), the number of outputs (#output), the total number of nodes in all hidden layers (#neuron), the number of hidden layers (depth), the number of total parameters (#parameter), and the width in each hidden layer (widthvec), the maximum width of all hidden layers (width), etc. For example, if $\phi\in\mathcal{NN}(\#\text{input}=2;\ \text{widthvec}=[100,100];\ \#\text{output}=1)$, then ϕ is a function satisfying the following conditions.

  - ϕ maps from $\mathbb{R}^2$ to $\mathbb{R}$.
  - ϕ is implemented by a ReLU FNN with two hidden layers and the number of nodes in each hidden layer being 100.

- Let $\sigma:\mathbb{R}\to\mathbb{R}$ denote the rectified linear unit (ReLU), i.e. $\sigma(x)=\max\{0,x\}$. With the abuse of notation, we define $\sigma:\mathbb{R}^d\to\mathbb{R}^d$ as $\sigma(\boldsymbol{x})=\begin{bmatrix}\max\{0,x_1\}\\ \vdots\\ \max\{0,x_d\}\end{bmatrix}$ for any $\boldsymbol{x}=[x_1,\cdots,x_d]^T\in\mathbb{R}^d$.

- For a function $\phi\in\mathcal{NN}(\#\text{input}=d;\ \text{widthvec}=[N_1,N_2,\cdots,N_L];\ \#\text{output}=1)$, if we set $N_0=d$ and $N_{L+1}=1$, then the architecture of the network implementing ϕ can be briefly described as follows:

$$\boldsymbol{x}=\widetilde{\boldsymbol{h}}_0 \xrightarrow{\boldsymbol{W}_0,\ \boldsymbol{b}_0} \boldsymbol{h}_1 \xrightarrow{\sigma} \widetilde{\boldsymbol{h}}_1 \quad \cdots \quad \xrightarrow{\boldsymbol{W}_{L-1},\ \boldsymbol{b}_{L-1}} \boldsymbol{h}_L \xrightarrow{\sigma} \widetilde{\boldsymbol{h}}_L \xrightarrow{\boldsymbol{W}_L,\ \boldsymbol{b}_L} \boldsymbol{h}_{L+1}=\phi(\boldsymbol{x}),$$

  where $\boldsymbol{W}_i\in\mathbb{R}^{N_{i+1}\times N_i}$ and $\boldsymbol{b}_i\in\mathbb{R}^{N_{i+1}}$ are the weight matrix and the bias vector in the i-th affine linear transform $\mathcal{L}_i$ in ϕ, respectively, i.e.,

$$\boldsymbol{h}_{i+1}=\boldsymbol{W}_i\cdot\widetilde{\boldsymbol{h}}_i+\boldsymbol{b}_i \eqqcolon \mathcal{L}_i(\widetilde{\boldsymbol{h}}_i),\quad \text{for } i=0,1,\cdots,L,$$

  and

$$\widetilde{\boldsymbol{h}}_i=\sigma(\boldsymbol{h}_i),\quad \text{for } i=1,\ \ldots,\ L.$$

  In particular, ϕ can be represented in a form of function compositions as follows

$$\phi=\mathcal{L}_L\circ\sigma\circ\mathcal{L}_{L-1}\circ\sigma\circ\ \cdots\ \circ\sigma\circ\mathcal{L}_1\circ\sigma\circ\mathcal{L}_0,$$

  which has been illustrated in Figure 2.

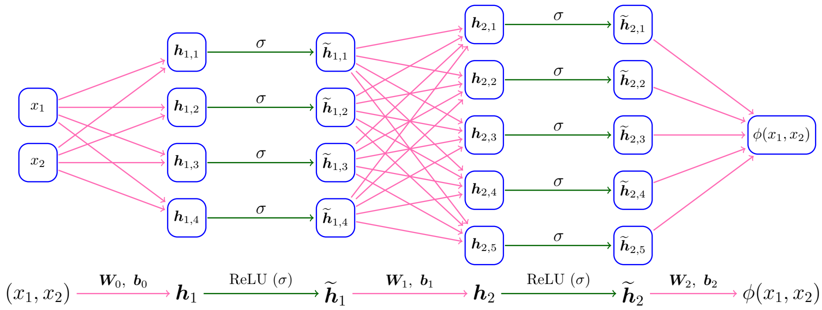


Figure 2: An example of a ReLU FNN with width 5 and depth 2.

- The expression "a network (architecture) with (of) width N and depth L" means
  - The maximum width of this network (architecture) for all **hidden** layers is no more than N.
  - The number of **hidden** layers of this network (architecture) is no more than L.
- For any $\theta \in [0,1)$, suppose its binary representation is $\theta = \sum_{\ell=1}^{\infty} \theta_\ell 2^{-\ell}$ with $\theta_\ell \in \{0,1\}$. We introduce a special notation $\text{bin}\,0.\theta_1\theta_2\cdots\theta_L$ to denote the L-term binary representation of θ, i.e., $\text{bin}\,0.\theta_1\theta_2\cdots\theta_L := \sum_{\ell=1}^{L} \theta_\ell 2^{-\ell} \approx \theta$.

2.2 Proof of Theorem 1.1

The introduction of the trifling region $\Omega([0,1]^d, K, \delta)$ is due to the fact that ReLU FNNs cannot approximate a step function uniformly well (as the ReLU activation function is continuous), which is also the reason for the main difficulty of obtaining approximation errors in the $L^\infty([0,1]^d)$-norm in our previous papers [40, 41]. The trifling region is a key technique to simplify the proofs of theories in [40, 41] as well as the proof of Theorem 1.1.

First, we present Theorem 2.1 to show that, as long as good uniform approximation by a ReLU FNN can be obtained outside the trifling region, the uniform approximation error can also be well controlled inside the trifling region when the network size is slightly increased. Second, as a simplified version of Theorem 1.1 ignoring the approximation error in the trifling region $\Omega([0,1]^d, K, \delta)$, Theorem 2.2 shows the existence of a ReLU FNN approximating a target smooth function uniformly well outside the trifling region. Finally, Theorems 2.1 and 2.2 immediately lead to Theorem 1.1. Theorem 2.1 can be applied to improve the theories in [40, 41] to obtain approximation errors in the $L^\infty([0,1]^d)$-norm.

Theorem 2.1. *Given any $\varepsilon > 0$, $N, L, K \in \mathbb{N}^+$, and $\delta \in (0, \frac{1}{3K}]$, assume $f \in C([0,1]^d)$ and $\widetilde{\phi}$ is a function implemented by a ReLU FNN with width N and depth L. If*

$$|\widetilde{\phi}(\boldsymbol{x}) - f(\boldsymbol{x})| \le \varepsilon, \quad \text{for any } \boldsymbol{x} \in [0,1]^d \backslash \Omega([0,1]^d, K, \delta),$$

then there exists a new function ϕ implemented by a ReLU FNN with width $3^d(N+4)$ and depth $L + 2d$ such that

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le \varepsilon + d \cdot \omega_f(\delta), \quad \text{for any } \boldsymbol{x} \in [0,1]^d.$$

Theorem 2.2. *Assume that $f \in C^s([0,1]^d)$ satisfies $\|\partial^{\boldsymbol{\alpha}} f\|_{L^\infty([0,1]^d)} \le 1$ for any $\boldsymbol{\alpha} \in \mathbb{N}^d$ with $\|\boldsymbol{\alpha}\|_1 \le s$. For any $N, L \in \mathbb{N}^+$, there exists a function ϕ implemented by a ReLU FNN with width $16s^{d+1}d(N+2)\log_2(8N)$ and depth $18s^2(L+2)\log_2(4L)$ such that*

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le 84(s+1)^d 8^s N^{-2s/d} L^{-2s/d}, \quad \text{for any } \boldsymbol{x} \in [0,1]^d \backslash \Omega([0,1]^d, K, \delta),$$

where $K = \lfloor N^{1/d} \rfloor^2 \lfloor L^{2/d} \rfloor$ and δ is an arbitrary number in $(0, \frac{1}{3K}]$.

We first prove Theorem 1.1 by assuming Theorems 2.1 and 2.2 are true. The proofs of Theorems 2.1 and 2.2 can be found in Sections 3 and 4, respectively.

Proof of Theorem 1.1. We may assume $\|f\|_{C^s([0,1]^d)} > 0$ since $\|f\|_{C^s([0,1]^d)} = 0$ is a trivial case. Define $\widetilde{f} \coloneqq \frac{f}{\|f\|_{C^s([0,1]^d)}} \in C_u^s([0,1]^d)$. Set $K = \lfloor N^{1/d} \rfloor^2 \lfloor L^{2/d} \rfloor$ and choose a small $\delta \in (0, \frac{1}{3K}]$ such that

$$d \cdot \omega_{\widetilde{f}}(\delta) \le N^{-2s/d} L^{-2s/d}.$$

Clearly, $\|\partial^{\boldsymbol{\alpha}} \widetilde{f}\|_{L^\infty([0,1]^d)} \le 1$ for any $\boldsymbol{\alpha} \in \mathbb{N}^d$ with $\|\boldsymbol{\alpha}\|_1 \le s$. By Theorem 2.2, there exists a function $\widehat{\phi}$ implemented by a ReLU FNN with width $16s^{d+1}d(N+2)\log_2(8N)$ and depth $18s^2(L+2)\log_2(4L)$ such that

$$|\widehat{\phi}(\boldsymbol{x}) - \widetilde{f}(\boldsymbol{x})| \le 84(s+1)^d 8^s N^{-2s/d} L^{-2s/d} \eqqcolon \varepsilon, \quad \text{for any } \boldsymbol{x} \in [0,1]^d \backslash \Omega([0,1]^d, K, \delta),$$

By Theorem 2.1, there exists a new function $\widetilde{\phi}$ implemented by a ReLU FNN with width

$$3^d\big(16s^{d+1}d(N+2)\log_2(8N) + 4\big) \le 17s^{d+1}3^d d(N+2)\log_2(8N)$$

and depth $18s^2(L+2)\log_2(4L) + 2d$ such that

$$\begin{aligned}\|\widetilde{\phi} - \widetilde{f}\|_{L^\infty([0,1]^d)} \le \varepsilon + d \cdot \omega_{\widetilde{f}}(\delta) &= 84(s+1)^d 8^s N^{-2s/d} L^{-2s/d} + d \cdot \omega_{\widetilde{f}}(\delta) \\ &\le 85(s+1)^d 8^s N^{-2s/d} L^{-2s/d}.\end{aligned}$$

Finally, set $\phi = \|f\|_{C^s([0,1]^d)} \cdot \widetilde{\phi}$; then

$$\begin{aligned}\|\phi - f\|_{L^\infty([0,1]^d)} &= \|f\|_{C^s([0,1]^d)} \cdot \|\widetilde{\phi} - \widetilde{f}\|_{L^\infty([0,1]^d)} \\ &\le 85(s+1)^d 8^s \|f\|_{C^s([0,1]^d)} N^{-2s/d} L^{-2s/d},\end{aligned}$$

and ϕ can also be implemented by a ReLU FNN with width $17s^{d+1}3^d d(N+2)\log_2(8N)$ and depth $18s^2(L+2)\log_2(4L) + 2d$. So we finish the proof. □

2.3 Optimality of Theorem 1.1

In this section, we will show that the approximation error in Theorem 1.1 is nearly tight in terms of VC-dimension. The key is the VC-dimension upper bound of ReLU FNNs in [22] will lead to a contradiction if our approximation is not optimal. This idea was used in [44] to prove its tightness for ReLU FNNs of width $\mathcal{O}(d)$ and depth sufficiently large to approximate smooth functions.

Let us first present the definitions of VC-dimension and related concepts. Let H be a class of functions mapping from a general domain $\mathcal{X}$ to $\{0,1\}$. We say H shatters the set $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m\} \subseteq \mathcal{X}$ if

$$\left|\left\{\left[h(\boldsymbol{x}_1), h(\boldsymbol{x}_2), \cdots, h(\boldsymbol{x}_m)\right]^T \in \{0,1\}^m : h \in H\right\}\right| = 2^m,$$

where $|\cdot|$ means the size of a set. This equation means, given any $\theta_i \in \{0,1\}$ for $i = 1, 2, \cdots, m$, there exists $h \in H$ such that $h(\boldsymbol{x}_i) = \theta_i$ for all i. For a general function set $\mathscr{F}$ mapping from $\mathcal{X}$ to $\mathbb{R}$, we say $\mathscr{F}$ shatters $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m\} \subseteq \mathcal{X}$ if $\mathcal{T} \circ \mathscr{F}$ does, where

$$\mathcal{T}(t) \coloneqq \begin{cases} 1, & t \ge 0, \\ 0, & t < 0 \end{cases} \quad \text{and} \quad \mathcal{T} \circ \mathscr{F} \coloneqq \{\mathcal{T} \circ f : f \in \mathscr{F}\}.$$

For any $m \in \mathbb{N}^+$, we define the growth function of H as

$$\Pi_H(m) \coloneqq \max_{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m \in \mathcal{X}} \left|\left\{\left[h(\boldsymbol{x}_1), h(\boldsymbol{x}_2), \cdots, h(\boldsymbol{x}_m)\right]^T \in \{0,1\}^m : h \in H\right\}\right|.$$

Definition 2.3 (VC-dimension)**.** Let H be a class of functions from $\mathcal{X}$ to $\{0,1\}$. The VC-dimension of H, denoted by $\mathrm{VCDim}(H)$, is the size of the largest shattered set, namely,

$$\mathrm{VCDim}(H) \coloneqq \sup\left(\{0\} \bigcup \left\{m \in \mathbb{N}^+ : \Pi_H(m) = 2^m\right\}\right).$$

Let $\mathscr{F}$ be a class of functions from $\mathcal{X}$ to $\mathbb{R}$. The VC-dimension of $\mathscr{F}$, denoted by $\mathrm{VCDim}(\mathscr{F})$, is defined by $\mathrm{VCDim}(\mathscr{F}) \coloneqq \mathrm{VCDim}(\mathcal{T} \circ \mathscr{F})$, where

$$\mathcal{T}(t) \coloneqq \begin{cases} 1, & t \ge 0, \\ 0, & t < 0 \end{cases} \quad \text{and} \quad \mathcal{T} \circ \mathscr{F} \coloneqq \{\mathcal{T} \circ f : f \in \mathscr{F}\}.$$

In particular, the expression "VC-dimension of a network (architecture)" means the VC-dimension of the function set that consists of all functions implemented by this network (architecture).

Recall that $C_u^s([0,1]^d)$ denotes the unit ball of $C^s([0,1]^d)$. Theorem 2.4 below shows that the best possible approximation error of functions in $C_u^s([0,1]^d)$ approximated by functions in $\mathscr{F}$ is bounded by a formula characterized by $\mathrm{VCDim}(\mathscr{F})$.

Theorem 2.4. *Given any $s, d \in \mathbb{N}^+$, there exists a (small) positive constant $C_{s,d}$ determined by s and d such that: For any $\varepsilon > 0$ and a function set $\mathscr{F}$ with all elements defined on $[0,1]^d$, if $\mathrm{VCDim}(\mathscr{F}) \ge 1$ and*

$$\inf_{\phi \in \mathscr{F}} \|\phi - f\|_{L^\infty([0,1]^d)} \le \varepsilon, \quad \text{for any } f \in C_u^s([0,1]^d), \tag{2.2}$$

then $\mathrm{VCDim}(\mathscr{F}) \ge C_{s,d}\, \varepsilon^{-d/s}$. [4]

[4] In fact, $C_{s,d}$ can be expressed by s and d with a **explicitly** formula as we remark in the proof of this theorem. However, the formula may be very complicated.

This theorem demonstrates the connection between the VC-dimension of $\mathscr{F}$ and the approximation error using elements of $\mathscr{F}$ to approximate functions in $C_u^s([0,1]^d)$. To be precise, the best possible approximation error is controlled by $\mathrm{VCDim}(\mathscr{F})^{-s/d}$ up to a constant. It is shown in [22] that the VC-dimension of ReLU FNNs with a fixed architecture with W parameters and L layers has an upper bound $\mathcal{O}(WL\ln W)$. It follows that the VC-dimension of ReLU FNNs with width N and depth L is bounded by $\mathcal{O}(N^2L \cdot L \cdot \ln(N^2L)) \le \mathcal{O}(N^2L^2\ln(NL))$. That is, $\mathrm{VCDim}(\mathscr{F}) \le \mathcal{O}(N^2L^2\ln(NL))$, where

$$\mathscr{F} = \mathcal{NN}(\#\text{input} = d;\ \text{width} \le N;\ \text{depth} \le L;\ \#\text{output} = 1).$$

Hence, the approximation error of functions in $C_u^s([0,1]^d)$, approximated by ReLU FNNs with width N and depth L, has a lower bound

$$C(s,d)\cdot\big(N^2L^2\ln(NL)\big)^{-s/d},$$

for some positive constant $C(s,d)$ determined by s and d. When the width and depth become $\mathcal{O}(N\ln N)$ and $\mathcal{O}(L\ln L)$, respectively, the lower bound of the approximation error becomes

$$C(s,d)\cdot\big(N^2L^2(\ln N)^3(\ln L)^3\big)^{-s/d},$$

for some positive constant $C(s,d)$ determined by s and d. These two lower bounds mean that our approximation errors in Theorem 1.1 and Corollary 1.2 are nearly optimal.

Now let us present the detailed proof of Theorem 2.4.

Proof of Theorem 2.4. To find a subset of $\mathscr{F}$ shattering $\mathcal{O}(\varepsilon^{-d/s})$ points in $[0,1]^d$, we divided the proof into two steps.

- Construct $\{f_\chi : \chi \in \mathscr{X}\} \subseteq C_u^s([0,1]^d)$ that scatters $\mathcal{O}(\varepsilon^{-d/s})$ points, where $\mathscr{X}$ is a function set defined later.

- Design $\phi_\chi \in \mathscr{F}$, for each $\chi \in \mathscr{X}$, based on f_χ and Equation (2.2) such that $\{\phi_\chi : \chi \in \mathscr{X}\} \subseteq \mathscr{F}$ also shatters $\mathcal{O}(\varepsilon^{-d/s})$ points.

The details of these two steps can be found below.

Step 1: Construct $\{f_\chi : \chi \in \mathscr{X}\} \subseteq C_u^s([0,1]^d)$ that scatters $\mathcal{O}(\varepsilon^{-d/s})$ points.

Let $K = \mathcal{O}(\varepsilon^{-1/s})$ be an integer determined later and divide $[0,1]^d$ into K^d non-overlapping sub-cubes $\{Q_{\boldsymbol{\beta}}\}_{\boldsymbol{\beta}}$ as follows:

$$Q_{\boldsymbol{\beta}} \coloneqq \Big\{\boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T \in [0,1]^d : x_i \in \big[\tfrac{\beta_i}{K}, \tfrac{\beta_i+1}{K}\big] \text{ for } i = 1,2,\cdots,d\Big\},$$

for any index vector $\boldsymbol{\beta} = [\beta_1, \beta_2, \cdots, \beta_d]^T \in \{0,1,\cdots,K-1\}^d$.

There exists $\widetilde{g} \in C^\infty(\mathbb{R}^d)$ such that $\widetilde{g}(\boldsymbol{0}) = 1$ and $\widetilde{g}(\boldsymbol{x}) = 0$ for $\|\boldsymbol{x}\|_2 \ge 1/3$.⑤ Then, $g \coloneqq \widetilde{g}/\widetilde{C}_{s,d} \in C_u^s([0,1]^d)$ by setting $\widetilde{C}_{s,d} \coloneqq \|\widetilde{g}\|_{C^s([0,1]^d)} > 0$.

Define

$$\mathscr{X} \coloneqq \big\{\chi : \chi \text{ is a map from } \{0,1,\cdots,K-1\}^d \text{ to } \{-1,1\}\big\}$$

⑤In fact, such a function $\widetilde{g}$ is called "bump function". An example can be attained by setting $\widetilde{g}(\boldsymbol{x}) = C\exp\big(\frac{1}{\|3\boldsymbol{x}\|_2^2-1}\big)$ if $\|\boldsymbol{x}\|_2 < 1/3$ and $\widetilde{g}(\boldsymbol{x}) = 0$ if $\|\boldsymbol{x}\|_2 \ge 1/3$, where C is a proper constant such that $\widetilde{g}(\boldsymbol{0}) = 1$.

and

$$g_{\boldsymbol{\beta}} \coloneqq K^{-s} g\big(K(\boldsymbol{x} - \boldsymbol{x}_{Q_{\boldsymbol{\beta}}})\big), \quad \text{for each } \boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d,$$

where $\boldsymbol{x}_{Q_\beta}$ is the center of $Q_{\boldsymbol{\beta}}$.

Next, for each $\chi \in \mathscr{X}$, we can define f_χ via

$$f_\chi(\boldsymbol{x}) \coloneqq \sum_{\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d} \chi(\boldsymbol{\beta}) g_{\boldsymbol{\beta}}(\boldsymbol{x}).$$

Then $f_\chi \in C_u^s([0,1]^d)$ for each $\chi \in \mathscr{X}$, since it satisfies the following two conditions.

- By the definition of $g_{\boldsymbol{\beta}}$ and χ, we have

$$\big\{\boldsymbol{x} : \chi(\boldsymbol{\beta}) g_{\boldsymbol{\beta}}(\boldsymbol{x}) \neq 0\big\} \subseteq \mathcal{B}(\boldsymbol{x}_{Q_{\boldsymbol{\beta}}}, \tfrac{1}{3K}) \subseteq \tfrac{2}{3} Q_{\boldsymbol{\beta}}, \quad \text{for each } \boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d,$$

  which implies that $f_\chi \in C^\infty([0,1]^d)$.

- For any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$, $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$, and $\boldsymbol{\alpha} \in \mathbb{N}^d$ with $\|\boldsymbol{\alpha}\|_1 \le s$,

$$\partial^{\boldsymbol{\alpha}} f_\chi(\boldsymbol{x}) = \chi(\boldsymbol{\beta}) \partial^{\boldsymbol{\alpha}} g_{\boldsymbol{\beta}}(\boldsymbol{x}) = K^{-s} \chi(\boldsymbol{\beta}) K^{\|\alpha\|_1} \partial^{\boldsymbol{\alpha}} g\big(K(\boldsymbol{x} - \boldsymbol{x}_{\boldsymbol{\beta}})\big),$$

  from which we deduce $|\partial^{\boldsymbol{\alpha}} f_\chi(\boldsymbol{x})| = \big|K^{-(s - \|\alpha\|_1)} \partial^{\boldsymbol{\alpha}} g\big(K(\boldsymbol{x} - \boldsymbol{x}_{\boldsymbol{\beta}})\big)\big| \le 1$.

It is easy to check that $\{f_\chi : \chi \in \mathscr{X}\} \subseteq C_u^s([0,1]^d)$ can shatter $K^d = \mathcal{O}(\varepsilon^{-d/s})$ points in $[0,1]^d$.

Step 2: Construct $\{\phi_\chi : \chi \in \mathscr{X}\}$ that also scatters $\mathcal{O}(\varepsilon^{-d/s})$ points.

By Equation (2.2), for each $\chi \in \mathscr{X}$, there exists $\phi_\chi \in \mathscr{F}$ such that

$$\|\phi_\chi - f_\chi\|_{L^\infty([0,1]^d)} \le \varepsilon + \varepsilon/2.$$

Let $\mu(\cdot)$ denote the Lebesgue measure of a set. Then, for each $\chi \in \mathscr{X}$, there exists $\mathcal{H}_\chi \subseteq [0,1]^d$ with $\mu(\mathcal{H}_\chi) = 0$ such that

$$|\phi_\chi(\boldsymbol{x}) - f_\chi(\boldsymbol{x})| \le \tfrac{3}{2}\varepsilon, \quad \text{for any } \boldsymbol{x} \in [0,1]^d \backslash \mathcal{H}_\chi.$$

Set $\mathcal{H} = \bigcup_{\chi \in \mathscr{X}} \mathcal{H}_\chi$; then we have $\mu(\mathcal{H}) = 0$ and

$$|\phi_\chi(\boldsymbol{x}) - f_\chi(\boldsymbol{x})| \le \tfrac{3}{2}\varepsilon, \quad \text{for any } \chi \in \mathscr{X} \text{ and } \boldsymbol{x} \in [0,1]^d \backslash \mathcal{H}. \tag{2.3}$$

Clearly, there exists $r \in (0,1)$ such that

$$g_{\boldsymbol{\beta}}(\boldsymbol{x}) \ge \tfrac{1}{2} g_{\boldsymbol{\beta}}(\boldsymbol{x}_{Q_{\boldsymbol{\beta}}}) > 0, \quad \text{for any } \boldsymbol{x} \in r Q_{\boldsymbol{\beta}},$$

where $\boldsymbol{x}_{Q_{\boldsymbol{\beta}}}$ is the center of $Q_{\boldsymbol{\beta}}$.

Note that $(rQ_{\boldsymbol{\beta}}) \backslash \mathcal{H}$ is not empty, since $\mu\big((rQ_{\boldsymbol{\beta}}) \backslash \mathcal{H}\big) > 0$ for each $\boldsymbol{\beta}$. Then, for any $\chi \in \mathscr{X}$ and $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$, there exists $\boldsymbol{x}_{\boldsymbol{\beta}} \in (rQ_{\boldsymbol{\beta}}) \backslash \mathcal{H}$ such that

$$|f_\chi(\boldsymbol{x}_{\boldsymbol{\beta}})| = |g_{\boldsymbol{\beta}}(\boldsymbol{x}_{\boldsymbol{\beta}})| \ge \tfrac{1}{2} |g_{\boldsymbol{\beta}}(\boldsymbol{x}_{Q_{\boldsymbol{\beta}}})| = \tfrac{1}{2} K^{-s} g(\boldsymbol{0}) = \tfrac{1}{2} K^{-s} / \widetilde{C}_{s,d} \ge 2\varepsilon, \tag{2.4}$$

where the last inequality is attained by setting $K = \lfloor (4\varepsilon \widetilde{C}_{s,d})^{-1/s} \rfloor$. Note that it is necessary to verify $K \neq 0$; we do this later in the proof.

By Equations (2.3) and (2.4), we have, for each $\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d$ and each $\chi \in \mathscr{X}$,

$$|f_\chi(\boldsymbol{x}_{\boldsymbol{\beta}})| \geq 2\varepsilon > \tfrac{3}{2}\varepsilon \geq |f_\chi(\boldsymbol{x}_{\boldsymbol{\beta}}) - \phi_\chi(\boldsymbol{x}_{\boldsymbol{\beta}})|,$$

implying $f_\chi(\boldsymbol{x}_{\boldsymbol{\beta}})$ and $\phi_\chi(\boldsymbol{x}_{\boldsymbol{\beta}})$ have the same sign. Then $\{\phi_\chi : \chi \in \mathscr{X}\}$ shatters $\big\{\boldsymbol{x}_{\boldsymbol{\beta}} : \boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d\big\}$ since $\{f_\chi : \chi \in \mathscr{X}\}$ shatters $\big\{\boldsymbol{x}_{\boldsymbol{\beta}} : \boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d\big\}$. Hence,

$$\mathrm{VCDim}(\mathscr{F}) \geq \mathrm{VCDim}\big(\{\phi_\chi : \chi \in \mathscr{X}\}\big) \geq K^d = \lfloor (4\varepsilon\widetilde{C}_{s,d})^{-1/s} \rfloor^d \geq 2^{-d}(4\varepsilon\widetilde{C}_{s,d})^{-d/s},$$

where the last inequality comes from the fact that $\lfloor x \rfloor \geq x/2$ for any $x \in [1,\infty)$.

Finally, by setting

$$C_{s,d} = 2^{-d}(4\widetilde{C}_{s,d})^{-d/s} = 2^{-d}\Big(4\|\widetilde{g}\|_{C^s([0,1]^d)}\Big)^{-d/s},$$

we have

$$\mathrm{VCDim}(\mathscr{F}) \geq 2^{-d}(4\varepsilon\widetilde{C}_{s,d})^{-d/s} = 2^{-d}(4\widetilde{C}_{s,d})^{-d/s}\varepsilon^{-d/s} = C_{s,d}\varepsilon^{-d/s}$$

and

$$K = \lfloor (4\varepsilon\widetilde{C}_{s,d})^{-1/s} \rfloor = \lfloor \varepsilon^{-1/s}(4\widetilde{C}_{s,d})^{-1/s} \rfloor = \lfloor \varepsilon^{-1/s}(2^d C_{s,d})^{1/d} \rfloor \geq 1,$$

where the last inequality comes from the assumption $\varepsilon \leq (2^d C_{s,d})^{s/d}$. Such an assumption is reasonable since $\varepsilon > (2^d C_{s,d})^{s/d}$ is a trivial case, which implies

$$\mathrm{VCDim}(\mathscr{F}) \geq 1 \geq 2^{-d} = C_{s,d}\Big((2^d C_{s,d})^{s/d}\Big)^{-d/s} > C_{s,d}\varepsilon^{-d/s}.$$

So we finish the proof. □

3 Proof of Theorem 2.1

Intuitively speaking, Theorem 2.1 shows that if a ReLU FNN can implement a function g approximating the target function f well except for the trifling region, then we can design a new ReLU network with a similar size to approximate f well on the whole domain. For example, if g approximates a one-dimensional continuous function f well except for a region in $\mathbb{R}$ with a sufficiently small measure δ, then $\mathrm{mid}\big(g(x+\delta), g(x), g(x-\delta)\big)$ can approximate f well on the whole domain, where $\mathrm{mid}(\cdot,\cdot,\cdot)$ is a function returning the middle value of three inputs and can be implemented via a ReLU FNN as shown in Lemma 3.1. This key idea is called the horizontal shift (translation) of g in this paper.

Lemma 3.1. *The middle value function* $\mathrm{mid}(x_1,x_2,x_3)$ *can be implemented by a ReLU FNN with width* 14 *and depth* 2.

Proof. Recall the fact that

$$x = \sigma(x) - \sigma(-x) \quad \text{and} \quad |x| = \sigma(x) + \sigma(-x), \quad \text{for any } x \in \mathbb{R}. \tag{3.1}$$

Therefore,

$$\begin{aligned}\max(x,y) &= \frac{x+y+|x-y|}{2} \\ &= \tfrac{1}{2}\sigma(x+y) - \tfrac{1}{2}\sigma(-x-y) + \tfrac{1}{2}\sigma(x-y) + \tfrac{1}{2}\sigma(-x+y),\end{aligned} \tag{3.2}$$

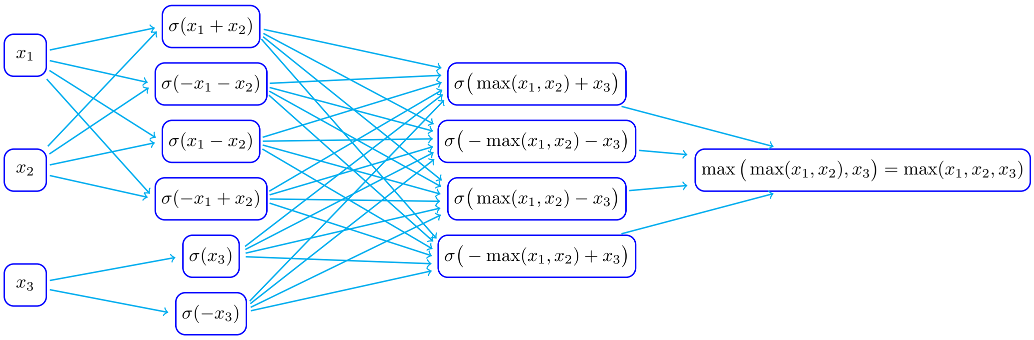


Figure 3: An illustration of the network architecture implementing $\max(x_1, x_2, x_3)$ based on Equations (3.1) and (3.2).

for any $x, y \in \mathbb{R}$. Thus, $\max(x_1, x_2, x_3)$ can be implemented by the network shown in Figure 3.

Clearly,

$$\max(x_1, x_2, x_3) \in \mathcal{NN}(\#\text{input} = 3;\ \text{widthvec} = [6, 4]).$$

Similarly, we have

$$\min(x_1, x_2, x_3) \in \mathcal{NN}(\#\text{input} = 3;\ \text{widthvec} = [6, 4]).$$

It is easy to check that

$$\begin{aligned}\operatorname{mid}(x_1, x_2, x_3) &= x_1 + x_2 + x_3 - \max(x_1, x_2, x_3) - \min(x_1, x_2, x_3)\\ &= \sigma(x_1 + x_2 + x_3) - \sigma(-x_1 - x_2 - x_3) - \max(x_1, x_2, x_3) - \min(x_1, x_2, x_3).\end{aligned}$$

Hence,

$$\operatorname{mid}(x_1, x_2, x_3) \in \mathcal{NN}(\#\text{input} = 3;\ \text{widthvec} = [14, 10]).$$

That is, $\operatorname{mid}(x_1, x_2, x_3)$ can be implemented by a ReLU FNN with width 14 and depth 2. So we finish the proof. □

The next lemma shows a simple but useful property of the $\operatorname{mid}(x_1, x_2, x_3)$ function that helps to exclude poor approximation in the trifling region.

Lemma 3.2. *For any $\varepsilon > 0$, if at least two elements of $\{x_1, x_2, x_3\}$ are in $\mathcal{B}(y, \varepsilon)$, then* $\operatorname{mid}(x_1, x_2, x_3) \in \mathcal{B}(y, \varepsilon)$.

Proof. Without loss of generality, we may assume $x_1, x_2 \in \mathcal{B}(y, \varepsilon)$ and $x_1 \le x_2$. Then the proof can be divided into three cases.

1. If $x_3 < x_1$, then $x_3 < x_1 \le x_2$, implying $\operatorname{mid}(x_1, x_2, x_3) = x_1 \in \mathcal{B}(y, \varepsilon)$.
2. If $x_1 \le x_3 \le x_2$, then $\operatorname{mid}(x_1, x_2, x_3) = x_3 \in \mathcal{B}(y, \varepsilon)$ since $y - \varepsilon \le x_1 \le x_3 \le x_2 \le y + \varepsilon$.
3. If $x_2 < x_3$, then $x_1 \le x_2 < x_3$, implying $\operatorname{mid}(x_1, x_2, x_3) = x_2 \in \mathcal{B}(y, \varepsilon)$.

So we finish the proof. □

Next, given a function g approximating f well on $[0,1]$ except for the trifling region, Lemma 3.3 below shows how to use the $\mathrm{mid}(x_1,x_2,x_3)$ function to construct a new function ϕ uniformly approximating f well on $[0,1]$, leveraging the useful property of $\mathrm{mid}(x_1,x_2,x_3)$ in Lemma 3.2.

Lemma 3.3. *Given any $\varepsilon>0$, $K\in\mathbb{N}^+$, and $\delta\in(0,\frac{1}{3K}]$, assume $f\in C([0,1])$ and $g:\mathbb{R}\to\mathbb{R}$ is a general function with*

$$|g(x)-f(x)|\le\varepsilon,\ \text{i.e., } g(x)\in\mathcal{B}\big(f(x),\varepsilon\big),\quad \text{for any } x\in[0,1]\backslash\Omega([0,1],K,\delta). \tag{3.3}$$

Then

$$|\phi(x)-f(x)|\le\varepsilon+\omega_f(\delta),\quad \text{for any } x\in[0,1],$$

where

$$\phi(x)\coloneqq\mathrm{mid}\big(g(x-\delta),g(x),g(x+\delta)\big),\quad \text{for any } x\in\mathbb{R}.$$

Proof. Divide $[0,1]$ into K small intervals denoted by $Q_k=[\frac{k}{K},\frac{k+1}{K}]$ for $k=0,1,\cdots,K-1$. For each $k\in\{0,1,\cdots,K-1\}$, we further divide Q_k into four small closed intervals as shown in Figure 4, i.e.,

$$Q_k=Q_{k,1}\bigcup Q_{k,2}\bigcup Q_{k,3}\bigcup Q_{k,4},$$

where $Q_{k,1}=[\frac{k}{K},\frac{k}{K}+\delta]$, $Q_{k,2}=[\frac{k}{K}+\delta,\frac{k+1}{K}-2\delta]$, $Q_{k,3}=[\frac{k+1}{K}-2\delta,\frac{k+1}{K}-\delta]$, and $Q_{k,4}=[\frac{k+1}{K}-\delta,\frac{k+1}{K}]$.

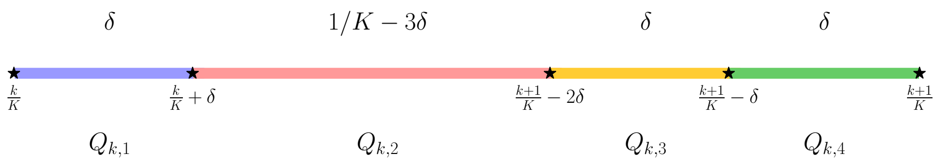


Figure 4: An illustration of $Q_{k,i}$ for $i=1,2,3,4$.

It is easy to verify that

- $Q_{k,i}\subseteq[0,1]\backslash\Omega([0,1],K,\delta)$ for $k=0,1,\cdots,K-1$ and $i=1,2,3$;
- $Q_{K-1,4}\subseteq[0,1]\backslash\Omega([0,1],K,\delta)$.

To estimate the difference between $\phi(x)$ and $f(x)$, we consider the following four cases of x in $[0,1]$ for each $k\in\{0,1,\cdots,K-1\}$.

Case 1: $x\in Q_{k,1}$.

If $x\in Q_{k,1}$, then $x\in[0,1]\backslash\Omega([0,1],K,\delta)$ and

$$x+\delta\in Q_{k,2}\bigcup Q_{k,3}\subseteq[0,1]\backslash\Omega([0,1],K,\delta).$$

It follows from Equation (3.3) that

$$g(x)\in\mathcal{B}\big(f(x),\varepsilon\big)\subseteq\mathcal{B}\big(f(x),\varepsilon+\omega_f(\delta)\big)$$

and

$$g(x+\delta)\in\mathcal{B}\big(f(x+\delta),\varepsilon\big)\subseteq\mathcal{B}\big(f(x),\varepsilon+\omega_f(\delta)\big).$$

By Lemma 3.2, we get

$$\mathrm{mid}\big(g(x-\delta), g(x), g(x+\delta)\big) \in \mathcal{B}\big(f(x), \varepsilon + \omega_f(\delta)\big).$$

Case 2: $x \in Q_{k,2}$.

If $x \in Q_{k,2}$, then

$$x-\delta, x, x+\delta \in Q_{k,1} \bigcup Q_{k,2} \bigcup Q_{k,3} \subseteq [0,1] \backslash \Omega([0,1], K, \delta).$$

It follows from Equation (3.3) that

$$g(x-\delta) \in \mathcal{B}\big(f(x-\delta), \varepsilon\big) \subseteq \mathcal{B}\big(f(x), \varepsilon + \omega_f(\delta)\big),$$

$$g(x) \in \mathcal{B}\big(f(x), \varepsilon\big) \subseteq \mathcal{B}\big(f(x), \varepsilon + \omega_f(\delta)\big),$$

and

$$g(x+\delta) \in \mathcal{B}\big(f(x+\delta), \varepsilon\big) \subseteq \mathcal{B}\big(f(x), \varepsilon + \omega_f(\delta)\big)$$

Then, by Lemma 3.2, we have

$$\mathrm{mid}\big(g(x-\delta), g(x), g(x+\delta)\big) \in \mathcal{B}\big(f(x), \varepsilon + \omega_f(\delta)\big).$$

Case 3: $x \in Q_{k,3}$.

If $x \in Q_{k,3}$, then $x \in [0,1] \backslash \Omega([0,1], K, \delta)$ and

$$x-\delta \in Q_{k,1} \bigcup Q_{k,2} \subseteq [0,1] \backslash \Omega([0,1], K, \delta).$$

It follows from Equation (3.3) that

$$g(x) \in \mathcal{B}\big(f(x), \varepsilon\big) \subseteq \mathcal{B}\big(f(x), \varepsilon + \omega_f(\delta)\big)$$

and

$$g(x-\delta) \in \mathcal{B}\big(f(x-\delta), \varepsilon\big) \subseteq \mathcal{B}\big(f(x), \varepsilon + \omega_f(\delta)\big).$$

By Lemma 3.2, we get

$$\mathrm{mid}\big(g(x-\delta), g(x), g(x+\delta)\big) \in \mathcal{B}\big(f(x), \varepsilon + \omega_f(\delta)\big).$$

Case 4: $x \in Q_{k,4}$.

If $x \in Q_{k,4}$, we can divide this case into two sub-cases.

- If $k \in \{0, 1, \cdots, K-2\}$, then $x - \delta \in Q_{k,3} \in [0,1] \backslash \Omega([0,1], K, \delta)$ and $x + \delta \in Q_{k+1,1} \subseteq [0,1] \backslash \Omega([0,1], K, \delta)$. It follows from Equation (3.3) that

$$g(x-\delta) \in \mathcal{B}\big(f(x-\delta), \varepsilon\big) \subseteq \mathcal{B}\big(f(x), \varepsilon + \omega_f(\delta)\big)$$

  and

$$g(x+\delta) \in \mathcal{B}\big(f(x+\delta), \varepsilon\big) \subseteq \mathcal{B}\big(f(x), \varepsilon + \omega_f(\delta)\big).$$

  By Lemma 3.2, we get

$$\mathrm{mid}\big(g(x-\delta), g(x), g(x+\delta)\big) \in \mathcal{B}\big(f(x), \varepsilon + \omega_f(\delta)\big).$$

- If $k = K-1$, then $x \in Q_{k,4} = Q_{K-1,4} \subseteq [0,1] \backslash \Omega([0,1], K, \delta)$ and $x - \delta \in Q_{k,3} \subseteq [0,1] \backslash \Omega([0,1], K, \delta)$. It follows from Equation (3.3) that

$$g(x) \in \mathcal{B}\big(f(x), \varepsilon\big) \subseteq \mathcal{B}\big(f(x), \varepsilon + \omega_f(\delta)\big)$$

  and

$$g(x-\delta) \in \mathcal{B}\big(f(x-\delta), \varepsilon\big) \subseteq \mathcal{B}\big(f(x), \varepsilon + \omega_f(\delta)\big).$$

  By Lemma 3.2, we get

$$\mathrm{mid}\big(g(x-\delta), g(x), g(x+\delta)\big) \in \mathcal{B}\big(f(x), \varepsilon + \omega_f(\delta)\big).$$

Since $[0,1] = \bigcup_{k=0}^{K-1} \Big(\bigcup_{i=1}^{4} Q_{k,i} \Big)$, we have

$$\mathrm{mid}\big(g(x-\delta), g(x), g(x+\delta)\big) \in \mathcal{B}\big(f(x), \varepsilon + \omega_f(\delta)\big), \quad \text{for any } x \in [0,1].$$

Recall that $\phi(x) = \mathrm{mid}\big(g(x-\delta), g(x), g(x+\delta)\big)$. Then we have

$$|\phi(x) - f(x)| \le \varepsilon + \omega_f(\delta), \quad \text{for any } x \in [0,1].$$

So we finish the proof. $\square$

The next lemma below extend Lemma 3.3 to the multidimensional case.

Lemma 3.4. *Given any $\varepsilon > 0$, $K \in \mathbb{N}^+$, and $\delta \in (0, \frac{1}{3K}]$, assume $f \in C([0,1]^d)$ and $g : \mathbb{R}^d \to \mathbb{R}$ is a general function with*

$$|g(\boldsymbol{x}) - f(\boldsymbol{x})| \le \varepsilon, \text{ i.e., } g(\boldsymbol{x}) \in \mathcal{B}\big(f(\boldsymbol{x}), \varepsilon\big), \quad \text{for any } \boldsymbol{x} \in [0,1]^d \backslash \Omega([0,1]^d, K, \delta).$$

Then

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le \varepsilon + d \cdot \omega_f(\delta), \quad \text{for any } \boldsymbol{x} \in [0,1]^d,$$

where $\phi := \phi_d$ is defined by induction through

$$\phi_{i+1}(\boldsymbol{x}) := \mathrm{mid}\big(\phi_i(\boldsymbol{x} - \delta \boldsymbol{e}_{i+1}), \phi_i(\boldsymbol{x}), \phi_i(\boldsymbol{x} + \delta \boldsymbol{e}_{i+1})\big), \quad \text{for } i = 0, 1, \cdots, d-1, \tag{3.4}$$

where $\phi_0 = g$ and $\{\boldsymbol{e}_i\}_{i=1}^d$ is the standard basis in $\mathbb{R}^d$.

Proof. For $\ell = 0, 1, \cdots, d$, we define

$$E_\ell := \left\{ \boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T : x_i \in \left\{ \begin{smallmatrix} [0,1], & \text{if } i \le \ell, \\ [0,1] \backslash \Omega([0,1], K, \delta), & \text{if } i > \ell \end{smallmatrix} \right. \right\}.$$

Clearly, $E_0 = [0,1]^d \backslash \Omega([0,1]^d, K, \delta)$ and $E_d = [0,1]^d$. See Figure 5 for the illustrations of E_ℓ for $\ell = 0, 1, \cdots, d$ when $K = 4$ and $d = 2$.

We would like to construct a sequence of functions $\phi_0, \phi_1, \cdots, \phi_d$ by induction, based on Equation (3.4), such that, for each $\ell \in \{0, 1, \cdots, d\}$,

$$\phi_\ell(\boldsymbol{x}) \in \mathcal{B}\big(f(\boldsymbol{x}), \varepsilon + \ell \cdot \omega_f(\delta)\big), \quad \text{for any } \boldsymbol{x} \in E_\ell. \tag{3.5}$$

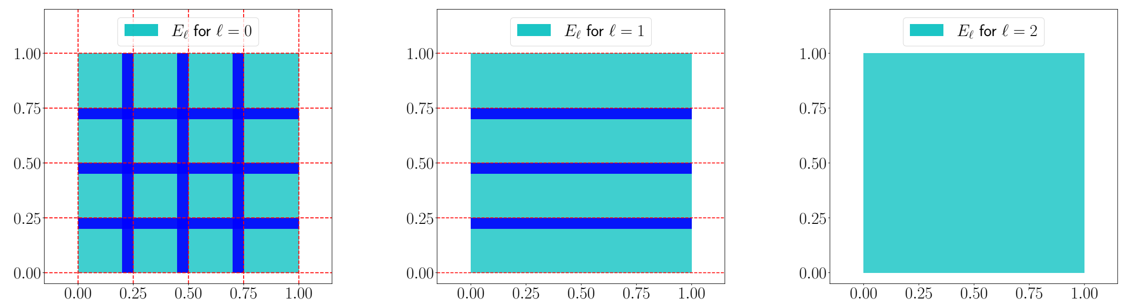


Figure 5: Illustrations of E_ℓ for $\ell = 0, 1, 2$ when $K = 4$ and $d = 2$.

Let us first consider the case $\ell = 0$. Note that $\phi_0 = g$, $E_0 = [0,1]^d \backslash \Omega([0,1]^d, K, \delta)$, and $|g(\boldsymbol{x}) - f(\boldsymbol{x})| \le \varepsilon$ for any $\boldsymbol{x} \in [0,1]^d \backslash \Omega([0,1]^d, K, \delta)$. Then we have

$$\phi_0(\boldsymbol{x}) = g(\boldsymbol{x}) \in \mathcal{B}\big(f(\boldsymbol{x}), \varepsilon\big), \quad \text{for any } \boldsymbol{x} \in E_0.$$

That is, Equation (3.5) is true for $\ell = 0$.

Now assume Equation (3.5) is true for $\ell = i$. We will prove that it also holds for $\ell = i + 1$. By the hypothesis of induction, we have

$$\phi_i(x_1, \cdots, x_i, t, x_{i+2}, \cdots, x_d) \in \mathcal{B}\Big(f(x_1, \cdots, x_i, t, x_{i+2}, \cdots, x_d), \varepsilon + i \cdot \omega_f(\delta)\Big), \tag{3.6}$$

for any $x_1, \cdots, x_i \in [0,1]$ and $t, x_{i+2}, \cdots, x_d \in [0,1] \backslash \Omega([0,1], K, \delta)$.

For fixed $x_1, \cdots, x_i \in [0,1]$ and $x_{i+2}, \cdots, x_d \in [0,1] \backslash \Omega([0,1], K, \delta)$, denote

$$\boldsymbol{x}^{[i]} \coloneqq [x_1, \cdots, x_i, x_{i+2}, \cdots, x_d]^T \in [0,1]^{d-1}.$$

Then define

$$\psi_{\boldsymbol{x}^{[i]}}(t) \coloneqq \phi_i(x_1, \cdots, x_i, t, x_{i+2}, \cdots, x_d), \quad \text{for any } t \in \mathbb{R},$$

and

$$f_{\boldsymbol{x}^{[i]}}(t) \coloneqq f(x_1, \cdots, x_i, t, x_{i+2}, \cdots, x_d), \quad \text{for any } t \in \mathbb{R}.$$

It follows from Equation (3.6) that

$$\psi_{\boldsymbol{x}^{[i]}}(t) \in \mathcal{B}\Big(f_{\boldsymbol{x}^{[i]}}(t), \varepsilon + i \cdot \omega_f(\delta)\Big), \quad \text{for any } t \in [0,1] \backslash \Omega([0,1], K, \delta).$$

Then by Lemma 3.3 (set $g = \psi_{\boldsymbol{x}^{[i]}}$ and $f = f_{\boldsymbol{x}^{[i]}}$ therein), we get, for any $t \in [0,1]$,

$$\begin{aligned}\mathrm{mid}\Big(\psi_{\boldsymbol{x}^{[i]}}(t-\delta), \psi_{\boldsymbol{x}^{[i]}}(t), \psi_{\boldsymbol{x}^{[i]}}(t+\delta)\Big) &\in \mathcal{B}\Big(f_{\boldsymbol{x}^{[i]}}(t), \varepsilon + i \cdot \omega_f(\delta) + \omega_{f_{\boldsymbol{x}^{[i]}}}(\delta)\Big) \\ &\subseteq \mathcal{B}\Big(f_{\boldsymbol{x}^{[i]}}(t), \varepsilon + (i+1)\omega_f(\delta)\Big).\end{aligned}$$

That is, for any $x_{i+1} = t \in [0,1]$,

$$\begin{aligned}&\mathrm{mid}\Big(\phi_i(x_1, \cdots, x_i, x_{i+1} - \delta, x_{i+2}, \cdots, x_d), \phi_i(x_1, \cdots, x_d), \phi_i(x_1, \cdots, x_i, x_{i+1} + \delta, x_{i+2}, \cdots, x_d)\Big) \\ &\in \mathcal{B}\Big(f(x_1, \cdots, x_d), \varepsilon + (i+1)\omega_f(\delta)\Big).\end{aligned}$$

Note that $x_1, \cdots, x_i \in [0,1]$, $x_{i+1} = t \in [0,1]$, and $x_{i+2}, \cdots, x_d \in [0,1] \backslash \Omega([0,1], K, \delta)$ are arbitrary. Thus, for any $\boldsymbol{x} \in E_{i+1}$, we have

$$\mathrm{mid}\big(\phi_i(\boldsymbol{x} - \delta \boldsymbol{e}_{i+1}), \phi_i(\boldsymbol{x}), \phi_i(\boldsymbol{x} + \delta \boldsymbol{e}_{i+1})\big) \in \mathcal{B}\big(f(\boldsymbol{x}), \varepsilon + (i+1)\omega_f(\delta)\big),$$

which implies

$$\phi_{i+1}(\boldsymbol{x}) \in \mathcal{B}\big(f(\boldsymbol{x}), \varepsilon + (i+1)\omega_f(\delta)\big), \quad \text{for any } \boldsymbol{x} \in E_{i+1}.$$

So Equation (3.5) is true for $\ell = i+1$, which means we finish the process of mathematical induction.

By the principle of induction, we have

$$\phi(\boldsymbol{x}) \coloneqq \phi_d(\boldsymbol{x}) \in \mathcal{B}\big(f(\boldsymbol{x}), \varepsilon + d \cdot \omega_f(\delta)\big), \quad \text{for any } \boldsymbol{x} \in E_d = [0,1]^d.$$

Therefore,

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le \varepsilon + d \cdot \omega_f(\delta), \quad \text{for any } \boldsymbol{x} \in [0,1]^d,$$

which means we finish the proof. □

With Lemma 3.4 in hand, we are ready to prove Theorem 2.1.

Proof of Theorem 2.1. Set $\phi_0 = \widetilde{\phi}$ and define ϕ_i for $i \in \{1, 2, \cdots, d\}$ by induction as follows:

$$\phi_{i+1}(\boldsymbol{x}) \coloneqq \mathrm{mid}\big(\phi_i(\boldsymbol{x} - \delta \boldsymbol{e}_{i+1}), \phi_i(\boldsymbol{x}), \phi_i(\boldsymbol{x} + \delta \boldsymbol{e}_{i+1})\big), \quad \text{for } i = 0, 1, \cdots, d-1,$$

where $\{\boldsymbol{e}_i\}_{i=1}^d$ is the standard basis in $\mathbb{R}^d$. Then by Lemma 3.4 with $\phi = \phi_d$, we have

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le \varepsilon + d \cdot \omega_f(\delta), \quad \text{for any } \boldsymbol{x} \in [0,1]^d.$$

It remains to determine the network architecture implementing $\phi = \phi_d$. Clearly, $\phi_0 = \widetilde{\phi} \in \mathcal{NN}(\text{width} \le N;\ \text{depth} \le L)$ implies

$$\phi_0(\cdot - \delta \boldsymbol{e}_1), \phi_0(\cdot), \phi_0(\cdot + \delta \boldsymbol{e}_1) \in \mathcal{NN}(\text{width} \le N;\ \text{depth} \le L).$$

By defining a vector-valued function $\boldsymbol{\Phi}_0 : \mathbb{R}^d \to \mathbb{R}^3$ as

$$\boldsymbol{\Phi}_0(\boldsymbol{x}) \coloneqq \big(\phi_0(\boldsymbol{x} - \delta \boldsymbol{e}_1), \phi_0(\boldsymbol{x}), \phi_0(\boldsymbol{x} + \delta \boldsymbol{e}_1)\big), \quad \text{for any } \boldsymbol{x} \in \mathbb{R}^d,$$

we have $\boldsymbol{\Phi}_0 \in \mathcal{NN}(\#\text{input} = d;\ \text{width} \le 3N;\ \text{depth} \le L;\ \#\text{output} = 3)$. Recall that $\mathrm{mid}(\cdot,\cdot,\cdot) \in \mathcal{NN}(\text{width} \le 14;\ \text{depth} \le 2)$ by Lemma 3.1. Therefore, $\phi_1 = \min(\cdot,\cdot,\cdot) \circ \boldsymbol{\Phi}_0$ can be implemented by a ReLU FNN with width $\max\{3N, 14\} \le 3(N+4)$ and depth $L+2$. Similarly, $\phi = \phi_d$ can be implemented by a ReLU FNN with width $3^d(N+4)$ and depth $L + 2d$. So we finish the proof. □

4 Proof of Theorem 2.2

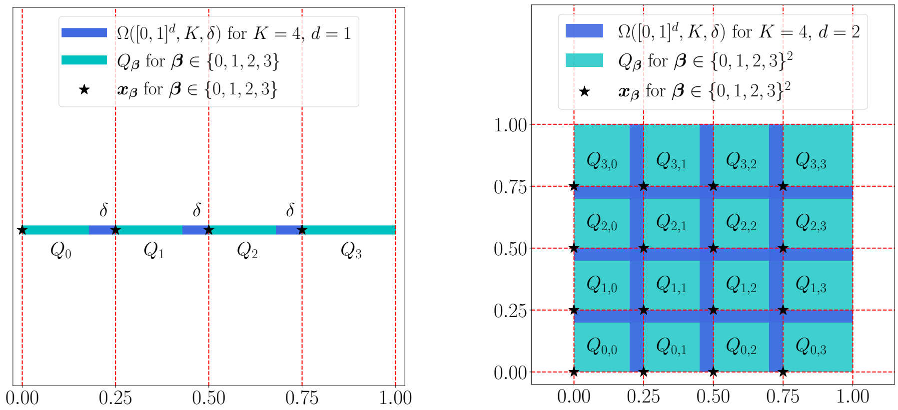
In this section, we prove Theorem 2.2, a weaker version of the main theorem of this paper (Theorem 1.1) targeting a ReLU FNN constructed to approximate a smooth function outside the trifling region. The main idea is to construct ReLU FNNs through Taylor expansions of smooth functions. We first discuss the proof sketch in Section 4.1 and give the detailed proof in Section 4.2.

4.1 Proof sketch of Theorem 2.2

Set $K = \mathcal{O}(N^{2/d}L^{2/d})$ and let $\Omega([0,1]^d, K, \delta)$ partition $[0,1]^d$ into K^d cubes $Q_{\boldsymbol{\beta}}$ for $\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d$. As we shall see later, the introduction of the trifling region $\Omega([0,1]^d, K, \delta)$ can reduce the difficulty of constructing ReLU FNNs to achieve the optimal approximation error simultaneously in width and depth, since it is only required to uniformly control the approximation error outside the trifling region and there is no requirement for the ReLU FNN inside the trifling region. In particular, for each $\boldsymbol{\beta} = [\beta_1, \beta_2, \cdots, \beta_d]^T \in \{0,1,\cdots,K-1\}^d$, we define $\boldsymbol{x}_{\boldsymbol{\beta}} := \boldsymbol{\beta}/K$ and

$$Q_{\boldsymbol{\beta}} := \Big\{\boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T : x_i \in \big[\tfrac{\beta_i}{K}, \tfrac{\beta_i+1}{K} - \delta \cdot \mathbb{1}_{\{\beta_i \le K-2\}}\big] \text{ for } i = 1,2,\cdots,d\Big\}.$$

Clearly, $[0,1]^d = \Omega([0,1]^d, K, \delta) \bigcup \big(\cup_{\boldsymbol{\beta}\in\{0,1,\cdots,K-1\}^d} Q_{\boldsymbol{\beta}}\big)$ and $\boldsymbol{x}_{\boldsymbol{\beta}}$ is the vertex of $Q_{\boldsymbol{\beta}}$ with minimum $\|\cdot\|_1$ norm. See Figure 6 for the illustrations of $Q_{\boldsymbol{\beta}}$ and $\boldsymbol{x}_{\boldsymbol{\beta}}$.



(a) (b)

Figure 6: Illustrations of $\Omega([0,1]^d, K, \delta)$, $Q_{\boldsymbol{\beta}}$, and $\boldsymbol{x}_{\boldsymbol{\beta}}$ for $\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d$. (a) $K = 4$ and $d = 1$. (b) $K = 4$ and $d = 2$.

For any $\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d$ and $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$, there exists $\xi_{\boldsymbol{x}} \in (0,1)$ such that

$$f(\boldsymbol{x}) = \underbrace{\sum_{\|\boldsymbol{\alpha}\|_1 \le s-1} \frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_{\boldsymbol{\beta}})}{\boldsymbol{\alpha}!}\boldsymbol{h}^{\boldsymbol{\alpha}}}_{\mathscr{T}_1} + \underbrace{\sum_{\|\boldsymbol{\alpha}\|_1 = s} \frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_{\boldsymbol{\beta}} + \xi_{\boldsymbol{x}}\boldsymbol{h})}{\boldsymbol{\alpha}!}\boldsymbol{h}^{\boldsymbol{\alpha}}}_{\mathscr{T}_2} =: \mathscr{T}_1 + \mathscr{T}_2, \text{⑥} \tag{4.1}$$

where $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{x}_{\boldsymbol{\beta}} = \boldsymbol{x} - \boldsymbol{\beta}/K$. Clearly, the magnitude of $\mathscr{T}_2$ is bounded by $\mathcal{O}(K^{-s}) = \mathcal{O}(N^{-2s/d}L^{-2s/d})$. So we only need to construct a ReLU FNN with width $\mathcal{O}(N \ln N)$ and depth $\mathcal{O}(L \ln L)$ to approximate

$$\mathscr{T}_1 = \sum_{\|\boldsymbol{\alpha}\|_1 \le s-1} \frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_{\boldsymbol{\beta}})}{\boldsymbol{\alpha}!}\boldsymbol{h}^{\boldsymbol{\alpha}}$$

within an error $\mathcal{O}(N^{-2s/d}L^{-2s/d})$. To approximate $\mathscr{T}_1$ well by ReLU FNNs, we need three key steps as follows.

⑥ $\sum_{\|\boldsymbol{\alpha}\|_1 = s}$ is short for $\sum_{\|\boldsymbol{\alpha}\|_1 = s,\, \boldsymbol{\alpha}\in\mathbb{N}^d}$. The same notation is used throughout this paper.

(i) Construct a ReLU FNN to implement a function $P_{\boldsymbol{\alpha}}:\mathbb{R}^d\to\mathbb{R}$ approximating the polynomial $\boldsymbol{h}^{\boldsymbol{\alpha}}$ well for each $\boldsymbol{\alpha}\in\mathbb{N}^d$ with $\|\boldsymbol{\alpha}\|_1\le s-1$.

(ii) Construct a ReLU FNN to implement a vector-valued function $\boldsymbol{\Psi}:\mathbb{R}^d\to\mathbb{R}^d$ projecting the whole cube $Q_{\boldsymbol{\beta}}$ to a point $\boldsymbol{x}_{\boldsymbol{\beta}}=\frac{\boldsymbol{\beta}}{K}$, i.e., $\boldsymbol{\Psi}(\boldsymbol{x})=\boldsymbol{x}_{\boldsymbol{\beta}}$ for any $\boldsymbol{x}\in Q_{\boldsymbol{\beta}}$ and each $\boldsymbol{\beta}\in\{0,1,\cdots,K-1\}^d$.

(iii) Construct a ReLU FNN to implement a function $\phi_{\boldsymbol{\alpha}}:\mathbb{R}^d\to\mathbb{R}$ approximating $\partial^{\boldsymbol{\alpha}}f$ via solving a point fitting problem, i.e., $\phi_{\boldsymbol{\alpha}}$ should fit $\partial^{\boldsymbol{\alpha}}f$ well at all points in $\big\{\boldsymbol{x}_{\boldsymbol{\beta}}:\boldsymbol{\beta}\in\{0,1,\cdots,K-1\}^d\big\}$ for each $\boldsymbol{\alpha}\in\mathbb{N}^d$ with $\|\boldsymbol{\alpha}\|_1\le s-1$. That is, for each $\boldsymbol{\alpha}\in\mathbb{N}^d$ with $\|\boldsymbol{\alpha}\|_1\le s-1$, we need to design $\phi_{\boldsymbol{\alpha}}$ satisfying

$$\big|\phi_{\boldsymbol{\alpha}}(\boldsymbol{x}_{\boldsymbol{\beta}})-\partial^{\boldsymbol{\alpha}}f(\boldsymbol{x}_{\boldsymbol{\beta}})\big|\le\mathcal{O}(N^{-2s/d}L^{-2s/d}),\quad\text{for any }\boldsymbol{\beta}\in\{0,1,\cdots,K-1\}^d.\tag{4.2}$$

We will establish three propositions corresponding to these three steps above. They will be applied to support the construction of the desired ReLU FNNs. Their proofs will be available in Section 5.

First, we establish a general proposition, Proposition 4.1 below, showing how to use ReLU FNNs to approximate multivariate polynomials. With Proposition 4.1 in hand, Step (i) is straightforward.

Proposition 4.1. *Assume $P(\boldsymbol{x})=\boldsymbol{x}^{\boldsymbol{\alpha}}=x_1^{\alpha_1}x_2^{\alpha_2}\cdots x_d^{\alpha_d}$ for $\boldsymbol{\alpha}\in\mathbb{N}^d$ with $\|\boldsymbol{\alpha}\|_1\le k\in\mathbb{N}^+$. For any $N,L\in\mathbb{N}^+$, there exists a function ϕ implemented by a ReLU FNN with width $9(N+1)+k-1$ and depth $7k^2L$ such that*

$$|\phi(\boldsymbol{x})-P(\boldsymbol{x})|\le 9k(N+1)^{-7kL},\quad\text{for any }\boldsymbol{x}\in[0,1]^d.$$

Proposition 4.1 shows that ReLU FNNs with width $\mathcal{O}(N)$ and depth $\mathcal{O}(L)$ is able to approximate polynomials with an error $\mathcal{O}(N^{-L})$. This reveals the power of depth in ReLU FNNs for approximating polynomials, from the perspective of function compositions. The starting point of a good approximation of functions is to approximate polynomials with high accuracy. In classical approximation theory, the approximation power of any numerical scheme depends on the degree of polynomials that can be locally reproduced. Being able to approximate polynomials by ReLU FNNs with high accuracy plays a vital role in the proof of Theorem 1.1. It is interesting to study whether there is any other function space with reasonable size, besides polynomial space, having an exponential error $\mathcal{O}(N^{-L})$ when approximated by ReLU FNNs. Obviously, the space of smooth function is too big due to the optimality of Theorem 1.1 as shown in Section 2.3.

Proposition 4.1 can be generalized to the case of polynomials defined on an arbitrary hypercube $[a,b]^d$. Let us give an example for the polynomial xy below. Its proof will be provided later in Section 5.1.

Lemma 4.2. *For any $N,L\in\mathbb{N}^+$ and $a,b\in\mathbb{R}$ with $a<b$, there exists a function ϕ implemented by a ReLU FNN with width $9N+1$ and depth L such that*

$$|\phi(x,y)-xy|\le 6(b-a)^2N^{-L},\quad\text{for any }x,y\in[a,b].$$

Second, our goal is to construct a step function $\boldsymbol{\Psi}$ mapping $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ to $\boldsymbol{x}_{\boldsymbol{\beta}} = \frac{\boldsymbol{\beta}}{K}$ for any $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$. We only need to approximate one-dimensional step functions, because in the multidimensional case we can simply set $\boldsymbol{\Psi}(\boldsymbol{x}) = [\psi(x_1), \psi(x_2), \cdots, \psi(x_d)]^T$, where ψ is a one-dimensional step function. Therefore, to implement Step (ii), we need to construct ReLU FNNs with width $\mathcal{O}(N)$ and depth $\mathcal{O}(L)$ to approximate one-dimensional step functions with $\mathcal{O}(K) = \mathcal{O}(N^{2/d}L^{2/d})$ "steps" as shown in Proposition 4.3 below.

Proposition 4.3. *For any $N, L, d \in \mathbb{N}^+$ and $\delta \in (0, \frac{1}{3K}]$ with $K = \lfloor N^{1/d} \rfloor^2 \lfloor L^{2/d} \rfloor$, there exists a one-dimensional function ϕ implemented by a ReLU FNN with width $4\lfloor N^{1/d} \rfloor + 3$ and depth $4L + 5$ such that*

$$\phi(x) = k, \quad \text{if } x \in \left[\tfrac{k}{K}, \tfrac{k+1}{K} - \delta \cdot \mathbb{1}_{\{k \le K-2\}}\right], \quad \text{for } k = 0, 1, \cdots, K-1.$$

Next, the aim of Step (iii) is to construct $\phi_{\boldsymbol{\alpha}}$ implemented by a ReLU FNN such that Equation (4.2) holds for each $\boldsymbol{\alpha}$. To this end, we establish a proposition, Proposition 4.4 below, to show that ReLU FNNs with width $\mathcal{O}(sN \ln N)$ and depth $\mathcal{O}(L \ln L)$ can be constructed to fit N^2L^2 points within an error $N^{-2s}L^{-2s}$.

Proposition 4.4. *Given any $N, L, s \in \mathbb{N}^+$ and $\xi_i \in [0, 1]$ for $i = 0, 1, \cdots, N^2L^2 - 1$, there exists a function ϕ implemented by a ReLU FNN with width $16s(N+1)\log_2(8N)$ and depth $5(L+2)\log_2(4L)$ such that*

(i) $|\phi(i) - \xi_i| \le N^{-2s}L^{-2s}$ *for* $i = 0, 1, \cdots, N^2L^2 - 1$*;*

(ii) $0 \le \phi(x) \le 1$ *for any* $x \in \mathbb{R}$.

The proofs of Propositions 4.1, 4.3, and 4.4 can be found in Sections 5.1, 5.2, and 5.3, respectively. The main ideas of proving Theorem 1.1 are summarized in Table 2.

Table 2: A list of sub-networks for approximating smooth functions. Recall that $\boldsymbol{h} = \boldsymbol{x} - \boldsymbol{\Psi}(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{x}_{\boldsymbol{\beta}}$ for $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$.

target function	function implemented by network	width	depth	approximation error
step function	$\boldsymbol{\Psi}(\boldsymbol{x})$	$\mathcal{O}(N)$	$\mathcal{O}(L)$	no error outside $\Omega([0,1]^d, K, \delta)$
$x_1 x_2$	$\varphi(x_1, x_2)$	$\mathcal{O}(N)$	$\mathcal{O}(L)$	$\mathscr{E}_1 = 216(N+1)^{-2s(L+1)}$
$\boldsymbol{h}^{\boldsymbol{\alpha}}$	$P_{\boldsymbol{\alpha}}(\boldsymbol{h})$	$\mathcal{O}(N)$	$\mathcal{O}(L)$	$\mathscr{E}_2 = 9s(N+1)^{-7sL}$
$\partial^{\boldsymbol{\alpha}} f(\boldsymbol{\Psi}(\boldsymbol{x}))$	$\phi_{\boldsymbol{\alpha}}(\boldsymbol{\Psi}(\boldsymbol{x}))$	$\mathcal{O}(N \ln N)$	$\mathcal{O}(L \ln L)$	$\mathscr{E}_3 = 2N^{-2s}L^{-2s}$
$\sum_{\|\boldsymbol{\alpha}\| \le s-1} \frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{\Psi}(\boldsymbol{x}))}{\boldsymbol{\alpha}!} \boldsymbol{h}^{\boldsymbol{\alpha}}$	$\sum_{\|\boldsymbol{\alpha}\| \le s-1} \varphi\Big(\frac{\phi_{\boldsymbol{\alpha}}(\boldsymbol{\Psi}(\boldsymbol{x}))}{\boldsymbol{\alpha}!}, P_{\boldsymbol{\alpha}}(\boldsymbol{h})\Big)$	$\mathcal{O}(N \ln N)$	$\mathcal{O}(L \ln L)$	$\mathcal{O}(\mathscr{E}_1 + \mathscr{E}_2 + \mathscr{E}_3)$
$f(\boldsymbol{x})$	$\phi(\boldsymbol{x}) \coloneqq \sum_{\|\boldsymbol{\alpha}\| \le s-1} \varphi\Big(\frac{\phi_{\boldsymbol{\alpha}}(\boldsymbol{\Psi}(\boldsymbol{x}))}{\boldsymbol{\alpha}!}, P_{\boldsymbol{\alpha}}(\boldsymbol{x} - \boldsymbol{\Psi}(\boldsymbol{x}))\Big)$	$\mathcal{O}(N \ln N)$	$\mathcal{O}(L \ln L)$	$\mathcal{O}(\|\boldsymbol{h}\|_2^{-s} + \mathscr{E}_1 + \mathscr{E}_2 + \mathscr{E}_3)$ $\le \mathcal{O}(K^{-s}) = \mathcal{O}(N^{-2s/d}L^{-2s/d})$

Finally, we would like to compare our analysis with that in [46]. Both [46] and our analysis rely on local Taylor expansions as in Equation (4.1) to approximate the target function f. Both analysis methods construct ReLU FNNs to approximate polynomials and encode the Taylor expansion coefficients into ReLU FNNs. However, the way to localize the Taylor expansion (i.e., defining the local neighborhood such that the expansion is valid) and the approach to constructing ReLU FNNs are different. We will discuss the details as follows.

Localization. In [46], a "two-scale" partition procedure and a standard triangulation divide $[0,1]^d$ into simplexes and a partition of unity is constructed using compactly supported functions that are linear on each simplex, which implies that these functions in the partition of unity can be represented by ReLU FNNs. Taylor expansions of f are constructed within each support of the functions in the partition of unity. In this paper, we simply divide the domain into small hypercubes of uniform size as visualized in Figure 6. Taylor expansions of f are constructed within each hypercubes. The reader can understand our approach as a simple way to construct a partition of unity using piecewise constant functions with binary values. The introduction of the trifling region allows us to simply construct ReLU FNNs to approximate these piecewise constant functions without caring about the approximation error within the trifling region. Hence, our construction can be much simplified and makes it easy to estimate all constant prefactors in our error estimates, which is challenging in [46].

ReLU FNNs for Taylor expansions. In [46], very deep ReLU FNNs with width $\mathcal{O}(1)$ are constructed to approximate polynomials in local Taylor expansions and, hence, the optimal approximation error in width was not explored in [46]. In this paper, we construct ReLU FNNs with arbitrary width and depth to approximate polynomials in local Taylor expansions using Proposition 4.1, which allows us to explore the optimal approximation error in width and is more challenging. In [46], the coefficients of adjacent local Taylor expansions, i.e., $\partial^{\boldsymbol{\alpha}} f$ in Equation (4.1), are encoded into ReLU FNNs via bit extraction, which is the key to achieving a better approximation error of ReLU FNNs to approximate f than the original local Taylor expansions, since the number of coefficients can be significantly reduced via encoding. Actually, the error in depth by bit extraction is nearly optimal. In this paper, the approximation to $\partial^{\boldsymbol{\alpha}} f$ is reduced to a point fitting problem that can be solved by constructing ReLU FNNs using bit extraction as sketched out in the previous paragraphs. Hence, we can also achieve the optimal approximation error in depth. The key to achieving the optimal approximation error in width in the above approximation is the application of Lemma 5.4 that essentially fits $\mathcal{O}(N^2)$ samples with ReLU FNNs of width $\mathcal{O}(N)$ and depth 2. Due to the simplicity of our analysis, we can construct ReLU FNNs with arbitrary width and depth to approximate f and specify all constant prefactors in our approximation error.

4.2 Constructive proof

According to the key ideas of proving Theorem 2.2 summarized in Section 4.1, let us present the detailed proof.

Proof of Theorem 2.2. The detailed proof can be divided into four steps as follows.

Step 1: Set up.

Set $K = \lfloor N^{1/d} \rfloor^2 \lfloor L^{2/d} \rfloor$ and let $\Omega([0,1]^d, K, \delta)$ partition $[0,1]^d$ into K^d cubes $Q_{\boldsymbol{\beta}}$ for $\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d$. In particular, for each $\boldsymbol{\beta} = [\beta_1, \beta_2, \cdots, \beta_d]^T \in \{0,1,\cdots,K-1\}^d$, we define $\boldsymbol{x}_{\boldsymbol{\beta}} \coloneqq \boldsymbol{\beta}/K$ and

$$Q_{\boldsymbol{\beta}} \coloneqq \Big\{ \boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T : x_i \in \big[\tfrac{\beta_i}{K}, \tfrac{\beta_i+1}{K} - \delta \cdot \mathbb{1}_{\{\beta_i \le K-2\}}\big] \text{ for } i = 1,2,\cdots,d \Big\}.$$

Clearly, $[0,1]^d = \Omega([0,1]^d, K, \delta) \bigcup \big(\cup_{\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d} Q_{\boldsymbol{\beta}} \big)$ and $\boldsymbol{x}_{\boldsymbol{\beta}}$ is the vertex of $Q_{\boldsymbol{\beta}}$ with minimum $\|\cdot\|_1$ norm. See Figure 6 for the illustrations of $Q_{\boldsymbol{\beta}}$ and $\boldsymbol{x}_{\boldsymbol{\beta}}$.

By Proposition 4.3, there exists $\psi \in \mathcal{NN}(\text{width} \le 4N+3;\ \text{depth} \le 4N+5)$ such that

$$\psi(x) = k, \quad \text{if } x \in \left[\tfrac{k}{K}, \tfrac{k+1}{K} - \delta \cdot \mathbb{1}_{\{k \le K-2\}}\right], \quad \text{for } k = 0, 1, \cdots, K-1.$$

Then for each $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$, $\psi(x_i) = \beta_i$ for all $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ for $i = 1, 2, \cdots, d$.

Define

$$\boldsymbol{\Psi}(\boldsymbol{x}) \coloneqq \big[\psi(x_1), \psi(x_2), \cdots, \psi(x_d)\big]^T / K, \quad \text{for any } \boldsymbol{x} \in [0,1]^d,$$

then

$$\boldsymbol{\Psi}(\boldsymbol{x}) = \boldsymbol{\beta}/K = \boldsymbol{x}_{\boldsymbol{\beta}}, \quad \text{if } \boldsymbol{x} \in Q_{\boldsymbol{\beta}}, \quad \text{for } \boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d.$$

For any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$, by the Taylor expansion, there exists $\xi_{\boldsymbol{x}} \in (0,1)$ such that

$$f(\boldsymbol{x}) = \sum_{\|\boldsymbol{\alpha}\|_1 \le s-1} \tfrac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{\Psi}(\boldsymbol{x}))}{\boldsymbol{\alpha}!} \boldsymbol{h}^{\boldsymbol{\alpha}} + \sum_{\|\boldsymbol{\alpha}\|_1 = s} \tfrac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{\Psi}(\boldsymbol{x}) + \xi_{\boldsymbol{x}} \boldsymbol{h})}{\boldsymbol{\alpha}!} \boldsymbol{h}^{\boldsymbol{\alpha}}, \quad \text{where } \boldsymbol{h} = \boldsymbol{x} - \boldsymbol{\Psi}(\boldsymbol{x}).$$

Step 2: Construct the desired function ϕ.

By Lemma 4.2, there exists

$$\varphi \in \mathcal{NN}\big(\text{width} \le 9(N+1)+1;\ \text{depth} \le 2s(L+1)\big)$$

such that

$$|\varphi(x_1, x_2) - x_1 x_2| \le 216(N+1)^{-2s(L+1)} \eqqcolon \mathscr{E}_1, \quad \text{for any } x_1, x_2 \in [-3, 3]. \tag{4.3}$$

For each $\boldsymbol{\alpha} \in \mathbb{N}^d$ with $\|\boldsymbol{\alpha}\|_1 \le s$, by Proposition 4.1, there exists

$$P_{\boldsymbol{\alpha}} \in \mathcal{NN}\big(\text{width} \le 9(N+1)+s-1;\ \text{depth} \le 7s^2 L\big)$$

such that

$$|P_{\alpha}(\boldsymbol{x}) - \boldsymbol{x}^{\boldsymbol{\alpha}}| \le 9s(N+1)^{-7sL} \eqqcolon \mathscr{E}_2, \quad \text{for any } \boldsymbol{x} \in [0,1]^d. \tag{4.4}$$

For each $i \in \{0, 1, \cdots, K^d - 1\}$, define

$$\boldsymbol{\eta}(i) = [\eta_1, \eta_2, \cdots, \eta_d]^T \in \{0, 1, \cdots, K-1\}^d$$

such that $\sum_{j=1}^d \eta_j K^{j-1} = i$. Such a map $\boldsymbol{\eta}$ is a bijection from $\{0, 1, \cdots, K^d - 1\}$ to $\{0, 1, \cdots, K-1\}^d$. For each $\boldsymbol{\alpha} \in \mathbb{N}^d$ with $\|\boldsymbol{\alpha}\|_1 \le s-1$, define

$$\xi_{\boldsymbol{\alpha}, i} = \big(\partial^{\boldsymbol{\alpha}} f(\tfrac{\boldsymbol{\eta}(i)}{K}) + 1\big)/2, \quad \text{for } i \in \{0, 1, \cdots, K^d - 1\}.$$

Then $\|\partial^{\boldsymbol{\alpha}} f\|_{L^\infty([0,1]^d)} \le 1$ implies $\xi_{\boldsymbol{\alpha}, i} \in [0,1]$ for $i = 0, 1, \cdots, K^d - 1$ and each $\boldsymbol{\alpha}$. Note that $K^d = \big(\lfloor N^{1/d} \rfloor^2 \lfloor L^{2/d} \rfloor\big)^d \le N^2 L^2$. By Proposition 4.4, there exists

$$\widetilde{\phi}_{\boldsymbol{\alpha}} \in \mathcal{NN}\big(\text{width} \le 16s(N+1)\log_2(8N);\ \text{depth} \le 5(L+2)\log_2(4L)\big)$$

such that, for each $\boldsymbol{\alpha} \in \mathbb{N}^d$ with $\|\boldsymbol{\alpha}\|_1 \le s-1$, we have

$$|\widetilde{\phi}_{\boldsymbol{\alpha}}(i) - \xi_{\boldsymbol{\alpha}, i}| \le N^{-2s} L^{-2s}, \quad \text{for } i = 0, 1, \cdots, K^d - 1.$$

For each $\boldsymbol{\alpha} \in \mathbb{N}^d$ with $\|\boldsymbol{\alpha}\|_1 \le s-1$, define

$$\phi_{\boldsymbol{\alpha}}(\boldsymbol{x}) \coloneqq 2\widetilde{\phi}_{\boldsymbol{\alpha}}\Big(\sum_{j=1}^{d} x_j K^{j-1}\Big) - 1, \quad \text{for any } \boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T \in \mathbb{R}^d.$$

It is easy to verify that

$$\phi_{\boldsymbol{\alpha}} \in \mathcal{NN}\big(\text{width} \le 16s(N+1)\log_2(8N);\ \text{depth} \le 5(L+2)\log_2(4L)\big).$$

Then, for each $\boldsymbol{\alpha} \in \mathbb{N}^d$ with $\|\boldsymbol{\alpha}\|_1 \le s-1$ and each $\boldsymbol{\eta} = \boldsymbol{\eta}(i) = [\eta_1, \eta_2, \cdots, \eta_d]^T \in \{0, 1, \cdots, K-1\}^d$ corresponding to $i = \sum_{j=1}^d \eta_j K^{j-1} \in \{0, 1, \cdots, K^d - 1\}$, we have

$$\begin{aligned} \big|\phi_{\boldsymbol{\alpha}}(\tfrac{\boldsymbol{\eta}}{K}) - \partial^{\boldsymbol{\alpha}} f(\tfrac{\boldsymbol{\eta}}{K})\big| &= \Big|2\widetilde{\phi}_{\boldsymbol{\alpha}}\Big(\sum_{j=1}^{d} \eta_j K^{j-1}\Big) - 1 - (2\xi_{\boldsymbol{\alpha},i} - 1)\Big| \\ &= 2\big|\widetilde{\phi}_{\boldsymbol{\alpha}}(i) - \xi_{\boldsymbol{\alpha},i}\big| \le 2N^{-2s}L^{-2s}. \end{aligned}$$

Therefore, for each $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$ and each $\boldsymbol{\alpha} \in \mathbb{N}^d$ with $\|\boldsymbol{\alpha}\|_1 \le s-1$, we have

$$\big|\phi_{\boldsymbol{\alpha}}(\boldsymbol{x}_{\boldsymbol{\beta}}) - \partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_{\boldsymbol{\beta}})\big| = \big|\phi_{\boldsymbol{\alpha}}(\tfrac{\boldsymbol{\beta}}{K}) - \partial^{\boldsymbol{\alpha}} f(\tfrac{\boldsymbol{\beta}}{K})\big| \le 2N^{-2s}L^{-2s} \eqqcolon \mathscr{E}_3. \tag{4.5}$$

Now we can construct the desired function ϕ as

$$\phi(\boldsymbol{x}) \coloneqq \sum_{\|\boldsymbol{\alpha}\|_1 \le s-1} \varphi\Big(\tfrac{\phi_{\boldsymbol{\alpha}}(\boldsymbol{\Psi}(\boldsymbol{x}))}{\boldsymbol{\alpha}!}, P_{\boldsymbol{\alpha}}\big(\boldsymbol{x} - \boldsymbol{\Psi}(\boldsymbol{x})\big)\Big), \quad \text{for any } \boldsymbol{x} \in \mathbb{R}^d. \tag{4.6}$$

It remains to estimate the approximation error and determine the size of the network implementing ϕ.

Step 3: Estimate approximation error.

Fix $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$, let us estimate the approximation error for a fixed $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$. See Table 2 for a summary of the approximation errors. Recall that $\boldsymbol{\Psi}(\boldsymbol{x}) = \boldsymbol{x}_{\boldsymbol{\beta}}$ and $\boldsymbol{h} = \boldsymbol{x} - \boldsymbol{\Psi}(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{x}_{\boldsymbol{\beta}}$. It is easy to check that $|f(\boldsymbol{x}) - \phi(\boldsymbol{x})|$ is bounded by

$$\begin{aligned} &\Big|\sum_{\|\boldsymbol{\alpha}\|_1 \le s-1} \tfrac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{\Psi}(\boldsymbol{x}))}{\boldsymbol{\alpha}!}\boldsymbol{h}^{\boldsymbol{\alpha}} + \sum_{\|\boldsymbol{\alpha}\|_1 = s} \tfrac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{\Psi}(\boldsymbol{x}) + \xi_{\boldsymbol{x}}\boldsymbol{h})}{\boldsymbol{\alpha}!}\boldsymbol{h}^{\boldsymbol{\alpha}} - \sum_{\|\boldsymbol{\alpha}\|_1 \le s-1} \varphi\Big(\tfrac{\phi_{\boldsymbol{\alpha}}(\boldsymbol{\Psi}(\boldsymbol{x}))}{\boldsymbol{\alpha}!}, P_{\boldsymbol{\alpha}}\big(\boldsymbol{x} - \boldsymbol{\Psi}(\boldsymbol{x})\big)\Big)\Big| \\ \le &\underbrace{\sum_{\|\boldsymbol{\alpha}\|_1 = s} \Big|\tfrac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_{\boldsymbol{\beta}} + \xi_{\boldsymbol{x}}\boldsymbol{h})}{\boldsymbol{\alpha}!}\boldsymbol{h}^{\boldsymbol{\alpha}}\Big|}_{\mathscr{I}_1} + \underbrace{\sum_{\|\boldsymbol{\alpha}\|_1 \le s-1} \Big|\tfrac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_{\boldsymbol{\beta}})}{\boldsymbol{\alpha}!}\boldsymbol{h}^{\boldsymbol{\alpha}} - \varphi\Big(\tfrac{\phi_{\boldsymbol{\alpha}}(\boldsymbol{x}_{\boldsymbol{\beta}})}{\boldsymbol{\alpha}!}, P_{\boldsymbol{\alpha}}(\boldsymbol{h})\Big)\Big|}_{\mathscr{I}_2} \eqqcolon \mathscr{I}_1 + \mathscr{I}_2. \end{aligned}$$

Recall the fact that

$$\sum_{\|\boldsymbol{\alpha}\|_1 = s} 1 = \Big|\big\{\boldsymbol{\alpha} \in \mathbb{N}^d : \|\boldsymbol{\alpha}\|_1 = s\big\}\Big| \le (s+1)^{d-1}$$ [7]

and

$$\sum_{\|\boldsymbol{\alpha}\|_1 \le s-1} 1 = \sum_{i=0}^{s-1}\Big(\sum_{\|\boldsymbol{\alpha}\|_1 = i} 1\Big) \le \sum_{i=0}^{s-1}(i+1)^{d-1} \le s \cdot (s-1+1)^{d-1} = s^d.$$

[7] In fact, we have $\big|\{\boldsymbol{\alpha} \in \mathbb{N}^d : \|\boldsymbol{\alpha}\|_1 = s\}\big| = \binom{s+d-1}{d-1}$, implying $(s/d+1)^{d-1} \le \sum_{\|\boldsymbol{\alpha}\|_1 = s} 1 \le (s+1)^{d-1}$. Thus, the lower bound of the estimate is still exponentially large in d. To the best of our knowledge, we cannot avoid a constant prefactor that is exponentially large in d when Taylor expansion is used in the analysis.

For the first part $\mathscr{I}_1$, we have

$$\mathscr{I}_1 = \sum_{\|\boldsymbol{\alpha}\|_1=s} \left| \frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_\beta+\xi_x\boldsymbol{h})}{\boldsymbol{\alpha}!}\boldsymbol{h}^{\boldsymbol{\alpha}} \right| \le \sum_{\|\boldsymbol{\alpha}\|_1=s} \left| \frac{1}{\boldsymbol{\alpha}!}\boldsymbol{h}^{\boldsymbol{\alpha}} \right| \le (s+1)^{d-1}K^{-s}.$$

For the second part $\mathscr{I}_2$, we have

$$\mathscr{I}_2 = \sum_{\|\boldsymbol{\alpha}\|_1\le s-1} \underbrace{\left| \frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!}\boldsymbol{h}^{\boldsymbol{\alpha}} - \varphi\Big(\frac{\phi_{\boldsymbol{\alpha}}(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!}, P_{\boldsymbol{\alpha}}(\boldsymbol{h})\Big)\right|}_{\mathscr{I}_2(\boldsymbol{\alpha})} =: \sum_{\|\boldsymbol{\alpha}\|_1\le s-1} \mathscr{I}_2(\boldsymbol{\alpha}).$$

Fix $\boldsymbol{\alpha}\in\mathbb{N}^d$ with $\|\boldsymbol{\alpha}\|_1\le s-1$, we have

$$\begin{aligned}
\mathscr{I}_2(\boldsymbol{\alpha}) &= \left| \frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!}\boldsymbol{h}^{\boldsymbol{\alpha}} - \varphi\Big(\frac{\phi_{\boldsymbol{\alpha}}(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!}, P_{\boldsymbol{\alpha}}(\boldsymbol{h})\Big)\right| \\
&\le \underbrace{\left| \frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!}\boldsymbol{h}^{\boldsymbol{\alpha}} - \varphi\Big(\frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!}, P_{\boldsymbol{\alpha}}(\boldsymbol{h})\Big)\right|}_{\mathscr{I}_{2,1}(\boldsymbol{\alpha})} + \underbrace{\left| \varphi\Big(\frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!}, P_{\boldsymbol{\alpha}}(\boldsymbol{h})\Big) - \varphi\Big(\frac{\phi_{\boldsymbol{\alpha}}(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!}, P_{\boldsymbol{\alpha}}(\boldsymbol{h})\Big)\right|}_{\mathscr{I}_{2,2}(\boldsymbol{\alpha})} \\
&=: \mathscr{I}_{2,1}(\boldsymbol{\alpha}) + \mathscr{I}_{2,2}(\boldsymbol{\alpha}).
\end{aligned}$$

Note that $\mathscr{E}_2 = 9s(N+1)^{-7sL} \le 9s(2)^{-7s} \le 2$. By $\boldsymbol{h}^{\boldsymbol{\alpha}}\in[0,1]$ and Equation (4.4), we have $P_{\boldsymbol{\alpha}}(\boldsymbol{h})\in[-2,3]\subseteq[-3,3]$. Then by $\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_\beta)\in[-1,1]$ and Equations (4.3) and (4.4), we have

$$\begin{aligned}
\mathscr{I}_{2,1}(\boldsymbol{\alpha}) &= \left| \frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!}\boldsymbol{h}^{\boldsymbol{\alpha}} - \varphi\Big(\frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!}, P_{\boldsymbol{\alpha}}(\boldsymbol{h})\Big)\right| \\
&\le \left| \frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!}\boldsymbol{h}^{\boldsymbol{\alpha}} - \frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!}P_{\boldsymbol{\alpha}}(\boldsymbol{h})\right| + \underbrace{\left| \frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!}P_{\boldsymbol{\alpha}}(\boldsymbol{h}) - \varphi\Big(\frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!}, P_{\boldsymbol{\alpha}}(\boldsymbol{h})\Big)\right|}_{\le \mathscr{E}_1 \text{ by Eq. (4.3)}} \\
&\le \frac{1}{\boldsymbol{\alpha}!}\underbrace{\left|\boldsymbol{h}^{\boldsymbol{\alpha}} - P_{\boldsymbol{\alpha}}(\boldsymbol{h})\right|}_{\le \mathscr{E}_2 \text{ by Eq. (4.4)}} + \mathscr{E}_1 \le \frac{1}{\boldsymbol{\alpha}!}\mathscr{E}_2 + \mathscr{E}_1 \le \mathscr{E}_1 + \mathscr{E}_2.
\end{aligned}$$

To estimate $\mathscr{I}_{2,2}(\boldsymbol{\alpha})$, we need the following fact derived from Equation (4.3):

$$\begin{aligned}
|\varphi(x_1,x_2) - \varphi(\widetilde{x}_1,x_2)| &\le \underbrace{|\varphi(x_1,x_2) - x_1x_2|}_{\le \mathscr{E}_1 \text{ by Eq. (4.3)}} + \underbrace{|\varphi(\widetilde{x}_1,x_2) - \widetilde{x}_1x_2|}_{\le \mathscr{E}_1 \text{ by Eq. (4.3)}} + |x_1x_2 - \widetilde{x}_1x_2| \\
&\le 2\mathscr{E}_1 + 3|x_1 - \widetilde{x}_1|,
\end{aligned} \tag{4.7}$$

for any $x_1, \widetilde{x}_1, x_2 \in [-3,3]$.

Since $\mathscr{E}_3 = 2N^{-2s}L^{-2s} \le 2$ and $\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_\beta)\in[-1,1]$, we have $\phi_{\boldsymbol{\alpha}}(\boldsymbol{x}_\beta)\in[-3,3]$ by Equation (4.5). Then by $P_{\boldsymbol{\alpha}}(\boldsymbol{h})\in[-3,3]$ and Equations (4.7) and (4.5), we have

$$\begin{aligned}
\mathscr{I}_{2,2}(\boldsymbol{\alpha}) &= \left| \varphi\Big(\frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!}, P_{\boldsymbol{\alpha}}(\boldsymbol{h})\Big) - \varphi\Big(\frac{\phi_{\boldsymbol{\alpha}}(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!}, P_{\boldsymbol{\alpha}}(\boldsymbol{h})\Big)\right| \\
&\le 2\mathscr{E}_1 + 3\underbrace{\left| \frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!} - \frac{\phi_{\boldsymbol{\alpha}}(\boldsymbol{x}_\beta)}{\boldsymbol{\alpha}!}\right|}_{\le \mathscr{E}_3 \text{ by Eq. (4.5)}} \le 2\mathscr{E}_1 + 3\mathscr{E}_3.
\end{aligned}$$

Therefore, we get

$$\begin{aligned}
|f(\boldsymbol{x})-\phi(\boldsymbol{x})| &\le \mathscr{I}_1+\mathscr{I}_2 \le \mathscr{I}_1+\sum_{\|\boldsymbol{\alpha}\|_1\le s-1}\mathscr{I}_2(\boldsymbol{\alpha}) \le \mathscr{I}_1+\sum_{\|\boldsymbol{\alpha}\|_1\le s-1}\Big(\mathscr{I}_{2,1}(\boldsymbol{\alpha})+\mathscr{I}_{2,2}(\boldsymbol{\alpha})\Big)\\
&\le (s+1)^{d-1}K^{-s}+s^d\Big((\mathscr{E}_1+\mathscr{E}_2)+(2\mathscr{E}_1+3\mathscr{E}_3)\Big)\\
&\le (s+1)^d(K^{-s}+3\mathscr{E}_1+\mathscr{E}_2+3\mathscr{E}_3).
\end{aligned}$$

Since $\boldsymbol{\beta}\in\{0,1,\cdots,K-1\}^d$ and $\boldsymbol{x}\in Q_{\boldsymbol{\beta}}$ are arbitrary and

$$[0,1]^d=\Omega([0,1]^d,K,\delta)\bigcup\Big(\cup_{\boldsymbol{\beta}\in\{0,1,\cdots,K-1\}^d}Q_{\boldsymbol{\beta}}\Big),$$

we have, for any $\boldsymbol{x}\in[0,1]^d\backslash\Omega([0,1]^d,K,\delta)$,

$$|f(\boldsymbol{x})-\phi(\boldsymbol{x})|\le(s+1)^d(K^{-s}+3\mathscr{E}_1+\mathscr{E}_2+3\mathscr{E}_3).$$

Recall that $K=\lfloor N^{1/d}\rfloor^2\lfloor L^{2/d}\rfloor\ge\frac{N^{2/d}L^{2/d}}{8}$ and

$$(N+1)^{-7sL}\le(N+1)^{-2s(L+1)}\le(N+1)^{-2s}2^{-2sL}\le N^{-2s}L^{-2s}.$$

Then we have

$$\begin{aligned}
&(s+1)^d(K^{-s}+3\mathscr{E}_1+\mathscr{E}_2+3\mathscr{E}_3)\\
&=(s+1)^d\Big(K^{-s}+648(N+1)^{-2s(L+1)}+9s(N+1)^{-7sL}+6N^{-2s}L^{-2s}\Big)\\
&\le(s+1)^d\Big(8^sN^{-2s/d}L^{-2s/d}+(654+9s)N^{-2s}L^{-2s}\Big)\\
&\le(s+1)^d(8^s+654+9s)N^{-2s/d}L^{-2s/d}\le 84(s+1)^d8^sN^{-2s/d}L^{-2s/d}.
\end{aligned}$$

Step 4: Determine the size of the network implementing ϕ.

It remains to estimate the width and depth of the network implementing ϕ. Recall that, for $\boldsymbol{\alpha}\in\mathbb{N}^d$ with $\|\boldsymbol{\alpha}\|_1\le s-1$,

$$\begin{cases}
\boldsymbol{\Psi}\in\mathcal{NN}\big(\text{width}\le d(4N+3);\ \text{depth}\le 4L+5\big),\\
\phi_{\boldsymbol{\alpha}}\in\mathcal{NN}\big(\text{width}\le 16s(N+1)\log_2(8N);\ \text{depth}\le 5(L+2)\log_2(4L)\big),\\
P_{\boldsymbol{\alpha}}\in\mathcal{NN}\big(\text{width}\le 9(N+1)+s-1;\ \text{depth}\le 7s^2L\big),\\
\varphi\in\mathcal{NN}\big(\text{width}\le 9(N+1)+1;\ \text{depth}\le 2s(L+1)\big).
\end{cases}$$

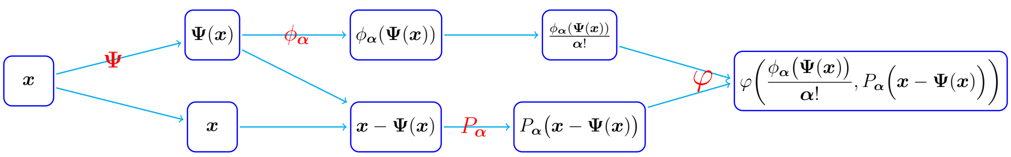


Figure 7: An illustration of the sub-network architecture implementing each component of ϕ, $\varphi\Big(\frac{\phi_{\boldsymbol{\alpha}}(\boldsymbol{\Psi}(\boldsymbol{x}))}{\boldsymbol{\alpha}!},P_{\boldsymbol{\alpha}}\big(\boldsymbol{x}-\boldsymbol{\Psi}(\boldsymbol{x})\big)\Big)$ for each $\boldsymbol{\alpha}\in\mathbb{N}^d$ with $\|\boldsymbol{\alpha}\|\le s-1$.

By Equation (4.6) and Figure 7, it easy to verify that ϕ can be implemented by a ReLU FNN with width

$$\sum_{\|\boldsymbol{\alpha}\|_1\le s-1} 16sd(N+2)\log_2(8N) \le s^d\cdot 16sd(N+2)\log_2(8N)$$
$$= 16s^{d+1}d(N+2)\log_2(8N)$$

and depth

$$(4L+5)+2s(L+1)+7s^2L+5(L+2)\log_2(4L)+3 \le 18s^2(L+2)\log_2(4L)$$

as desired. So we finish the proof. □

5 Proofs of Propositions in Section 4.1

In this section, we will prove all propositions in Section 4.1.

5.1 Proof of Proposition 4.1 for polynomial approximation

To prove Proposition 4.1, we will construct ReLU FNNs to approximate multivariate polynomials following the four steps below.

- $f(x)=x^2$. We approximate $f(x)=x^2$ by the combinations and compositions of "sawtooth" functions as shown in Figures 8 and 9.
- $f(x,y)=xy$. To approximate $f(x,y)=xy$, we use the result of the previous step and the fact that $xy = 2\big((\frac{x+y}{2})^2-(\frac{x}{2})^2-(\frac{y}{2})^2\big)$.
- $f(x_1,x_2,\cdots,x_k)=x_1x_2\cdots x_k$. We approximate $f(x_1,x_2,\cdots,x_k)=x_1x_2\cdots x_k$ for any $k\ge 2$ via mathematical induction based on the result of the previous step.
- A general polynomial $P(\boldsymbol{x})=\boldsymbol{x}^{\boldsymbol{\alpha}}=x_1^{\alpha_1}x_2^{\alpha_2}\cdots x_d^{\alpha_d}$ with $\|\boldsymbol{\alpha}\|_1\le k$. Any one-term polynomial of degree $\le k$ can be written as $Cz_1z_2\cdots z_k$ with some entries equaling 1, where C is a constant and $\boldsymbol{z}=[z_1,z_2,\cdots,z_k]^T$ can be attained via an affine linear map with $\boldsymbol{x}$ as the input. Then use the result of the previous step.

The idea of using "sawtooth" functions (see Figure 8) was first raised in [44] for approximating x^2 using FNNs with width 6 and depth $\mathcal{O}(L)$ and achieving an error $\mathcal{O}(2^{-L})$; our construction is different from and more general than that in [44], working for ReLU FNNs of width $\mathcal{O}(N)$ and depth $\mathcal{O}(L)$ for any N and L, and achieving an error $\mathcal{O}(N^{-L})$. As discussed below Proposition 4.1, this $\mathcal{O}(N^{-L})$ approximation error of polynomial functions shows the power of depth in ReLU FNNs via function composition.

First, let us show how to construct ReLU FNNs to approximate $f(x)=x^2$.

Lemma 5.1. *For any $N,L\in\mathbb{N}^+$, there exists a function ϕ implemented by a ReLU FNN with width $3N$ and depth L such that*

$$|\phi(x)-x^2|\le N^{-L},\quad \text{for any } x\in[0,1].$$

Proof. Define a set of "sawtooth" functions $T_i : [0,1] \to [0,1]$ by induction as follows. Set

$$T_1(x) = \begin{cases} 2x, & \text{if } x \in [0, \frac{1}{2}], \\ 2(1-x), & \text{if } x \in (\frac{1}{2}, 1], \end{cases}$$

and

$$T_i = T_{i-1} \circ T_1, \quad \text{for } i = 2, 3, \cdots.$$

It is easy to check that T_i has 2^{i-1} "sawteeth" and

$$T_{m+n} = T_m \circ T_n, \quad \text{for any } m, n \in \mathbb{N}^+.$$

See Figure 8 for illustrations of T_i for $i = 1, 2, 3, 4$.

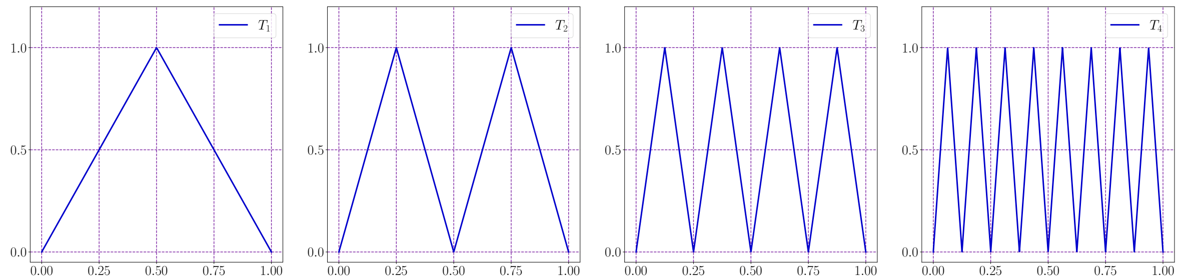


Figure 8: Examples of "sawtooth" functions T_1, T_2, T_3, and T_4.

Define piecewise linear functions $f_s : [0,1] \to [0,1]$ for $s \in \mathbb{N}^+$ satisfying the following two requirements (see Figure 9 for several examples of f_s).

- $f_s(\frac{j}{2^s}) = \left(\frac{j}{2^s}\right)^2$ for $j = 0, 1, 2, \cdots, 2^s$.
- $f_s(x)$ is linear between any two adjacent points of $\{\frac{j}{2^s} : j = 0, 1, 2, \cdots, 2^s\}$.

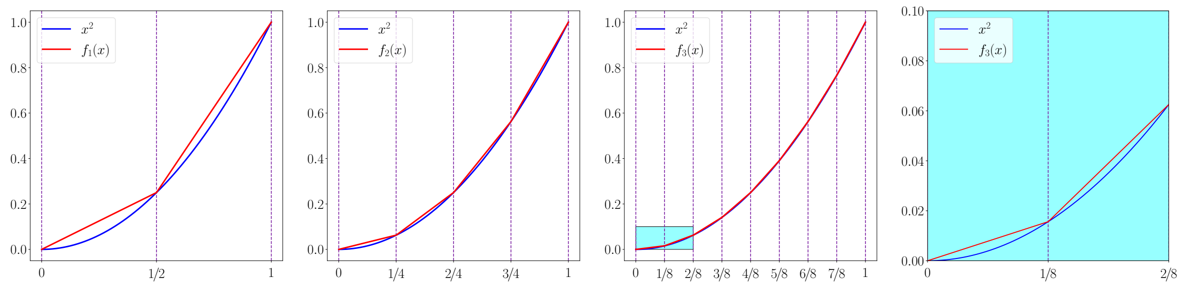


Figure 9: Illustrations of f_1, f_2, and f_3 for approximating x^2.

Recall the fact

$$0 \le t x_1^2 + (1-t) x_2^2 - \Big(t x_1 + (1-t) x_2\Big)^2 \le \frac{(x_2 - x_1)^2}{4}, \quad \text{for any } t, x_1, x_2 \in [0,1].$$

Thus, we have

$$0 \le f_s(x) - x^2 \le \frac{(2^{-s})^2}{4} = 2^{-2(s+1)}, \quad \text{for any } x \in [0,1] \text{ and } s \in \mathbb{N}^+. \tag{5.1}$$

Note that $f_{i-1}(x) = f_i(x) = x^2$ for $x \in \{\frac{j}{2^{i-1}} : j = 0,1,2,\cdots,2^{i-1}\}$ and the graph of $f_{i-1} - f_i$ is a symmetric "sawtooth" between any two adjacent points of $\{\frac{j}{2^{i-1}} : j = 0,1,2,\cdots,2^{i-1}\}$. It is easy to verify that

$$f_{i-1}(x) - f_i(x) = \frac{T_i(x)}{2^{2i}}, \quad \text{for any } x \in [0,1] \text{ and } i = 2,3,\cdots.$$

Therefore, for any $x \in [0,1]$ and $s \in \mathbb{N}^+$, we have

$$f_s(x) = f_1(x) + \sum_{i=2}^{s}(f_i - f_{i-1}) = x - (x - f_1(x)) - \sum_{i=2}^{s}\frac{T_i(x)}{2^{2i}} = x - \sum_{i=1}^{s}\frac{T_i(x)}{2^{2i}}.$$

Given $N \in \mathbb{N}^+$, there exists a unique $k \in \mathbb{N}^+$ such that $(k-1)2^{k-1} + 1 \le N \le k2^k$. For this k, using $s = Lk$, we can construct a ReLU FNN as shown in Figure 10 to implement a function $\phi = f_{Lk}$ approximating x^2 well. Note that T_i can be implemented by a one-hidden-layer ReLU FNN with width 2^i. Hence, the network in Figure 10 has width $k2^k + 1 \le 3N$[8] and depth $2L$.

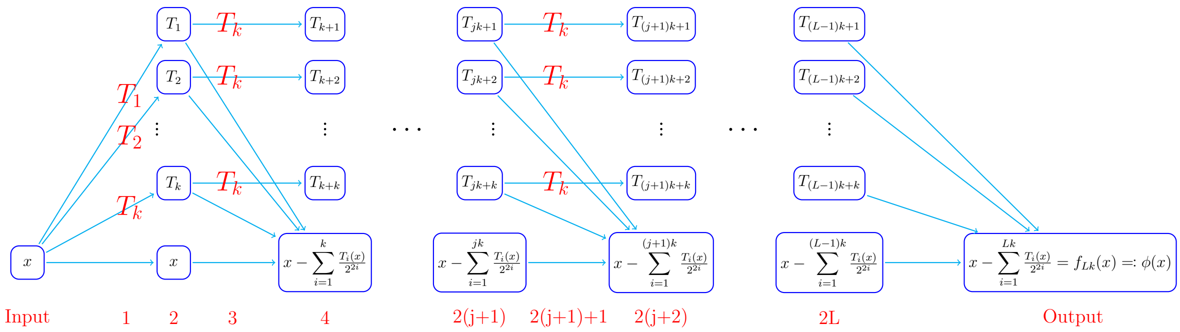


Figure 10: An illustration of the target network architecture for approximating x^2 on $[0,1]$. T_i can be implemented by a one-hidden-layer ReLU FNN with width 2^i for $i = 1,2,\cdots,K$. The red numbers below the architecture indicate the order of hidden layers.

As shown in Figure 10, the (2ℓ)-th hidden layer of the network has the identify function as activation functions for $\ell = 1,2,\cdots,L$. Thus, the network in Figure 10 can be interpreted as a ReLU FNN with width $3N$ and depth L. In fact, if all activation functions in a certain hidden layer are identity maps, the depth can be reduced by one via combining two adjacent linear transforms into one. For example, suppose $\boldsymbol{W}_1 \in \mathbb{R}^{N_1\times N_2}$, $\boldsymbol{W}_2 \in \mathbb{R}^{N_2\times N_3}$, and ϱ is an identity map that can be applied to vectors or matrices elementwisely; then $\boldsymbol{W}_1\varrho(\boldsymbol{W}_2\boldsymbol{x}) = \boldsymbol{W}_3\boldsymbol{x}$ for any $\boldsymbol{x} \in \mathbb{R}^{N_3}$, where $\boldsymbol{W}_3 = \boldsymbol{W}_1 \cdot \boldsymbol{W}_2 \in \mathbb{R}^{N_1\times N_3}$.

It remains to estimate the approximation error of $\phi(x) \approx x^2$. By Equation (5.1), for any $x \in [0,1]$, we have

$$|\phi(x) - x^2| = |f_{Lk}(x) - x^2| \le 2^{-2(Lk+1)} \le 2^{-2Lk} \le N^{-L},$$

where the last inequality comes from $N \le k2^k \le 2^{2k}$. So we finish the proof. $\square$

[8] This inequality is clear for $k = 1,2,3,4$. In the case $k \ge 5$, we have $k2^k + 1 \le \frac{k2^k+1}{N}N \le \frac{(k+1)2^k}{(k-1)2^{k-1}}N \le 2\frac{k+1}{k-1}N \le 3N$.

We have constructed a ReLU FNN to approximate $f(x) = x^2$. By the fact that $xy = 2\big((\frac{x+y}{2})^2 - (\frac{x}{2})^2 - (\frac{y}{2})^2\big)$, it is easy to construct a new ReLU FNN to approximate $f(x,y) = xy$ as follows.

Lemma 5.2. *For any $N, L \in \mathbb{N}^+$, there exists a function ϕ implemented by a ReLU FNN with width $9N$ and depth L such that*

$$|\phi(x,y) - xy| \le 6N^{-L}, \quad \text{for any } x, y \in [0,1].$$

Proof. By Lemma 5.1, there exists a function ψ implemented by a ReLU FNN with width $3N$ and depth L such that

$$|x^2 - \psi(x)| \le N^{-L}, \quad \text{for any } x \in [0,1].$$

Inspired by the fact

$$xy = 2\big((\tfrac{x+y}{2})^2 - (\tfrac{x}{2})^2 - (\tfrac{y}{2})^2\big), \quad \text{for any } x, y \in \mathbb{R},$$

we construct the desired function ϕ as

$$\phi(x,y) := 2\big(\psi(\tfrac{x+y}{2}) - \psi(\tfrac{x}{2}) - \psi(\tfrac{y}{2})\big), \quad \text{for any } x, y \in \mathbb{R}. \tag{5.2}$$

Then ϕ can be implemented by the network architecture in Figure 11.

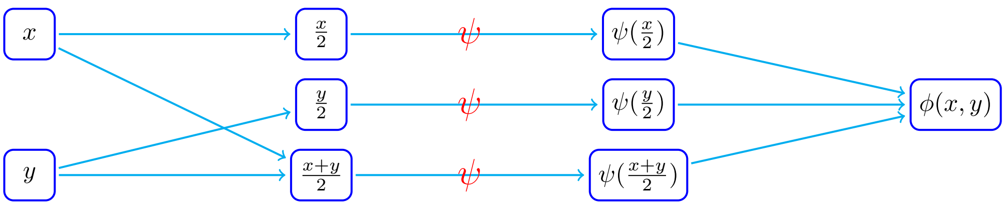


Figure 11: An illustration of the network architecture implementing ϕ for approximating xy on $[0,1]^2$.

It follows from $\psi \in \mathcal{NN}(\text{width} \le 3N;\ \text{depth} \le L)$ that the network in Figure 11 is with width $9N$ and depth $L+2$. Similar to the discussion in the proof of Lemma 5.1, the network in Figure 11 can be interpreted as a ReLU FNN with width $9N$ and depth L, since two of hidden layers have the identify function as their activation functions. Moreover, for any $x, y \in [0,1]$,

$$\begin{aligned}|xy - \phi(x,y)| &= \Big|2\big((\tfrac{x+y}{2})^2 - (\tfrac{x}{2})^2 - (\tfrac{y}{2})^2\big) - 2\big(\psi(\tfrac{x+y}{2}) - \psi(\tfrac{x}{2}) - \psi(\tfrac{y}{2})\big)\Big| \\ &\le 2\Big|(\tfrac{x+y}{2})^2 - \psi(\tfrac{x+y}{2})\Big| + 2\Big|(\tfrac{x}{2})^2 - \psi(\tfrac{x}{2})\Big| + 2\Big|(\tfrac{y}{2})^2 - \psi(\tfrac{y}{2})\Big| \le 6N^{-L}.\end{aligned}$$

Therefore, we have finished the proof. □

Now let us prove Lemma 4.2, which shows how to construct a ReLU FNN to approximate $f(x,y) = xy$ on $[a,b]^2$ with arbitrary $a < b$, i.e., a rescaled version of Lemma 5.2.

Proof of Lemma 4.2. By Lemma 5.2, there exists a function ψ implemented by a ReLU FNN with width $9N$ and depth L such that

$$|\psi(\widetilde{x},\widetilde{y}) - \widetilde{x}\widetilde{y}| \le 6N^{-L}, \quad \text{for any } \widetilde{x},\widetilde{y} \in [0,1].$$

By setting $\widetilde{x} = \frac{x-a}{b-a}$ and $\widetilde{y} = \frac{y-a}{b-a}$ for any $x,y \in [a,b]$, we have $\widetilde{x},\widetilde{y} \in [0,1]$, implying

$$\left|\psi\left(\tfrac{x-a}{b-a}, \tfrac{y-a}{b-a}\right) - \tfrac{x-a}{b-a}\tfrac{y-a}{b-a}\right| \le 6N^{-L}, \quad \text{for any } x,y \in [a,b].$$

It follows that, for any $x,y \in [a,b]$,

$$\left|(b-a)^2\psi\left(\tfrac{x-a}{b-a}, \tfrac{y-a}{b-a}\right) + a(x+y) - a^2 - xy\right| \le 6(b-a)^2N^{-L}.$$

Define, for any $x,y \in \mathbb{R}$,

$$\phi(x,y) \coloneqq (b-a)^2\psi\left(\tfrac{x-a}{b-a}, \tfrac{y-a}{b-a}\right) + a\cdot\sigma(x+y+2|a|) - a^2 - 2a|a|.$$

Then ϕ can be implemented by the network architecture in Figure 12.

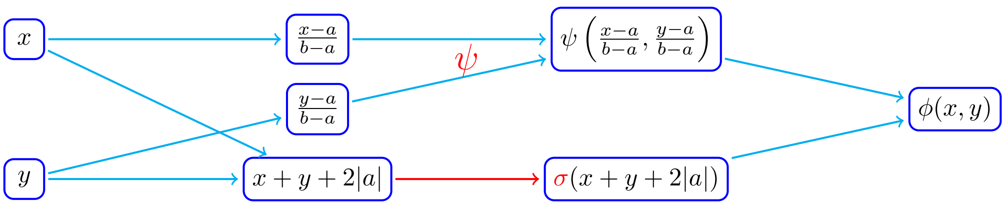


Figure 12: An illustration of the network architecture implementing ϕ for approximating xy on $[a,b]^2$. Two of hidden layers have the identify function as their activation functions, since the red "σ" comes from the red arrow "$\longrightarrow$", where the red arrow "$\longrightarrow$" is a ReLU FNN with width 1 and depth L.

It follows from $\psi \in \mathcal{NN}(\text{width} \le 9N;\ \text{depth} \le L)$ that the network in Figure 12 is with width $9N+1$ and depth $L+2$. Similar to the discussion in the proof of Lemma 5.1, the network in Figure 12 can be interpreted as a ReLU FNN with width $9N+1$ and depth L, since two of hidden layers have the identify function as their activation functions.

Note that $x+y+2|a| \ge 0$ for any $x,y \in [a,b]$, implying

$$\phi(x,y) = (b-a)^2\psi\left(\tfrac{x-a}{b-a}, \tfrac{y-a}{b-a}\right) + a(x+y) - a^2, \quad \text{for any } x,y \in [a,b].$$

Hence,

$$\left|\phi(x,y) - xy\right| \le 6(b-a)^2N^{-L}, \quad \text{for any } x,y \in [a,b].$$

So we finish the proof. □

The next lemma shows how to construct a ReLU FNN to approximate a multivariate function $f(x_1,x_2,\cdots,x_k) = x_1x_2\cdots x_k$ on $[0,1]^k$.

Lemma 5.3. *For any $N,L,k \in \mathbb{N}^+$ with $k \ge 2$, there exists a function ϕ implemented by a ReLU FNN with width $9(N+1)+k-1$ and depth $7kL(k-1)$ such that*

$$|\phi(\boldsymbol{x}) - x_1x_2\cdots x_k| \le 9(k-1)(N+1)^{-7kL}, \quad \text{for any } \boldsymbol{x} = [x_1,x_2,\cdots,x_k]^T \in [0,1]^k.$$

Proof. By Lemma 4.2, there exists a function ϕ_1 implemented by a ReLU FNN with width $9(N+1)+1$ and depth $7kL$ such that

$$|\phi_1(x,y) - xy| \le 6(1.2)^2(N+1)^{-7kL} \le 9(N+1)^{-7kL}, \quad \text{for any } x,y \in [-0.1, 1.1]. \tag{5.3}$$

Next, we construct a sequence of functions $\phi_i : [0,1]^{i+1} \to [0,1]$ for $i \in \{1,2,\cdots,k-1\}$ by induction such that

(i) ϕ_i can be implemented by a ReLU FNN with width $9(N+1)+i$ and depth $7kLi$ for each $i \in \{1,2,\cdots,k-1\}$.

(ii) For any $i \in \{1,2,\cdots,k-1\}$ and $x_1, x_2, \cdots, x_{i+1} \in [0,1]$, it holds that

$$|\phi_i(x_1,\cdots,x_{i+1}) - x_1x_2\cdots x_{i+1}| \le 9i(N+1)^{-7kL}. \tag{5.4}$$

First, let us consider the case $i=1$, it is obvious that the two required conditions are true: 1) $9(N+1)+i = 9(N+1)+1$ and $7kLi = 7kL$ if $i=1$; 2) Equation (5.3) implies Equation (5.4) for $i=1$.

Now assume ϕ_i has been defined, we define

$$\phi_{i+1}(x_1,\cdots,x_{i+2}) \coloneqq \phi_1\big(\phi_i(x_1,\cdots,x_{i+1}), \sigma(x_{i+2})\big), \quad \text{for any } x_1,\cdots,x_{i+2} \in \mathbb{R}.$$

Note that $\phi_i \in \mathcal{NN}(\text{width} \le 9(N+1)+i;\ \text{depth} \le 7kLi)$ and $\phi_1 \in \mathcal{NN}(\text{width} \le 9(N+1)+1;\ \text{depth} \le 7kL)$. Then ϕ_{i+1} can be implemented via a ReLU FNN with width

$$\max\{9(N+1)+i+1, 9(N+1)+1\} = 9(N+1)+(i+1)$$

and depth $7kLi + 7kL = 7kL(i+1)$.

By the hypothesis of induction, we have

$$|\phi_i(x_1,\cdots,x_{i+1}) - x_1x_2\cdots x_{i+1}| \le 9i(N+1)^{-7kL}. \tag{5.5}$$

Recall the fact that $9i(N+1)^{-7kL} \le 9k2^{-7k} \le 9k\frac{2^{-7}}{k} \le 0.1$ for any $N, L, k \in \mathbb{N}^+$ and $i \in \{1,2,\cdots,k-1\}$. It follows that

$$\phi_i(x_1,\cdots,x_{i+1}) \in [-0.1, 1.1], \quad \text{for any } x_1,\cdots,x_{i+1} \in [0,1].$$

Therefore, by Equations (5.3) and (5.5), we have

$$\begin{aligned}
&|\phi_{i+1}(x_1,\cdots,x_{i+2}) - x_1x_2\cdots x_{i+2}| \\
&= \big|\phi_1\big(\phi_i(x_1,\cdots,x_{i+1}), \sigma(x_{i+2})\big) - x_1x_2\cdots x_{i+2}\big| \\
&\le \big|\phi_1\big(\phi_i(x_1,\cdots,x_{i+1}), x_{i+2}\big) - \phi_i(x_1,\cdots,x_{i+1})x_{i+2}\big| + \big|\phi_i(x_1,\cdots,x_{i+1})x_{i+2} - x_1x_2\cdots x_{i+2}\big| \\
&\le 9(N+1)^{-7kL} + 9i(N+1)^{-7kL} = 9(i+1)(N+1)^{-7kL},
\end{aligned}$$

for any $x_1, x_2, \cdots, x_{i+2} \in [0,1]$, which means we finish the process of induction.

Now let $\phi \coloneqq \phi_{k-1}$, by the principle of induction, we have

$$|\phi(x_1,\cdots,x_k) - x_1x_2\cdots x_k| \le 9(k-1)(N+1)^{-7kL}, \quad \text{for any } x_1,\cdots,x_k \in [0,1].$$

So ϕ is the desired function implemented by a ReLU FNN with width $9(N+1)+k-1$ and depth $7kL(k-1)$, which means we finish the proof. ∎

With Lemma 5.3 in hand, we are ready to prove Proposition 4.1 for approximating general multivariate polynomials by ReLU FNNs.

Proof of Proposition 4.1. The case $k = 1$ is trivial, so we assume $k \geq 2$ below. Set $\widetilde{k} = \|\boldsymbol{\alpha}\|_1 \leq k$, denote $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_d]^T$, and let $[z_1, z_2, \cdots, z_{\widetilde{k}}]^T \in \mathbb{R}^{\widetilde{k}}$ be the vector such that

$$z_\ell = x_j, \quad \text{if } \sum_{i=1}^{j-1} \alpha_i < \ell \leq \sum_{i=1}^{j} \alpha_i, \quad \text{for } j = 1, 2, \cdots, d.$$

That is,

$$[z_1, z_2, \cdots, z_{\widetilde{k}}]^T = \Big[\overbrace{x_1, \cdots, x_1}^{\alpha_1 \text{ times}}, \overbrace{x_2, \cdots, x_2}^{\alpha_2 \text{ times}}, \cdots, \overbrace{x_d, \cdots, x_d}^{\alpha_d \text{ times}}\Big]^T \in \mathbb{R}^{\widetilde{k}}.$$

Then we have $P(\boldsymbol{x}) = \boldsymbol{x}^{\boldsymbol{\alpha}} = z_1 z_2 \cdots z_{\widetilde{k}}$.

We construct the target ReLU FNN in two steps. First, there exists an affine linear map $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}^k$ that duplicates $\boldsymbol{x}$ to form a new vector $[z_1, z_2, \cdots, z_{\widetilde{k}}, 1, \cdots, 1]^T \in \mathbb{R}^k$, i.e., $\mathcal{L}(\boldsymbol{x}) = [z_1, z_2, \cdots, z_{\widetilde{k}}, 1, \cdots, 1]^T \in \mathbb{R}^k$. Second, by Lemma 5.3, there exists a function $\psi : \mathbb{R}^k \to \mathbb{R}$ implemented by a ReLU FNN with width $9(N+1) + k - 1$ and depth $7kL(k-1)$ such that ψ maps $[z_1, z_2, \cdots, z_{\widetilde{k}}, 1, \cdots, 1]^T \in \mathbb{R}^k$ to $z_1 z_2 \cdots z_{\widetilde{k}}$ within an error $9(k-1)(N+1)^{-7kL}$. Hence, we can construct the desired function via $\phi \coloneqq \psi \circ \mathcal{L}$. Then ϕ can implemented by a ReLU FNN with width $9(N+1)+k-1$ and depth $7kL(k-1) \leq 7k^2 L$, and

$$\begin{aligned}
|\phi(\boldsymbol{x}) - P(\boldsymbol{x})| = |\phi(\boldsymbol{x}) - \boldsymbol{x}^{\boldsymbol{\alpha}}| &= |\psi \circ \mathcal{L}(\boldsymbol{x}) - x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d}| \\
&= |\psi(z_1, z_2, \cdots, z_{\widetilde{k}}, 1, \cdots, 1) - z_1 z_2 \cdots z_{\widetilde{k}}| \\
&\leq 9(k-1)(N+1)^{-7kL} \leq 9k(N+1)^{-7kL},
\end{aligned}$$

for any $x_1, x_2, \cdots, x_d \in [0, 1]$. So, we finish the proof. □

5.2 Proof of Proposition 4.3 for step function approximation

To prove Proposition 4.3 in this sub-section, we will discuss how to pointwisely approximate step functions by ReLU FNNs except for the trifling region. Before proving Proposition 4.3, let us first introduce a basic lemma about fitting $\mathcal{O}(N_1 N_2)$ samples using a two-hidden-layer ReLU FNN with $\mathcal{O}(N_1 + N_2)$ neurons.

Lemma 5.4. *For any $N_1, N_2 \in \mathbb{N}^+$, given $N_1(N_2+1) + 1$ samples $(x_i, y_i) \in \mathbb{R}^2$ with $x_0 < x_1 < \cdots < x_{N_1(N_2+1)}$ and $y_i \geq 0$ for $i = 0, 1, \cdots, N_1(N_2+1)$, there exists $\phi \in \mathcal{NN}(\#\text{input} = 1; \text{widthvec} = [2N_1, 2N_2+1])$ satisfying the following conditions.*

1. *$\phi(x_i) = y_i$ for $i = 0, 1, \cdots, N_1(N_2+1)$;*
2. *ϕ is linear on each interval $[x_{i-1}, x_i]$ for $i \notin \{(N_2+1)j : j = 1, 2, \cdots, N_1\}$.*

The above lemma is Lemma 2.2 of [40] and the reader is referred to [40] for its proof. Essentially, this lemma shows the equivalence of one-hidden-layer ReLU FNNs of size $\mathcal{O}(N^2)$ and two-hidden-layer ones of size $\mathcal{O}(N)$ to fit $\mathcal{O}(N^2)$ samples.

The next lemma below shows that special shallow and wide ReLU FNNs can be represented by deep and narrow ones. This lemma was proposed as Proposition 2.2 in [41].

Lemma 5.5. *For any $N, L, d \in \mathbb{N}^+$, it holds that*

$$\begin{aligned}&\mathcal{NN}(\#\text{input} = d;\ \text{widthvec} = [N, NL];\ \#\text{output} = 1)\\ \subseteq\ &\mathcal{NN}(\#\text{input} = d;\ \text{width} \le 2N + 2;\ \text{depth} \le L + 1;\ \#\text{output} = 1).\end{aligned}$$

With Lemmas 5.4 and 5.5 in hand, let us present the detailed proof of Proposition 4.3.

Proof of Proposition 4.3. We divide the proof into two cases: $d = 1$ and $d \ge 2$.

Case 1: $d = 1$.

In this case, $K = \lfloor N^{1/d}\rfloor^2 \lfloor L^{2/d}\rfloor = N^2L^2$. Denote $M = N^2L$ and consider the sample set

$$\begin{aligned}\big\{(1, M-1), (2, 0)\big\} &\bigcup \big\{(\tfrac{m}{M}, m) : m = 0, 1, \cdots, M-1\big\}\\ &\bigcup \big\{(\tfrac{m+1}{M} - \delta, m) : m = 0, 1, \cdots, M-2\big\}.\end{aligned}$$

Its size is $2M + 1 = N \cdot \big((2NL - 1) + 1\big) + 1$. By Lemma 5.4 (set $N_1 = N$ and $N_2 = 2NL - 1$ therein), there exists

$$\begin{aligned}\phi_1 &\in \mathcal{NN}(\text{widthvec} = [2N, 2(2NL-1)+1])\\ &= \mathcal{NN}(\text{widthvec} = [2N, 4NL-1])\end{aligned}$$

such that

- $\phi_1(\frac{M-1}{M}) = \phi_1(1) = M - 1$ and $\phi_1(\frac{m}{M}) = \phi_1(\frac{m+1}{M} - \delta) = m$ for $m = 0, 1, \cdots, M-2$;
- ϕ_1 is linear on $[\frac{M-1}{M}, 1]$ and each interval $[\frac{m}{M}, \frac{m+1}{M} - \delta]$ for $m = 0, 1, \cdots, M-2$.

Then

$$\phi_1(x) = m, \quad \text{if } x \in [\tfrac{m}{M}, \tfrac{m+1}{M} - \delta \cdot \mathbb{1}_{\{m \le M-2\}}], \quad \text{for } m = 0, 1, \cdots, M-1. \tag{5.6}$$

Now consider the another sample set

$$\begin{aligned}\big\{(\tfrac{1}{M}, L-1), (2, 0)\big\} &\bigcup \big\{(\tfrac{\ell}{ML}, \ell) : \ell = 0, 1, \cdots, L-1\big\}\\ &\bigcup \big\{(\tfrac{\ell+1}{ML} - \delta, \ell) : \ell = 0, 1, \cdots, L-2\big\}.\end{aligned}$$

Its size is $2L + 1 = 1 \cdot \big((2L - 1) + 1\big) + 1$. By Lemma 5.4 (set $N_1 = 1$ and $N_2 = 2L - 1$ therein), there exists

$$\begin{aligned}\phi_2 &\in \mathcal{NN}(\text{widthvec} = [2, 2(2L-1)+1])\\ &= \mathcal{NN}(\text{widthvec} = [2, 4L-1])\end{aligned}$$

such that

- $\phi_2(\frac{L-1}{ML}) = \phi_2(\frac{1}{M}) = L - 1$ and $\phi_2(\frac{\ell}{ML}) = \phi_2(\frac{\ell+1}{ML} - \delta) = \ell$ for $\ell = 0, 1, \cdots, L-2$;
- ϕ_2 is linear on $[\frac{L-1}{ML}, \frac{1}{M}]$ and each interval $[\frac{\ell}{ML}, \frac{\ell+1}{ML} - \delta]$ for $\ell = 0, 1, \cdots, L-2$.

It follows that, for $m = 0,1,\cdots,M-1$ and $\ell = 0,1,\cdots,L-1$,

$$\phi_2(x - \tfrac{m}{M}) = \ell, \quad \text{for } x \in [\tfrac{mL+\ell}{ML}, \tfrac{mL+\ell+1}{ML} - \delta\cdot\mathbb{1}_{\{\ell\le L-2\}}]. \tag{5.7}$$

$K = ML$ implies any $k \in \{0,1,\cdots,K-1\}$ can be unique represented by $k = mL+\ell$ for $m \in \{0,1,\cdots,M-1\}$ and $\ell \in \{0,1,\cdots,L-1\}$. Then the desired function ϕ can be implemented by ReLU FNN shown in Figure 13.

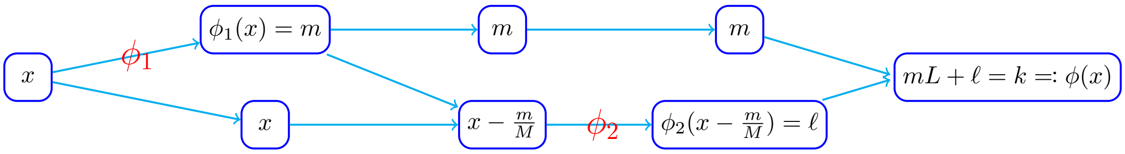


Figure 13: An illustration of the network architecture implementing ϕ based on Equations (5.6) and (5.7) with $x \in [\frac{k}{K}, \frac{k+1}{K} - \delta\cdot\mathbb{1}_{\{k\le K-2\}}] = [\frac{mL+\ell}{ML}, \frac{mL+\ell+1}{ML} - \delta\cdot\mathbb{1}_{\{m\le M-2 \text{ or } \ell\le L-2\}}]$, where $k = mL+\ell$ for $m = 0,1,\cdots,M-1$ and $\ell = 0,1,\cdots,L-1$.

Clearly,

$$\phi(x) = k, \quad \text{if } x \in [\tfrac{k}{K}, \tfrac{k+1}{K} - \delta\cdot\mathbb{1}_{\{k\le K-2\}}], \quad \text{for } k \in \{0,1,\cdots,K-1\}.$$

By Lemma 5.5, $\phi_1 \in \mathcal{NN}(\text{widthvec} = [2N, 4NL-1]) \subseteq \mathcal{NN}(\text{width} \le 4N+2;\ \text{depth} \le 2L+1)$ and $\phi_2 \in \mathcal{NN}(\text{widthvec} = [2, 4L-1]) \subseteq \mathcal{NN}(\text{width} \le 6;\ \text{depth} \le 2L+1)$, implying $\phi \in \mathcal{NN}(\text{width} \le \max\{4N+2+1, 6+1\} = 4N+3;\ \text{depth} \le (2L+1)+2+(2L+1)+1 = 4L+5)$. So we finish the proof for the case $d = 1$

Case 2: $d \ge 2$.

Now we consider the case when $d \ge 2$. Consider the sample set

$$\begin{aligned}\big\{(1, K-1), (2,0)\big\} &\bigcup \big\{(\tfrac{k}{K}, k) : k = 0,1,\cdots,K-1\big\} \\ &\bigcup \big\{(\tfrac{k+1}{K} - \delta, k) : k = 0,1,\cdots,K-2\big\},\end{aligned}$$

whose size is $2K+1 = \lfloor N^{1/d}\rfloor\big((2\lfloor N^{1/d}\rfloor\lfloor L^{2/d}\rfloor - 1)+1\big)+1$. By Lemma 5.4 (set $N_1 = \lfloor N^{1/d}\rfloor$ and $N_2 = 2\lfloor N^{1/d}\rfloor\lfloor L^{2/d}\rfloor - 1$ therein), there exists

$$\begin{aligned}\phi &\in \mathcal{NN}(\text{widthvec} = [2\lfloor N^{1/d}\rfloor, 2(2\lfloor N^{1/d}\rfloor\lfloor L^{2/d}\rfloor - 1)+1]) \\ &= \mathcal{NN}(\text{widthvec} = [2\lfloor N^{1/d}\rfloor, 4\lfloor N^{1/d}\rfloor\lfloor L^{2/d}\rfloor - 1])\end{aligned}$$

such that

- $\phi(\frac{K-1}{K}) = \phi(1) = K-1$, and $\phi(\frac{k}{K}) = \phi(\frac{k+1}{K} - \delta) = k$ for $k = 0,1,\cdots,K-2$;
- ϕ is linear on $[\frac{K-1}{K}, 1]$ and each interval $[\frac{k}{K}, \frac{k+1}{K} - \delta]$ for $k = 0,1,\cdots,K-2$.

Then

$$\phi(x) = k, \quad \text{if } x \in [\tfrac{k}{K}, \tfrac{k+1}{K} - \delta\cdot\mathbb{1}_{\{k\le K-2\}}], \quad \text{for } k = 0,1,\cdots,K-1.$$

By Lemma 5.5,

$$\begin{aligned}\phi &\in \mathcal{NN}(\text{widthvec} = [2\lfloor N^{1/d}\rfloor, 4\lfloor N^{1/d}\rfloor\lfloor L^{2/d}\rfloor - 1]) \\ &\subseteq \mathcal{NN}(\text{width} \le 4\lfloor N^{1/d}\rfloor + 2;\ \text{depth} \le 2\lfloor L^{2/d}\rfloor + 1) \\ &\subseteq \mathcal{NN}(\text{width} \le 4\lfloor N^{1/d}\rfloor + 3;\ \text{depth} \le 4L+5).\end{aligned}$$

which means we finish the proof for the case $d \ge 2$. □

5.3 Proof of Proposition 4.4 for point fitting

In this sub-section, we will discuss how to use ReLU FNNs to fit a collection of points in $\mathbb{R}^2$.⑨ It is trivial to fit n points via one-hidden-layer ReLU FNNs with $\mathcal{O}(n)$ parameters. However, to prove Proposition 4.4, we need to fit $\mathcal{O}(n)$ points with much fewer parameters, which is the main difficulty of our proof. Our proof below is mainly based on the "bit extraction" technique and the composition architecture of neural networks.

Let us first introduce a basic lemma based on the "bit extraction" technique, which is actually Lemma 2.6 of [41].

Lemma 5.6. *For any $N, L \in \mathbb{N}^+$, any $\theta_{m,\ell} \in \{0,1\}$ for $m = 0,1,\cdots,M-1$ and $\ell = 0,1,\cdots,L-1$, where $M = N^2L$, there exists a function ϕ implemented by a ReLU FNN with width $4N+3$ and depth $3L+3$ such that*

$$\phi(m,\ell) = \sum_{j=0}^{\ell} \theta_{m,j}, \qquad \text{for } m = 0,1,\cdots,M-1 \text{ and } \ell = 0,1,\cdots,L-1.$$

Next, let us introduce Lemma 5.7, a variant of Lemma 5.6 for a different mapping for the "bit extraction". Its proof is based on Lemmas 5.4, 5.5, and 5.6.

Lemma 5.7. *For any $N, L \in \mathbb{N}^+$ and any $\theta_i \in \{0,1\}$ for $i = 0,1,\cdots,N^2L^2-1$, there exists a function ϕ implemented by a ReLU FNN with width $8N+6$ and depth $5L+7$ such that*

$$\phi(i) = \theta_i, \quad \text{for } i = 0,1,\cdots,N^2L^2-1.$$

Proof. The case $L = 1$ is clear. We assume $L \ge 2$ below.

Denote $M = N^2L$, for each $i \in \{0,1,\cdots,N^2L^2-1\}$, there exists a unique representation $i = mL + \ell$ for $m \in \{0,1,\cdots,M-1\}$ and $\ell \in \{0,1,\cdots,L-1\}$. Thus, we can define, for $m = 0,1,\cdots,M-1$ and $\ell = 0,1,\cdots,L-1$,

$$a_{m,\ell} \coloneqq \theta_i, \quad \text{where } i = mL + \ell.$$

Then, for $m = 0,1,\cdots,M-1$, we set $b_{m,0} = 0$ and $b_{m,\ell} = a_{m,\ell-1}$ for $\ell = 1,2,\cdots,L-1$.

By Lemma 5.6, there exist $\phi_1, \phi_2 \in \mathcal{NN}(\text{width} \le 4N+3;\ \text{depth} \le 3L+3)$ such that

$$\phi_1(m,\ell) = \sum_{j=0}^{\ell} a_{m,j} \quad \text{and} \quad \phi_2(m,\ell) = \sum_{j=0}^{\ell} b_{m,j},$$

for $m = 0,1,\cdots,M-1$ and $\ell = 0,1,\cdots,L-1$.

We consider the sample set

$$\{(mL,m) : m = 0,1,\cdots,M\} \bigcup \Big\{\big((m+1)L-1,m\big) : m = 0,1,\cdots,M-1\Big\}.$$

Its size is $2M+1 = N\cdot\big((2NL-1)+1\big)+1$. By Lemma 5.4 (set $N_1 = N$ and $N_2 = 2NL-1$ therein), there exists

$$\begin{aligned}\psi &\in \mathcal{NN}(\text{widthvec} = [2N, 2(2NL-1)+1]) \\ &= \mathcal{NN}(\text{widthvec} = [2N, 4NL-1])\end{aligned}$$

such that

⑨Fitting a collection of points $\{(x_i,y_i)\}_i$ in $\mathbb{R}^2$ means that the target ReLU FNN takes a value close to y_i at the location x_i.

- $\psi(ML) = M$ and $\psi(mL) = \psi\big((m+1)L-1\big) = m$ for $m = 0, 1, \cdots, M-1$;
- ψ is linear on each interval $[mL, (m+1)L-1]$ for $m = 0, 1, \cdots, M-1$.

It follows that

$$\psi(x) = m, \quad \text{if } x \in [mL, (m+1)L-1], \qquad \text{for } m = 0, 1, \cdots, M-1,$$

implying

$$\psi(mL+\ell) = m, \quad \text{for } m = 0, 1, \cdots, M-1 \text{ and } \ell = 0, 1, \cdots, L-1.$$

For $i = 0, 1, \cdots, N^2L^2 - 1$, by representing $i = mL + \ell$ for $m = 0, 1, \cdots, M-1$ and $\ell = 0, 1, \cdots, L-1$, we have $\psi(i) = \psi(mL+\ell) = m$ and $i - L\psi(i) = \ell$, from which we deduce

$$\begin{aligned} &\phi_1\big(\psi(i), i - L\psi(i)\big) - \phi_2\big(\psi(i), i - L\psi(i)\big) \\ &= \phi_1(m, \ell) - \phi_2(m, \ell) = \sum_{j=0}^{\ell} a_{m,j} - \sum_{j=0}^{\ell} b_{m,j} \\ &= \sum_{j=0}^{\ell} a_{m,j} - \sum_{j=1}^{\ell} a_{m,j-1} - b_0 = a_{m,\ell} = \theta_i. \end{aligned} \tag{5.8}$$

Therefore, the desired function ϕ can be implemented by the network architecture described in Figure 14.

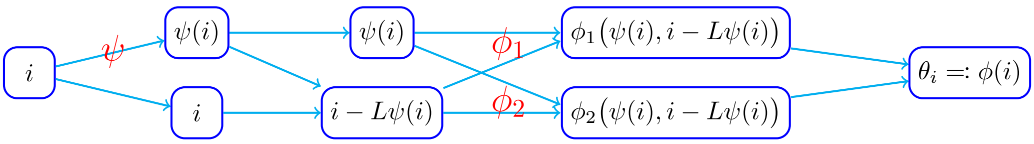


Figure 14: An illustration of the network architecture implementing the desired function ϕ based on Equation (5.8).

Note that

$$\phi_1, \phi_2 \in \mathcal{NN}(\text{width} \le 4N+3;\ \text{depth} \le 3L+3).$$

And by Lemma 5.5,

$$\begin{aligned} \psi &\in \mathcal{NN}(\text{widthvec} = [2N, 4NL-1]) \\ &\subseteq \mathcal{NN}(\text{width} \le 4N+2;\ \text{depth} \le 2L+1). \end{aligned}$$

Hence, the network architecture shown in Figure 14 is with width $\max\{4L+2+1, 2(4L+3)\} = 8N+6$ and depth $(2L+1)+2+(3L+3)+1 = 5L+7$, implying $\phi \in \mathcal{NN}(\text{width} \le 8N+6;\ \text{depth} \le 5L+7)$. So we finish the proof. $\square$

With Lemma 5.7 in hand, we are now ready to prove Proposition 4.4.

Proof of Proposition 4.4. Set $J = \lceil 2s \log_2(NL+1) \rceil \in \mathbb{N}^+$. For each $\xi_i \in [0,1]$, there exist $\xi_{i,1}, \xi_{i,2}, \cdots, \xi_{i,J} \in \{0,1\}$ such that

$$\big|\xi_i - \text{bin}\, 0.\xi_{i,1}\xi_{i,2}\cdots\xi_{i,J}\big| \le 2^{-J}, \quad \text{for } i = 0, 1, \cdots, N^2L^2 - 1.$$

By Lemma 5.7, there exist

$$\phi_1, \phi_2, \cdots, \phi_J \in \mathcal{NN}(\text{width} \le 8N + 6;\ \text{depth} \le 5L + 7)$$

such that

$$\phi_j(i) = \xi_{i,j}, \quad \text{for } i = 0, 1, \cdots, N^2L^2 - 1 \text{ and } \ j = 1, 2, \cdots, J.$$

Define

$$\widetilde{\phi}(x) := \sum_{j=1}^{J} 2^{-j}\phi_j(x), \quad \text{for any } x \in \mathbb{R}.$$

It follows that, for $i = 0, 1, \cdots, N^2L^2 - 1$,

$$\begin{aligned} |\widetilde{\phi}(i) - \xi_i| &= \Big| \sum_{j=1}^{J} 2^{-j}\phi_j(i) - \xi_i \Big| = \Big| \sum_{j=1}^{J} 2^{-j}\xi_{i,j} - \xi_i \Big| \\ &= \big| \text{bin}\, 0.\xi_{i,1}\xi_{i,2}\cdots\xi_{i,J} - \xi_i \big| \le 2^{-J} \le N^{-2s}L^{-2s}, \end{aligned}$$

where the last inequality comes from

$$2^{-J} = 2^{-\lceil 2s \log_2(NL+1)\rceil} \le 2^{-2s\log_2(NL+1)} = (NL+1)^{-2s} \le N^{-2s}L^{-2s}.$$

Now let us estimate the width and depth of the network implementing $\widetilde{\phi}$. Recall that

$$\begin{aligned} J = \lceil 2s \log_2(NL+1)\rceil &\le 2s\big(1 + \log_2(NL+1)\big) \le 2s\big(1 + \log_2(2N) + \log_2 L\big) \\ &\le 2s\big(1 + \log_2(2N)\big)\big(1 + \log_2 L\big) \le 2s\lceil \log_2(4N)\rceil \lceil \log_2(2L)\rceil, \end{aligned}$$

and $\phi_j \in \mathcal{NN}(\text{width} \le 8N + 6;\ \text{depth} \le 5L + 7)$ for each j.

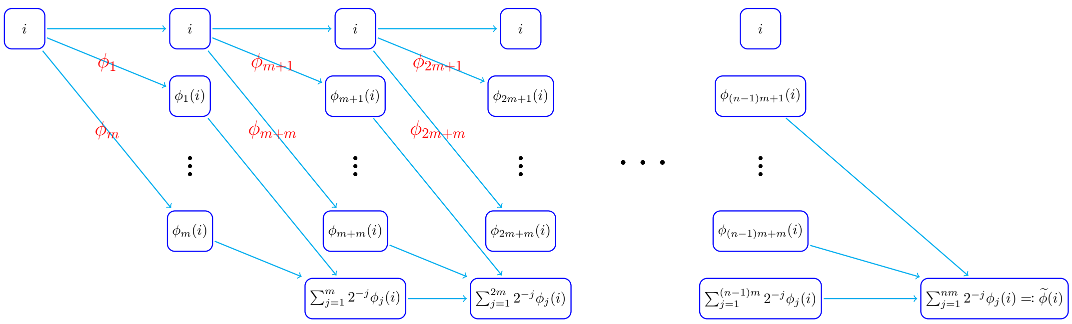


Figure 15: An illustration of the network architecture implementing $\widetilde{\phi} = \sum_{j=1}^{J} 2^{-j}\phi_j$ for any $i \in \{0, 1, \cdots, N^2L^2 - 1\}$. We assume $J = mn$, where $m = 2s\lceil \log_2(4N)\rceil$ and $n = \lceil \log_2(2L)\rceil$, since we can set $\phi_{J+1} = \cdots = \phi_{nm} = 0$ if $J < nm$.

As we can see from Figure 15, $\widetilde{\phi} = \sum_{j=1}^{J} 2^{-j}\phi_j$ can be implemented by a ReLU FNN with width

$$\begin{aligned} (8N+6)m + (1 + m + 1) &= (8N+6)2s\lceil \log_2(4N)\rceil + 2s\lceil \log_2(4N)\rceil + 2 \\ &\le 16s(N+1)\log_2(8N) \end{aligned}$$

and depth

$$\big((5L+7)+1\big)n = (5L+8)\lceil \log_2(2L)\rceil \le (5N+8)\log_2(4L).$$

Finally, we define

$$\phi(x) \coloneqq \min\left\{\sigma\big(\widetilde{\phi}(x)\big), 1\right\} = \min\left\{\max\{0, \widetilde{\phi}(x)\}, 1\right\}, \quad \text{for any } x \in \mathbb{R}.$$

Then $0 \le \phi(x) \le 1$ for any $x \in \mathbb{R}$ and ϕ can be implemented by a ReLU FNN with width $16s(N+1)\log_2(8N)$ and depth $(5L+8)\log_2(4L) + 3 \le 5(L+2)\log_2(4L)$. See Figure 16 for the network architecture implementing ϕ. Note that

$$\widetilde{\phi}(i) = \sum_{j=1}^{J} 2^{-j}\phi_j(i) = \sum_{j=1}^{J} 2^{-j}\xi_{i,j} \in [0,1], \quad \text{for } i = 0, 1, \cdots, N^2L^2 - 1.$$

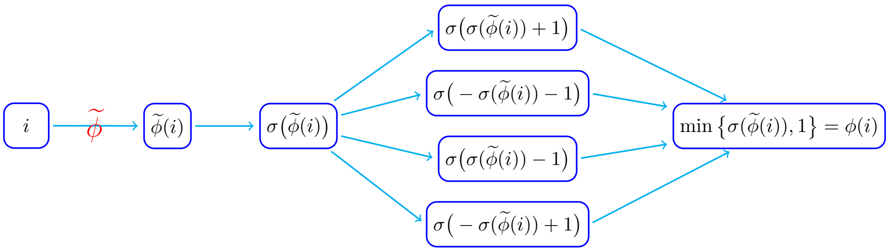


Figure 16: An illustration of the network architecture implementing the desired function ϕ based on the fact that $\min\{x_1, x_2\} = \frac{x_1+x_2-|x_1-x_2|}{2} = \frac{\sigma(x_1+x_2)-\sigma(-x_1-x_2)-\sigma(x_1-x_2)-\sigma(-x_1+x_2)}{2}$.

It follows that

$$|\phi(i) - \xi_i| = \left|\min\left\{\max\{0, \widetilde{\phi}(i)\}, 1\right\} - \xi_i\right| = |\widetilde{\phi}(i) - \xi_i| \le N^{-2s}L^{-2s},$$

for $i = 0, 1, \cdots, N^2L^2 - 1$. The proof is complete. □

6 Conclusions

This paper has established a nearly optimal approximation error of ReLU FNNs in terms of both width and depth to approximate smooth functions. It is shown that ReLU FNNs with width $\mathcal{O}(N \ln N)$ and depth $\mathcal{O}(L \ln L)$ can approximate functions in the unit ball of $C^s([0,1]^d)$ with an approximation error $\mathcal{O}(N^{-2s/d}L^{-2s/d})$. Through VC-dimension, it is also proved that this approximation error is asymptotically nearly tight for the closed unit ball of $C^s([0,1]^d)$.

We would like to remark that our analysis is for the fully connected feed-forward neural networks with the ReLU activation function. It would be an interesting direction to generalize our results to neural networks with other architectures (e.g., convolutional neural networks and ResNet) and activation functions (e.g., tanh and sigmoid functions). These will be subjects of future work.

Acknowledgments

The work of J. Lu is supported in part by the National Science Foundation via grants DMS-1415939, CCF-1934964, and DMS-2012286. Z. Shen is supported by Tan Chin Tuan Centennial Professorship. H. Yang H. Yang was partially supported by the National Science Foundation under award DMS-1945029.

References

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. *arXiv e-prints*, page arXiv:1811.04918, November 2018.

[2] Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[3] Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *ICML*, 2019.

[4] Chenglong Bao, Qianxiao Li, Zuowei Shen, Cheng Tai, Lei Wu, and Xueshuang Xiang. Approximation analysis of convolutional neural networks. 2019.

[5] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, May 1993.

[6] Andrew R. Barron and Jason M. Klusowski. Approximation and estimation for high-dimensional deep learning networks. *arXiv e-prints*, page arXiv:1809.03090, September 2018.

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

[8] M. Bianchini and F. Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems*, 25(8):1553–1565, Aug 2014.

[9] Helmut. Bölcskei, Philipp. Grohs, Gitta. Kutyniok, and Philipp. Petersen. Optimal approximation with sparsely connected deep neural networks. *SIAM Journal on Mathematics of Data Science*, 1(1):8–45, 2019.

[10] Yuan Cao and Quanquan Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. *CoRR*, abs/1905.13210, 2019.

[11] Liang Chen and Congwei Wu. A note on the expressive power of deep rectified linear unit networks in high-dimensional spaces. *Mathematical Methods in the Applied Sciences*, 42(9):3400–3404, 2019.

[12] Minshuo Chen, Haoming Jiang, Wenjing Liao, and Tuo Zhao. Efficient approximation of deep ReLU networks for functions on low dimensional manifolds. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8174–8184. Curran Associates, Inc., 2019.

[13] Zixiang Chen, Yuan Cao, Difan Zou, and Quanquan Gu. How much over-parameterization is sufficient to learn deep ReLU networks? *CoRR*, arXiv:1911.12360, 2019.

[14] Charles K. Chui, Shao-Bo Lin, and Ding-Xuan Zhou. Construction of neural networks for realization of localized deep learning. *Frontiers in Applied Mathematics and Statistics*, 4:14, 2018.

[15] George Cybenko. Approximation by superpositions of a sigmoidal function. *MCSS*, 2:303–314, 1989.

[16] Ronald A. Devore. Optimal nonlinear approximation. *Manuskripta Math*, pages 469–478, 1989.

[17] Weinan E, Chao Ma, and Qingcan Wang. A priori estimates of the population risk for residual networks. *ArXiv*, abs/1903.02154, 2019.

[18] Weinan E, Chao Ma, and Lei Wu. A priori estimates of the population risk for two-layer neural networks. *Communications in Mathematical Sciences*, 17(5):1407–1425, 2019.

[19] Weinan E and Qingcan Wang. Exponential convergence of the deep neural network approximation for analytic functions. *CoRR*, abs/1807.00297, 2018.

[20] Rémi Gribonval, Gitta Kutyniok, Morten Nielsen, and Felix Voigtlaender. Approximation spaces of deep neural networks. *arXiv e-prints*, page arXiv:1905.01208, May 2019.

[21] Ingo Gühring, Gitta Kutyniok, and Philipp Petersen. Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms. *arXiv e-prints*, page arXiv:1902.07896, Feb 2019.

[22] Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight VC-dimension bounds for piecewise linear neural networks. In Satyen Kale and Ohad Shamir, editors, *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 1064–1068, Amsterdam, Netherlands, 07–10 Jul 2017. PMLR.

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

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

[25] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *CoRR*, abs/1806.07572, 2018.

[26] Ziwei Ji and Matus Telgarsky. Polylogarithmic width suffices for gradient descent to achieve arbitrarily small test error with shallow ReLU networks. *ArXiv*, abs/1909.12292, 2020.

[27] Michael J. Kearns and Robert E. Schapire. Efficient distribution-free learning of probabilistic concepts. *J. Comput. Syst. Sci.*, 48(3):464–497, June 1994.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105. Curran Associates, Inc., 2012.

[29] Qianxiao Li, Ting Lin, and Zuowei Shen. Deep learning via dynamical systems: An approximation perspective. *Journal of European Mathematical Society*, to appear.

[30] Shiyu Liang and R. Srikant. Why deep neural networks? *CoRR*, abs/1610.04161, 2016.

[31] Hadrien Montanelli and Qiang Du. New error bounds for deep networks using sparse grids. *SIAM Journal on Mathematics of Data Science*, 1(1):78–92, 2019.

[32] Hadrien Montanelli and Haizhao Yang. Error bounds for deep ReLU networks using the Kolmogorov–Arnold superposition theorem. *Neural Networks*, 129:1–6, 2020.

[33] Hadrien Montanelli, Haizhao Yang, and Qiang Du. Deep ReLU networks overcome the curse of dimensionality for bandlimited functions. *arXiv e-prints*, page arXiv:1903.00735, March 2019.

[34] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2924–2932. Curran Associates, Inc., 2014.

[35] Ryumei Nakada and Masaaki Imaizumi. Adaptive approximation and generalization of deep neural network with intrinsic dimensionality. *Journal of Machine Learning Research*, 21(174):1–38, 2020.

[36] J. A. A. Opschoor, Ch. Schwab, and J. Zech. Exponential ReLU DNN expression of holomorphic maps in high dimension. Technical Report 2019-35, Seminar for Applied Mathematics, ETH Zürich, Switzerland., 2019.

[37] Philipp Petersen and Felix Voigtlaender. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Networks*, 108:296–330, 2018.

[38] T. Poggio, H. N. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao. Why and when can deep—but not shallow—networks avoid the curse of dimensionality: A review. *International Journal of Automation and Computing*, 14:503–519, 2017.

[39] Akito Sakurai. Tight bounds for the VC-dimension of piecewise polynomial networks. In *Advances in Neural Information Processing Systems*, pages 323–329. Neural information processing systems foundation, 1999.

[40] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Nonlinear approximation via compositions. *Neural Networks*, 119:74–84, 2019.

[41] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network approximation characterized by number of neurons. *Communications in Computational Physics*, 28(5):1768–1811, 2020.

[42] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Optimal approximation rate of ReLU networks in terms of width and depth. *Journal de Mathématiques Pures et Appliquées*, to appear.

[43] Taiji Suzuki. Adaptivity of deep ReLU network for learning in Besov and mixed smooth Besov spaces: optimal rate and curse of dimensionality. In *International Conference on Learning Representations*, 2019.

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

[45] Dmitry Yarotsky. Optimal approximation of continuous functions by very deep ReLU networks. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 639–649. PMLR, 06–09 Jul 2018.

[46] Dmitry Yarotsky and Anton Zhevnerchuk. The phase diagram of approximation rates for deep neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13005–13015. Curran Associates, Inc., 2020.

[47] Ding-Xuan Zhou. Universality of deep convolutional neural networks. *Applied and Computational Harmonic Analysis*, 48(2):787–794, 2020.