ReLU Network Approximation in Terms of Intrinsic Parameters

Zuowei Shen* Haizhao Yang† Shijun Zhang‡

Abstract

This paper studies the approximation error of ReLU networks in terms of the number of intrinsic parameters (i.e., those depending on the target function f). First, we prove by construction that, for any Lipschitz continuous function f on $[0,1]^d$ with a Lipschitz constant $\lambda > 0$, a ReLU network with $n+2$ intrinsic parameters can approximate f with an exponentially small error $5\lambda\sqrt{d}\,2^{-n}$ measured in the L^p-norm for $p \in [1,\infty)$. More generally for an arbitrary continuous function f on $[0,1]^d$ with a modulus of continuity $\omega_f(\cdot)$, the approximation error is $\omega_f(\sqrt{d}\,2^{-n}) + 2^{-n+2}\omega_f(\sqrt{d})$. Next, we extend these two results from the L^p-norm to the L^∞-norm at a price of $3^d n + 2$ intrinsic parameters. Finally, by using a high-precision binary representation and the bit extraction technique via a fixed ReLU network independent of the target function, we design, theoretically, a ReLU network with only three intrinsic parameters to approximate Hölder continuous functions with an arbitrarily small error.

Key words: ReLU Neural Networks; Intrinsic Parameters; Approximation Error; Exponential Convergence; Transfer Learning.

1 Introduction

Deep learning has been a powerful tool in science and engineering. As the workhorses of deep learning, deep neural networks are treated as an important regression tool in many successful applications. Understanding the approximation capacity of deep neural networks has become a key question for revealing the power of deep learning. One of the fundamental problems is the characterization of the approximation error of deep neural networks in terms of the network size measured in the width, the depth, the number of neurons, or the number of parameters.

It was shown in [38, 42, 45, 48] that the approximation error $\mathcal{O}(n^{-2/d})$ is (nearly) optimal for ReLU networks with $\mathcal{O}(n)$ parameters to approximate Lipschitz continuous functions on $[0,1]^d$. To gain better approximation errors, existing results either consider smaller target function spaces (e.g., [3, 11, 15, 25, 29, 44, 47]) or introduce new activation

*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).

functions (e.g., [39, 40, 41, 46]). This paper proposes a new perspective to study the approximation error in terms of the number of parameters depending on the target function, which are called the **intrinsic parameters**, excluding those independent of the target function.

1.1 Main results

Let $C([0,1]^d)$ denote the space of continuous functions defined on $[0,1]^d$. For simplicity, let $\mathcal{H}_W(d_1,d_2)$ denote the function space consisting of all ReLU networks with W parameters mapping from $\mathbb{R}^{d_1}$ to $\mathbb{R}^{d_2}$, i.e.,

$$\mathcal{H}_W(d_1,d_2) \coloneqq \Big\{ g : \ g:\mathbb{R}^{d_1}\to\mathbb{R}^{d_2} \text{ is realized by a ReLU network with } W \text{ parameters} \Big\}.$$

Let $\mathcal{H}(d_1,d_2) \coloneqq \bigcup_{W=1}^{\infty} \mathcal{H}_W(d_1,d_2)$.

For any $f \in C([0,1]^d)$, our goal is to construct two f-independent functions $\phi_1 \in \mathcal{H}(d,1)$ and $\phi_2 \in \mathcal{H}(n,1)$, and use $s\cdot(\phi_2\circ\phi_f\circ\phi_1)+b$ to approximate f, where $s\in[0,\infty)$, $b\in\mathbb{R}$, and $\phi_f \in \mathcal{H}_n(1,n)$ are learned from f. Under these settings, an approximation error $\omega_f(\sqrt{d}\,2^{-n})+2^{-n+2}\omega_f(\sqrt{d})$ is attained as shown in the theorem below, where the modulus of continuity of a continuous function $f\in C([0,1]^d)$ is defined as

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

Theorem 1.1. *Given any $n\in\mathbb{N}^+$ and $p\in[1,\infty)$, there exist $\phi_1\in\mathcal{H}_{2^{dn+4}}(d,1)$ and $\phi_2\in\mathcal{H}_{2^{dn+5}n}(n,1)$ such that: For any $f\in C([0,1]^d)$, there exists a linear map $\mathcal{L}:\mathbb{R}\to\mathbb{R}^n$ satisfying*

$$\big\|s\cdot(\phi_2\circ\mathcal{L}\circ\phi_1)+b-f\big\|_{L^p([0,1]^d)} \le \omega_f(\sqrt{d}\,2^{-n})+2^{-n+2}\omega_f(\sqrt{d}),$$

where $s=2\omega_f(\sqrt{d})$, $b=f(\mathbf{0})-\omega_f(\sqrt{d})$, and $\mathcal{L}$ is a linear map given by $\mathcal{L}(t)=(a_1t,a_2t,\cdots,a_nt)$ with $a_1,a_2,\cdots,a_n\in[0,\frac{1}{3})$ determined by f and n.

In Theorem 1.1, s is a scale factor, b is the bias for a vertical shift, and $a_1,a_2,\cdots,a_n\in[0,\frac{1}{3})$ are the key intrinsic parameters storing most of information of f. Clearly, $s\cdot(\phi_2\circ\mathcal{L}\circ\phi_1)+b$ can be implemented by a ReLU network with $n+2$ intrinsic parameters. We call ϕ_1 and ϕ_2 **inner-function** and **outer-function**, respectively. They are independent of the target function f and can be implemented by ReLU networks.

Note that the approximation error in Theorem 1.1 is characterized by the L^p-norm for $p\in[1,\infty)$. In fact, we can extend such a result to a similar one measured in the L^∞-norm.

Theorem 1.2. *Given any $n\in\mathbb{N}^+$, there exist $\boldsymbol{\phi}_1\in\mathcal{H}_{3^d2^{dn+5}}(d,3^d)$ and $\phi_2\in\mathcal{H}_{3^d2^{dn+8}n}(3^dn,1)$ such that: For any $f\in C([0,1]^d)$, there exists a linear map $\mathcal{L}:\mathbb{R}^{3^d}\to\mathbb{R}^{3^dn}$ satisfying*

$$\|s\cdot(\phi_2\circ\mathcal{L}\circ\boldsymbol{\phi}_1)+b-f\|_{L^\infty([0,1]^d)} \le \omega_f(\sqrt{d}\,2^{-n})+2^{-n+2}\omega_f(\sqrt{d}),$$

where $s=2\omega_f(\sqrt{d})$, $b=f(\mathbf{0})-\omega_f(\sqrt{d})$, and $\mathcal{L}$ is given by

$$\mathcal{L}(x_1,\cdots,x_{3^d}) = \Big(\mathcal{L}_0(x_1),\cdots,\mathcal{L}_0(x_{3^d})\Big) \quad \text{for any } \boldsymbol{x}=(x_1,\cdots,x_{3^d})\in\mathbb{R}^{3^d},$$

where $\mathcal{L}_0:\mathbb{R}\to\mathbb{R}^n$ is a linear map given by $\mathcal{L}_0(t)=(a_1t,a_2t,\cdots,a_nt)$ with $a_1,a_2,\cdots,a_n\in[0,\frac{1}{3})$ determined by f and n.

Simplifying the implicit approximation error in Theorem 1.1 (or 1.2) to make it explicitly depending on n is challenging in general, since the modulus of continuity $\omega_f(\cdot)$ may be complicated. However, if f is a Hölder continuous function on $[0,1]^d$ of order $\alpha \in (0,1]$ with a Hölder constant $\lambda > 0$. That is, f satisfies

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \le \lambda \|\boldsymbol{x} - \boldsymbol{y}\|_2^\alpha \quad \text{for any } \boldsymbol{x}, \boldsymbol{y} \in [0,1]^d,$$

implying $\omega_f(r) \le \lambda r^\alpha$ for any $r \ge 0$. This means we can get an exponentially small approximation error $5\lambda d^{\alpha/2} 2^{-\alpha n}$. In particular, in the special case of $\alpha = 1$, i.e., f is a Lipschitz continuous function with a Lipschitz constant $\lambda > 0$, then the approximation error is simplified to $5\lambda\sqrt{d}\, 2^{-n}$.

Though the linear map $\mathcal{L}$ in Theorem 1.2 is essentially determined by n key parameters $a_1, a_2, \cdots, a_n$, these n key parameters are repeated 3^d times in the final network architecture as shown in Figure 3. Therefore, $s \cdot (\phi_2 \circ \mathcal{L} \circ \boldsymbol{\phi}_1) + b$ can be implemented by a ReLU network with $3^d n + 2$ intrinsic parameters. Remark that we can reduce the number of intrinsic parameters to $n + 2$ via using a fixed ReLU network to copy n key parameters 3^d times. The idea is similar to that of Theorem 1.3 introduced later.

Furthermore, the number of intrinsic parameters can be reduced to three in the case of Hölder continuous functions. In fact, three intrinsic parameters are enough to achieve an arbitrary pre-specified error if sufficiently high precision is provided, as shown in the theorem below.

Theorem 1.3. *Given any $\varepsilon > 0$, $\alpha \in (0,1]$, and $\lambda > 0$, there exists $\phi \in \mathcal{H}(d+1, 1)$ such that: For any Hölder continuous function f on $[0,1]^d$ of order $\alpha \in (0,1]$ with a Hölder constant $\lambda > 0$, there exist three parameters $s \in [0, \infty)$, $v \in [0,1)$, and $b \in \mathbb{R}$ satisfying*

$$\left|s\phi(\boldsymbol{x}, v) + b - f(\boldsymbol{x})\right| \le \varepsilon \quad \text{for any } \boldsymbol{x} \in [0,1]^d.$$

In Theorem 1.3, s is a scale factor, b is the bias for a vertical shift, and v is the key intrinsic parameter storing sufficient information of the target function f. Clearly, s, b, and v are learned from f, while ϕ is independent of f. Let $\phi_2 = \phi$, $\boldsymbol{\phi}_1$ be the identity map on $\mathbb{R}^d$, and $\mathcal{L}_v : \mathbb{R}^d \to \mathbb{R}^{d+1}$ be an affine linear transform mapping $\boldsymbol{x}$ to $(\boldsymbol{x}, v)$. Then $s\phi(\boldsymbol{x}, v) + b$ can also be represented as $s\phi_2 \circ \mathcal{L}_v \circ \boldsymbol{\phi}_1(\boldsymbol{x}) + b$.

Remark that Theorem 1.3 is just a theoretical result since the key intrinsic parameter "v" requires extremely high precision, which is necessary for storing the values of f at sufficiently many points within a sufficiently small error. Via the idea of the binary representation, we can extract the values of f stored in "v" via an f-independent ReLU network (as a sub-network of the final network realizing ϕ in Theorem 1.3). In fact, there is a balance between the precision requirement and the number of intrinsic parameters. For example, if we store the values of f in two intrinsic parameters (not one), then the precision requirement is greatly lessened.

1.2 Contributions and further interpretation

Our key contributions can be summarized as follows.

(i) First, we prove by construction in Theorem 1.1 that a ReLU network with $n + 2$ intrinsic parameters can approximate a continuous function f on $[0,1]^d$ with an

error $\omega_f(\sqrt{d}\,2^{-n})+2^{-n+2}\omega_f(\sqrt{d})$ measured in the L^p-norm for $p\in[1,\infty)$. In the case of Hölder continuous functions, the approximation error is simplified to $5\lambda d^{\alpha/2}2^{-\alpha n}$, where $\alpha\in(0,1]$ and $\lambda>0$ are the Hölder order and constant, respectively.

(ii) Next, we generalize the approximation error in Theorem 1.1 from the L^p-norm for $p\in[1,\infty)$ to the L^∞-norm, as shown in Theorem 1.2. Such an generalization is at a price of more intrinsic parameters. To be precise, the final network in Theorem 1.2 has $3^d n+2$ intrinsic parameters, compared to $n+2$ ones in Theorem 1.1.

(iii) Finally, we show in Theorem 1.3 that the number of intrinsic parameters in Theorems 1.1 and 1.2 can be further reduced to three in the case of Hölder continuous functions. To be precise, ReLU networks with three intrinsic parameters can achieve an arbitrary error for approximating Hölder continuous function on $[0,1]^d$. In this scenario, extremely high precision is required as we shall see later.

Approximation with inner-function and outer-function

The composition architecture $\phi_2\circ\mathcal{L}\circ\boldsymbol{\phi}_1$ is the key part of the final ReLU-network-realized function $s\cdot(\phi_2\circ\mathcal{L}\circ\boldsymbol{\phi}_1)+b$ in Theorem 1.1 (or 1.2). Given a target function f, the composition architecture $\phi_2\circ\mathcal{L}\circ\boldsymbol{\phi}_1$ can be generalized to $\phi_{\text{out}}\circ\boldsymbol{\phi}_{\boldsymbol{\theta}}\circ\boldsymbol{\phi}_{\text{in}}$, where $\boldsymbol{\phi}_{\boldsymbol{\theta}}$ an f-dependent function parameterized by $\boldsymbol{\theta}=\boldsymbol{\theta}(f)\in\mathbb{R}^n$, and $\boldsymbol{\phi}_{\text{in}}$ and ϕ_{out} are two f-independent functions, called the inner-function and the outer-function, respectively.

Let $\Gamma_n(d_1,d_2)$ denote a general space consisting of vector-valued functions mapping from $\mathbb{R}^{d_1}$ to $\mathbb{R}^{d_2}$ and pamameterized with n parameters. Each element of $\Gamma_n(d_1,d_2)$ can be denoted by $\boldsymbol{\phi}_{\boldsymbol{\theta}}:\mathbb{R}^{d_1}\to\mathbb{R}^{d_2}$, parameterized with $\boldsymbol{\theta}\in\mathbb{R}^n$. Existing literature uses $\boldsymbol{\phi}_{\boldsymbol{\theta}}\in\Gamma_n(d_1,d_2)$ with proper d_1 and d_2 to directly approximate the target function f in a given function space. This paper proposes a new perspective to study the function approximation for $f:[0,1]^d\to\mathbb{R}$. To be precise, we design an inner-function $\boldsymbol{\phi}_{\text{in}}:\mathbb{R}^d\to\mathbb{R}^{d_1}$ and an outer-function $\phi_{\text{out}}:\mathbb{R}^{d_2}\to\mathbb{R}$, both of which are independent of f, and use $s\cdot(\phi_{\text{out}}\circ\boldsymbol{\phi}_{\boldsymbol{\theta}}\circ\boldsymbol{\phi}_{\text{in}})+b$ with $\boldsymbol{\phi}_{\boldsymbol{\theta}}\in\Gamma_n(d_1,d_2)$ to approximate the target function f. Here, s is a scale factor, b is the bias for a vertical shift, and $\boldsymbol{\theta}\in\mathbb{R}^n$ is the key parameter vector learned from f.

In particular, let Γ_n denote the function of all functions realized by a ReLU network with n parameters and a pre-specified architecture. It is proved in [38, 45] that, when using elements in Γ_n to approximate Hölder continuous functions on $[0,1]^d$, the (nearly) optimal approximation error is $\mathcal{O}(\lambda n^{-2\alpha/d})$, where $\alpha\in(0,1]$ and $\lambda>0$ are the Hölder order and constant, respectively. Clearly, an error like $\mathcal{O}(\lambda n^{-2\alpha/d})$ suffers from the curse of dimensionaltiy. However, with inner-function $\boldsymbol{\phi}_{\text{in}}$ and outer-function ϕ_{out} in hand, we can use $s\cdot(\phi_{\text{out}}\circ\boldsymbol{\phi}_{\boldsymbol{\theta}}\circ\boldsymbol{\phi}_{\text{in}})+b$ with $\boldsymbol{\phi}_{\boldsymbol{\theta}}\in\Gamma_n$ to approximate a Hölder continuous function on $[0,1]^d$ with an exponentially small error $5\lambda d^{\alpha/2}2^{-\alpha n}$, where $\alpha\in(0,1]$ and $\lambda>0$ are the Hölder order and constant, respectively. This means that the approximation error can be greatly improved by pre-designing two (vector-valued) functions $\boldsymbol{\phi}_{\text{in}}$ and ϕ_{out}, which are independent of f and can be implemented by ReLU networks.

Connection with transfer learning

Transfer learning dates back to 1970s [9, 10]. It is a research direction in machine learning that applies knowledge gained in one problem to solve a different but related problem. Typically in deep learning, transfer learning uses a pre-trained neural network obtained for one task as an initial guess of the neural network for another task to achieve a short training time. Our theory in this paper could provide insights into the success of transfer learning using neural networks, though the setting of our theory is different from realistic transfer learning. In our theory, $\boldsymbol{\phi}_{\text{in}}$ and ϕ_{out} are universally useful for all learning tasks for continuous functions, which can be understood as the part of networks that can be transferred to different tasks. Suppose f_1 and f_2 are the target functions for two different but related tasks. If f_1 has been learned via an architecture $s_1 \cdot (\phi_{\text{out}} \circ \phi_{\boldsymbol{\theta}_1} \circ \boldsymbol{\phi}_{\text{in}}) + b_1$, then we can "transfer" the knowledge (ϕ_{out} and $\boldsymbol{\phi}_{\text{in}}$) to another task. This means that, by only learning $s_2, b_2, \boldsymbol{\theta}_2$ from f_2, we can use $s_2 \cdot (\phi_{\text{out}} \circ \phi_{\boldsymbol{\theta}_2} \circ \boldsymbol{\phi}_{\text{in}}) + b_2$ to approximate f_2 well. Therefore, the total number of parameters that need to be learned again is not large. Our theory may provide a certain theoretical understanding in the spirit of transfer learning from a network approximation perspective. To gain a deeper understanding, one can refer to [6, 22, 26, 27, 32, 34, 35]. It would be interesting to test the proposed network architecture in the context of transfer learning in the future.

Error analysis of deep learning

In supervised learning, an unknown target function f defined on a domain $\mathcal{X}$ is learned through its finitely many samples $\{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i=1}^n$. For simplicity, denote $\phi(\boldsymbol{x}; \boldsymbol{\theta})$ as a network-generated function with $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ as the set of parameters, where $\boldsymbol{\Theta}$ is the parameter domain typically taken as $[-M, M]^W$ for two pre-specified constants $M > 0$ and $W \in \mathbb{N}^+$. If $\phi(\boldsymbol{x}; \boldsymbol{\theta})$ is used to infer $f(\boldsymbol{x})$ for unseen data samples $\boldsymbol{x}$, then we need to identify the empirical risk minimizer $\boldsymbol{\theta}_{\mathcal{S}}$, which is given by

$$\boldsymbol{\theta}_{\mathcal{S}} \in \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} R_{\mathcal{S}}(\boldsymbol{\theta}), \quad \text{where } R_{\mathcal{S}}(\boldsymbol{\theta}) := \frac{1}{n} \sum_{i=1}^{n} \ell\big(\phi(\boldsymbol{x}_i; \boldsymbol{\theta}), f(\boldsymbol{x}_i)\big) \tag{1.1}$$

with a loss function typically taken as $\ell(y, y') = \frac{1}{2}|y - y'|^2$.

In fact, the best network-generated function to infer $f(\boldsymbol{x})$ is $\phi(\boldsymbol{x}; \boldsymbol{\theta}_{\mathcal{D}})$, but not $\phi(\boldsymbol{x}; \boldsymbol{\theta}_{\mathcal{S}})$, where $\boldsymbol{\theta}_{\mathcal{D}}$ is the expected risk minimizer given by

$$\boldsymbol{\theta}_{\mathcal{D}} \in \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} R_{\mathcal{D}}(\boldsymbol{\theta}), \quad \text{where } R_{\mathcal{D}}(\boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{x} \sim U(\mathcal{X})} \left[\ell(\phi(\boldsymbol{x}; \boldsymbol{\theta}), f(\boldsymbol{x}))\right],$$

where U is a unknown data distribution over $\mathcal{X}$. The best possible inference error is $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$. In real applications, $U(\mathcal{X})$ is unknown and only finitely many samples from this distribution are available. Hence, the empirical risk $R_{\mathcal{S}}(\boldsymbol{\theta})$ is minimized, hoping to obtain $\phi(\boldsymbol{x}; \boldsymbol{\theta}_{\mathcal{S}})$, instead of minimizing the expected risk $R_{\mathcal{D}}(\boldsymbol{\theta})$ to obtain $\phi(\boldsymbol{x}; \boldsymbol{\theta}_{\mathcal{D}})$. In practice, a numerical optimization method to solve (1.1) may result in a numerical solution (denoted as $\boldsymbol{\theta}_{\mathcal{N}}$) that may not be a global minimizer $\boldsymbol{\theta}_{\mathcal{S}}$. Therefore, the actually learned network-generated function to infer $f(\boldsymbol{x})$ is $\phi(\boldsymbol{x}; \boldsymbol{\theta}_{\mathcal{N}})$ and the corresponding

inference error is measured by $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}})$, which is bounded by

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

The constructive approximation established in this paper and the literature provides an upper bound of $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$. The second term of (1.2) is bounded by the optimization error of the numerical algorithm applied to solve the empirical risk minimization problem in (1.1). If the numerical algorithm can find a global minimizer, the second term is equal to zero. The theoretical guarantee of the convergence of an optimization algorithm to a global minimizer $\boldsymbol{\theta}_{\mathcal{S}}$ and the characterization of the convergence belong to the optimization analysis of neural networks. The study of the bounds for the third and fourth terms of (1.2) is referred to as the generalization error analysis of neural networks.

Theorems 1.1, 1.2, and 1.3 provide upper bounds of $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$. These bounds only depends on the number of intrinsic parameters of ReLU networks and the modulus of continuity $\omega_f(\cdot)$. Hence, these bounds are independent of the empirical risk minimization in (1.1) and the optimization algorithm used to compute the numerical solution of (1.1). In other words, Theorems 1.1, 1.2, and 1.3 quantify the approximation power of ReLU networks in terms of the nubmer of intrinsic parameters. Designing efficient optimization algorithms and analyzing the generalization erre for ReLU networks are two other separate future directions.

1.3 Related work

The expressiveness of deep neural networks has been studied extensively from many perspectives, e.g., in terms of combinatorics [30], topology [7], Vapnik-Chervonenkis (VC) dimension [5, 18, 36], fat-shattering dimension [1, 23], information theory [33], classical approximation theory [2, 3, 8, 12, 13, 14, 16, 17, 20, 24, 25, 28, 31, 37, 38, 43, 44, 45, 48, 49], etc. In the early works of approximation theory for neural networks, the universal approximation theorem [14, 19, 20] 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 sufficiently smooth functions, Barron [3, 4] 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.

Recently, it is proved in [38, 45, 48] that the (nearly) optimal approximation error would be $\mathcal{O}(n^{-2/d})$ when using ReLU networks with n parameters to approximate functions in the unit ball of Lipschitz continuous function space. Clearly, such an error suffers from the curse of dimensionality. To bridge this gap, one could either consider smaller function spaces, e.g., smooth functions [25, 47] and band-limited functions [29], or introducing new networks, e.g., Floor-ReLU networks [40], Floor-Exponential-Step (FLES) networks [41], and (Sin, ReLU, 2^x)-activated networks [21]. This paper proposes a new perspective to characterize the approximation error in terms of the number of intrinsic parameters. Such a method is inspired by an observation that most parameters of the

ReLU network approximating a target function are independent of the target function. Thus, most parameters can be assigned or computed in advance. As shown in Theorem 1.1, we can first design an inner-function ϕ_1 and an outer-function ϕ_2, both of which can be implemented by ReLU networks. Then, for any continuous function $f \in C([0,1]^d)$, $s \cdot (\phi_2 \circ \mathcal{L} \circ \phi_1) + b$ can approximate f with an error $\omega_f(\sqrt{d}\,2^{-n}) + 2^{-n+2}\omega_f(\sqrt{d})$ by the following two steps: 1) determining s and b, 2) designing a linear map $\mathcal{L}$ defined by $\mathcal{L}(t) = (a_1 t, \cdots, a_n t)$, where $a_1, \cdots, a_n$ are determined by the target function f. Therefore, we overcome the curse of dimensionality in the sense of the approximation error characterized by the number of intrinsic parameters when the variation of $\omega_f(r)$ as $r \to 0$ is moderate (e.g., $\omega_f(r) \lesssim r^\alpha$ for Hölder continuous functions).

Organization: The rest of this paper is organized as follows. In Section 2, we prove Theorems 1.1, 1.2, and 1.3 based on an auxiliary theorem, Theorem 2.1. Next, the auxiliary theorem is proved in Section 3 based on Proposition 3.1, which is proved later at the end of Section 3. Finally, Section 4 concludes this paper with a short discussion.

2 Constructive proof

In this section, we first list all notations used throughout this paper. Then, we prove Theorems 1.1, 1.2, and 1.3 based on an auxiliary theorem, Theorem 2.1, which will be proved in Section 3.

2.1 Notations

Firstly, let us summarize the main notations 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 the matrix and vector computation. For example, adding a scalar and a vector means adding the scalar to each entry of the vector.
- For $\theta \in [0,1)$, 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 $\mathrm{bin}0.\theta_1\theta_2\cdots\theta_L$ to denote the L-term binary representation of θ, i.e., $\mathrm{bin}0.\theta_1\theta_2\cdots\theta_L \coloneqq \sum_{\ell=1}^{L} \theta_\ell 2^{-\ell}$.
- For any $p \in [1,\infty)$, the p-norm of a vector $\boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d$ is defined by

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

- The expression "a network with width N and depth L" means
  - The maximum width of this network for all **hidden** layers is no more than N.

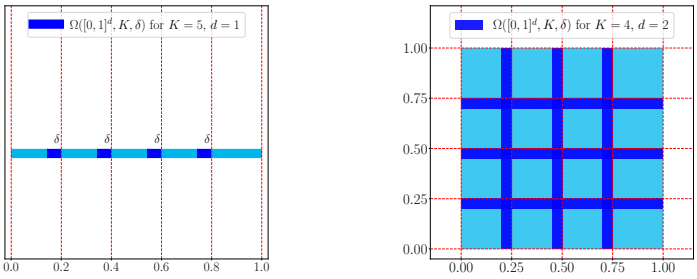
– The number of **hidden** layers of this network is no more than L.

- 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. For example, $\mathrm{mid}(2, 1, 3) = 2$ and $\mathrm{mid}(3, 2, 3) = 3$.

- Given any $K \in \mathbb{N}^+$ and $\delta \in (0, \frac{1}{K}]$, define a trifling region $\Omega([0,1]^d, K, \delta)$ of $[0,1]^d$ as

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

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



(a) (b)

Figure 1: Two examples of trifling regions. (a) $K = 5, d = 1$. (b) $K = 4, d = 2$.

- Given a univariate activation function σ, let us introduce the architecture of a σ-activated network, i.e., a network with each hidden neuron activated by σ. To be precise, a σ-activated network with a vector input $\boldsymbol{x} \in \mathbb{R}^d$, an output $\phi(\boldsymbol{x}) \in \mathbb{R}$, and $L \in \mathbb{N}^+$ hidden layers can be briefly described as follows:

$$\boldsymbol{x} = \widetilde{\boldsymbol{h}}_0 \xrightarrow[\mathcal{L}_0]{\boldsymbol{A}_0,\, \boldsymbol{b}_0} \boldsymbol{h}_1 \xrightarrow{\sigma} \widetilde{\boldsymbol{h}}_1 \quad \cdots \quad \xrightarrow[\mathcal{L}_{L-1}]{\boldsymbol{A}_{L-1},\, \boldsymbol{b}_{L-1}} \boldsymbol{h}_L \xrightarrow{\sigma} \widetilde{\boldsymbol{h}}_L \xrightarrow[\mathcal{L}_L]{\boldsymbol{A}_L,\, \boldsymbol{b}_L} \boldsymbol{h}_{L+1} = \phi(\boldsymbol{x}), \tag{2.2}$$

where $N_0 = d \in \mathbb{N}^+$, $N_1, N_2, \cdots, N_L \in \mathbb{N}^+$, $N_{L+1} = 1$, $\boldsymbol{A}_i \in \mathbb{R}^{N_{i+1} \times N_i}$ 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 $\mathcal{L}_i$, respectively, i.e.,

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

and

$$\widetilde{h}_{i,j} = \sigma(h_{i,j}) \quad \text{for } j = 1, 2, \cdots, N_i \text{ and } i = 1, 2, \cdots, L.$$

Here, $\widetilde{h}_{i,j}$ and $h_{i,j}$ are the j-th entry of $\widetilde{\boldsymbol{h}}_i$ and $\boldsymbol{h}_i$, respectively, for $j = 1, 2, \cdots, N_i$ and $i = 1, 2, \cdots, L$. If σ is applied to a vector entrywisely, i.e.,

$$\sigma(\boldsymbol{y}) = \big(\sigma(y_1), \cdots, \sigma(y_d)\big) \quad \text{for any } \boldsymbol{y} = (y_1, \cdots, y_d) \in \mathbb{R}^d,$$

then ϕ can be represented in a form of function compositions as follows:

$$\phi(\boldsymbol{x}) = \mathcal{L}_L \circ \sigma \circ \quad \cdots \quad \circ \sigma \circ \mathcal{L}_1 \circ \sigma \circ \mathcal{L}_0(\boldsymbol{x}) \quad \text{for any } \boldsymbol{x} \in \mathbb{R}^d.$$

See Figure 2 for an example.

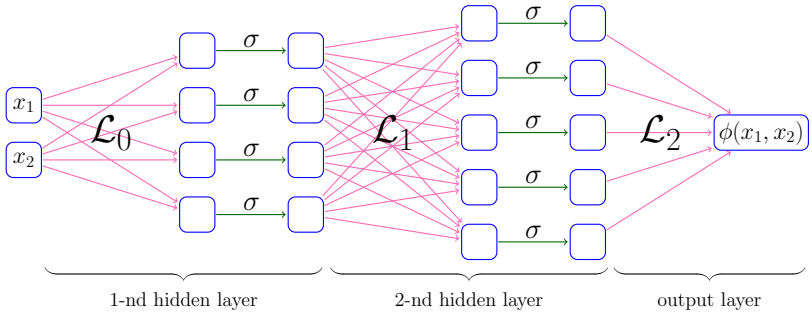


Figure 2: An example of a σ-activated network with width 5 and depth 2.

2.2 Proof of Theorem 1.1

To prove Theorems 1.1 and 1.2, we introduce an auxiliary theorem below with a similar result ignoring the approximation inside the trifling region.

Theorem 2.1. *Given any $n \in \mathbb{N}^+$, there exist $\phi_1 \in \mathcal{H}_{2^{dn+4}}(d, 1)$ and $\phi_2 \in \mathcal{H}_{2^{dn+5}n}(n, 1)$ such that: For any continuous function $f : [0,1]^d \to [0,1]$, there exists a linear map $\mathcal{L} : \mathbb{R} \to \mathbb{R}^n$ satisfying $\|\phi_2 \circ \mathcal{L} \circ \phi_1\|_{L^\infty(\mathbb{R}^d)} \le 1$ and*

$$|\phi_2 \circ \mathcal{L} \circ \phi_1(\boldsymbol{x}) - f(\boldsymbol{x})| \le \omega_f(\sqrt{d}\, 2^{-n}) + 2^{-n} \quad \text{for any } \boldsymbol{x} \in [0,1]^d \backslash \Omega([0,1]^d, K, \delta),$$

where $K = 2^n$, δ is an arbitrary number in $(0, \frac{1}{3K}]$, and $\mathcal{L}$ is given by $\mathcal{L}(t) = (a_1 t, a_2 t, \cdots, a_n t)$. Here, $a_i \in [0, \frac{1}{3})$ is determined by f and n, and $2^m a_i \in \mathbb{N}$ for $i = 1, 2, \cdots, n$, where $m = 2^{dn+1}$.

The proof of Theorem 2.1 can be found later in Section 3. Let us first prove Theorem 1.1 based on Theorem 2.1.

Proof of Theorem 1.1. We may assume f is not a constant function since it is a trivial case. Then $\omega_f(r) > 0$ for any $r > 0$. Set $s = 2\omega_f(\sqrt{d}) > 0$ and $b = f(\mathbf{0}) - \omega_f(\sqrt{d})$. Then, by defining

$$\widetilde{f} \coloneqq \frac{f - b}{s} = \frac{f - f(\mathbf{0}) + \omega_f(\sqrt{d})}{2\omega_f(\sqrt{d})},$$

we have $\widetilde{f}(\boldsymbol{x}) \in [0,1]$ for any $\boldsymbol{x} \in [0,1]^d$. By applying Theorem 2.1 to $\widetilde{f}$, there exist two functions, $\phi_1 : \mathbb{R}^d \to \mathbb{R}$ and $\phi_2 : \mathbb{R}^n \to \mathbb{R}$, both of which are independent of f and can be implemented by ReLU networks with $\le 2^{dn+4}$ and $\le 2^{dn+5}n$ parameters, respectively, and a linear map $\mathcal{L} : \mathbb{R} \to \mathbb{R}^n$ satisfying $\|\phi_2 \circ \mathcal{L} \circ \phi_1\|_{L^\infty(\mathbb{R}^d)} \le 1$ and

$$|\phi_2 \circ \mathcal{L} \circ \phi_1(\boldsymbol{x}) - \widetilde{f}(\boldsymbol{x})| \le \omega_{\widetilde{f}}(\sqrt{d}\, 2^{-n}) + 2^{-n} \quad \text{for any } \boldsymbol{x} \in [0,1]^d \backslash \Omega([0,1]^d, K, \delta),$$

where $K = 2^n$, δ is an arbitrary number in $(0, \frac{1}{3K}]$, and $\mathcal{L}$ is given by $\mathcal{L}(t) = (a_1 t, a_2 t, \cdots, a_n t)$ with $a_1, a_2, \cdots, a_n \in [0, \frac{1}{3})$ determined by $\widetilde{f}$ and n. Since $\widetilde{f}$ is derived from f, $a_1, a_2, \cdots, a_n$ are essentially determined by f and n.

Then choose a small δ satisfying

$$dK\delta 2^p \le 2^{-pn} = (2^{-n})^p.$$

Note that the Lebesgue measure of $\Omega([0,1]^d,K,\delta)$ is bounded by $dK\delta$ and

$$|\phi_2\circ\mathcal{L}\circ\phi_1(\boldsymbol{x})-\widetilde{f}(\boldsymbol{x})|\le|\phi_2\circ\mathcal{L}\circ\phi_1(\boldsymbol{x})|+|\widetilde{f}(\boldsymbol{x})|\le 1+1=2\quad\text{for any }\boldsymbol{x}\in[0,1]^d.$$

Then, $\|\phi_2\circ\mathcal{L}\circ\phi_1-\widetilde{f}\|^p_{L^p([0,1]^d)}$ is bounded by

$$\begin{aligned}&\int_{[0,1]^d\setminus\Omega([0,1]^d,K,\delta)}|\phi_2\circ\mathcal{L}\circ\phi_1(\boldsymbol{x})-\widetilde{f}(\boldsymbol{x})|^p d\boldsymbol{x}+\int_{\Omega([0,1]^d,K,\delta)}|\phi_2\circ\mathcal{L}\circ\phi_1(\boldsymbol{x})-\widetilde{f}(\boldsymbol{x})|^p d\boldsymbol{x}\\&\le\big(\omega_{\widetilde{f}}(\sqrt{d}\,2^{-n})+2^{-n}\big)^p+dK\delta 2^p\le\big(\omega_{\widetilde{f}}(\sqrt{d}\,2^{-n})+2^{-n}\big)^p+(2^{-n})^p\le\big(\omega_{\widetilde{f}}(\sqrt{d}\,2^{-n})+2^{-n+1}\big)^p,\end{aligned}$$

implying $\|\phi_2\circ\mathcal{L}\circ\phi_1-\widetilde{f}\|_{L^p([0,1]^d)}\le\omega_{\widetilde{f}}(\sqrt{d}\,2^{-n})+2^{-n+1}$. Note that $\omega_f(r)=s\cdot\omega_{\widetilde{f}}(r)$ for any $r\ge 0$. Therefore, we have

$$\begin{aligned}&\|s\cdot(\phi_2\circ\mathcal{L}\circ\phi_1)+b-f\|_{L^p([0,1]^d)}=\Big\|s\cdot(\phi_2\circ\mathcal{L}\circ\phi_1)+b-\big(s\cdot\widetilde{f}+b\big)\Big\|_{L^p([0,1]^d)}\\&=s\|\phi_2\circ\mathcal{L}\circ\phi_1-\widetilde{f}\|_{L^p([0,1]^d)}\le s\cdot\omega_{\widetilde{f}}(\sqrt{d}\,2^{-n})+2^{-n+1}s=\omega_f(\sqrt{d}\,2^{-n})+2^{-n+2}\omega_f(\sqrt{d}).\end{aligned}$$

So we finish the proof. □

2.3 Proof of Theorem 1.2

Next, let us prove Theorem 1.2. To this end, we need to introduce the following lemma, which is actually Lemma 3.4 of [25] (or Lemma 3.11 of [48]).

Lemma 2.2 (Lemma 3.4 of [25]). *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*

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

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

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

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

With Lemma 2.2 in hand, we are ready to present the proof of Theorem 1.2.

Proof of Theorem 1.2. We may assume f is not a constant function since it is a trivial case. Then $\omega_f(r)>0$ for any $r>0$. Set $s=2\omega_f(\sqrt{d})>0$ and $b=f(\boldsymbol{0})-\omega_f(\sqrt{d})$. Then, by defining

$$\widetilde{f}:=\frac{f-b}{s}=\frac{f-f(\boldsymbol{0})+\omega_f(\sqrt{d})}{2\omega_f(\sqrt{d})},$$

we have $\widetilde{f}(\boldsymbol{x})\in[0,1]$ for any $\boldsymbol{x}\in[0,1]^d$. By applying Theorem 2.1 to $\widetilde{f}$, there exist two functions, $\psi_{1,0}:\mathbb{R}^d\to\mathbb{R}$ and $\psi_{2,0}:\mathbb{R}^n\to\mathbb{R}$, both of which are independent of $\widetilde{f}$ (or

f) and can be implemented by ReLU networks with $\le 2^{dn+4}$ and $\le 2^{dn+5}n$ parameters, respectively, and a linear map $\mathcal{L}_{\boldsymbol{a},0}:\mathbb{R}\to\mathbb{R}^n$ satisfying $\|\psi_0\|_{L^\infty(\mathbb{R}^d)}\le 1$ and

$$|\psi_0(\boldsymbol{x})-\widetilde{f}(\boldsymbol{x})|\le \omega_{\widetilde{f}}(\sqrt{d}\,2^{-n})+2^{-n}=:\varepsilon \quad \text{for any } \boldsymbol{x}\in[0,1]^d\backslash\Omega([0,1]^d,K,\delta),$$

where $\psi_0:=\psi_{2,0}\circ\mathcal{L}_{\boldsymbol{a},0}\circ\psi_{1,0}$, $K=2^n$, δ is an arbitrary number in $(0,\frac{1}{3K}]$, and $\mathcal{L}_{\boldsymbol{a},0}$ is given by $\mathcal{L}_{\boldsymbol{a},0}(t)=(a_1t,a_2t,\cdots,a_nt)$ with $\boldsymbol{a}=(a_1,a_2,\cdots,a_n)$ determined by f and n. Moreover, $a_i\in[0,\frac{1}{3})$ and $2^m a_i\in\mathbb{N}$ for $i=1,2,\cdots,n$, where $m=2^{dn+1}$.①

Choose a small δ satisfying $d\cdot\omega_{\widetilde{f}}(\delta)\le 2^{-n}$. With $\psi_0=\psi_{2,0}\circ\mathcal{L}_{\boldsymbol{a},0}\circ\psi_{1,0}$ in hand, we can define $\psi_1,\cdots,\psi_d$ by induction via

$$\psi_{i+1}(\boldsymbol{x}):=\mathrm{mid}\Big(\psi_i(\boldsymbol{x}-\delta\boldsymbol{e}_{i+1}),\psi_i(\boldsymbol{x}),\psi_i(\boldsymbol{x}+\delta\boldsymbol{e}_{i+1})\Big) \quad \text{for } i=0,1,\cdots,d-1.$$

The detailed iterative equations for $\psi_i:\mathbb{R}^d\to\mathbb{R}$, $\boldsymbol{\psi}_{1,i}:\mathbb{R}^d\to\mathbb{R}^{3^i}$, $\mathcal{L}_{\boldsymbol{a},i}:\mathbb{R}^{3^i}\to\mathbb{R}^{3^id}$, and $\psi_{2,i}:\mathbb{R}^{3^id}\to\mathbb{R}$, for $i=1,2,\cdots,d$, are listed as follows.

- $\psi_i=\psi_{2,i}\circ\mathcal{L}_{\boldsymbol{a},i}\circ\boldsymbol{\psi}_{1,i}$.
- $\boldsymbol{\psi}_{1,i}(\boldsymbol{y})=\Big(\boldsymbol{\psi}_{1,i-1}(\boldsymbol{y}-\delta\boldsymbol{e}_i),\ \boldsymbol{\psi}_{1,i-1}(\boldsymbol{y}),\ \boldsymbol{\psi}_{1,i-1}(\boldsymbol{y}+\delta\boldsymbol{e}_i)\Big)$ for any $\boldsymbol{y}\in\mathbb{R}^d$.
- $\mathcal{L}_{\boldsymbol{a},i}(\boldsymbol{y}_1,\boldsymbol{y}_2,\boldsymbol{y}_3)=\Big(\mathcal{L}_{\boldsymbol{a},i-1}(\boldsymbol{y}_1),\ \mathcal{L}_{\boldsymbol{a},i-1}(\boldsymbol{y}_2),\ \mathcal{L}_{\boldsymbol{a},i-1}(\boldsymbol{y}_3)\Big)$ for any $\boldsymbol{y}_1,\boldsymbol{y}_2,\boldsymbol{y}_3\in\mathbb{R}^{3^{i-1}}$.
- $\psi_{2,i}(\boldsymbol{y}_1,\boldsymbol{y}_2,\boldsymbol{y}_3)=\mathrm{mid}\Big(\psi_{2,i-1}(\boldsymbol{y}_1),\ \psi_{2,i-1}(\boldsymbol{y}_2),\ \psi_{2,i-1}(\boldsymbol{y}_3)\Big)$ for any $\boldsymbol{y}_1,\boldsymbol{y}_2,\boldsymbol{y}_3\in\mathbb{R}^{3^{i-1}d}$.

See the illustrations in Figure 3.

By Lemma 2.2, we have

$$|\phi(\boldsymbol{x})-\widetilde{f}(\boldsymbol{x})|\le\varepsilon+d\cdot\omega_{\widetilde{f}}(\delta)\le\omega_{\widetilde{f}}(\sqrt{d}\,2^{-n})+2^{-n+1} \quad \text{for any } \boldsymbol{x}\in[0,1]^d,$$

where $\phi:=\psi_d=\psi_{2,d}\circ\mathcal{L}_{\boldsymbol{a},d}\circ\boldsymbol{\psi}_{1,d}$. By defining $\mathcal{L}:=\mathcal{L}_{\boldsymbol{a},d}$, $\boldsymbol{\phi}_1:=\boldsymbol{\psi}_{1,d}$, and $\phi_2:=\psi_{2,d}$, we have

$$|\phi_2\circ\mathcal{L}\circ\boldsymbol{\phi}_1(\boldsymbol{x})-\widetilde{f}(\boldsymbol{x})|\le\omega_{\widetilde{f}}(\sqrt{d}\,2^{-n})+2^{-n+1} \quad \text{for any } \boldsymbol{x}\in[0,1]^d,$$

As shown in Figure 3, $\mathcal{L}:=\mathcal{L}_{\boldsymbol{a},d}$ is a linear map from $\mathbb{R}^{3^d}$ to $\mathbb{R}^{3^dn}$ determined by $\boldsymbol{a}=(a_1,a_2,\cdots,a_n)\in[0,\frac{1}{3})^n$, which depends on f and n. Moreover, as shown in Figure 3, $\boldsymbol{\phi}_1:=\boldsymbol{\psi}_{1,d}$ and $\phi_2:=\psi_{2,d}$ are independent of f and can be implemented by ReLU networks with

$$\le 3^d(2^{dn+4})+3d(d+1)(3^{d-1}+3^{d-2}+\cdots+3^0)\le 3^d2^{dn+5}$$

and

$$\le 3^d(2^{dn+5}n)+280(3^{d-1}+3^{d-2}+\cdots+3^0)\le 3^d2^{dn+8}n$$ ②

parameters, respectively.

Note that $\omega_f(r)=s\cdot\omega_{\widetilde{f}}(r)$ for any $r\ge 0$. Therefore, we have

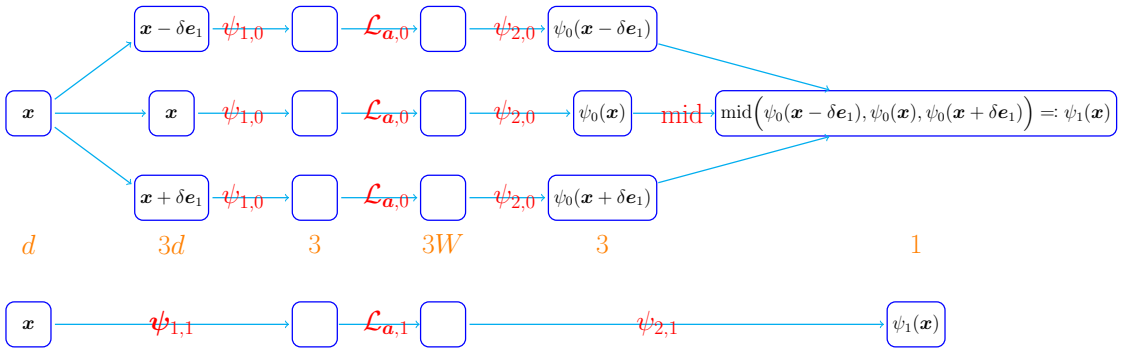
$$\|s\cdot(\phi_2\circ\mathcal{L}\circ\boldsymbol{\phi}_1)+b-f\|_{L^\infty([0,1]^d)}=\Big\|s\cdot(\phi_2\circ\mathcal{L}\circ\boldsymbol{\phi}_1)+b-\big(s\cdot\widetilde{f}+b\big)\Big\|_{L^\infty([0,1]^d)}$$

$$=s\|\phi_2\circ\mathcal{L}\circ\boldsymbol{\phi}_1-\widetilde{f}\|_{L^\infty([0,1]^d)}\le s\cdot\omega_{\widetilde{f}}(\sqrt{d}\,2^{-n})+2^{-n+1}s=\omega_f(\sqrt{d}\,2^{-n})+2^{-n+2}\omega_f(\sqrt{d}).$$

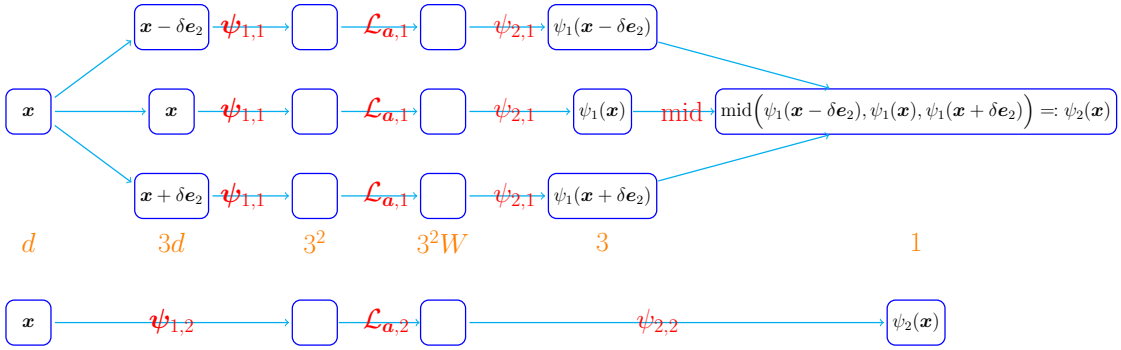
So we finish the proof. □

①This property will be used in the proof of Theorem 1.3.

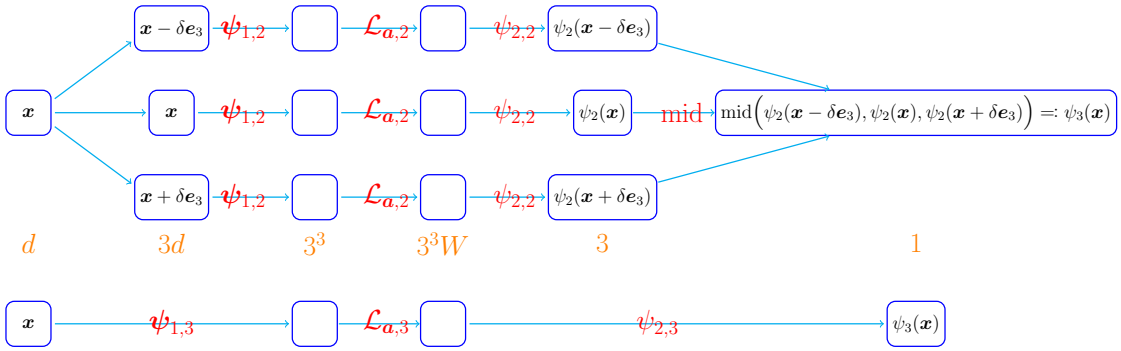
②As shown Lemma 3.1 of [25], "mid$(\cdot,\cdot,\cdot)$" can be implemented by a ReLU network with width 14 and depth 2, which has $\le(3+1)\times14+(14+1)\times14+(14+1)=280$ parameters.



(a) An illustration of the network architecture implementing $\psi_1 = \psi_{2,1} \circ \boldsymbol{\mathcal{L}}_{\boldsymbol{a},1} \circ \boldsymbol{\psi}_{1,1}$ based on $\psi_0 = \psi_{2,0} \circ \boldsymbol{\mathcal{L}}_{\boldsymbol{a},0} \circ \psi_{1,0}$. The top architecture is in detail, while the bottom one is just a sketch of the top one. The orange numbers indicate the number of neurons in each layer.



(b) An illustration of the network architecture implementing $\psi_2 = \psi_{2,2} \circ \boldsymbol{\mathcal{L}}_{\boldsymbol{a},2} \circ \boldsymbol{\psi}_{1,2}$ based on $\psi_1 = \psi_{2,1} \circ \boldsymbol{\mathcal{L}}_{\boldsymbol{a},1} \circ \boldsymbol{\psi}_{1,1}$. The top architecture is in detail, while the bottom one is just a sketch of the top one. The orange numbers indicate the number of neurons in each layer.



(c) An illustration of the network architecture implementing $\psi_3 = \psi_{2,3} \circ \boldsymbol{\mathcal{L}}_{\boldsymbol{a},3} \circ \boldsymbol{\psi}_{1,3}$ based on $\psi_2 = \psi_{2,2} \circ \boldsymbol{\mathcal{L}}_{\boldsymbol{a},2} \circ \boldsymbol{\psi}_{1,2}$. The top architecture is in detail, while the bottom one is just a sketch of the top one. The orange numbers indicate the number of neurons in each layer.

Figure 3: Illustrations of the implementations of ϕ_1, ϕ_2, and ϕ_3. The inductive implementations of $\phi_4, \cdots, \phi_d$ are similar.

2.4 Proof of Theorem 1.3

To simplify the proof of Theorem 1.3, we introduce two lemmas below. First, we need to establish a lemma showing how to store many parameters in one intrinsic parameter via a fixed network.

Lemma 2.3. *Given any $m, n \in \mathbb{N}$, there exists a vector-valued function $\boldsymbol{\phi} : \mathbb{R} \to \mathbb{R}^n$ realized by a ReLU network such that: For any $a_i \in [0,1)$ with $2^m a_i \in \mathbb{N}$ for $i = 0, 1, \cdots, n$, there exists a real number $v \in [0,1)$ such that*

$$\boldsymbol{\phi}(v) = (a_1, a_2, \cdots, a_n).$$

Next, we establish another lemma using a ReLU network to uniformly approximate multiplication operation $\psi(x, y) = xy$ well.

Lemma 2.4. *For any $M > 0$ and $\eta > 0$, there exists a function $\psi_\eta : \mathbb{R}^2 \to \mathbb{R}$ realized by a ReLU network such that*

$$\psi_\eta(x, y) \rightrightarrows \psi(x, y) = xy \quad \text{on } [-M, M]^2 \quad \text{as } \eta \to 0^+,$$

where $\rightrightarrows$ denotes the uniform convergence.

The proof of Lemma 2.3 is placed later in this section. Lemma 2.4 is just a direct result of Lemma 4.2 of [25]. With Lemmas 2.3 and 2.4 in hand, we are ready to prove Theorem 1.3.

Proof of Theorem 1.3. For any $\varepsilon > 0$, choose a large $n = n(\varepsilon, \alpha, \lambda) \in \mathbb{N}^+$ such that

$$5\lambda d^{\alpha/2} 2^{-\alpha n} \le \varepsilon/2.$$

Since f is a Hölder continuous function on $[0,1]^d$ of order $\alpha \in (0,1]$ with a Hölder constant $\lambda > 0$, we have $\omega_f(r) \le \lambda r^\alpha$ for any $r \ge 0$. By Theorem 1.2, there exist two functions $\boldsymbol{\phi}_1 : \mathbb{R}^d \to \mathbb{R}^{3^d}$ and $\phi_2 : \mathbb{R}^{3^d n} \to \mathbb{R}$, implemented by f-independent ReLU networks, such that

$$\|s(\phi_2 \circ \mathcal{L} \circ \boldsymbol{\phi}_1) + b - f\|_{L^\infty([0,1]^d)} \le \omega_f(\sqrt{d}\, 2^{-n}) + 2^{-n+2}\omega_f(\sqrt{d}) \le 5\lambda d^{\alpha/2} 2^{-\alpha n} \le \varepsilon/2, \quad (2.3)$$

where $s = 2\omega_f(\sqrt{d}) \le 2\lambda d^{\alpha/2}$, $b = f(\mathbf{0}) - \omega_f(\sqrt{d})$, and $\mathcal{L} : \mathbb{R}^{3^d} \to \mathbb{R}^{3^d n}$ is a linear map given by

$$\mathcal{L}(y_1, \cdots, y_{3^d}) = \Big(\mathcal{L}_0(y_1), \cdots, \mathcal{L}_0(y_{3^d})\Big) \quad \text{for any } \boldsymbol{y} = (y_1, \cdots, y_{3^d}) \in \mathbb{R}^{3^d},$$

where $\mathcal{L}_0 : \mathbb{R} \to \mathbb{R}^n$ is a linear map given by $\mathcal{L}_0(t) = (a_1 t, a_2 t, \cdots, a_n t)$ with $a_1, a_2, \cdots, a_n \in [0, \frac{1}{3})$ determined by f and n. Since $a_1, a_2, \cdots, a_n$ are repeated 3^d times in the definition of $\mathcal{L}$, there are $3^d n$ parameters in total. We will show how to store these $3^d n$ parameters in one intrinsic parameter v via an f-independent ReLU network as shown in the following two steps.

- Regard $a_1, a_2, \cdots, a_n$ as inputs, but not parameters. See the difference in Figure 4. Then, we only need to store $a_1, a_2, \cdots, a_n$ one time by copying them 3^d times.

- As stated in the proof of Theorem 1.2, $a_1, a_2, \cdots, a_n$ have finite binary representations. Then, we can store them in a key parameter v and use an f-independent ReLU network to extract them from v.

The details of these two steps can be found below.

Step 1: Regard $a_1, a_2, \cdots, a_n$ as inputs and copy them 3^d times.

Since $a_1, a_2, \cdots, a_n$ are regraded as inputs, the implementation of $\mathcal{L}_0(t) = (a_1 t, a_2 t, \cdots, a_n t)$ requires multiplication operations. This means that we need to approximate $\psi(x, y) = xy$ well via an f-independent ReLU network. See Figure 4 for illustrations.

Denote $\boldsymbol{a} = (a_1, a_2, \cdots, a_n)$ and

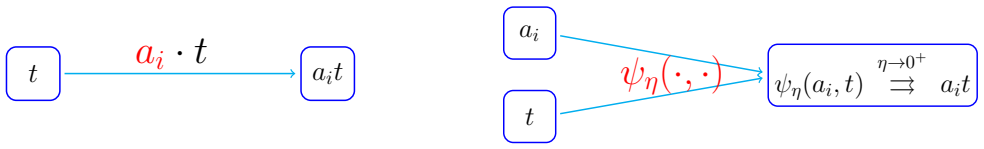
$$\boldsymbol{\phi}_1(\boldsymbol{x}) = \Big(\phi_{1,1}(\boldsymbol{x}),\ \phi_{1,2}(\boldsymbol{x}),\ \cdots,\ \phi_{1,3^d}(\boldsymbol{x})\Big) \quad \text{for any } \boldsymbol{x} \in [0,1]^d.$$

Then define

$$M \coloneqq 1 + \sup\Big\{|\phi_{1,j}(\boldsymbol{x})| : \boldsymbol{x} \in [0,1]^d,\ j = 1, 2, \cdots, 3^d\Big\}.$$

By Lemma 2.4, there exists a function $\psi_\eta : \mathbb{R}^2 \to \mathbb{R}$ realized by a ReLU network such that

$$\psi_\eta(x, y) \rightrightarrows \psi(x, y) = xy \quad \text{on } [-M, M]^2 \quad \text{as } \eta \to 0^+.$$



(a) (b)

Figure 4: Illustrations of two methods getting/approximating $a_i t$ for $i = 1, 2, \cdots, n$ and $t \in \{\phi_{1,j}(\boldsymbol{x}) : \boldsymbol{x} \in [0,1]^d,\ j = 1, 2, \cdots, 3^d\}$. (a) By regarding a_i as a parameter, one can easily get the product of an input t and an parameter a_i. (b) By regarding a_i as an input, one needs to use a ReLU network to approximate the multiplication operation for approximating $a_i t$ well.

Note that $\mathcal{L}$ and $\mathcal{L}_0$ depend on $\boldsymbol{a} = (a_1, a_2, \cdots, a_n)$. For clarity, we denote $\mathcal{L}_{\boldsymbol{a}} = \mathcal{L}$ and $\mathcal{L}_{\boldsymbol{a},0} = \mathcal{L}_0$. For any $a_1, a_2, \cdots, a_n \in [0, \frac{1}{3}) \subseteq [-M, M]$ and

$$t \in \Big\{\phi_{1,j}(\boldsymbol{x}) : \boldsymbol{x} \in [0,1]^d,\ j = 1, 2, \cdots, 3^d\Big\} \subseteq [-M, M],$$

we can use

$$\mathcal{L}_{\boldsymbol{a},0,\eta}(t) \coloneqq \Big(\psi_\eta(a_1, t),\ \psi_\eta(a_2, t),\ \cdots,\ \psi_\eta(a_n, t)\Big)$$

to approximate

$$\mathcal{L}_{\boldsymbol{a},0}(t) = (a_1 t, a_2 t, \cdots, a_n t) = \Big(\psi(a_1, t),\ \psi(a_2, t),\ \cdots,\ \psi(a_n, t)\Big).$$

Then

$$\mathcal{L}_{\boldsymbol{a},\eta}(y_1, \cdots, y_{3^d}) \coloneqq \Big(\mathcal{L}_{\boldsymbol{a},0,\eta}(y_1),\ \cdots,\ \mathcal{L}_{\boldsymbol{a},0,\eta}(y_{3^d})\Big)$$

can also approximate

$$\mathcal{L}_{\boldsymbol{a}}(y_1,\cdots,y_{3^d}) = \Big(\mathcal{L}_{\boldsymbol{a},0}(y_1),\ \cdots,\ \mathcal{L}_{\boldsymbol{a},0}(y_{3^d})\Big)$$

well. Define $\widetilde{\phi}_\eta : \mathbb{R}^{n+3^d} \to \mathbb{R}^{3^d n}$ and $\widetilde{\phi} : \mathbb{R}^{n+3^d} \to \mathbb{R}^{3^d n}$ via

$$\widetilde{\phi}_\eta(\boldsymbol{y},\boldsymbol{a}) \coloneqq \mathcal{L}_{\boldsymbol{a},\eta}(y_1,\cdots,y_{3^d}) \quad \text{for any } \boldsymbol{a}\in[0,\tfrac{1}{3})^n \text{ and } \boldsymbol{y}=(y_1,\cdots,y_{3^d})\in[-M,M]^{3^d}$$

and

$$\widetilde{\phi}(\boldsymbol{y},\boldsymbol{a}) \coloneqq \mathcal{L}_{\boldsymbol{a}}(y_1,\cdots,y_{3^d}) \quad \text{for any } \boldsymbol{a}\in[0,\tfrac{1}{3})^n \text{ and } \boldsymbol{y}=(y_1,\cdots,y_{3^d})\in[-M,M]^{3^d}.$$

Since

$$\psi_\eta(x,y) \rightrightarrows \psi(x,y)=xy \quad \text{on } [-M,M]^2 \quad \text{as } \eta\to 0^+,$$

it is easy to verify that

$$\widetilde{\phi}_\eta(\boldsymbol{y},\boldsymbol{a}) \rightrightarrows \widetilde{\phi}(\boldsymbol{y},\boldsymbol{a}) \quad \text{for } \boldsymbol{y}\in[-M,M]^{3^d} \text{ and } \boldsymbol{a}\in[0,\tfrac{1}{3})^n \quad \text{as } \eta\to 0^+.$$

Note that $\boldsymbol{\phi}_1(\boldsymbol{x})\in[-M,M]^{3^d}$ for any $\boldsymbol{x}\in[0,1]^d$. Then,

$$\phi_2\circ\widetilde{\phi}_\eta\Big(\boldsymbol{\phi}_1(\boldsymbol{x}),\boldsymbol{a}\Big) \rightrightarrows \phi_2\circ\widetilde{\phi}\Big(\boldsymbol{\phi}_1(\boldsymbol{x}),\boldsymbol{a}\Big) \quad \text{for } \boldsymbol{x}\in[0,1]^d \text{ and } \boldsymbol{a}\in[0,\tfrac{1}{3})^n \quad \text{as } \eta\to 0^+.$$

The fact $\widetilde{\phi}(\boldsymbol{y},\boldsymbol{a})=\mathcal{L}_{\boldsymbol{a}}(\boldsymbol{y})=\mathcal{L}(\boldsymbol{y})$ implies $\phi_2\circ\widetilde{\phi}\Big(\boldsymbol{\phi}_1(\boldsymbol{x}),\boldsymbol{a}\Big)=\phi_2\circ\mathcal{L}\circ\boldsymbol{\phi}_1(\boldsymbol{x})$. Therefore,

$$\phi_2\circ\widetilde{\phi}_\eta\Big(\boldsymbol{\phi}_1(\boldsymbol{x}),\boldsymbol{a}\Big) \rightrightarrows \phi_2\circ\mathcal{L}\circ\boldsymbol{\phi}_1(\boldsymbol{x}) \quad \text{for } \boldsymbol{x}\in[0,1]^d \text{ and } \boldsymbol{a}\in[0,\tfrac{1}{3})^n \quad \text{as } \eta\to 0^+.$$

Choose a small $\eta=\eta(n)>0$ such that

$$\Big|\phi_2\circ\widetilde{\phi}_\eta\Big(\boldsymbol{\phi}_1(\boldsymbol{x}),\boldsymbol{a}\Big)-\phi_2\circ\mathcal{L}\circ\boldsymbol{\phi}_1(\boldsymbol{x})\Big|\le 2^{-n} \quad \text{for any } \boldsymbol{x}\in[0,1]^d \text{ and } \boldsymbol{a}\in[0,\tfrac{1}{3})^n\ . \tag{2.4}$$

Recall that ψ_η can be realized by an f-independent ReLU network. It is easy to verify that $\widetilde{\phi}_\eta$ can also be realized by an f-independent ReLU network.

Step 2: Store $a_1,a_2,\cdots,a_n$ in a key parameter v.

As we can see from the proof of Theorem 1.2, $a_i\in[0,\frac{1}{3})$ with $2^m a_i\in\mathbb{N}$ for $i=1,2,\cdots,n$, where $m=2^{dn+1}$. That is,

$$a_i\in\Big\{\mathrm{bin}0.\theta_1\cdots\theta_m:\ \theta_\ell\in\{0,1\},\ \ell=1,2,\cdots,m\Big\}.$$

Then, by Lemma 2.3, there exists a real number $v\in[0,1)$ and a vector function $\boldsymbol{\phi}_0:\mathbb{R}\to\mathbb{R}^n$ implemented by a ReLU network independent of $a_1,a_2,\cdots,a_n$ such that

$$\boldsymbol{\phi}_0(v)=(a_1,a_2,\cdots,a_n)=\boldsymbol{a}.$$

Next, we can define

$$\phi(\boldsymbol{x},v)\coloneqq\phi_2\circ\widetilde{\phi}_\eta\Big(\boldsymbol{\phi}_1(\boldsymbol{x}),\boldsymbol{\phi}_0(v)\Big)=\phi_2\circ\widetilde{\phi}_\eta\Big(\boldsymbol{\phi}_1(\boldsymbol{x}),\boldsymbol{a}\Big)$$

for any $\boldsymbol{x} \in [0,1]^d$ and $v \in [0,1)$.

Since the ReLU network realizing $\boldsymbol{\phi}_0$ is independent of $a_1, a_2, \cdots, a_n$, and hence independent of f. Recall that $\boldsymbol{\phi}_1$, ϕ_2, and $\widetilde{\phi}_\eta$ can be implemented by f-independent ReLU networks. Hence,

$$\phi(\boldsymbol{x}, v) = \phi_2 \circ \widetilde{\phi}_\eta\Big(\boldsymbol{\phi}_1(\boldsymbol{x}), \boldsymbol{\phi}_0(v)\Big)$$

can also implemented by an f-independent ReLU network. It remains to estimate the error. By Equations (2.3) and (2.4), we have

$$\begin{aligned}
\big|s\phi(\boldsymbol{x}, v) + b - f(\boldsymbol{x})\big| &\le \Big|s\phi(\boldsymbol{x}, v) + b - \big(s\phi_2 \circ \mathcal{L} \circ \boldsymbol{\phi}_1(\boldsymbol{x}) + b\big)\Big| + \Big|s\phi_2 \circ \mathcal{L} \circ \boldsymbol{\phi}_1(\boldsymbol{x}) + b - f(\boldsymbol{x})\Big| \\
&\le s\big|\phi(\boldsymbol{x}, v) - \phi_2 \circ \mathcal{L} \circ \boldsymbol{\phi}_1(\boldsymbol{x})\big| + \varepsilon/2 \\
&\le 2\lambda d^{\alpha/2}\Big|\phi_2 \circ \widetilde{\phi}_\eta\Big(\boldsymbol{\phi}_1(\boldsymbol{x}), \boldsymbol{a}\Big) - \phi_2 \circ \mathcal{L} \circ \boldsymbol{\phi}_1(\boldsymbol{x})\Big| + \varepsilon/2 \\
&\le 2\lambda d^{\alpha/2} 2^{-n} + \varepsilon/2 \le 5\lambda d^{\alpha/2} 2^{-\alpha n} + \varepsilon/2 \le \varepsilon/2 + \varepsilon/2 = \varepsilon.
\end{aligned}$$

So we finish the proof. □

Finally, let us prove Lemma 2.3 to end this section.

Proof of Lemma 2.3. Since $a_i \in [0,1)$ with $2^m a_i \in \mathbb{N}$ for $i = 1, 2, \cdots, n$, a_i can be represented as a binary form

$$a_i = \operatorname{bin} 0.a_{i,1}a_{i,2}\cdots a_{i,m}.$$

Denote

$$v = \sum_{i=1}^{n} 2^{-m(i-1)} a_i = \operatorname{bin} 0.\underbrace{a_{1,1}\cdots a_{1,m}}_{\text{store } a_1}\underbrace{a_{2,1}\cdots a_{2,m}}_{\text{store } a_2} \cdots \underbrace{a_{n,1}\cdots a_{n,m}}_{\text{store } a_n},$$

which requires pretty high precision. It is easy to extract a_i from v via the floor function ($\lfloor\cdot\rfloor$), i.e.,

$$a_i = \lfloor 2^{mi} v \rfloor / 2^m - \lfloor 2^{m(i-1)} v \rfloor \quad \text{for } i = 1, 2, \cdots, n.$$

Next, we need to use a ReLU network to replace the floor function. Let $g : \mathbb{R} \to \mathbb{R}$ be the continuous piecewise linear function with the following breakpoints:

$$(\ell, \ell) \quad \text{and} \quad (\ell + 1 - \delta, \ell) \quad \text{for } \ell = 0, 1, \cdots, 2^{mn} - 1, \quad \text{where } \delta = 2^{-mn}.$$

Clearly, g can be realized by a ReLU network independent of $a_1, a_2, \cdots, a_n$. By defining

$$g_i(t) \coloneqq g(2^{mi} t)/2^m - g(2^{m(i-1)} t) \quad \text{for } i = 1, 2, \cdots, n \text{ and any } t \in \mathbb{R},$$

we have

$$a_i = g(2^{mi} v)/2^m - g(2^{m(i-1)} v) = g_i(v) \quad \text{for } i = 1, 2, \cdots, n.$$

Next, The target function $\boldsymbol{\phi}$ can be defined via

$$\boldsymbol{\phi}(t) = \Big(g_1(t), g_2(t), \cdots, g_n(t)\Big) \quad \text{for any } t \in \mathbb{R}.$$

Thus, we have

$$\boldsymbol{\phi}(v) = \Big(g_1(v), g_2(v), \cdots, g_n(v)\Big) = (a_1, a_2, \cdots, a_n).$$

and $\boldsymbol{\phi}$ can be realized by a ReLU network independent of $a_1, a_2, \cdots, a_n$. So we finish the proof. □

3 Proof of Theorem 2.1

In this section, we first present the proof sketch of Theorem 2.1 in Section 3.1, and then give the detailed proof in Section 3.2 based on Proposition 3.1, which will be proved later in Section 3.3.

3.1 Sketch of proof

Before proving Theorem 2.1, let us present the key steps as follows.

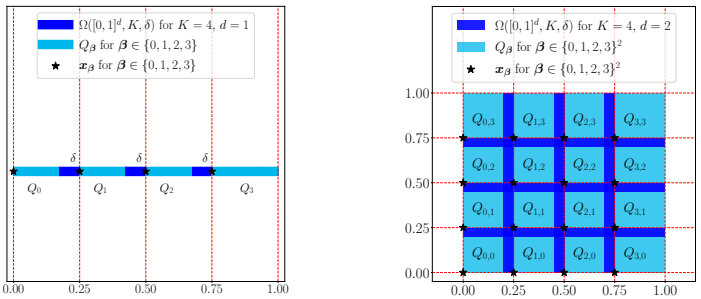
1. Set $K = 2^n$, divide $[0,1]^d$ into K^d cubes $Q_{\boldsymbol{\beta}}$ for $\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d$ and the trifling region $\Omega([0,1]^d, K, \delta)$, and denote $\boldsymbol{x}_{\boldsymbol{\beta}}$ as the vertex of $Q_{\boldsymbol{\beta}}$ with minimum $\|\cdot\|_1$ norm for each $\boldsymbol{\beta}$. See Figure 5 for illustrations.

2. Design a ReLU sub-network to implement a function $\phi_1 : \mathbb{R}^d \to \mathbb{R}$, independent of f, projecting the whole $Q_{\boldsymbol{\beta}}$ to a number determined by $\boldsymbol{\beta}$ in $\{4^j : j = 1,2,\cdots,K^d\}$ for each $\boldsymbol{\beta}$.

3. Design a linear map $\mathcal{L} : \mathbb{R} \to \mathbb{R}^n$, given by $\mathcal{L}(t) = (a_1 t, a_2 t, \cdots, a_n t)$, for later use, where $a_1, a_2, \cdots, a_n$ are determined by f and n.

4. Design a ReLU sub-network to implement a function $\phi_2 : \mathbb{R}^n \to \mathbb{R}$, independent of f, such that $\phi_2 \circ \mathcal{L} \circ \phi_1(\boldsymbol{x}) \approx f(\boldsymbol{x}_{\boldsymbol{\beta}})$ for any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and each $\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d$. Then $\phi_2 \circ \mathcal{L} \circ \phi_1$ approximates f well outside of $\Omega([0,1]^d, K, \delta)$.

5. Estimate the approximation error of $\phi_2 \circ \mathcal{L} \circ \phi_1 \approx f$.



(a) (b)

Figure 5: 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) $K = 4$, $d = 1$. (b) $K = 4$, $d = 2$.

As we shall see later, the constructions of ϕ_1 and $\mathcal{L}$ are not difficult. The most technical part is to design ϕ_2 implemented by a ReLU network, which relies on the following proposition.

Proposition 3.1. *Given any $J \in \mathbb{N}^+$, there exists a function ϕ implemented by a ReLU network with width 2 and depth $2J+2$ such that: For any $\theta_1, \theta_2, \cdots, \theta_J \in \{0,1\}$, we have $\phi(x) \in [0,1]$ for any $x \in \mathbb{R}$ and*

$$\phi(4^j a) = \theta_j \quad \text{for } j = 1,2,\cdots,J, \quad \text{where } a = \sum_{j=1}^{J} \theta_j 4^{-j}. \tag{3.1}$$

The proof of this proposition can be found in Section 3.3. We shall point out that the function ϕ in this proposition is independent of $\theta_1, \theta_2, \cdots, \theta_J \in \{0,1\}$.

3.2 Constructive proof

Now we are ready to give the detailed proof of Theorem 2.1.

Proof of Theorem 2.1. The proof consists of five steps.

Step 1: Set up.

Set $K = 2^n$ and let $\delta > 0$ be a small number determined later. Then define $\boldsymbol{x}_{\boldsymbol{\beta}} \coloneqq \boldsymbol{\beta}/K$ and divide $[0,1]^d$ into K^d cubes $Q_{\boldsymbol{\beta}}$ for $\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d$ and a small region $\Omega([0,1]^d, K, \delta)$. Namely,

$$Q_{\boldsymbol{\beta}} \coloneqq \Big\{\boldsymbol{x} = (x_1, x_2, \cdots, x_d) : x_i \in [\tfrac{\beta_i}{K}, \tfrac{\beta_i+1}{K} - \delta] \text{ for } i = 1, \cdots, d\Big\},$$

for $\boldsymbol{\beta} = (\beta_1, \beta_2, \cdots, \beta_d) \in \{0,1,\cdots,K-1\}^d$. Clearly, $\Omega([0,1]^d, K, \delta) = [0,1]^d \backslash (\cup_{\boldsymbol{\beta}\in\{0,1,\cdots,K-1\}^d} Q_{\boldsymbol{\beta}})$. See Figure 5 for illustrations.

Step 2: Construct ϕ_1.

Let g be a "step function" such that

- $g(\frac{k}{K}) = g(\frac{k+1}{K} - \delta) = k$ for $k = 0, 1, \cdots, K-1$ and $g(1) = K-1$.
- g is linear between any two adjacent points of

$$\{\tfrac{k}{K} : k = 0, 1, \cdots, K\} \cup \{\tfrac{k+1}{K} - \delta : k = 0, 1, \cdots, K-1\}.$$

Then, for any $\boldsymbol{x} = (x_1, \cdots, x_d) \in Q_{\boldsymbol{\beta}}$ and $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_d) \in \{0,1,\cdots,K-1\}^d$, we have

$$g(x_i) = \beta_i \quad \text{for } i = 1, 2, \cdots, d.$$

Also, such a function g can be easily realized by a one-hidden-layer ReLU network with width $2K$.

Let h be a function satisfying $h(j) = 4^j$ for $j = 1, 2, \cdots, K^d$. Such a function h can be easily realized by a one-hidden-layer ReLU network with width K^d. Then the desired function $\phi_1 : \mathbb{R}^d \to \mathbb{R}$ can be defined via

$$\phi_1(\boldsymbol{x}) \coloneqq h\Big(1 + \sum_{i=1}^{d} g(x_i) K^{i-1}\Big) = h \circ \varrho\Big(g(x_1), \cdots, g(x_d)\Big) \quad \text{for any } \boldsymbol{x} = (x_1, \cdots, x_d) \in \mathbb{R}^d,$$

where $\varrho : \mathbb{R}^d \to \mathbb{R}$ is a linear function defined by

$$\varrho(\boldsymbol{y}) = 1 + \sum_{i=1}^{d} y_i K^{i-1} \quad \text{for any } \boldsymbol{y} = (y_1, \cdots, y_d) \in \mathbb{R}^d.$$

Clearly, ϱ is a bijection (one-to-one map) from $\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d$ to $\varrho(\boldsymbol{\beta}) = 1 + \sum_{i=1}^{d} \beta_i K^{i-1} \in \{1, 2, \cdots, K^d\}$.

Then, for any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$, we have

$$\phi_1(\boldsymbol{x}) = h \circ \varrho\Big(g(x_1), \cdots, g(x_d)\Big) = h \circ \varrho\big(\beta_1, \cdots, \beta_d\big) = 4^{\varrho(\boldsymbol{\beta})} = 4^{1+\sum_{i=1}^d \beta_i K^{i-1}}. \tag{3.2}$$

Apparently, ϕ_1 is independent of f and it can be realized by a ReLU network with

$$\leq d(2K \times 3 + 1) \ + \ (K+1) \ + \ 3K^d + 1 \leq 11K^d + 2 = 11 \times 2^{dn} + 2 \leq 2^{dn+4}$$

parameters.

Step 3: Construct $\mathcal{L}$.

For each $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$, it follows from $f(\boldsymbol{x}_{\boldsymbol{\beta}}) \in [0, 1]$ that there exist $\xi_{\boldsymbol{\beta},1}, \cdots, \xi_{\boldsymbol{\beta},n}$ such that

$$|f(\boldsymbol{x}_{\boldsymbol{\beta}}) - \mathrm{bin}0.\xi_{\boldsymbol{\beta},1}\cdots\xi_{\boldsymbol{\beta},n}| \leq 2^{-n}. \tag{3.3}$$

Given any $j \in \{1, 2, \cdots, K^d\}$, there exists a unique $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$ such that $j = 1 + \sum_{i=1}^d \beta_i K^{i-1} = \varrho(\boldsymbol{\beta})$. Thus, for any $\ell \in \{1, 2, \cdots, n\}$, we can define

$$\theta_{j,\ell} \coloneqq \xi_{\boldsymbol{\beta},\ell}, \quad \text{for } j = \varrho(\boldsymbol{\beta}) \text{ and } \boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d. \tag{3.4}$$

Then the desired linear map $\mathcal{L}$ can be defined via

$$\mathcal{L}(t) \coloneqq (a_1 t, a_2 t, \cdots, a_n t) \quad \text{for any } t \in \mathbb{R},$$

where $a_\ell = \sum_{j=1}^{K^d} \theta_{j,\ell} 4^{-j}$ for $\ell = 1, 2, \cdots, n$. Clearly, for $\ell = 1, 2, \cdots, n$, we have

$$a_\ell = \sum_{j=1}^{K^d} \theta_{j,\ell} 4^{-j} \in [0, \tfrac{1}{3})$$

and

$$2^m a_\ell = 2^{2^{dn+1}} a_\ell = 4^{2^{dn}} a_\ell = 4^{K^d} a_\ell \in \mathbb{N}, \quad \text{where } m = 2^{dn+1}.$$

Step 4: Construct ϕ_2.

Fix $\ell \in \{1, 2, \cdots, n\}$, by Proposition 3.1 (set $J = K^d$ and $\theta_j = \theta_{j,\ell}$ therein), there exists a function $\phi_{2,\ell}$ implemented by a ReLU network with width 2 and depth $2K^d + 2$ such that $\phi_{2,\ell}(t) \in [0, 1]$ for any $t \in \mathbb{R}$ and

$$\phi_{2,\ell}(4^j a_\ell) = \theta_{j,\ell} \quad \text{for } j = 1, 2, \cdots, K^d, \quad \text{where } a_\ell = \sum_{j=1}^{K^d} \theta_{j,\ell} 4^{-j}. \tag{3.5}$$

Note that $\phi_{2,\ell}$ is independent of $\theta_{j,\ell}$ for $j = 1, 2, \cdots, K^d$, so it is also independent of f. Then the desired function $\phi_2 : \mathbb{R}^n \to \mathbb{R}$ can be defined via

$$\phi_2(\boldsymbol{y}) \coloneqq \sum_{\ell=1}^{n} 2^{-\ell} \phi_{2,\ell}(y_\ell) \quad \text{for any } \boldsymbol{y} = (y_1, \cdots, y_n) \in \mathbb{R}^n.$$

Then $\phi_2(\boldsymbol{y}) \in [0, 1]$ for any $\boldsymbol{y} \in \mathbb{R}^n$ and ϕ_2 can be implemented by a ReLU network, independent of f, with

$$\leq \Big(6(2K^n + 2) + (2 + 1)\Big)n + n + 1 = 6n(2^{nd+1} + 2) + 4n + 1 \leq 2^{nd+5} n$$

parameters.

Step 5: Estimate the approximation error.

It remains to estimate the approximation error. By Equations (3.2), (3.4), and (3.5), for $\ell \in \{1,2,\cdots,n\}$, $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$, $j = \varrho(\boldsymbol{\beta}) = 1 + \sum_{i=1}^{d} \beta_i K^{i-1} \in \{1,2,\cdots,K^d\}$, and $\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d$, we have

$$\begin{aligned}\phi_2 \circ \mathcal{L} \circ \phi_1(\boldsymbol{x}) &= \phi_2 \circ \mathcal{L} \circ h \circ \varrho\Big(g(x_1),\cdots,g(x_d)\Big) = \phi_2 \circ \mathcal{L} \circ h \circ \varrho(\boldsymbol{\beta}) \\ &= \phi_2 \circ \mathcal{L}(4^{\varrho(\boldsymbol{\beta})}) = \phi_2 \circ \mathcal{L}(4^j) = \phi_2(4^j a_1, 4^j a_2, \cdots, 4^j a_n) \\ &= \sum_{\ell=1}^{n} 2^{-\ell} \phi_{2,\ell}(4^j a_\ell) = \sum_{\ell=1}^{n} 2^{-\ell} \theta_{j,\ell} = \sum_{\ell=1}^{n} 2^{-\ell} \xi_{\boldsymbol{\beta},\ell}.\end{aligned}$$

Then by Equation (3.3), for any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and $\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d$, we get

$$\begin{aligned}|f(\boldsymbol{x}) - \phi_2 \circ \mathcal{L} \circ \phi_1(\boldsymbol{x})| &= |f(\boldsymbol{x}) - f(\boldsymbol{x}_{\boldsymbol{\beta}})| + |f(\boldsymbol{x}_{\boldsymbol{\beta}}) - \phi_2 \circ \mathcal{L} \circ \phi_1(\boldsymbol{x})| \\ &\le \omega_f\big(\tfrac{\sqrt{d}}{K}\big) + |f(\boldsymbol{x}_{\boldsymbol{\beta}}) - \sum_{\ell=1}^{n} 2^{-\ell} \xi_{\boldsymbol{\beta},\ell}| \\ &\le \omega_f\big(\tfrac{\sqrt{d}}{K}\big) + |f(\boldsymbol{x}_{\boldsymbol{\beta}}) - \mathrm{bin}0.\xi_{\boldsymbol{\beta},1}\cdots\xi_{\boldsymbol{\beta},n}| \le \omega_f\big(\sqrt{d}\,2^{-n}\big) + 2^{-n}.\end{aligned}$$

That is,

$$|f(\boldsymbol{x}) - \phi_2 \circ \mathcal{L} \circ \phi_1(\boldsymbol{x})| \le \omega_f\big(\sqrt{d}\,2^{-n}\big) + 2^{-n} \quad \text{for any } \boldsymbol{x} \in [0,1]^d \backslash \Omega([0,1]^d, K, \delta).$$

Moreover, the fact $\phi_2(\boldsymbol{y}) \in [0,1]$ for any $\boldsymbol{y} \in \mathbb{R}^n$ implies $\|\phi_2 \circ \mathcal{L} \circ \phi_1\|_{L^\infty(\mathbb{R}^d)} \le 1$. So we finish the proof. □

3.3 Proof of Proposition 3.1

Before proving Proposition 3.1, let us introduce a notation to simplify the proof. We use $\mathcal{T}_m$ for $m \in \mathbb{N}^+$ to denote a "sawtooth" function satisfying the following conditions.

- $\mathcal{T}_m : [0, 2m] \to [0,1]$ is linear between any two adjacent integers of $\{0,1,\cdots,2m\}$.
- $\mathcal{T}_m(2j) = 0$ for $j = 0,1,\cdots,m$ and $\mathcal{T}_m(2j+1) = 1$ for $j = 0,1,\cdots,m-1$.

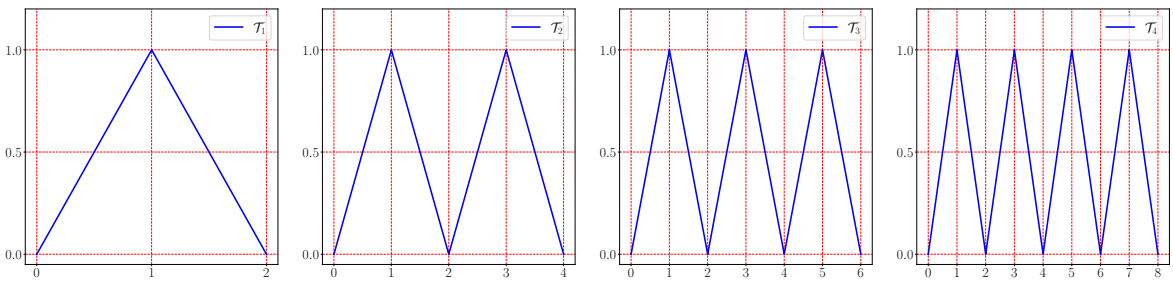


Figure 6: Illustrations of teeth functions $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$ and $\mathcal{T}_4$.

To simplify the proof of Proposition 3.1, we first introduce a lemma based on the "sawtooth" function.

Lemma 3.2. *Given any $J \in \mathbb{N}^+$ and $\theta_j \in \{0,1\}$ for $j = 1, 2, \cdots, J$, set $a = \sum_{j=1}^{J} \theta_j 4^{-j}$. Then*

$$\mathcal{T}_{4^J}(4^j a) \in [0, 1/3] \quad \text{if } \theta_j = 0 \quad \text{and} \quad \mathcal{T}_{4^J}(4^j a) \in [2/3, 1] \quad \text{if } \theta_j = 1, \tag{3.6}$$

where $\mathcal{T}_{4^J} : [0, 2\times 4^J] \to [0,1]$ is a "sawtooth" function with 4^J "teeth" defined just above.

Proof. Fix $j \in \{1, 2, \cdots, J\}$, we have

$$4^j a = 4^j \sum_{i=1}^{J} \theta_i 4^{-i} = \underbrace{\sum_{i=1}^{j-1} \theta_i 4^{j-i}}_{= 4k \text{ for some } k \in \mathbb{N} \text{ with } k \le \frac{4^{J-1}}{3}} + \overbrace{\theta_j}^{0 \text{ or } 1} + \underbrace{\sum_{i=j+1}^{J} \theta_i 4^{j-i}}_{\in [0, \frac{1}{3})}. \tag{3.7}$$

Clearly,

$$\sum_{i=1}^{j-1} \theta_i 4^{j-i} \in \{4k : k \in \mathbb{N},\, k \le 4^{J-1}/3\} \quad \text{and} \quad 0 \le \sum_{i=j+1}^{J} \theta_i 4^{j-i} \le \sum_{i=j+1}^{J} 4^{j-i} \le 1/3.$$

If $\theta_j = 0$, then Equation (3.7) implies

$$4^j a \in [4k, 4k + 1/3] \quad \text{for some } k \in \mathbb{N} \text{ with } k \le 4^{J-1}/3 \le 4^J - 1,$$

which implies $\mathcal{T}_{4^J}(4^j a) \in [0, 1/3]$.

Similarly, if $\theta_j = 1$, then Equation (3.7) implies

$$4^j a \in [4k + 1, 4k + 1 + 1/3] \quad \text{for some } k \in \mathbb{N} \text{ with } k \le 4^{J-1}/3 \le 4^J - 1,$$

which implies $\mathcal{T}_{4^J}(4^j a) \in [2/3, 1]$. So we finish the proof. □

It is worth mentioning that the "sawtooth" function $\mathcal{T}_m$ can be replaced by other functions that also have a key property "the function values near even integers are much larger than the ones near odd integers".

With Lemma 3.2 in hand, we are ready to prove Proposition 3.1.

Proof of Proposition 3.1. By Lemma 3.2, we have

$$\mathcal{T}_{4^J}(4^j a) \in [0, 1/3] \quad \text{if } \theta_j = 0 \quad \text{and} \quad \mathcal{T}_{4^J}(4^j a) \in [2/3, 1] \quad \text{if } \theta_j = 1.$$

Define $g(x) \coloneqq 3\sigma(x - 1/3) - 3\sigma(x - 2/3)$ for any $x \in \mathbb{R}$, where σ is ReLU, i.e., $\sigma(x) = \max\{0, x\}$. See an illustration of g in Figure 7. Clearly,

$$g(x) = 0 \text{ if } x \le 1/3 \quad \textit{and} \quad g(x) = 1 \text{ if } x \ge 2/3.$$

Hence, $\phi \coloneqq g \circ \mathcal{T}_{4^J}$ is the desired function. Obviously, $\phi(x) \in [0,1]$ for any $x \in \mathbb{R}$. To verify Equation (3.1), we fix $j \in \{1, 2, \cdots, J\}$. If $\theta_j = 0$, then $\mathcal{T}_{4^J}(4^j a) \in [0, 1/3]$, implying $\phi(4^j a) = g \circ \mathcal{T}_{4^J}(4^j a) = 0 = \theta_j$. If $\theta_j = 1$, then $\mathcal{T}_{4^J}(4^j a) \in [2/3, 1]$, implying $\phi(4^j a) = g \circ \mathcal{T}_{4^J}(4^j a) = 1 = \theta_j$.

It remains to show $\phi = g \circ \mathcal{T}_{4^J}$ can be realized by a ReLU network with the expected width and depth. Clearly, $\mathcal{T}_{4^J}$ is a continuous piecewise linear function, which means

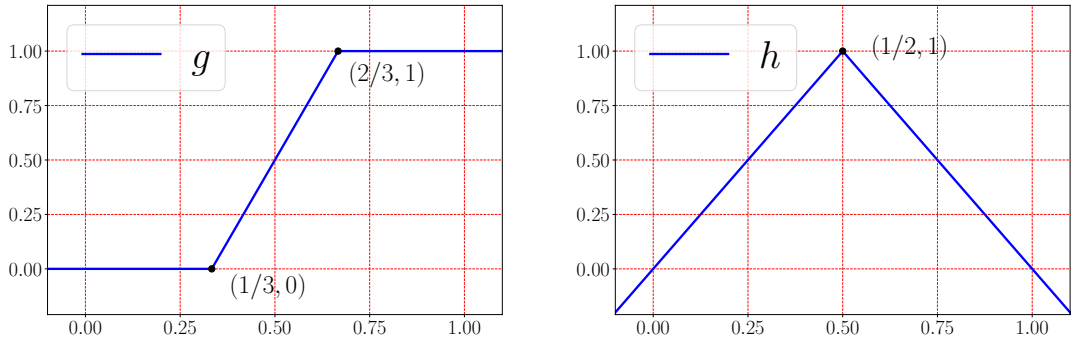


Figure 7: Illustrations of g and h on $[0,1]$.

it can be implemented by a one-hidden-layer ReLU network. To make our construction more efficient, we introduce another method to implement ϕ.

Define $h(x) \coloneqq 1 - 2\sigma(x - 1/2) - 2\sigma(1/2 - x)$ for any $x \in [0,1]$. See an illustration of h in Figure 7. Then, it is easy to verify that $\mathcal{T}_1(2x) = h(x)$ for any $x \in [0,1]$ and h can be implemented by a one-hidden-layer ReLU network with width 2. For any $J \in \mathbb{N}^+$, it is easy to verify that

$$\mathcal{T}_{4^J}(2^{2J+1}x) = \underbrace{h \circ h \circ \cdots \circ h}_{2J+1 \text{ times}}(x), \quad \text{for any } x \in [0,1].$$

So, $\mathcal{T}_{4^J}$ limited on $[0, 2 \times 4^J] = [0, 2^{2J+1}]$ can be realized by a ReLU network with width 2 and depth $2J+1$. It follows that $\phi = g \circ \mathcal{T}_{4^J}$ can be implemented by a ReLU network with width 2 and depth $2J+2$, which means we finish the proof. □

4 Conclusion

This paper studies the approximation error of ReLU networks in terms of the number of intrinsic parameters. Theorem 1.1 implies that, for any Hölder continuous function on $[0,1]^d$, a ReLU network with $n+2$ intrinsic parameters can approximate f with an error $5\lambda d^{\alpha/2} 2^{-\alpha n}$ measured in the L^p-norm for $p \in [1, \infty)$, where $\alpha \in (0,1]$ and $\lambda > 0$ are the Hölder order and constant, respectively. Moreover, such a result can be generalized from the L^p-norm to the L^∞-norm with a price of adding $\mathcal{O}(n)$ intrinsic parameters, as shown in Theorem 1.2. Finally, we show in Theorem 1.3 that three intrinsic parameters are enough to achieve an arbitrary error in the case of Hölder continuous functions, though this result requires high precision to encode these three parameters on computers. Remark that this paper only focuses on the approximation error characterized by the number of intrinsic parameters, the study of the optimization error and generalization error will be left as future work.

Acknowledgments

Z. Shen is supported by Tan Chin Tuan Centennial Professorship. H. Yang was partially supported by the US National Science Foundation under award DMS-1945029.

S. Zhang is supported by a Postdoctoral Fellowship under NUS ENDOWMENT FUND (EXP WBS) (01 651).

References

[1] M. Anthony and P. L. Bartlett, *Neural Network Learning: Theoretical Foundations*, Cambridge University Press, New York, NY, USA, 1st ed., 2009.

[2] C. Bao, Q. Li, Z. Shen, C. Tai, L. Wu, and X. Xiang, *Approximation analysis of convolutional neural networks*, Semantic Scholar e-Preprint, (2019), p. Corpus ID: 204762668.

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

[4] A. R. Barron and J. M. Klusowski, *Approximation and estimation for high-dimensional deep learning networks*, 2018.

[5] P. Bartlett, V. Maiorov, and R. Meir, *Almost linear VC-dimension bounds for piecewise polynomial networks*, Neural Computation, 10 (1998), pp. 217–3.

[6] J. Baxter, *Theoretical models of learning to learn*, in Learning to Learn, Springer, 1998.

[7] 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 (2014), pp. 1553–1565.

[8] H. Bölcskei, P. Grohs, G. Kutyniok, and P. Petersen, *Optimal approximation with sparsely connected deep neural networks*, SIAM Journal on Mathematics of Data Science, 1 (2019), pp. 8–45.

[9] S. Bozinovski, *Reminder of the first paper on transfer learning in neural networks, 1976*, Informatica, 44 (2020), pp. 291–302.

[10] S. Bozinovski and A. Fulgosi, *The influence of pattern similarity and transfer learning upon the training of a base perceptron B2. (original in croatian)*, in Proceedings of the Symposium Informatica 3-121-5, Bled, Slovenia, 1976.

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

[12] M. Chen, H. Jiang, W. Liao, and T. Zhao, *Efficient approximation of deep relu networks for functions on low dimensional manifolds*, in Advances in Neural Information Processing Systems, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds., vol. 32, Curran Associates, Inc., 2019.

[13] C. K. Chui, S.-B. Lin, and D.-X. Zhou, *Construction of neural networks for realization of localized deep learning*, Frontiers in Applied Mathematics and Statistics, 4 (2018), p. 14.

[14] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, MCSS, 2 (1989), pp. 303–314.

[15] W. E, C. Ma, and L. Wu, *A priori estimates of the population risk for two-layer neural networks*, Communications in Mathematical Sciences, 17 (2019), pp. 1407–1425.

[16] R. Gribonval, G. Kutyniok, M. Nielsen, and F. Voigtlaender, *Approximation spaces of deep neural networks*, arXiv e-prints, (2019), p. arXiv:1905.01208.

[17] I. Gühring, G. Kutyniok, and P. Petersen, *Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms*, arXiv e-prints, (2019), p. arXiv:1902.07896.

[18] N. Harvey, C. Liaw, and A. Mehrabian, *Nearly-tight VC-dimension bounds for piecewise linear neural networks*, in Proceedings of the 2017 Conference on Learning Theory, S. Kale and O. Shamir, eds., vol. 65 of Proceedings of Machine Learning Research, Amsterdam, Netherlands, 07–10 Jul 2017, PMLR, pp. 1064–1068.

[19] K. Hornik, *Approximation capabilities of multilayer feedforward networks*, Neural Networks, 4 (1991), pp. 251–257.

[20] K. Hornik, M. Stinchcombe, and H. White, *Multilayer feedforward networks are universal approximators*, Neural Networks, 2 (1989), pp. 359–366.

[21] Y. Jiao, Y. Lai, X. Lu, F. Wang, J. Zhijian Yang, and Y. Yang, *Deep neural networks with ReLU-Sine-Exponential activations break curse of dimensionality on hölder class*, arXiv e-prints, (2021), p. arXiv:2103.00542.

[22] T. G. Karimpanal and R. Bouffanais, *Self-organizing maps for storage and transfer of knowledge in reinforcement learning*, Adaptive Behavior, 27 (2019), pp. 111–126.

[23] M. J. Kearns and R. E. Schapire, *Efficient distribution-free learning of probabilistic concepts*, J. Comput. Syst. Sci., 48 (1994), pp. 464–497.

[24] Q. Li, T. Lin, and Z. Shen, *Deep learning via dynamical systems: An approximation perspective*, arXiv e-prints, (2019), p. arXiv:1912.10382.

[25] J. Lu, Z. Shen, H. Yang, and S. Zhang, *Deep network approximation for smooth functions*, SIAM Journal on Mathematical Analysis, 53 (2021), pp. 5465–5506.

[26] L. Mihalkova, T. Huynh, and R. J. Mooney, *Mapping and revising markov logic networks for transfer learning*, in Proceedings of the 22nd Conference on Artificial Intelligence (AAAI-07), vol. 1, Vancouver, Canada, 2007, pp. 608–614.

[27] T. Mikolov, A. Joulin, and M. Baroni, *A roadmap towards machine intelligence*, in Computational Linguistics and Intelligent Text Processing, A. Gelbukh, ed., Cham, 2018, Springer International Publishing, pp. 29–61.

[28] H. Montanelli and H. Yang, *Error bounds for deep ReLU networks using the Kolmogorov-Arnold superposition theorem*, Neural Networks, 129 (2020), pp. 1–6.

[29] H. Montanelli, H. Yang, and Q. Du, *Deep ReLU networks overcome the curse of dimensionality for bandlimited functions*, Journal of Computational Mathematics, (2020).

[30] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, *On the number of linear regions of deep neural networks*, in Advances in Neural Information Processing Systems 27, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds., Curran Associates, Inc., 2014, pp. 2924–2932.

[31] R. Nakada and M. Imaizumi, *Adaptive approximation and generalization of deep neural network with intrinsic dimensionality*, Journal of Machine Learning Research, 21 (2020), pp. 1–38.

[32] A. Niculescu-Mizil and R. Caruana, *Inductive transfer for bayesian network structure learning*, in Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, M. Meila and X. Shen, eds., vol. 2 of Proceedings of Machine Learning Research, San Juan, Puerto Rico, 21–24 Mar 2007, PMLR, pp. 339–346.

[33] P. Petersen and F. Voigtlaender, *Optimal approximation of piecewise smooth functions using deep ReLU neural networks*, Neural Networks, 108 (2018), pp. 296–330.

[34] L. Y. Pratt, *Discriminability-based transfer between neural networks*, in Advances in Neural Information Processing Systems, S. Hanson, J. Cowan, and C. Giles, eds., vol. 5, Morgan-Kaufmann, 1993.

[35] A. A. Rusu, M. Vecerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, *Sim-to-real robot learning from pixels with progressive nets*, in 1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings, vol. 78 of Proceedings of Machine Learning Research, PMLR, 2017, pp. 262–270.

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

[37] Z. Shen, H. Yang, and S. Zhang, *Nonlinear approximation via compositions*, Neural Networks, 119 (2019), pp. 74–84.

[38] ——, *Deep network approximation characterized by number of neurons*, Communications in Computational Physics, 28 (2020), pp. 1768–1811.

[39] ——, *Deep network approximation: Achieving arbitrary accuracy with fixed number of neurons*, arXiv e-prints, (2021), p. arXiv:2107.02397.

[40] ——, *Deep network with approximation error being reciprocal of width to power of square root of depth*, Neural Computation, 33 (2021), pp. 1005–1036.

[41] ——, *Neural network approximation: Three hidden layers are enough*, Neural Networks, 141 (2021), pp. 160–173.

[42] ——, *Optimal approximation rate of ReLU networks in terms of width and depth*, Journal de Mathématiques Pures et Appliquées, (to appear).

[43] T. 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] D. Yarotsky, *Error bounds for approximations with deep ReLU networks*, Neural Networks, 94 (2017), pp. 103–114.

[45] ——, *Optimal approximation of continuous functions by very deep ReLU networks*, in Proceedings of the 31st Conference On Learning Theory, S. Bubeck, V. Perchet, and P. Rigollet, eds., vol. 75 of Proceedings of Machine Learning Research, PMLR, 06–09 Jul 2018, pp. 639–649.

[46] D. Yarotsky, *Elementary superexpressive activations*, in Proceedings of the 38th International Conference on Machine Learning, M. Meila and T. Zhang, eds., vol. 139 of Proceedings of Machine Learning Research, PMLR, 18–24 Jul 2021, pp. 11932–11940.

[47] D. Yarotsky and A. Zhevnerchuk, *The phase diagram of approximation rates for deep neural networks*, in Advances in Neural Information Processing Systems, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds., vol. 33, Curran Associates, Inc., 2020, pp. 13005–13015.

[48] S. Zhang, *Deep neural network approximation via function compositions*, PhD Thesis, National University of Singapore, (2020). URL: https://scholarbank.nus.edu.sg/handle/10635/186064.

[49] D.-X. Zhou, *Universality of deep convolutional neural networks*, Applied and Computational Harmonic Analysis, 48 (2020), pp. 787–794.