Journal of Computer Science and Cybernetics, V.37, N.3 (2021), 291–320
DOI 10.15625/1813-9663/37/3/15902

COMPUTATION COMPLEXITY OF DEEP RELU NEURAL NETWORKS IN HIGH-DIMENSIONAL APPROXIMATION

DINH DŨNG[1], VAN KIEN NGUYEN[2,*], MAI XUAN THAO[3]

[1]*Information Technology Institute, Vietnam National University, Hanoi*
144 Xuan Thuy, Cau Giay, Hanoi, Vietnam

[2]*Faculty of Basic Sciences, University of Transport and Communications,*
No.3 Cau Giay Street, Lang Thuong Ward, Dong Da District, Hanoi, Vietnam

[3]*Department of Natural Sciences, Hong Duc University,*
565 Quang Trung, Thanh Hoa, Vietnam

Abstract. The purpose of the present paper is to study the computation complexity of deep ReLU neural networks for approximation of functions in Hölder-Nikol'skii spaces of mixed smoothness $H^\alpha_\infty(\mathbb{I}^d)$ on the unit cube $\mathbb{I}^d := [0,1]^d$. In this context, for any function $f \in H^\alpha_\infty(\mathbb{I}^d)$, we explicitly construct nonadaptive and adaptive deep ReLU neural networks having an output that approximates f with a prescribed accuracy ε, and prove dimension-dependent bounds for the computation complexity of the approximation, characterized by the size and depth of this deep ReLU neural network, explicitly in d and ε. Our results show the advantage of the adaptive method of approximation by deep ReLU neural networks over nonadaptive one.

Keywords. High-dimensional approximation; Deep ReLU neural networks; Nonadaptive approximation; Adaptive approximation; Hölder-Nikol'skii space of mixed smoothness.

1. INTRODUCTION

Neural networks and their applications have been of great interest for almost 80 years, started from the foundational works of McCulloch and Pitts [25], Hebb [19] and of Rosenblatt [31]. Recently, deep neural networks have been successfully applied to a wide variety of Machine Learning problems [22, 23, 38]. A typical deep neural network model is based on a hierarchy of composition of linear functions and a given nonlinear activation function. Among all the activation functions, Rectified Linear Unit (ReLU) activation $\max(0, x)$ is the most commonly used in deep neural networks. Compared to traditional nonlinear activation functions (tanh and sigmoid), the ReLU function provides the same benefits but accelerates the training speed of deep neural networks since its gradient is very simple (either 0 or 1

Dedicated to Professor Phan Dinh Dieu on the occasion of his 85th birth anniversary.

*Corresponding author.
E-mail addresses: dinhzung@gmail.com (D. Dũng); kiennv@utc.edu.vn (V.K. Nguyen); maixuanthao@hdu.edu.vn (M.X. Thao)

depending on the sign of input). Deep ReLU neural networks do not need to take additional time for computing error terms during training phase. Deep ReLU neural networks has found many applications in different fields such as computer vision [10, 18] and speech recognition [24, 35].

It is observed that there is a closed relation between approximating by deep ReLU networks and B-spline interpolation and quasi-interpolation representation, in particular by piecewise linear functions. There have been numerous papers addressing the expressive power of deep ReLU neural networks in approximating different sets of functions such as analytic functions [8, 26], differentiable functions [29, 39], oscillatory functions [14], functions in isotropic Sobolev or Besov spaces [1, 7, 11, 16, 40], functions with dominating mixed smoothness [27, 34] or in approximating solutions to partial differential equations [9, 28, 33], to mention just a few. The main advantage of deep ReLU neural networks in approximating functions is that they can output compositions of functions cheaply and consequently improve the convergence rate of approximation error, see [7, 8, 39]. We refer the reader to recent surveys [14, 30] for results in deep ReLU neural network approximation theory.

Function spaces having mixed smoothness appear naturally in many models of real world problem in mathematical physics, finance and other fields. For instance, in a recent work on regularity properties of solutions of the electronic Schrödinger equation, Yserentant [41] has shown that the eigenfunctions of the electronic Schrödinger operator have a certain mixed smoothness. Triebel [37, Chapter 6] has indicated a relation between Faber bases and sampling recovery in the context of spaces with mixed smoothness and solutions of Navier-Stokes equations. In particular, when initial data belongs to spaces with mixed smoothness, Navier-Stokes equations admit a unique solution. In mathematical finance, many problems are expressed as the expectation of some payoff function depending on quantities, such as stock prices, which are solutions of stochastic equations governed by Brownian motions. The payoff function normally has kinks and jumps and belongs to a very high dimensional space. To approximate the expected value one can apply preliminary integration method with respect to a single well chosen variable to obtain a function of $d-1$ variables which belongs to appropriate mixed Sobolev spaces in which Quasi-Monte Carlo can be applied efficiently, see [13] and references therein. For a survey on various aspects of high-dimensional approximation of functions having a mixed smoothness we refer the reader to the book [5].

In approximation theory, modern problems driven by a lot of applications in Information Technology, Mathematical Finance, Chemistry, Quantum Mechanics, Meteorology, and, in particular, in Uncertainty Quantification and Deep Machine Learning are being formulated in very high dimensions. Many times, numerical methods for such problems may demand computational cost increasing exponentially in dimension when the accuracy increases and as a consequence the method becomes intractable when the dimension of input data is large. This phenomenon is called "curse of dimensionality". Hence, the problem of estimating dimension-dependent error in high-dimensional approximation problems arises naturally. Hyperbolic crosses and sparse grids promise to rid the "curse of dimensionality" in some problems when high-dimensional data belongs to certain classes of functions having mixed smoothness. Approximation methods and sampling algorithms for functions having mixed smoothness constructed on hyperbolic crosses and sparse grids give a surprising effect since hyperbolic crosses and sparse grids have the number of elements much less than those of standard domains and grids but give the same approximation error. This essentially reduces

the computational cost, and therefore makes the problem tractable.

In the recent paper [3], we have studied the approximation by deep ReLU neural networks, of functions from the Hölder-Zygmund space of mixed smoothness defined on the unit cube $\mathbb{I}^d$ when the dimension d may be very large. The approximation error is measured in the norm of the isotropic Sobolev space. For any function f from this space, we explicitly constructed a deep ReLU neural network having an output that approximates f with a prescribed accuracy ε, and proved tight dimension-dependent estimates of the computation complexity of this approximation, characterized as the size and depth of this deep ReLU neural network, explicitly in d and ε.

The purpose of the present paper is to study the computation complexity of deep ReLU neural networks for approximation of Hölder-Nikol'skii functions having mixed smoothness on the unit cube $\mathbb{I}^d := [0,1]^d$. Let us introduce the space $H^\alpha_\infty(\mathbb{I}^d)$ of our interest. For univariate functions f on $\mathbb{I} := [0,1]$, the difference operator Δ_h is defined by

$$\Delta_h f(x) := f(x+h) - f(x),$$

for all x and $h \geqslant 0$ such that $x, x+h \in \mathbb{I}$. If u is a subset of $\{1,\dots,d\}$, for multivariate functions f on $\mathbb{I}^d$ the mixed difference operator $\Delta_{\boldsymbol{h},u}$ is defined by

$$\Delta_{\boldsymbol{h},u} := \prod_{i\in u} \Delta_{h_i}, \quad \Delta_{\boldsymbol{h},\varnothing} = \mathrm{Id},$$

for all $\boldsymbol{x} = (x_1,\dots,x_d)$ and $\boldsymbol{h} = (h_1,\dots,h_d)$ such that $\boldsymbol{x}, \boldsymbol{x}+\boldsymbol{h} \in \mathbb{I}^d$. Here the univariate operator Δ_{h_i} is applied to the univariate function f by considering f as a function of variable x_i with the other variables held fixed. If $0 < \alpha \leqslant 1$, we introduce the semi-norm $|f|_{H^\alpha_\infty(u)}$ for functions $f \in C(\mathbb{I}^d)$ by

$$|f|_{H^\alpha_\infty(u)} := \sup_{\boldsymbol{h}>0} \prod_{i\in u} h_i^{-\alpha} \|\Delta_{\boldsymbol{h},u}(f)\|_{C(\mathbb{I}^d(\boldsymbol{h},u))}$$

(in particular, $|f|_{H^\alpha_\infty(\varnothing)} = \|f\|_{C(\mathbb{I}^d)}$), where $\mathbb{I}^d(\boldsymbol{h},u) := \{\boldsymbol{x} \in \mathbb{I}^d : x_i + h_i \in \mathbb{I},\, i \in u\}$. The Hölder-Nikol'skii space $H^\alpha_\infty(\mathbb{I}^d)$ of mixed smoothness α then is defined as the set of functions $f \in C(\mathbb{I}^d)$ for which the norm

$$\|f\|_{H^\alpha_\infty(\mathbb{I}^d)} := \max_{u\subset\{1,\dots,d\}} |f|_{H^\alpha_\infty(u)}$$

is finite. From the definition we have that $H^\alpha_\infty(\mathbb{I}^d) \subset C(\mathbb{I}^d)$. Denote by $\mathring{C}(\mathbb{I}^d)$ the set of all functions $f \in C(\mathbb{I}^d)$ vanishing on the boundary $\partial\mathbb{I}^d$ of $\mathbb{I}^d$, i.e., the set of all functions $f \in C(\mathbb{I}^d)$ such that $f(x) = 0$ if $x_j = 0$ or $x_j = 1$ for some index $j \in \{1,\dots,d\}$. Denote by $\mathring{U}^{\alpha,d}_\infty$ the set of all functions f in the intersection $\mathring{H}^\alpha_\infty(\mathbb{I}^d) := H^\alpha_\infty(\mathbb{I}^d) \cap \mathring{C}(\mathbb{I}^d)$ such that $\|f\|_{H^\alpha_\infty(\mathbb{I}^d)} \leqslant 1$.

As a continuation of [3], the present paper investigates nonadaptive and adaptive high-dimensional approximation by deep ReLU neural networks for functions from the classes $\mathring{U}^{\alpha,d}_\infty$. The approximation error is measured in the norm of $L_\infty(\mathbb{I}^d)$. In this context, we pay attention on the computation complexity of the deep ReLU networks, characterized by the size and depth of this deep ReLU neural network, explicitly in d and tolerance ε. A key tool for explicit construction of approximation methods by deep ReLU networks for functions in $H^\alpha_\infty(\mathbb{I}^d)$ is truncations of tensorized Faber series.

The main contribution of the present paper is as follows.

Based on the decomposition of continuous functions by tensorized Faber series, for any $f \in \mathring{U}^{\alpha,d}_\infty$ we explicitly construct a deep ReLU neural network $\Phi_\varepsilon(f)$ having the output that

approximates f in the $L_\infty(\mathbb{I}^d)$-norm with a prescribed accuracy ε and having computation complexity expressing the dimension-dependent size

$$W(\Phi_\varepsilon(f)) \leqslant C_\alpha d \Big(\frac{K_1^d}{(d-1)!}\Big)^{\frac{1}{\alpha}+1} \varepsilon^{-\frac{1}{\alpha}} \log(2\varepsilon^{-1})^{(d-1)(\frac{1}{\alpha}+1)+1}, \tag{1}$$

where $K_1 = B^{1/(\alpha+1)} 4\alpha^{-1}$ with $B = (2^\alpha - 1)^{-1}$. The idea in proving the above result is to use truncation of Faber series $R_n(f)$ as an intermediate approximation. Precisely, we first approximate function $f \in \mathring{U}_\infty^{\alpha,d}$ by $R_n(f)$ and then approximate $R_n(f)$ by a deep ReLU network.

The advantage of this method is that the deep ReLU neural networks are easily constructed and they have the same architecture for all functions in $\mathring{U}_\infty^{\alpha,d}$, i.e., it is nonadaptive. However, since this method uses $R_n(f)$ as an intermediate approximation, a disadvantage of it is that the computation complexity of deep ReLU networks is not better than that when approximating functions $f \in \mathring{U}_\infty^{\alpha,d}$ by the linear method $R_n(f)$.

To overcome this disadvantage we develop a technique used in [40] and [7] for the univariate case. By this, we first represent the difference $f - R_n(f)$ in a special form and then approximate terms in this representation by deep ReLU networks. We emphasize that extension of technique in [40] and [7] to multivariate case and mixed smoothness is non-trivial task since one needs to construct a set of finite cardinality to approximate functions in $\mathring{U}_\infty^{\alpha,d}$. For any $f \in \mathring{U}_\infty^{\alpha,d}$ we explicitly construct a deep ReLU neural network $\Phi_\varepsilon(f)$ of adaptive architecture having the output that approximates f in the $L_\infty(\mathbb{I}^d)$-norm with a prescribed accuracy ε and having a size estimated by

$$W(\Phi_\varepsilon(f)) \leqslant C_\alpha d^2 \Big(\frac{K_2^d}{(d-1)!}\Big)^{\frac{2}{\alpha}+2} \varepsilon^{-\frac{1}{\alpha}} \big(\log(2\varepsilon^{-1}) \log\log(2\varepsilon^{-1})\big)^{(1+\frac{1}{\alpha})(d-1)}, \tag{2}$$

where $K_2 = 4(2^{\alpha+3}B)^{\frac{1}{2\alpha+2}} (\alpha^{-1}\log(2\alpha^{-1}))^{1/2}$. Comparing (1) and (2) we find the later estimation improves $\log(2\varepsilon^{-1})$. Notice that the terms in right-hand side of both (1) and (2) which depend on dimension d only decay as fast as super exponential in d.

The outline of this paper is as follows. In Section 2, we introduce necessary definitions and elementary facts on deep ReLU neural networks. Section 3 is devoted to recall a decomposition of continuous functions on the unit cube $\mathbb{I}^d$ by Faber system and approximation of functions $f \in \mathring{U}_\infty^{\alpha,d}$ by truncations of Faber series $R_n(f)$ as well as by sets of finite cardinality. In Section 4, we explicitly construct nonadaptive deep ReLU neural networks that approximate functions in $\mathring{U}_\infty^{\alpha,d}$ and prove upper estimates for size and the depth required. Section 5 presents an improvement for approximation by adaptive deep ReLU neural networks of the results obtained in Section 4. In Section 6, we give an application of our results in numerical approximation of solutions to elliptic partial differential equations. Conclusions are given in Section 7.

Notation. As usual, $\mathbb{N}$ is the natural numbers, $\mathbb{Z}$ the integers, $\mathbb{R}$ the real numbers and $\mathbb{N}_0 := \{s \in \mathbb{Z} : s \geqslant 0\}$; $\mathbb{N}_{-1} = \mathbb{N}_0 \cup \{-1\}$. The letter d is reserved for the underlying dimension of $\mathbb{R}^d$, $\mathbb{N}^d$, etc. If $x \in \mathbb{R}$, $\lfloor x \rfloor$ is defined to be the largest integer no larger than x. Vectorial quantities are denoted by boldface letters and x_i denotes the ith coordinate of $\boldsymbol{x} \in \mathbb{R}^d$, i.e., $\boldsymbol{x} := (x_1, \ldots, x_d)$. For $\boldsymbol{x} \in \mathbb{R}^d$, we denote $|\boldsymbol{x}|_p := (|x_1|^p + \ldots + |x_d|^p)^{1/p}$ if $0 < p < \infty$, $|\boldsymbol{x}|_\infty = \max\{|x_1|, \ldots, |x_d|\}$, $\operatorname{supp}(\boldsymbol{x}) = \{j : x_j \neq 0\}$ and $2^{\boldsymbol{x}} := (2^{x_1}, \ldots, 2^{x_d})$. If $\boldsymbol{k}, \boldsymbol{s} \in \mathbb{N}_0^d$, we denote $2^{-\boldsymbol{k}}\boldsymbol{s} := (2^{-k_1}s_1, \ldots, 2^{-k_d}s_d)$. We use the abbreviation: $L_\infty := L_\infty(\mathbb{I}^d)$

and $\|\cdot\|_\infty := \|\cdot\|_{L_\infty}$. Universal constants or constants depending on parameters α, d are denoted by C or $C_{\alpha,d}$, respectively. Values of constants C and $C_{\alpha,d}$ in general, are not specified except the case when they are precisely given, and may be different in various places. $|A|$ denotes the cardinality of the finite set $|A|$.

2. DEEP RELU NEURAL NETWORKS

In this section we introduce necessary definitions and elementary facts on deep ReLU neural networks. There is a wide variety of neural network architectures and each of them is adapted to specific tasks. We only consider feed-forward deep ReLU neural networks such that only connections between neighboring layers are allowed.

Definition 1. Let $d, L \in \mathbb{N}$, $L \geqslant 2$, $N_0 = d$, and $N_1, \dots, N_L \in \mathbb{N}$. Let $\boldsymbol{W}^\ell = (w^\ell_{i,j})$, $\ell = 1, \dots, L$, be $N_\ell \times N_{\ell-1}$ matrix, and $\boldsymbol{b}^\ell = (b^\ell_j) \in \mathbb{R}^{N_\ell}$.

- A neural network Φ with input dimension d and L layers is a sequence of matrix-vector tuples
$$\Phi = \big((\boldsymbol{W}^1, \boldsymbol{b}^1), \dots, (\boldsymbol{W}^L, \boldsymbol{b}^L)\big).$$
We will use the following terminology.
  - The number of layers $L(\Phi) = L$ is the depth of Φ;
  - $N_w(\Phi) = \max_{\ell=0,\dots,L}\{N_\ell\}$ is the width of Φ; $\boldsymbol{N}(\Phi) = (N_0, N_1, \dots, N_L)$ the dimension of Φ;
  - The real numbers $w^\ell_{i,j}$ and b^ℓ_j are edge and node weights of Φ, respectively;
  - The number of nonzero weights $w^\ell_{i,j}$ and b^ℓ_j is the size of Φ and denoted by $W(\Phi)$;
  - When $L(\Phi) \geqslant 3$, Φ is called a deep neural network, and otherwise, a shallow neural network.
- A neural network architecture $\mathbb{A}$ with input dimension d and L layers is a neural network
$$\mathbb{A} = \big((\boldsymbol{W}^1, \boldsymbol{b}^1), \dots, (\boldsymbol{W}^L, \boldsymbol{b}^L)\big),$$
where elements of $\boldsymbol{W}^\ell$ and $\boldsymbol{b}^\ell$, $\ell = 1, \dots, L$, are in $\{0, 1\}$.

The above defined networks are sometimes called standard networks to distinguish with networks allowing for connections of neurons in non-neighboring layers. A deep neural network can be visualized in a graph. The graph associated with a deep neural network Φ defined in Definition 1 consists of $L + 1$ layers which are numbered from 0 to L. The ℓth layer has N_ℓ nodes which are numbered from 1 to N_ℓ. If $w^\ell_{i,j} \neq 0$, then there is an edge connecting the node j in the layer $\ell - 1$ to the node i in the layer ℓ. In Figure 1 we illustrate a deep neural network with input dimension 3 and 5 layers.

Definition 2. Given $L \in \mathbb{N}$, $L \geqslant 2$, and a neural network architecture $\mathbb{A} = \big((\overline{\boldsymbol{W}}^1, \overline{\boldsymbol{b}}^1), \dots, (\overline{\boldsymbol{W}}^L, \overline{\boldsymbol{b}}^L)\big)$. We say that a neural network $\Phi = \big((\boldsymbol{W}^1, \boldsymbol{b}^1), \dots, (\boldsymbol{W}^L, \boldsymbol{b}^L)\big)$ has architecture $\mathbb{A}$ if

- $\boldsymbol{N}(\Phi) = \boldsymbol{N}(\mathbb{A})$;
- $\overline{w}^\ell_{i,j} = 0$ implies $w^\ell_{i,j} = 0$, $\overline{b}^\ell_i = 0$ implies $b^\ell_i = 0$ for all $i = 1, \dots, N_\ell$, $j = 1, \dots, N_{\ell-1}$, and $\ell = 1, \dots, L$. Here $\overline{w}^\ell_{i,j}$ are entries of $\overline{\boldsymbol{W}}^\ell$ and $\overline{b}^\ell_i$ are elements of $\overline{\boldsymbol{b}}^\ell$, $\ell = 1, \dots, L$.

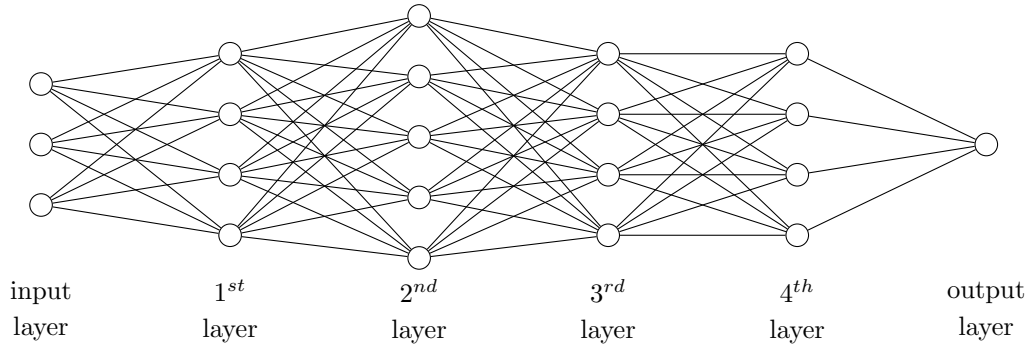

Figure 1: The graph associated to a deep neural network with input dimension 3 and 5 layers

For a given deep neural network $\Phi = \big((\boldsymbol{W}^1, \boldsymbol{b}^1), \dots, (\boldsymbol{W}^L, \boldsymbol{b}^L)\big)$, there exists a unique deep neural network architecture $\mathbb{A} = \big((\overline{\boldsymbol{W}}^1, \overline{\boldsymbol{b}}^1), \dots, (\overline{\boldsymbol{W}}^L, \overline{\boldsymbol{b}}^L)\big)$ such that

- $\boldsymbol{N}(\Phi) = \boldsymbol{N}(\mathbb{A})$;
- $\overline{w}^\ell_{i,j} = 0 \iff w^\ell_{i,j} = 0$, $\overline{b}^\ell_i = 0 \iff b^\ell_i = 0$ for all $i = 1, \dots, N_\ell$, $j = 1, \dots, N_{\ell-1}$, and $\ell = 1, \dots, L$.

We call this architecture $\mathbb{A}$ the minimal architecture of Φ (this definition is proper in the sense that any architecture of Φ is also an architecture of $\mathbb{A}$).

A deep neural network is associated with an activation function which calculates output at each node. The choice of activation function depends on the problem under consideration. In this paper we focus our attention on ReLU activation function defined by $\sigma(t) := \max\{t, 0\}$, $t \in \mathbb{R}$. We will use the notation $\sigma(\boldsymbol{x}) := (\sigma(x_1), \dots, \sigma(x_d))$ for $\boldsymbol{x} \in \mathbb{R}^d$.

Definition 3. A deep ReLU neural network with input dimension d and L layers is a neural network

$$\Phi = \big((\boldsymbol{W}^1, \boldsymbol{b}^1), \dots, (\boldsymbol{W}^L, \boldsymbol{b}^L)\big),$$

in which the following computation scheme is implemented

$$\begin{aligned} &\boldsymbol{z}^0 := \boldsymbol{x} \in \mathbb{R}^d, \\ &\boldsymbol{z}^\ell := \sigma(\boldsymbol{W}^\ell \boldsymbol{z}^{\ell-1} + \boldsymbol{b}^\ell), \quad \ell = 1, \dots, L-1, \\ &\boldsymbol{z}^L := \boldsymbol{W}^L \boldsymbol{z}^{L-1} + \boldsymbol{b}^L. \end{aligned}$$

We call $\boldsymbol{z}^0$ the input and with an ambiguity denote $\Phi(\boldsymbol{x}) := \boldsymbol{z}^L$ the output of Φ and in some places we identify a deep ReLU neural network with its output.

Several deep ReLU neural networks can be combined to form a larger deep ReLU neural network whose output is a linear combination or composition of outputs of sub-networks. In the following, we introduce parallelization, concatenation and special construction.

Lemma 1 (Parallelization)**.** *Let $N \in \mathbb{N}$, $\Omega \subset \mathbb{R}^d$ be a bounded set, $\lambda_j \in \mathbb{R}$, $j = 1, \dots, N$. Let Φ_j, $j = 1, \dots, N$ be deep ReLU neural networks with input dimension d. Then we can explicitly construct a deep ReLU neural network denoted by Φ so that*

$$\Phi(\boldsymbol{x}) = \sum_{j=1}^N \lambda_j \Phi_j(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega,$$

with $L(\Phi) = \max_{j=1,\dots,N}\{L(\Phi_j)\}$ *and*

$$W(\Phi) = \sum_{j=1}^{N} W(\Phi_j) + \sum_{j:L(\Phi_j)<L(\Phi)} (L(\Phi) - L(\Phi_j) + 2) \leqslant 3N \max_{j=1,\dots,N} W(\Phi_j).$$

The network Φ *is called the Parallelization network of* Φ_j, $j = 1, \dots, N$.

A proof of Lemma 1 can be found in [3]. The last estimate in Lemma 1 is due to $2 \leqslant L \leqslant \max_{j=1,\dots,N} W(\Phi_j)$.

Another way to construct a ReLU network whose output is a linear combination of outputs of other ReLU networks is to use special networks. A special deep ReLU neural network with input dimension d can be defined as follows. In each hidden layer a special role is reserved for d first (top) nodes and the last (bottom) node. Concatenation of top d nodes and the bottom node in each layer to the corresponding nodes in the next layer form $d + 1$ parallel channels. The nodes in these $d + 1$ channel are free of activation. The top d parallel channels are called the source channels and just carry $\boldsymbol{x} = (x_1, \dots, x_d)$ forward. The bottom channel is called collation channel. The nodes in the bottom channel are used to collect intermediate outputs by addition. This channel never feeds forward into subsequent calculation, it only accepts previous calculations. It has been shown in [3] that if Φ is a special deep ReLU neural network with input dimension d depth L and $\boldsymbol{x} \in \mathbb{I}^d$, then there is a deep ReLU neural network Φ' such that

$$L(\Phi') = L(\Phi) \tag{3}$$

and $\Phi'(\boldsymbol{x}) = \Phi(\boldsymbol{x})$. In view of the proof of [3, Lemma 4.2] we find only node weights in the collation channel of Φ and Φ' are different. Therefore we deduce

$$W(\Phi') \leqslant W(\Phi) + L(\Phi) \leqslant 2W(\Phi). \tag{4}$$

Lemma 2 (Special Construction)**.** *Let* $N \in \mathbb{N}$, $\Omega \subset \mathbb{R}^d$ *be a bounded set,* $\lambda_j \in \mathbb{R}$, $j = 1, \dots, N$. *Let* Φ_j, $j = 1, \dots, N$ *be deep ReLU neural networks with input dimension* d. *Then we can explicitly construct a deep special ReLU neural network denoted by* Φ *so that*

$$\Phi(\boldsymbol{x}) = \sum_{j=1}^{N} \lambda_j \Phi_j(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega,$$

with $L(\Phi) = \sum_{j=1}^{N} L(\Phi_j)$ *and* $W(\Phi) \leqslant \sum_{j=1}^{N} W(\Phi_j) + (d + 1)L(\Phi)$.

An illustration of a special network Φ whose output is a linear combination of network Φ_j, $j = 1, \dots, N$ is given in Figure 2.

Lemma 3 (Concatenation)**.** *Let* Φ_1 *and* Φ_2 *be two ReLU neural networks such that output layer of* Φ_1 *has the same dimension as input layer of* Φ_2. *Then, we can explicitly construct a ReLU neural network* Φ *such that* $\Phi(\boldsymbol{x}) = \Phi_2(\Phi_1(\boldsymbol{x}))$ *for* $\boldsymbol{x} \in \mathbb{R}^d$. *Moreover we have* $L(\Phi) = L(\Phi_1) + L(\Phi_2)$ *and* $W(\Phi) \leqslant 2W(\Phi_1) + 2W(\Phi_2)$.

A proof of the above lemma can be found in [29]. The network Φ in this lemma is called the concatenation network of Φ_1 and Φ_2.

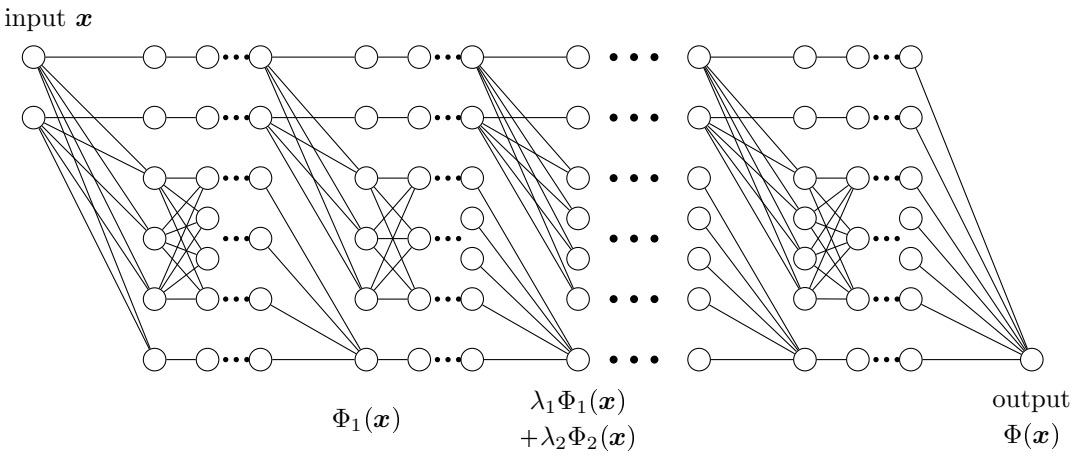

Figure 2: Illustration of a special deep ReLU neural network ($d = 2$)

3. APPROXIMATION BY SETS OF FINITE CARDINALITY

In this section we recall a decomposition of continuous functions on the unit cube $\mathbb{I}^d$ by Faber series, interpolation approximation by truncated Faber series and by set of finite cardinality. They are a key tool for explicit construction of approximation methods by deep ReLU networks for functions in Hölder-Nikol'skii spaces of mixed smoothness.

Let $\varphi(x) = (1 - |x - 1|)_+$, $x \in \mathbb{R}$, be the hat function (the piece-wise linear B-spline with knots at $0, 1, 2$), where $x_+ := \max(x, 0)$ for $x \in \mathbb{R}$. For $k \in \mathbb{N}_{-1}$ we define the functions $\varphi_{k,s}$ by

$$\varphi_{k,s}(x) := \varphi(2^{k+1}x - 2s), \quad k \geqslant 0,\ s \in Z(k) := \{0, 1, \dots, 2^k - 1\},$$

and

$$\varphi_{-1,s}(x) := \varphi(x - s + 1),\ s \in Z(-1) := \{0, 1\}.$$

For a univariate function f on $\mathbb{I}$, $k \in \mathbb{N}_{-1}$, and $s \in Z(k)$ we define

$$\lambda_{k,s}(f) := -\frac{1}{2}\Delta^2_{2^{-k-1}} f\big(2^{-k}s\big),\ k \geqslant 0, \quad \lambda_{-1,s}(f) := f(s),$$

where

$$\Delta^2_h f(x) := f(x + 2h) - 2f(x + h) + f(x),$$

for all x and $h \geqslant 0$ such that $x, x + h \in \mathbb{I}$. If $m \in \mathbb{N}_0$ we put

$$R_m(f) := \sum_{k=0}^{m} q_k(f), \quad q_k(f) := \sum_{s \in Z(k)} \lambda_{k,s}(f)\varphi_{k,s}. \tag{5}$$

For $k \in \mathbb{N}_0$, we define the functions $\varphi^*_{k,s} \in \mathring{C}(\mathbb{I})$ by

$$\varphi^*_{k,s}(x) := \varphi(2^{k+1}x - s + 1), \quad s \in Z_*(k) := \{1, \dots, 2^{k+1} - 1\}, \tag{6}$$

and for $f \in \mathring{C}(\mathbb{I})$ one can check

$$R_m(f) = \sum_{s \in Z_*(m)} f(2^{-m-1}s)\varphi^*_{m,s}\,.$$

Hence $R_m(f) \in \mathring{C}(\mathbb{I})$ interpolates f at the points $2^{-m-1}s$, $s \in Z_*(m)$, that is,

$$R_m(f)(2^{-m-1}s) \;=\; f(2^{-m-1}s), \quad s \in Z_*(m).$$

Put $Z(\boldsymbol{k}) := \times_{j=1}^{d} Z(k_j)$. For $\boldsymbol{k} \in \mathbb{N}_{-1}^d$, $\boldsymbol{s} \in Z(\boldsymbol{k})$, we introduce the tensorized Faber basis by

$$\varphi_{\boldsymbol{k},\boldsymbol{s}}(\boldsymbol{x}) \;:=\; \varphi_{k_1,s_1}(x_1) \cdot \ldots \cdot \varphi_{k_d,s_d}(x_d), \quad \boldsymbol{x} \in \mathbb{I}^d. \tag{7}$$

We also define the linear functionals $\lambda_{\boldsymbol{k},\boldsymbol{s}}$ for multivariate function f on $\mathbb{I}^d$, $\boldsymbol{k} \in \mathbb{N}_{-1}^d$, and $\boldsymbol{s} \in Z(\boldsymbol{k})$ by

$$\lambda_{\boldsymbol{k},\boldsymbol{s}}(f) \;:= \prod_{i=1}^{d} \lambda_{k_i,s_i}(f),$$

where the univariate functional λ_{k_i,s_i} is applied to the univariate function f by considering f as a function of variable x_i with the other variables held fixed.

We have the following representation by Faber series for continuous functions on $\mathbb{I}^d$ (see [2, Section 4] and [36, Theorem 3.10]).

Lemma 4. *The tensorized Faber system* $\{\varphi_{\boldsymbol{k},\boldsymbol{s}} : \boldsymbol{k} \in \mathbb{N}_{-1}^d, \boldsymbol{s} \in Z(\boldsymbol{k})\}$ *is a basis in* $C(\mathbb{I}^d)$. *Moreover, every function* $f \in C(\mathbb{I}^d)$ *can be represented by the Faber series*

$$f \;=\; \sum_{\boldsymbol{k} \in \mathbb{N}_{-1}^d} q_{\boldsymbol{k}}(f), \quad q_{\boldsymbol{k}}(f) := \sum_{\boldsymbol{s} \in Z(\boldsymbol{k})} \lambda_{\boldsymbol{k},\boldsymbol{s}}(f)\varphi_{\boldsymbol{k},\boldsymbol{s}}$$

converging in the norm of $C(\mathbb{I}^d)$.

When $f \in \mathring{U}_{\infty}^{\alpha,d}$, $\lambda_{\boldsymbol{k},\boldsymbol{s}}(f) = 0$ if $k_j = -1$ for some $j \in \{1, \ldots, d\}$, hence we can write

$$f \;=\; \sum_{\boldsymbol{k} \in \mathbb{N}_0^d} q_{\boldsymbol{k}}(f)$$

with unconditional convergence in $C(\mathbb{I}^d)$, see [36, Theorem 3.13]. In this case it holds the following estimate

$$\begin{aligned} |\lambda_{\boldsymbol{k},\boldsymbol{s}}(f)| &= 2^{-d} \Big| \prod_{i=1}^{d} \Delta_{2^{-k_i-1}}^2 f\big(2^{-\boldsymbol{k}}\boldsymbol{s}\big) \Big| \\ &= 2^{-d} \Big| \prod_{i=1}^{d} \Big[\Delta_{2^{-k_i-1}} f\big(2^{-\boldsymbol{k}}\boldsymbol{s} + 2^{-k_i-1}\boldsymbol{e}^i\big) - \Delta_{2^{-k_i-1}} f\big(2^{-\boldsymbol{k}}\boldsymbol{s}\big) \Big] \Big| \leqslant 2^{-\alpha d} 2^{-\alpha |\boldsymbol{k}|_1}, \end{aligned} \tag{8}$$

for $\boldsymbol{k} \in \mathbb{N}_0^d$, $\boldsymbol{s} \in Z(\boldsymbol{k})$. Here $\{\boldsymbol{e}^i\}_{i=1,\ldots,d}$ is the standard basis of $\mathbb{R}^d$.

For $f \in \mathring{C}(\mathbb{I}^d)$, we define the operator R_m by

$$R_m(f) := \sum_{|\boldsymbol{k}|_1 \leqslant m} q_{\boldsymbol{k}}(f) \;=\; \sum_{|\boldsymbol{k}|_1 \leqslant m} \; \sum_{s \in Z(\boldsymbol{k})} \lambda_{\boldsymbol{k},\boldsymbol{s}}(f)\varphi_{\boldsymbol{k},\boldsymbol{s}}.$$

The truncated Faber series $R_m(f) \in \mathring{C}(\mathbb{I}^d)$ completely determined by values of f at the points $2^{-\boldsymbol{k}-\mathbf{1}}\boldsymbol{s}$, for $(\boldsymbol{k}, \boldsymbol{s}) \in G^d(m)$, where

$$G^d(m) := \big\{(\boldsymbol{k}, \boldsymbol{s}) : \; |\boldsymbol{k}|_1 \leqslant m, \; \boldsymbol{s} \in Z_*(\boldsymbol{k})\big\},$$

$Z_*(\boldsymbol{k}) := \prod_{j=1}^d Z_*(k_j)$ and $\mathbf{1} = (1,\dots,1) \in \mathbb{N}^d$. Moreover, $R_m(f)$ interpolates f at the points $2^{-\boldsymbol{k}-\mathbf{1}}\boldsymbol{s}$, for $(\boldsymbol{k},\boldsymbol{s}) \in G^d(m)$, i.e.,

$$R_m(f)(2^{-\boldsymbol{k}-\mathbf{1}}\boldsymbol{s}) \;=\; f(2^{-\boldsymbol{k}-\mathbf{1}}\boldsymbol{s}), \quad (\boldsymbol{k},\boldsymbol{s}) \in G^d(m).$$

The following lemma gives a d-dependent estimate of the approximation error by $R_m(f)$ of $f \in \mathring{U}_\infty^{\alpha,d}$, see [4].

Lemma 5. *Let $d \geqslant 2$, $m \in \mathbb{N}$, and $0 < \alpha \leqslant 1$. Then we have*

$$\sup_{f \in \mathring{U}_\infty^{\alpha,d}} \|f - R_m(f)\|_\infty \leqslant 2^{-\alpha} B^d\, 2^{-\alpha m} \binom{m+d}{d-1},\ B = (2^\alpha - 1)^{-1}.$$

We make use the abbreviations: $\boldsymbol{x}_j := (x_1,\dots,x_j) \in \mathbb{R}^j$; $\bar{\boldsymbol{x}}_j := (x_{j+1},\dots,x_d) \in \mathbb{R}^{d-j}$ with the convention $\boldsymbol{x}_0 := 0$ for $\boldsymbol{x} \in \mathbb{R}^d$ and $j = 0, 1, \dots, d-1$. When $j = 1$ we denote x_1 instead of $\boldsymbol{x}_1$.

For $f \in \mathring{U}_\infty^{\alpha,1}$ we explicitly construct the function $S_f \in \mathring{C}(\mathbb{I})$ by

$$S_f := \sum_{s \in Z_*(m)} 2^{-\alpha(m+1)} l_s(f) \varphi^*_{m,s}, \tag{9}$$

where we put $l_0(f) = 0$ and assign the values $S_f(2^{-m-1}s) = 2^{-\alpha(m+1)} l_s(f)$ from left to right closest to $f(2^{-m-1}s)$ for $s = 1, \dots, 2^{m+1} - 1$. If there are two possible choices for $l_s(f)$ we choose $l_s(f)$ that is closest to the already determined $l_{s-1}(f)$. We define

$$\mathcal{S}^\alpha(m) := \big\{S_f : f \in \mathring{U}_\infty^{\alpha,1}\big\}. \tag{10}$$

It has been proved that the set $\mathcal{S}^\alpha(m)$ is finite and it holds the estimate $|\mathcal{S}^\alpha(m)| \leqslant 3^{2^{m+1}}$, see [4]. Moreover, by Lemma 5 and [4, Lemma 2.3] for $f \in \mathring{U}_\infty^{\alpha,1}$ and $m \in \mathbb{N}_0$ we have

$$\|f - S_f\|_\infty \leqslant \|f - R_m(f)\|_\infty + \|R_m(f) - S_f\|_\infty \leqslant 2^{-(m+1)\alpha - \frac{1}{2}} + \frac{2^{-(m+1)\alpha}}{2^\alpha - 1}\,. \tag{11}$$

In case of high dimensions we have the following.

Lemma 6. *Let $m > 1$, $d \geqslant 2$ and $0 < \alpha \leqslant 1$. For $f \in \mathring{U}_\infty^{\alpha,d}$, let the function $S_m(f)$ be defined by*

$$S_m(f)(\boldsymbol{x}) := \sum_{|\bar{\boldsymbol{k}}_1| \leqslant m} 2^{-\alpha(|\bar{\boldsymbol{k}}_1|_1 + d - 1)} \sum_{\bar{\boldsymbol{s}}_1 \in Z(\bar{\boldsymbol{k}}_1)} \varphi_{\bar{\boldsymbol{k}}_1, \bar{\boldsymbol{s}}_1}(\bar{\boldsymbol{x}}_1) S_{K_{\bar{\boldsymbol{k}}_1, \bar{\boldsymbol{s}}_1}(f)}(x_1), \tag{12}$$

where $S_{K_{\bar{\boldsymbol{k}}_1, \bar{\boldsymbol{s}}_1}(f)} \in \mathcal{S}^\alpha(m - |\bar{\boldsymbol{k}}_1|_1)$ is as in (9) for the function $K_{\bar{\boldsymbol{k}}_1, \bar{\boldsymbol{s}}_1}(f)$. Then it holds the inequality

$$\|f - S_m(f)\|_\infty \leqslant B^d 2^{-\alpha m} \binom{m+d}{d-1}.$$

Moreover, for the set

$$\mathcal{S}^{\alpha,d}(m) := \big\{S_m(f) :\ f \in \mathring{U}_\infty^{\alpha,d}\big\},$$

we have $N_d(m) := |\mathcal{S}^{\alpha,d}(m)| \leqslant 3^{2^{m+1}\binom{m+d-1}{d-1}}$.

For a proof of the above lemma we refer the reader to [4].

4. DEEP RELU NEURAL NETWORK APPROXIMATION A NONADAPTIVE METHOD

In this section, we explicitly construct a nonadaptive deep ReLU neural network having an output that approximates every function $f \in \mathring{U}_\infty^{\alpha,d}$ in the $L_\infty(\mathbb{I}^d)$-norm with a prescribed accuracy ε and prove dimension-dependent error estimates of its size and depth. Nonadaptivity means that its architecture is the same for all $f \in \mathring{U}_\infty^{\alpha,d}$. Our technique is first to approximate f by its truncation of Faber series $R_n(f)$ and then approximate $R_n(f)$ by a deep ReLU network. Since the case $d = 1$ was already considered (see, e.g., [1, 7, 11]), we study the high dimension case when $d \geqslant 2$. Our main result in this section is read as follows.

Theorem 7. *Let $d \in \mathbb{N}$, $d \geqslant 2$ and $\alpha \in (0, 1]$. Then there is $\varepsilon_0 = \varepsilon_0(d, \alpha) \in (0, 1]$ such that for every $\varepsilon \in (0, \varepsilon_0)$ we can explicitly construct a deep neural network architecture $\mathbb{A}_\varepsilon$ with the following property. For every $f \in \mathring{U}_\infty^{\alpha,d}$, we can explicitly construct a deep ReLU neural network $\Phi_\varepsilon(f)$ having the architecture $\mathbb{A}_\varepsilon$ such that*

$$\|f - \Phi_\varepsilon(f)\|_\infty \leqslant \varepsilon.$$

Moreover, we have

$$W(\mathbb{A}_\varepsilon) \leqslant C_\alpha d \left(\frac{K_1^d}{(d-1)!}\right)^{\frac{1}{\alpha}+1} \varepsilon^{-\frac{1}{\alpha}} \log(2\varepsilon^{-1})^{(d-1)(\frac{1}{\alpha}+1)+1} \tag{13}$$

and

$$L(\mathbb{A}_\varepsilon) \leqslant C \log d \log(2\varepsilon^{-1}),$$

where $K_1 = B^{1/(\alpha+1)} 4\alpha^{-1}$ with B given in Lemma 5 and C_α depends only on α.

To prepare for proving Theorem 7 we recall results of approximating the product $\prod_{j=1}^d x_j$ and $\varphi_{\boldsymbol{k},\boldsymbol{s}}$ by deep ReLU neural networks, see [33] and [3].

Lemma 8. *For every $\delta \in (0, 1)$, $d \in \mathbb{N}$, $d \geqslant 2$, we can explicitly construct a deep ReLU neural network Φ_P so that*

$$\sup_{\boldsymbol{x} \in [-1,1]^d} \left| \prod_{j=1}^d x_j - \Phi_P(\boldsymbol{x}) \right| \leqslant \delta.$$

Furthermore, if $x_j = 0$ for some $j \in \{1, \ldots, d\}$ then $\Phi_P(\boldsymbol{x}) = 0$ and there exists a constant $C > 0$ independent of δ and d such that

$$W(\Phi_P) \leqslant Cd \log(d\delta^{-1}) \quad \text{and} \quad L(\Phi_P) \leqslant C \log d \log(d\delta^{-1}).$$

Lemma 9. *For every dimension $d \geqslant 2$, $\delta \in (0, 1)$ and for the d-variate hat functions $\varphi_{\boldsymbol{k},\boldsymbol{s}}$, $\boldsymbol{k} \in \mathbb{N}_0^d$, $\boldsymbol{s} \in Z(\boldsymbol{k})$, defined as in (7), we can explicitly construct a deep neural network $\Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}})$ so that*

$$\|\varphi_{\boldsymbol{k},\boldsymbol{s}} - \Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}})\|_\infty \leqslant \delta$$

and

$$W(\Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}})) \leqslant Cd \log(d\delta^{-1}) \quad \text{and} \quad L(\Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}})) \leqslant C \log d \log(d\delta^{-1}). \tag{14}$$

Moreover, $\operatorname{supp} \Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}}) \subset \operatorname{supp} \varphi_{\boldsymbol{k},\boldsymbol{s}}$.

The above result allows us to construct a deep ReLU network $\Phi_\varepsilon\big(R_n(f)\big)$ to approximate $R_n(f)$.

Lemma 10. *Let $d \in \mathbb{N}$, $d \geqslant 2$, $n \in \mathbb{N}$, $\alpha \in (0,1]$ and $\varepsilon \in (0,1)$. Then for every $f \in \mathring{U}_\infty^{\alpha,d}$ we can explicitly construct a deep ReLU network $\Phi_\varepsilon\big(R_n(f)\big)$ of the same architecture $\mathbb{A}_\varepsilon$ so that*

$$\big\|R_n(f) - \Phi_\varepsilon\big(R_n(f)\big)\big\|_\infty \leqslant \varepsilon. \tag{15}$$

Moreover, we have

$$W\big(\Phi_\varepsilon(R_n(f))\big) \leqslant Cd2^n \binom{n+d-1}{d-1} \log(dB^d\varepsilon^{-1}) \tag{16}$$

and

$$L\big(\Phi_\varepsilon(R_n(f))\big) \leqslant C \log d \log(dB^d\varepsilon^{-1}). \tag{17}$$

The estimates (16) *and* (17) *also hold for $W(\mathbb{A}_\varepsilon)$ and $L(\mathbb{A}_\varepsilon)$ respectively.*

Proof. For every pair $\boldsymbol{k}, \boldsymbol{s}$ with $|\boldsymbol{k}|_1 \leqslant n$ and $\boldsymbol{s} \in Z(\boldsymbol{k})$, by applying Lemma 9 with $\delta := B^{-d}\varepsilon$, we explicitly construct a deep ReLU neural network $\Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}})$ so that $\operatorname{supp}\Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}}) \subset \operatorname{supp}\varphi_{\boldsymbol{k},\boldsymbol{s}}$,

$$\|\varphi_{\boldsymbol{k},\boldsymbol{s}} - \Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}})\|_\infty \leqslant B^{-d}\varepsilon, \tag{18}$$

and it holds the estimates (14) for $W(\Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}}))$ and $L(\Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}}))$. We approximate $R_n(f)$ by the output

$$\Phi_\varepsilon\big(R_n(f)\big)(\boldsymbol{x}) = \sum_{|\boldsymbol{k}|_1 \leqslant n} \sum_{\boldsymbol{s} \in Z(\boldsymbol{k})} \lambda_{\boldsymbol{k},\boldsymbol{s}}(f)\Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}})(\boldsymbol{x})$$

of the network $\Phi_\varepsilon\big(R_n(f)\big)$ which is a parallelization of the networks $\{\Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}})\}_{|\boldsymbol{k}|_1 \leqslant n,\, \boldsymbol{s} \in Z(\boldsymbol{k})}$. Notice that the interiors of $\operatorname{supp}\Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}})$ and $\operatorname{supp}\Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}'})$ have empty intersection if $\boldsymbol{s} \neq \boldsymbol{s}'$. Moreover, for every $\boldsymbol{x} \in \mathbb{I}^d$, there is an $\boldsymbol{s} \in Z(\boldsymbol{k})$ such that $\boldsymbol{x} \in \operatorname{supp}\varphi_{\boldsymbol{k},\boldsymbol{s}}$, and hence, by using (8) and (18) we get the estimates

$$\begin{aligned}
|R_n(f)(\boldsymbol{x}) - \Phi_\varepsilon\big(R_n(f)\big)(\boldsymbol{x})| &= \sum_{|\boldsymbol{k}|_1 \leqslant n} \big|\lambda_{\boldsymbol{k},\boldsymbol{s}}(f)\big(\varphi_{\boldsymbol{k},\boldsymbol{s}}(\boldsymbol{x}) - \Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}})(\boldsymbol{x})\big)\big| \\
&\leqslant 2^{-\alpha d} \sum_{|\boldsymbol{k}|_1 \leqslant n} 2^{-\alpha|\boldsymbol{k}|_1}\varepsilon B^{-d} \\
&\leqslant \varepsilon(1-2^{-\alpha})^d \sum_{j=0}^{n} 2^{-\alpha j} \binom{j+d-1}{d-1}.
\end{aligned}$$

From

$$\sum_{j=0}^{\infty} \binom{j+m}{m} t^j \leqslant (1-t)^{-m-1},\ t \in (0,1), \tag{19}$$

see [6, Lemma 2.2], we obtain (15).

By using Lemma 1 and the estimates (14), the size and the depth of $\Phi_\varepsilon\big(R_n(f)\big)$ can be estimated as

$$\begin{aligned}
W(\Phi_\varepsilon\big(R_n(f)\big)) &\leqslant C|\{(\boldsymbol{k},\boldsymbol{s}) : |\boldsymbol{k}|_1 \leqslant n, \boldsymbol{s} \in Z(\boldsymbol{k})\}| \max_{|\boldsymbol{k}|_1 \leqslant n, \boldsymbol{s} \in Z(\boldsymbol{k})} W(\Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}})) \\
&= C\sum_{\ell=0}^{n} 2^\ell \binom{\ell+d-1}{d-1} d\log(dB^d\varepsilon^{-1}) \\
&\leqslant Cd2^n \binom{n+d-1}{d-1} \log(dB^d\varepsilon^{-1}),
\end{aligned}$$

and

$$L(\Phi_\varepsilon\big(R_n(f)\big)) \leqslant \max_{|\boldsymbol{k}|_1\leqslant n, \boldsymbol{s}\in Z(\boldsymbol{k})} L(\Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}})) \leqslant C\log d\log(dB^d\varepsilon^{-1}).$$

The proof is completed by noticing that $\Phi_\varepsilon\big(R_n(f)\big)$ has the architecture $\mathbb{A}_\varepsilon$ (independent of f) which is defined as the minimal architecture of the deep ReLU neural network Φ_ε obtained by parallelization of the networks $\{\Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}})\}_{|\boldsymbol{k}|_1\leqslant n,\, \boldsymbol{s}\in Z(\boldsymbol{k})}$ with the output

$$\Phi_\varepsilon(\boldsymbol{x}) = \sum_{|\boldsymbol{k}|_1\leqslant n}\sum_{\boldsymbol{s}\in Z(\boldsymbol{k})} \Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}})(\boldsymbol{x}),\ \boldsymbol{x}\in\mathbb{I}^d.$$

Hence, the estimates (16) and (17) also hold for $W(\mathbb{A}_\varepsilon)$ and $L(\mathbb{A}_\varepsilon)$ respectively. ∎

We are ready to prove Theorem 7.

Proof. Denote n_0 the natural point from which the function $h(n) = 2^{-\alpha}B^d 2^{-\alpha n}\binom{n+d}{d-1}$ is decreasing and $h(n-1) \leqslant 2^{-\alpha n/2}$ for all $n > n_0$. We put $\varepsilon_0 = \min\{h(n_0), h(d)\}$. For $\varepsilon\in(0,\varepsilon_0)$ we define $n > \max\{n_0, d\}$ by

$$2^{-\alpha}B^d 2^{-\alpha n}\binom{n+d}{d-1} \leqslant \frac{\varepsilon}{2} < 2^{-\alpha}B^d 2^{-\alpha(n-1)}\binom{n-1+d}{d-1}. \tag{20}$$

With $\varepsilon' = \varepsilon/2$ in Lemma 10 and $\Phi_\varepsilon(f) = \Phi_{\varepsilon'}(R_n(f))$ we have

$$\|f-\Phi_\varepsilon(f)\|_\infty \leqslant \|f-R_n(f)\|_\infty + \|R_n - \Phi_{\varepsilon'}(R_n(f))\|_\infty \leqslant 2^{-\alpha}B^d 2^{-\alpha n}\binom{n+d}{d-1} + \frac{\varepsilon}{2} \leqslant \varepsilon.$$

We define $\mathbb{A}_\varepsilon$ as the minimal architecture of the deep ReLU neural network Φ_ε obtained by parallelization of the networks $\{\Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}})\}_{|\boldsymbol{k}|_1\leqslant n,\, \boldsymbol{s}\in Z(\boldsymbol{k})}$ with the output

$$\Phi_\varepsilon(\boldsymbol{x}) = \sum_{|\boldsymbol{k}|_1\leqslant n}\sum_{\boldsymbol{s}\in Z(\boldsymbol{k})} \Phi_\delta(\varphi_{\boldsymbol{k},\boldsymbol{s}})(\boldsymbol{x}), \quad \boldsymbol{x}\in\mathbb{I}^d.$$

Then $\Phi_\varepsilon(f)$ has the architecture for all $f\in \mathring{U}_\infty^{\alpha,d}$. From Lemma 10 we have

$$W(\mathbb{A}_\varepsilon) \leqslant Cd2^n\log(2d\varepsilon^{-1}B^d)\binom{n+d-1}{d-1}.$$

From the choice of n we have

$$2d\varepsilon^{-1}B^d \leqslant d2^\alpha 2^{\alpha n}\binom{n+d}{d-1}^{-1} \leqslant 2^{\alpha n}2d\binom{n+d}{d-1}^{-1} \leqslant 2^{\alpha n}.$$

By this and (20) we get

$$\begin{aligned} W(\mathbb{A}_\varepsilon) &\leqslant Cd\bigg(2\varepsilon^{-1}B^d\binom{n+d-1}{d-1}\bigg)^{1/\alpha}\alpha n\binom{n+d-1}{d-1} \\ &\leqslant Cd\big(2\varepsilon^{-1}B^d\big)^{1/\alpha} n\binom{n+d-1}{d-1}^{\frac{1}{\alpha}+1} \\ &\leqslant Cd\big(\varepsilon^{-1}B^d\big)^{1/\alpha} n\bigg(\frac{(2n)^{d-1}}{(d-1)!}\bigg)^{\frac{1}{\alpha}+1}. \end{aligned}$$

Now $h(n-1) \leqslant 2^{-\alpha n/2}$ and (20) lead to $\frac{\varepsilon}{2} \leqslant 2^{-\alpha n/2}$ which implies $n \leqslant \frac{2}{\alpha}\log(2\varepsilon^{-1})$. Therefore we get

$$W(\mathbb{A}_\varepsilon) \leqslant Cd\big(\varepsilon^{-1}B^d\big)^{1/\alpha}\log(2\varepsilon^{-1})\left(\frac{(4\alpha^{-1}\log(2\varepsilon^{-1}))^{d-1}}{(d-1)!}\right)^{\frac{1}{\alpha}+1}$$
$$= Cd\big(B^d\big)^{1/\alpha}\left(\frac{(4\alpha^{-1})^{d-1}}{(d-1)!}\right)^{\frac{1}{\alpha}+1}\varepsilon^{-\frac{1}{\alpha}}\log(2\varepsilon^{-1})^{(d-1)(\frac{1}{\alpha}+1)+1},$$

and (13) follows. We also have

$$L(\mathbb{A}_\varepsilon) \leqslant C\log d\log\big(d2\varepsilon^{-1}B^d\big) \leqslant C\alpha n\log d \leqslant C\log d\log(2\varepsilon^{-1}).$$

■

5. DEEP RELU NEURAL NETWORK APPROXIMATION - AN ADAPTIVE METHOD

In this section, we explicitly construct an adaptive method of approximation with accuracy $\varepsilon > 0$ by deep ReLU neural networks of functions $f \in \mathring{U}_\infty^{\alpha,d}$. This method reduces the computation complexity expressing as the size and depth of the approximating deep ReLU networks comparing with the computation complexity of the nonadaptive method given in Theorem 7. As mentioned the univariate case was already considered in [7] ($0 < \alpha < 1$)) and [40] ($\alpha = 1$), we focus our attention on multivariate case when $d \geqslant 2$. The main result of this section is read as follows.

Theorem 11. *Let $d \in \mathbb{N}$, $d \geqslant 2$, $\alpha \in (0,1]$. Then there is $\varepsilon_0 = \varepsilon_0(d,\alpha) \in (0,1/2]$ such that for every $\varepsilon \in (0,\varepsilon_0)$ and for every $f \in \mathring{U}_\infty^{\alpha,d}$ we can explicitly construct an adaptive deep ReLU neural network $\Phi_\varepsilon(f)$ so that*

$$\|f - \Phi_\varepsilon(f)\|_\infty \leqslant \varepsilon.$$

Moreover, we have

$$W(\Phi_\varepsilon(f)) \leqslant C_\alpha d^2\left(\frac{K_2^d}{(d-1)!}\right)^{\frac{2}{\alpha}+2}\varepsilon^{-\frac{1}{\alpha}}\big(\log(2\varepsilon^{-1})\log\log(2\varepsilon^{-1})\big)^{(1+\frac{1}{\alpha})(d-1)} \tag{21}$$

and

$$L(\Phi_\varepsilon(f)) \leqslant C'_\alpha\varepsilon^{-\frac{1}{d\alpha}}(\log(2\varepsilon^{-1}))^{\frac{d-1-\alpha}{d\alpha}}(\log\log(2\varepsilon^{-1}))^{\frac{(\alpha+1)(d-1)}{d\alpha}}, \tag{22}$$

where

$$K_2 := 4(2^{\alpha+3}B)^{\frac{1}{2\alpha+2}}(\alpha^{-1}\log(2\alpha^{-1}))^{1/2}$$

and positive constants C_α, C'_α depend on α only.

Let us explain the idea of the proof. Let $f \in \mathring{U}_\infty^{\alpha,d}$ and $\varepsilon \in (0,\varepsilon_0)$ (ε_0 will be specified latter) be given. Using the writing

$$f \;=\; R_n(f) \;+\; (f - R_n(f)),$$

we explicitly construct deep ReLU neural networks to approximate with accuracy $\varepsilon/2$ the terms $R_n(f)$ and $f - R_n(f)$ and evaluate the dimension-dependent computation complexity

separately, and then take their sum to get an approximation with accuracy ε to f and its dimension-dependent computation complexity. For approximation of the first term $R_n(f)$, we take the deep ReLU neural network $\Phi_{\varepsilon/2}(R_n(f))$ which has been constructed in Lemma 10.

Thus, our main task is to explicitly construct a desired deep ReLU neural network $\Phi_{\varepsilon/2}\big(f-R_n(f)\big)$ for approximation of the second term $f-R_n(f)$. Our strategy is to represent the difference $f-R_n(f)$ in a special form and then approximate terms in this representation by deep ReLU networks. To this end, we need some auxiliary preparation.

For univariate functions $f\in \mathring{C}(\mathbb{I})$, let the operator T_k, $k\in\mathbb{N}_0$, be defined by

$$T_k(f):=f-R_{k-1}(f)$$

with the operator R_k defined as in (5) and the convention $R_{-1}:=0$. From this definition we have T_0 is the identity operator. Notice that for $f\in \mathring{U}_\infty^{\alpha,1}$, it holds the inequality $\|T_k(f)\|_{H^\alpha_\infty(\mathbb{I})}\leqslant 2$.

For a multivariate function $f\in \mathring{C}(\mathbb{I}^d)$, the tensor product operator $T_{\boldsymbol{k}}$, $\boldsymbol{k}=(k_1,\dots,k_d)\in\mathbb{N}_0^d$, is defined by

$$T_{\boldsymbol{k}}(f):=\prod_{j=1}^d T_{k_j}(f),$$

where the univariate operator T_{k_j} is applied to the univariate function f by considering f as a function of variable x_j with the other variables held fixed.

For $n\in\mathbb{N}$, it has been shown in [4] that $f-R_n(f)$ can be represented in the following special form

$$f-R_n(f)=\sum_{j=0}^{d-1}\sum_{|\boldsymbol{k}_j|_1\leqslant n} F_{\boldsymbol{k}_j}, \tag{23}$$

where $F_{\boldsymbol{k}_0}:=T_{(n+1)\boldsymbol{e}^1}$ and

$$F_{\boldsymbol{k}_j}:=T_{(n+1-|\boldsymbol{k}_j|_1)\boldsymbol{e}^{j+1}}\big(q_{\boldsymbol{k}_j}(f)\big),\quad j=1,\dots,d-1,$$

or equivalently,

$$F_{\boldsymbol{k}_j}=\prod_{i=1}^j\big(T_{(k_i-1)\boldsymbol{e}^j}-T_{k_i\boldsymbol{e}^j}\big)T_{(n+1-|\boldsymbol{k}_j|_1)\boldsymbol{e}^{j+1}}(f),\qquad j=1,\dots,d-1. \tag{24}$$

We shall explicitly construct deep ReLU neural networks $\Phi_{\varepsilon'}(F_{\boldsymbol{k}_j})$ to approximate each term $F_{\boldsymbol{k}_j}$ in the sum in (23). Due to (24) this is reduced to construct deep ReLU networks that approximate $T_{\boldsymbol{k}}(f)$, $\boldsymbol{k}\in\mathbb{N}_0^d$. Put

$$I_{\boldsymbol{k},\boldsymbol{s}}:=\bigtimes_{j=1}^d I_{k_j,s_j}=\bigtimes_{j=1}^d[2^{-k_j}s_j,2^{-k_j}(s_j+1)],\ \boldsymbol{k}\in\mathbb{N}_0^d,\ \boldsymbol{s}\in Z(\boldsymbol{k}),$$

and

$$T_{\boldsymbol{k},\boldsymbol{s}}(f)(\boldsymbol{x}):=2^{\alpha|\boldsymbol{k}|_1-d}\big(T_{\boldsymbol{k}}(f)\chi_{I_{\boldsymbol{k},\boldsymbol{s}}}\big)\big(2^{-\boldsymbol{k}}(\boldsymbol{x}+\boldsymbol{s})\big).$$

Since $\operatorname{supp}\big(T_{\boldsymbol{k}}(f)\chi_{I_{\boldsymbol{k},\boldsymbol{s}}}\big)\subset I_{\boldsymbol{k},\boldsymbol{s}}$ and $\|T_{\boldsymbol{k}}(f)\chi_{I_{\boldsymbol{k},\boldsymbol{s}}}\|_{H^\alpha_\infty(\mathbb{I}^d)}\leqslant 2^d$, we have that

$$\operatorname{supp}\big(T_{\boldsymbol{k},\boldsymbol{s}}(f)\big)\subset\mathbb{I}^d,\quad T_{\boldsymbol{k},\boldsymbol{s}}(f)\in\mathring{U}_\infty^{\alpha,d}.$$

Take the function $S_m(T_{\boldsymbol{k},\boldsymbol{s}}(f))$ defined as in (12) for $T_{\boldsymbol{k},\boldsymbol{s}}(f) \in \mathring{U}_\infty^{\alpha,d}$. By Lemma 6 it holds the estimate

$$\big\|T_{\boldsymbol{k},\boldsymbol{s}}(f) - S_m(T_{\boldsymbol{k},\boldsymbol{s}}(f))\big\|_\infty \leqslant B^d 2^{-\alpha m}\binom{m+d}{d-1}.$$

Define

$$S_{\boldsymbol{k},m}(f)(\boldsymbol{x}) := 2^{-\alpha|\boldsymbol{k}|_1+d}\sum_{\boldsymbol{s}\in Z(\boldsymbol{k})} S_m\big(T_{\boldsymbol{k},\boldsymbol{s}}(f)\big)\big(2^{\boldsymbol{k}}\boldsymbol{x}-\boldsymbol{s}\big). \tag{25}$$

We then get

$$\begin{aligned}\big\|T_{\boldsymbol{k}}(f) - S_{\boldsymbol{k},m}(f)\big\|_\infty &= \Big\|\sum_{\boldsymbol{s}\in Z(\boldsymbol{k})}\Big[T_{\boldsymbol{k}}(f)\chi_{I_{\boldsymbol{k},\boldsymbol{s}}}(\cdot) - 2^{-\alpha|\boldsymbol{k}|_1+d}S_m\big(T_{\boldsymbol{k},\boldsymbol{s}}(f)\big)\big(2^{\boldsymbol{k}}\cdot-\boldsymbol{s}\big)\Big]\Big\|_\infty\\ &= 2^{-\alpha|\boldsymbol{k}|_1+d}\Big\|\sum_{\boldsymbol{s}\in Z(\boldsymbol{k})}\Big[T_{\boldsymbol{k},\boldsymbol{s}}(f) - S_m\big(T_{\boldsymbol{k},\boldsymbol{s}}(f)\big)\Big]\big(2^{\boldsymbol{k}}\cdot-\boldsymbol{s}\big)\Big\|_\infty.\end{aligned}$$

Since support of $T_{\boldsymbol{k},\boldsymbol{s}}(f) - S_m\big(T_{\boldsymbol{k},\boldsymbol{s}}(f)\big)$ is contained in $\mathbb{I}^d$, we finally obtain

$$\big\|T_{\boldsymbol{k}}(f) - S_{\boldsymbol{k},m}(f)\big\|_\infty \leqslant (2B)^d\big(2^m 2^{|\boldsymbol{k}|_1}\big)^{-\alpha}\binom{m+d}{d-1}. \tag{26}$$

Considering $S_{\boldsymbol{k},m}(f)$ as an intermediate approximation of $T_{\boldsymbol{k}}(f)$, we shall construct deep ReLU networks approximating $S_{\boldsymbol{k},m}(f)$. Since $S_{\boldsymbol{k},m}(f)$ is a sum of functions in $\mathcal{S}^{\alpha,d}(m)$, we shall construct a deep ReLU neural network $\Phi_\varepsilon(S)$ for approximating $S \in \mathcal{S}^{\alpha,d}(m)$ with accuracy ε and estimate its size.

Lemma 12. *Let $d \in \mathbb{N}$, $d \geqslant 2$, $m \in \mathbb{N}$, $\alpha \in (0,1]$, and $\varepsilon \in (0,1)$. Then for every $S \in \mathcal{S}^{\alpha,d}(m)$, we can explicitly construct a deep ReLU neural network $\Phi_\varepsilon(S)$ so that* $\operatorname{supp}\Phi_\varepsilon(S) \subset \mathbb{I}^d$ *and*

$$\|S - \Phi_\varepsilon(S)\|_\infty \leqslant \varepsilon. \tag{27}$$

Moreover, there is a positive constant C such that

$$W(\Phi_\varepsilon(S)) \leqslant Cd\log d\, 2^m\binom{m+d-1}{d-1}\log(dB^d\varepsilon^{-1}) \tag{28}$$

and

$$L(\Phi_\varepsilon(S)) \leqslant C2^m\log d\log(dB^d\varepsilon^{-1}), \tag{29}$$

where B is given in Lemma 5.

Proof. By Lemma 6, for every function $S \in \mathcal{S}^{\alpha,d}(m)$, there is a function $f \in \mathring{U}_\infty^{\alpha,d}$ such that

$$S(\boldsymbol{x}) = S_m(f)(\boldsymbol{x}) = \sum_{|\bar{\boldsymbol{k}}_1|_1\leqslant m} 2^{-\alpha(|\bar{\boldsymbol{k}}_1|_1+d-1)}\sum_{\bar{\boldsymbol{s}}_1\in Z(\bar{\boldsymbol{k}}_1)}\varphi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}(\bar{\boldsymbol{x}}_1)S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}(x_1),$$

where $S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1} := S_{K_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}(f)} \in \mathcal{S}^\alpha(m-|\bar{\boldsymbol{k}}_1|_1)$. Since $S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}$ is a piecewise linear continuous function, see (10) and (9), according to [7, Theorem 3.1] we can explicitly construct a deep ReLU neural network $\Phi\big(S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}\big)$ with one-dimensional input so that $\Phi\big(S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}\big)(x_1) = S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}(x_1)$, $x_1 \in \mathbb{I}$, and

$$W\big(\Phi\big(S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}\big)\big) \leqslant C2^{m-|\bar{\boldsymbol{k}}_1|_1}, \quad L\big(\Phi\big(S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}\big)\big) \leqslant C2^{m-|\bar{\boldsymbol{k}}_1|_1}. \tag{30}$$

Each univariate function φ_{k_j,s_j} in the tensor product $\varphi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1} = \bigotimes_{j=2}^{d}\varphi_{k_j,s_j}$ can be expressed as an output of a neural network $\Phi(\varphi_{k_j,s_j})$ with one-dimensional input, deep 3 and 8 weights. Adding layers (with one node in each layer) putting forward x_j to each network $\Phi(\varphi_{k_j,s_j})$ such that it has the length $L\big(\Phi\big(S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}\big)\big)$. We still denote these new networks by $\Phi(\varphi_{k_j,s_j})$. Then we obtain

$$W(\Phi(\varphi_{k_j,s_j})) \leqslant C2^{m-|\bar{\boldsymbol{k}}_1|_1}.$$

We approximate the d-univariate function $\varphi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}(\bar{\boldsymbol{x}}_1)S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}(x_1)$ by the output of the network $\Phi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}$ with d-dimensional input which is explicitly constructed as a concatenation of the networks $\Phi\big(S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}\big)$, $\Phi(\varphi_{k_j,s_j})$, $j = 2,\dots,d$, with product network Φ_P in Lemma 8. With $\delta = \varepsilon B^{1-d}$ in Lemma 8 we have

$$\big\|\varphi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1} - \Phi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}\big\|_\infty \leqslant \varepsilon B^{1-d}. \tag{31}$$

Since $|\varphi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}(\bar{\boldsymbol{x}}_1)| \leqslant 1$ for $\bar{\boldsymbol{x}}_1 \in \mathbb{I}^{d-1}$ and $|S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}(x_1)| \leqslant 4$ for $x_1 \in \mathbb{I}$ by (11), from Lemmata 3, 8 and (30) we derive that

$$\begin{aligned} W(\Phi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}) &\leqslant C\bigg(\sum_{j=2}^{d} W\big(\Phi(\varphi_{k_j,s_j})\big) + W\big(\Phi(S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1})\big) + W(\Phi_P)\bigg) \\ &\leqslant Cd\big(2^{m-|\bar{\boldsymbol{k}}_1|_1} + \log(dB^d\varepsilon^{-1})\big), \end{aligned} \tag{32}$$

and

$$L(\Phi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}) \leqslant L\big(\Phi\big(S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}\big)\big) + L(\Phi_P) \leqslant C\big(2^{m-|\bar{\boldsymbol{k}}_1|_1} + \log d\log(dB^d\varepsilon^{-1})\big). \tag{33}$$

Moreover $\operatorname{supp}(\Phi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}) \subset \operatorname{supp}(\varphi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1})$ by Lemma 8.

Let the network $\Phi_{\bar{\boldsymbol{k}}_1}$ with output

$$\Phi_{\bar{\boldsymbol{k}}_1}(\boldsymbol{x}) = \sum_{\bar{\boldsymbol{s}}_1\in Z(\bar{\boldsymbol{k}}_1)} \Phi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}(\boldsymbol{x})$$

be explicitly constructed as a combination of the networks $\big\{\Phi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}\big\}_{\bar{\boldsymbol{s}}_1\in Z(\bar{\boldsymbol{k}}_1)}$ by the special construction. Then by Lemma 2, (32) and (33) we obtain that

$$\begin{aligned} L(\Phi_{\bar{\boldsymbol{k}}_1}) &\leqslant \sum_{\bar{\boldsymbol{s}}_1\in Z(\bar{\boldsymbol{k}}_1)} L(\Phi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}) \leqslant C2^{|\bar{\boldsymbol{k}}_1|_1}\big(2^{m-|\bar{\boldsymbol{k}}_1|_1} + \log d\log(dB^d\varepsilon^{-1})\big) \\ &\leqslant C2^m \log d\log(dB^d\varepsilon^{-1}) \end{aligned} \tag{34}$$

and

$$\begin{aligned} W(\Phi_{\bar{\boldsymbol{k}}_1}) &\leqslant \sum_{\bar{\boldsymbol{s}}_1\in Z(\bar{\boldsymbol{k}}_1)} W(\Phi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}) + (d+1)L(\Phi_{\bar{\boldsymbol{k}}_1}) \\ &\leqslant Cd2^{|\bar{\boldsymbol{k}}_1|_1}\big(2^{m-|\bar{\boldsymbol{k}}_1|_1} + \log(dB^d\varepsilon^{-1})\big) + C(d\log d)2^m\log(dB^d\varepsilon^{-1}) \\ &\leqslant C(d\log d)2^m\log(dB^d\varepsilon^{-1}). \end{aligned} \tag{35}$$

Since $\boldsymbol{x} \in \mathbb{I}^d$, we can construct a standard network with the same output as $\Phi_{\bar{\boldsymbol{k}}_1}$ and the estimates (34) and (35) hold, see (3) and (4). We still denote this network by $\Phi_{\bar{\boldsymbol{k}}_1}$. Now we define the network $\Phi_\varepsilon(S)$ as a parallelization of the networks $(\Phi_{\bar{\boldsymbol{k}}_1})_{|\bar{\boldsymbol{k}}_1|_1\leqslant m}$ with output

$$\Phi_\varepsilon(S)(\boldsymbol{x}) = \sum_{|\bar{\boldsymbol{k}}_1|_1\leqslant m} 2^{-\alpha(|\bar{\boldsymbol{k}}_1|_1+d-1)}\Phi_{\bar{\boldsymbol{k}}_1}(\boldsymbol{x}).$$

Since $\operatorname{supp}\big(\Phi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}\big) \subset \operatorname{supp}(\varphi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1} S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1})$ and for a given $\bar{\boldsymbol{k}}_1$, $\operatorname{supp}(\varphi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1} S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1})$ and $\operatorname{supp}(\varphi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1'} S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1'})$ are disjoint if $\bar{\boldsymbol{s}}_1' = \bar{\boldsymbol{s}}_1$, it holds by (31) and (19) that

$$\begin{aligned}
\|S - \Phi_\varepsilon(S)\|_\infty &\leqslant \Big\| \sum_{|\bar{\boldsymbol{k}}_1|_1 \leqslant m} \sum_{\bar{\boldsymbol{s}}_1 \in Z(\bar{\boldsymbol{k}}_1)} 2^{-\alpha(|\bar{\boldsymbol{k}}_1|_1+d-1)} |\varphi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1} S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1} - \Phi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}| \Big\|_\infty \\
&= \sum_{|\bar{\boldsymbol{k}}_1|_1 \leqslant m} 2^{-\alpha(|\bar{\boldsymbol{k}}_1|_1+d-1)} \max_{\bar{\boldsymbol{s}}_1 \in Z(\bar{\boldsymbol{k}}_1)} \big\|\varphi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1} S_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1} - \Phi_{\bar{\boldsymbol{k}}_1,\bar{\boldsymbol{s}}_1}\big\|_\infty \\
&\leqslant \sum_{|\bar{\boldsymbol{k}}_1|_1 \leqslant m} 2^{-\alpha(|\bar{\boldsymbol{k}}_1|_1+d-1)} \varepsilon B^{1-d} \\
&= \varepsilon (1-2^{-\alpha})^{d-1} \sum_{\ell=0}^{m} 2^{-\ell\alpha} \binom{\ell+d-2}{d-2} \leqslant \varepsilon.
\end{aligned}$$

By Lemma 1 and (34), (35) we obtain

$$W(\Phi_\varepsilon(S)) \leqslant 3\big|\{\bar{\boldsymbol{k}}_1 : |\bar{\boldsymbol{k}}_1|_1 \leqslant m\}\big| \max_{|\bar{\boldsymbol{k}}_1|_1 \leqslant m} W(\Phi_{\bar{\boldsymbol{k}}_1}) \leqslant C(d\log d) 2^m \binom{m+d-1}{d-1} \log(dB^d\varepsilon^{-1}),$$

and

$$L(\Phi_\varepsilon(S)) \leqslant \max_{|\bar{\boldsymbol{k}}_1|_1 \leqslant m} L(\Phi_{\bar{\boldsymbol{k}}_1}) \leqslant C(\log d) 2^m \log(dB^d\varepsilon^{-1}).$$

Finally, the inclusion $\operatorname{supp}\Phi_\varepsilon(S) \subset \mathbb{I}^d$ follows from Lemmata 8 and 9. ∎

The following result is a generalization of [7, Lemma 5.1] to d-dimensional case.

Lemma 13. *Let $k \in \mathbb{N}$, $\Lambda \subset Z(k)$ and $j \in \{1,\dots,d\}$. Let Φ be a deep ReLU network with input dimension d such that* $\operatorname{supp}\Phi \subset \mathbb{I}^d$. *Denote*

$$f(\boldsymbol{x}) := \sum_{s\in\Lambda} \Phi(x_1,\dots,2^k x_j - s,\dots,x_d),\ \ \boldsymbol{x} \in \mathbb{I}^d.$$

Then we can explicitly construct a deep ReLU network Φ_Λ with output $f(\boldsymbol{x})$ and

$$W(\Phi_\Lambda) \leqslant C(d|\Lambda| + W(\Phi)),\ \ L(\Phi_\Lambda) \leqslant 5 + L(\Phi). \tag{36}$$

Proof. Without loss of generality we assume that $j = 1$.

Set $H_{2^k-1}(t) := \sigma(t - 2^{-k}s)/(1-2^{-k}s)$, $H_{2^k}(t) := 0$ and $H_s := \varphi^*_{k,s+1}$ for $s \in Z(k)\backslash\{2^k-1\}$, where $\varphi^*_{k,s+1}$ is defined as in (6). Let

$$Z_i(k) := \{s \in Z(k) : s = 3r + i,\ r \in \mathbb{N}_0\},\quad i = 0,1,2.$$

To make the proof simple, we divide it into several cases of Λ and $\Phi(\cdot)$.

Case 1. The case $\Lambda \subset Z_i(k)$ for some $i \in \{0,1,2\}$ and $\Phi(\boldsymbol{x}) \geqslant 0$ for all $\boldsymbol{x} \in \mathbb{I}^d$. We will show that

$$f(\boldsymbol{x}) = \sigma\bigg(\Phi\Big(\sum_{s\in\Lambda} H_s(x_1), \bar{\boldsymbol{x}}_1\Big) - \Phi\Big(1 - \sum_{s\in\Lambda} H_{s+1}(x_1), \bar{\boldsymbol{x}}_1\Big)\bigg), \tag{37}$$

for all $\boldsymbol{x} \in \mathbb{I}^d$. Indeed, if $x_1 \notin \cup_{s\in\Lambda}[2^{-k}s, 2^{-k}(s+3)]$ we have $\sum_{s\in\Lambda} H_{s+1}(x_1) = \sum_{s\in\Lambda} H_s(x_1) = 0$. Since $\operatorname{supp}\Phi(\cdot) \subset \mathbb{I}^d$ we get

$$f(\boldsymbol{x}) = 0 = \sigma\big(\Phi\big(0, \bar{\boldsymbol{x}}_1\big) - \Phi\big(1, \bar{\boldsymbol{x}}_1\big)\big).$$

If $x_1 \in [2^{-k}s_0, 2^{-k}(s_0+1)]$ for some $s_0 \in \Lambda$ we have $\sum_{s\in\Lambda} H_{s+1}(x_1) = 0$ and $\sum_{s\in\Lambda} H_s(x_1) = 2^k x_1 - s_0$. Since $\Phi(\boldsymbol{x}) \geqslant 0$ and $\operatorname{supp}\Phi(\cdot) \subset \mathbb{I}^d$ we obtain

$$f(\boldsymbol{x}) = \Phi\big(2^k x_1 - s_0, \bar{\boldsymbol{x}}_1\big) = \sigma\big(\Phi\big(2^k x_1 - s_0, \bar{\boldsymbol{x}}_1\big) - \Phi\big(1, \bar{\boldsymbol{x}}_1\big)\big).$$

If $x_1 \in [2^{-k}(s_0+2), 2^{-k}(s_0+3)]$ for some $s_0 \in \Lambda$ we have $\sum_{s\in\Lambda} H_s(x_1) = 0$. Again from $\Phi(\boldsymbol{x}) \geqslant 0$ and $\operatorname{supp}\Phi(\cdot) \subset \mathbb{I}^d$ we get

$$f(\boldsymbol{x}) = 0 = \sigma\bigg(\Phi\big(0, \bar{\boldsymbol{x}}_1\big) - \Phi\bigg(1 - \sum_{s\in\Lambda} H_{s+1}(x_1), \bar{\boldsymbol{x}}_1\bigg)\bigg).$$

If $x_1 \in [2^{-k}(s_0+1), 2^{-k}(s_0+2)]$, $s_0 \in \Lambda$, it is easy to see that $\sum_{s\in\Lambda} H_s(x_1) = 1 - \sum_{s\in\Lambda} H_{s+1}(x_1)$. Hence, the equality (37) holds. We have

$$H_s(x_1) = \sigma\big(1 - \sigma\big(2^k x_1 - s - 1\big) - \sigma\big(s + 1 - 2^k x_1\big)\big)$$

for $s \in Z(k)\backslash\{2^k - 1\}$ and $H_{2^k-1}(x_1) = \frac{1}{1-2^{-k}s}\sigma(x_1 - 2^{-k}s)$.

Denote the neural networks on the right side by $\Phi(H_s)$. Then the functions $\sum_{s\in\Lambda} H_s(x_1)$ and $1 - \sum_{s\in\Lambda} H_{s+1}(x_1)$ can be realized exactly by two networks Φ_1 and Φ_2 constructed by parallelization of Φ_{H_s}. By Lemma 1, the length of Φ_1 and Φ_2 is 3 and their sizes are bounded $C|\Lambda|$. Since $\Phi_1(x_1) \geqslant 0$ and $\Phi_2(x_1) \geqslant 0$ when $x_1 \in \mathbb{I}$, we can write

$$f(\boldsymbol{x}) = \sigma\big[\Phi\big(\sigma(\Phi_1(x_1)), \sigma(\sigma(\sigma(\bar{\boldsymbol{x}}_1)))\big) - \Phi\big(\sigma(\Phi_2(x_1)), \sigma(\sigma(\sigma(\bar{\boldsymbol{x}}_1)))\big)\big].$$

Therefore, the network Φ_Λ is a concatenation of Φ_1, Φ_2, $\sigma(\sigma(\sigma(\bar{\boldsymbol{x}}_1)))$, and Φ. It is clear that we have the estimate

$$W(\Phi_\Lambda) \leqslant C(d|\Lambda| + W(\Phi)), \quad L(\Phi_\Lambda) \leqslant 4 + L(\Phi).$$

Case 2. The case $\Lambda \subset Z_i(\boldsymbol{k})$ for some $i \in \{0, 1, 2\}$ and $\Phi(\boldsymbol{x})$ changing sign when $\boldsymbol{x} \in \mathbb{I}^d$. In this case, we write $\Phi(\boldsymbol{x}) = \sigma\big(\Phi(\boldsymbol{x})\big) - \sigma\big(-\Phi(\boldsymbol{x})\big)$. Hence

$$f(\boldsymbol{x}) := \sum_{s\in\Lambda} \sigma\big(\Phi(2^k x_1 - s, \bar{\boldsymbol{x}}_1)\big) - \sum_{s\in\Lambda} \sigma\big(-\Phi(2^k x_1 - s, \bar{\boldsymbol{x}}_1)\big), \quad \boldsymbol{x} \in \mathbb{I}^d.$$

Applying the construction in Case 1 for each sum on the right side with Φ replaced by $\mathrm{Id}(\sigma(\Phi(\cdot)))$ and $\mathrm{Id}(\sigma(-\Phi(\cdot)))$ respectively we obtain two neural networks Φ_Λ^+ and Φ_Λ^-. Here Id is the identity operator. Concatenating these two network by parallelization, see Lemma 1, we obtain Φ_Λ. Note that

$$W\big(\mathrm{Id}(\sigma(\Phi(\cdot)))\big) = W\big(\mathrm{Id}(\sigma(-\Phi(\cdot)))\big) = W(\Phi) + 1$$

and

$$L\big(\mathrm{Id}(\sigma(\Phi(\cdot)))\big) = L\big(\mathrm{Id}(\sigma(-\Phi(\cdot)))\big) = L(\Phi) + 1.$$

Therefore, the estimates (36) still hold true.

Case 3. General case. We rewrite f in the form:

$$f(\boldsymbol{x}) = \sum_{j=0,1,2} \sum_{s\in\Lambda\cap Z_j(k)} \Phi(2^k x_1 - s, \bar{\boldsymbol{x}}_1).$$

To construct the network Φ_Λ, we first construct the network Φ_{Λ_j}, $j = 0, 1, 2$, by using the procedure in Case 2 to have that

$$\Phi_{\Lambda_j}(\boldsymbol{x}) = \sum_{s\in\Lambda\cap Z_j(k)} \Phi(2^k x_1 - s, \bar{\boldsymbol{x}}_1).$$

Then by parallelizing $(\Phi_{\Lambda_j})_{j=0,1,2}$ we obtain the network Φ_Λ. From Lemma 1 we prove (36). ∎

Lemma 14. *Let $d, m \in \mathbb{N}$, $d \geqslant 2$, $\boldsymbol{k} \in \mathbb{N}^d$, $\alpha \in (0, 1]$ and $\varepsilon \in (0, 1)$. Assume that $\Phi_\varepsilon(S)$ is the neural network constructed in Lemma 12 to approximate $S \in \mathcal{S}^{\alpha,d}(m)$ with accuracy ε and computation complexity as in* (27) *and* (28), (29). *Then for every $f \in \mathring{U}_\infty^{\alpha,d}$ we can explicitly construct a deep ReLU neural network $\Phi_\varepsilon\big(S_{\boldsymbol{k},m}(f)\big)$ so that*

$$\big\|\Phi_\varepsilon\big(S_{\boldsymbol{k},m}(f)\big) - S_{\boldsymbol{k},m}(f)\big\|_\infty \leqslant 2^{-\alpha|\boldsymbol{k}|_1+d}\varepsilon\,. \tag{38}$$

Moreover,

$$W(\Phi_\varepsilon\big(S_{\boldsymbol{k},m}(f)\big)) \leqslant Cd\bigg(2^{|\boldsymbol{k}|_1} + \log d 2^{|\boldsymbol{k}|_1-|\boldsymbol{k}|_\infty} N_d(m) 2^m \binom{m+d-1}{d-1} \log(dB^d\varepsilon^{-1})\bigg), \tag{39}$$

and

$$L\big(\Phi_\varepsilon\big(S_{\boldsymbol{k},m}(f)\big)\big) \leqslant C \log d\, N_d(m) 2^m \log(dB^d\varepsilon^{-1}), \tag{40}$$

where $N_d(m)$ is given in Lemma 6 and B is given in Lemma 5.

Proof. We can assume without loss of generality that $k_1 = |\boldsymbol{k}|_\infty$. By the definition (25), for $f \in \mathring{U}_\infty^{\alpha,d}$ we have that

$$S_{\boldsymbol{k},m}(f)(\boldsymbol{x}) := 2^{-\alpha|\boldsymbol{k}|_1+d} \sum_{\bar{\boldsymbol{s}}_1\in Z(\bar{\boldsymbol{k}}_1)} \sum_{s_1\in Z(k_1)} S_m\big(T_{\boldsymbol{k},\boldsymbol{s}}(f)\big)\big(2^{\boldsymbol{k}}\boldsymbol{x} - \boldsymbol{s}\big).$$

We number the elements of the set $\mathcal{S}^{\alpha,d}(m)$ from 1 to $N_d(m)$ as $S_1, \dots, S_{N_d(m)}$. For $\bar{\boldsymbol{s}}_1 \in Z(\bar{\boldsymbol{k}}_1)$ and $\eta = 1, \dots, N_d(m)$, we define

$$\Lambda_\eta(\bar{\boldsymbol{s}}_1) := \big\{s_1 \in Z(k_1) : S_m\big(T_{\boldsymbol{k},\boldsymbol{s}}(f)\big) = S_\eta \in \mathcal{S}^{\alpha,d}(m)\big\}.$$

Hence, we can write

$$S_{\boldsymbol{k},m}(f)(\boldsymbol{x}) = 2^{-\alpha|\boldsymbol{k}|_1+d} \sum_{\bar{\boldsymbol{s}}_1\in Z(\bar{\boldsymbol{k}}_1)} \sum_{\eta=1}^{N_d(m)} \sum_{s_1\in\Lambda_\eta(\bar{\boldsymbol{s}}_1)} S_\eta\big(2^{\boldsymbol{k}}\boldsymbol{x} - \boldsymbol{s}\big).$$

To approximate $S_{\boldsymbol{k},m}(f)$ we use the output

$$\Phi_\varepsilon\big(S_{\boldsymbol{k},m}(f)\big)(\boldsymbol{x}) := 2^{-\alpha|\boldsymbol{k}|_1+d} \sum_{\bar{\boldsymbol{s}}_1\in Z(\bar{\boldsymbol{k}}_1)} \sum_{\eta=1}^{N_d(m)} \sum_{s_1\in\Lambda_\eta(\bar{\boldsymbol{s}}_1)} \Phi_\varepsilon(S_\eta)\big(2^{\boldsymbol{k}}\boldsymbol{x} - \boldsymbol{s}\big) \tag{41}$$

of a deep ReLU neural network $\Phi_\varepsilon\big(S_{\boldsymbol{k},m}(f)\big)$. Let us first show explicitly how to construct such a network $\Phi_\varepsilon\big(S_{\boldsymbol{k},m}(f)\big)$ and then estimate its size and depth. Denote by $\Phi_{S_\eta,\bar{\boldsymbol{s}}_1}$ the network constructed by adding a layer of d nodes before the input layer of $\Phi_\varepsilon(S_\eta)$. Computations at nodes in the first layer of $\Phi_{S_\eta,\bar{\boldsymbol{s}}_1}$ are $\sigma(x_1)$ and $\sigma(2^{k_j}x_j - s_j)$, $j = 2,\dots,d$. Then by (28) and (29) we have

$$W(\Phi_{S_\eta,\bar{\boldsymbol{s}}_1}) \leqslant 2(d-1) + 1 + W(\Phi_\varepsilon(S_\eta)) \leqslant CW(\Phi_\varepsilon(S_\eta))$$

and

$$L(\Phi_{S_\eta,\bar{\boldsymbol{s}}_1}) \leqslant 1 + L(\Phi_\varepsilon(S_\eta))\,.$$

Since $\operatorname{supp}\Phi_\varepsilon(S_\eta) \subset \mathbb{I}^d$, we have $\Phi_{S_\eta,\bar{\boldsymbol{s}}_1}(\boldsymbol{x}) = \Phi_\varepsilon(S_\eta)\big(x_1, 2^{\bar{\boldsymbol{k}}_1}\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{s}}_1\big)$. Hence we can write

$$\Phi_\varepsilon\big(S_{\boldsymbol{k},m}(f)\big)(\boldsymbol{x}) = 2^{d-\alpha|\boldsymbol{k}|_1} \sum_{\bar{\boldsymbol{s}}_1 \in Z(\bar{\boldsymbol{k}}_1)} \sum_{\eta=1}^{N_d(m)} \sum_{s_1 \in \Lambda_\eta(\bar{\boldsymbol{s}}_1)} \Phi_{S_\eta,\bar{\boldsymbol{s}}_1}\big(2^{k_1}x_1 - s_1, \bar{\boldsymbol{x}}_1\big).$$

Applying Lemma 13 to the function $\sum_{s_1\in\Lambda_\eta(\bar{\boldsymbol{s}}_1)} \Phi_{S_\eta,\bar{\boldsymbol{s}}_1}\big(2^{k_1}x_1 - s_1, \bar{\boldsymbol{x}}_1\big)$, we can explicitly construct a network $\Phi_{\Lambda_\eta(\bar{\boldsymbol{s}}_1)}$ with the output

$$\Phi_{\Lambda_\eta(\bar{\boldsymbol{s}}_1)}(\boldsymbol{x}) = \sum_{s_1\in\Lambda_\eta(\bar{\boldsymbol{s}}_1)} \Phi_{S_\eta,\bar{\boldsymbol{s}}_1}\big(2^{k_1}x_1 - s_1, \bar{\boldsymbol{x}}_1\big),$$

so that its size and depth satisfy

$$W\big(\Phi_{\Lambda_\eta(\bar{\boldsymbol{s}}_1)}\big) \leqslant C\big(d|\Lambda_\eta(\bar{\boldsymbol{s}}_1)| + W(\Phi_{S_\eta,\bar{\boldsymbol{s}}_1})\big) \leqslant C\big(d|\Lambda_\eta(\bar{\boldsymbol{s}}_1)| + W(\Phi_\varepsilon(S_\eta))\big)$$

and

$$L(\Phi_{\Lambda_\eta(\bar{\boldsymbol{s}}_1)}) \leqslant CL(\Phi_{S_\eta,\bar{\boldsymbol{s}}_1}) \leqslant CL(\Phi_\varepsilon(S_\eta)).$$

Let $\Phi_{\bar{\boldsymbol{s}}_1}$ be the special network combining $(\Phi_{\Lambda_\eta(\bar{\boldsymbol{s}}_1)})_{\eta=1,\dots,N_d(m)}$ with output

$$\Phi_{\bar{\boldsymbol{s}}_1}(\boldsymbol{x}) = \sum_{\eta=1}^{N_d(m)} \Phi_{\Lambda_\eta(\bar{\boldsymbol{s}}_1)}(\boldsymbol{x}).$$

By Lemmata 2 and 12 its length is bounded as

$$L(\Phi_{\bar{\boldsymbol{s}}_1}) \leqslant \sum_{\eta=1}^{N_d(m)} L\big(\Phi_{\Lambda_\eta(\bar{\boldsymbol{s}}_1)}\big) \leqslant C \sum_{\eta=1}^{N_d(m)} L(\Phi_\varepsilon(S_\eta)) \leqslant C\log dN_d(m)2^m \log(dB^d\varepsilon^{-1})$$

and its size

$$\begin{aligned}
W(\Phi_{\bar{\boldsymbol{s}}_1}) &\leqslant \sum_{\eta=1}^{N_d(m)} W\big(\Phi_{\Lambda_\eta(\bar{\boldsymbol{s}}_1)}\big) + (d+1)L(\Phi_{\bar{\boldsymbol{s}}_1}) \\
&\leqslant \sum_{\eta=1}^{N_d(m)} C\Big(d|\Lambda_\eta(\bar{\boldsymbol{s}}_1)| + W(\Phi_\varepsilon(S_\eta))\Big) + (d+1)L(\Phi_{\bar{\boldsymbol{s}}_1}) \\
&\leqslant C\left(\sum_{\eta=1}^{N_d(m)} d|\Lambda_\eta(\bar{\boldsymbol{s}}_1)| + (d\log d)N_d(m)2^m\log(dB^d\varepsilon^{-1})\binom{m+d-1}{d-1}\right) \\
&\leqslant Cd\left(2^{k_1} + (\log d)N_d(m)2^m\binom{m+d-1}{d-1}\log(dB^d\varepsilon^{-1})\right).
\end{aligned}$$

Since $\boldsymbol{x} \in \mathbb{I}^d$, the network $\Phi_{\bar{\boldsymbol{s}}_1}$ can be transformed to a standard ReLU neural network with the same output and estimation for depth and size (by adjusting the constants), see (3) and (4). We still denote this new network by $\Phi_{\bar{\boldsymbol{s}}_1}$.

The network $\Phi_\varepsilon\big(S_{\boldsymbol{k},m}(f)\big)$ is a parallelization of $\big(\Phi_{\bar{\boldsymbol{s}}_1}\big)_{\bar{\boldsymbol{s}}_1 \in \bar{\boldsymbol{k}}_1}$ which has output (41) and by Lemma 1

$$\begin{aligned} W\big(\Phi_\varepsilon\big(S_{\boldsymbol{k},m}(f)\big)\big) &\leqslant 2 \cdot 2^{|\bar{\boldsymbol{k}}_1|_1} \max_{\bar{\boldsymbol{s}}_1 \in \bar{\boldsymbol{k}}_1} W(\Phi_{\bar{\boldsymbol{s}}_1}) \\ &\leqslant C d 2^{|\bar{\boldsymbol{k}}_1|_1} \left(2^{k_1} + \log d N_d(m) 2^m \binom{m+d-1}{d-1} \log(dB^d \varepsilon^{-1})\right) \end{aligned}$$

and

$$L\big(\Phi_\varepsilon\big(S_{\boldsymbol{k},m}(f)\big)\big) \leqslant \max_{\bar{\boldsymbol{s}}_1 \in Z(\bar{\boldsymbol{k}}_1)} L(\Phi_{\bar{\boldsymbol{s}}_1}) \leqslant C \log d\, N_d(m) 2^m \log(dB^d \varepsilon^{-1}).$$

Thus, (39) and (40) have been proven. Next, we prove the estimate of the approximation error (38). Notice that by the assumptions of the lemma and Lemma 12 $\operatorname{supp} S \subset \mathbb{I}^d$ and $\operatorname{supp} \Phi_\varepsilon(S) \subset \mathbb{I}^d$ for all $S \in \mathcal{S}^{\alpha,d}(m)$, and it holds the estimate (27). Moreover, for different pairs $(\boldsymbol{s}, \eta)$ and $(\boldsymbol{s}', \eta')$, the supports of the functions $(\Phi_\varepsilon(S_\eta) - S_\eta)\big(2^{\boldsymbol{k}} \cdot - \boldsymbol{s}\big)$ and $(\Phi_\varepsilon(S_{\eta'}) - S_{\eta'})\big(2^{\boldsymbol{k}} \cdot - \boldsymbol{s}'\big)$ are disjoint. Hence, by (27) we obtain

$$\begin{aligned} \big\|\Phi_\varepsilon\big(S_{\boldsymbol{k},m}(f)\big) - S_{\boldsymbol{k},m}(f)\big\|_\infty &= 2^{d-\alpha|\boldsymbol{k}|_1} \left\| \sum_{\bar{\boldsymbol{s}}_1 \in Z(\bar{\boldsymbol{k}}_1)} \sum_{\eta=1}^{N_d(m)} \sum_{s_1 \in \Lambda_\eta(\bar{\boldsymbol{s}}_1)} (\Phi_\varepsilon(S_\eta) - S_\eta)\big(2^{\boldsymbol{k}} \cdot - \boldsymbol{s}\big) \right\|_\infty \\ &= 2^{d-\alpha|\boldsymbol{k}|_1} \max_{\bar{\boldsymbol{s}}_1 \in Z(\bar{\boldsymbol{k}}_1)} \max_{1 \leqslant \eta \leqslant N_d(m)} \max_{s_1 \in \Lambda_\eta(\bar{\boldsymbol{s}}_1)} \big\|(\Phi_\varepsilon(S_\eta) - S_\eta)\big(2^{\boldsymbol{k}} \cdot - \boldsymbol{s}\big)\big\|_\infty \\ &\leqslant 2^{d-\alpha|\boldsymbol{k}|_1} \varepsilon \end{aligned}$$

which proves (38). ∎

We are now in position to prove Theorem 11.

Proof. For convenience, we divide the proof into several steps.

Step 1. Let us recall our plan of the proof. To approximate $f \in \mathring{U}_\infty^{\alpha,d}$, we will construct a deep ReLU neural network with an output of the form

$$\Phi_\varepsilon(f) \;=\; \Phi_{\varepsilon/2}\big(R_n(f)\big) \;+\; \Phi_{\varepsilon/2}\big(f - R_n(f)\big), \tag{42}$$

where $\Phi_{\varepsilon/2}\big(R_n(f)\big)$ and $\Phi_{\varepsilon/2}\big(f - R_n(f)\big)$ are deep ReLU neural networks approximating $R_n(f)$ and $f - R_n(f)$ with accuracy $\varepsilon/2$, respectively. Then we have

$$\|f - \Phi_\varepsilon(f)\|_\infty \leqslant \big\|R_n(f) - \Phi_{\varepsilon/2}\big(R_n(f)\big)\big\|_\infty + \big\|\big(f - R_n(f)\big) - \Phi_{\varepsilon/2}\big(f - R_n(f)\big)\big\|_\infty \leqslant \varepsilon. \tag{43}$$

For approximation of the first term $R_n(f)$, we take the deep ReLU neural network $\Phi_{\varepsilon/2}\big(R_n(f)\big)$ which has been constructed in Lemma 10. In the following we construct a deep ReLU neural network $\Phi_{\varepsilon/2}\big(f - R_n(f)\big)$ for approximating $f - R_n(f)$ with accuracy $\varepsilon/2$.

As noticed above, since the difference $f - R_n(f)$ is represented as in (23), we shall explicitly construct deep ReLU neural networks $\Phi_{\varepsilon'}(F_{\boldsymbol{k}_j})$ to approximate each term $F_{\boldsymbol{k}_j}$ with accuracy ε' in the sum in (23), where the value of ε' will be chosen latter. For ease of

notation we consider the case $\operatorname{supp}(\boldsymbol{k}_j) = j$ with $1 \leqslant j \leqslant d-1$. The other cases are carried out similarly with a slight modification. From (24) we have

$$\begin{aligned} F_{\boldsymbol{k}_j} &= \prod_{i=1}^{j} \big(T_{(k_i-1)\boldsymbol{e}^j} - T_{k_i\boldsymbol{e}^j}\big) T_{(n+1-|\boldsymbol{k}_j|_1)\boldsymbol{e}^{j+1}}(f) \\ &= \sum_{\boldsymbol{e}\in\{0,1\}^j} (-1)^{|\boldsymbol{e}|_1} T_{\boldsymbol{k}_j-\boldsymbol{e}} T_{(n+1-|\boldsymbol{k}_j|_1)\boldsymbol{e}^{j+1}}(f) = \sum_{\boldsymbol{\ell}\in\Lambda(\boldsymbol{k}_j)} c_{\boldsymbol{\ell}} T_{\boldsymbol{\ell}}(f)\,, \end{aligned}$$

where

$$\Lambda(\boldsymbol{k}_j) := \Big\{\boldsymbol{\ell}\in\mathbb{N}_0^d,\ \operatorname{supp}(\boldsymbol{\ell})\subset\{1,\dots,j+1\}, \boldsymbol{\ell}_j = \boldsymbol{k}_j - \boldsymbol{e}, \ell_{j+1} = n+1-|\boldsymbol{k}_j|_1, \boldsymbol{e}\in\{0,1\}^j\Big\}$$

and $c_{\boldsymbol{\ell}}$ is either 1 or -1. It is easy to see that $|\Lambda(\boldsymbol{k}_j)| \leqslant 2^j$ for all $\boldsymbol{k}_j$ and if $\boldsymbol{\ell}\in\Lambda(\boldsymbol{k}_j)$ then $n+1-d \leqslant |\boldsymbol{\ell}|_1 \leqslant n+1$.

We approximate $F_{\boldsymbol{k}_j}$ by the output

$$\Phi_{\varepsilon'}\big(F_{\boldsymbol{k}_j}\big)(\boldsymbol{x}) := \sum_{\boldsymbol{\ell}\in\Lambda(\boldsymbol{k}_j)} c_{\boldsymbol{\ell}} \Phi_{\varepsilon'}(S_{\boldsymbol{\ell},m}(f)),$$

where the networks $\Phi_{\varepsilon'}(S_{\boldsymbol{\ell},m}(f))$ are constructed as in Lemma 14. The network $\Phi_{\varepsilon'}\big(F_{\boldsymbol{k}_j}\big)$ is a parallelization of $\Phi_{\varepsilon'}(S_{\boldsymbol{\ell},m}(f))$, $\boldsymbol{\ell}\in\Lambda(\boldsymbol{k}_j)$.

We define $\Phi_{\varepsilon/2}\big(f - R_n(f)\big)$ as a deep ReLU neural network with the output

$$\Phi_{\varepsilon/2}\big(f - R_n(f)\big) := \sum_{j=0}^{d-1} \sum_{|\boldsymbol{k}_j|_1\leqslant n} \Phi_{\varepsilon'}\big(F_{\boldsymbol{k}_j}\big)(\boldsymbol{x}) \tag{44}$$

which is a parallelization of $\Phi_{\varepsilon'}\big(F_{\boldsymbol{k}_j}\big)$, $|\boldsymbol{k}_j|_1 \leqslant n$, $j = 0,\dots,d-1$. It approximates $f - R_n(f)$ with accuracy $\varepsilon/2$ by an appropriate choice of ε'.

We put

$$\varepsilon' = B^d 2^{-\alpha m} \binom{m+d}{d-1}$$

with m will be chosen later such that $\varepsilon'\in(0,1)$. We have from (26) and Lemma 14

$$\begin{aligned} \big\|F_{\boldsymbol{k}_j} - \Phi_{\varepsilon'}(F_{\boldsymbol{k}_j})\big\|_\infty &\leqslant \sum_{\boldsymbol{\ell}\in\Lambda(\boldsymbol{k}_j)} \Big(\big\|T_{\boldsymbol{\ell}}(f) - S_{\boldsymbol{\ell},m}(f)\big\|_\infty + \big\|\Phi_{\varepsilon'}\big(S_{\boldsymbol{\ell},m}(f)\big) - S_{\boldsymbol{\ell},m}(f)\big\|_\infty\Big) \\ &\leqslant \sum_{\boldsymbol{\ell}\in\Lambda(\boldsymbol{k}_j)} \left((2B)^d \big(2^m 2^{|\boldsymbol{\ell}|_1}\big)^{-\alpha} \binom{m+d}{d-1} + 2^{-\alpha|\boldsymbol{\ell}|_1+d}\varepsilon'\right) \\ &\leqslant \sum_{\boldsymbol{\ell}\in\Lambda(\boldsymbol{k}_j)} \left((2B)^d \big(2^m 2^{n+1-d}\big)^{-\alpha} \binom{m+d}{d-1} + 2^{-\alpha(n+1-d)+d}\varepsilon'\right) \\ &\leqslant 2^{j+1-\alpha}(2^{\alpha+1}B)^d \big(2^m 2^n\big)^{-\alpha} \binom{m+d}{d-1}. \end{aligned}$$

This leads to

$$\begin{aligned}\big\|\big(f-R_n(f)\big)-\Phi_{\varepsilon/2}\big(f-R_n(f)\big)\big\|_\infty &\leqslant \sum_{j=0}^{d-1}\sum_{|\boldsymbol{k}_j|_1\leqslant n}\big\|\Phi_{\varepsilon'}\big(F_{\boldsymbol{k}_j}\big)-F_{\boldsymbol{k}_j}\big\|_\infty\\ &\leqslant \sum_{j=0}^{d-1}\sum_{|\boldsymbol{k}_j|_1\leqslant n} 2^{j+1-\alpha}(2^{\alpha+1}B)^d\big(2^m2^n\big)^{-\alpha}\binom{m+d}{d-1}\\ &\leqslant \sum_{j=0}^{d-1}2^j\binom{n+j}{j}2^{1-\alpha}(2^{\alpha+1}B)^d\big(2^m2^n\big)^{-\alpha}\binom{m+d}{d-1}\\ &\leqslant 2^{1-\alpha}(2^{\alpha+2}B)^d\big(2^m2^n\big)^{-\alpha}\binom{m+d}{d-1}\binom{n+d-1}{d-1}.\end{aligned}$$

We denote the last term by $A_{n,m}$. In the next step, our task is to choose n, m (and therefore, ε') depending on ε such that $A_{n,m}\leqslant\varepsilon/2$. Then we define the deep ReLU neural network $\Phi_\varepsilon(f)$ as a parallelization of the networks $\Phi_{\varepsilon/2}\big(R_n(f)\big)$ and $\Phi_{\varepsilon/2}(f-R_n(f))$ with the output (42). From this (43) follows. The size and depth of $\Phi_\varepsilon(f)$ are estimated explicitly in d and ε from the estimation of sizes and depths of $\Phi_{\varepsilon/2}\big(R_n(f)\big)$ and $\Phi_{\varepsilon'}(F_{\boldsymbol{k}_j})$ by the choice of m, n.

Step 2. The choices of ε_0 and n, m. Define $m_0\geqslant d$ as the smallest integer such that $B^d2^{-\alpha m_0}\binom{m_0+d}{d-1}<1$. Denote $n_0\in\mathbb{N}$ from which the function

$$h(n):=K_{d,\alpha}2^{-\alpha n}n^{d-1-\alpha}(\log n)^{(\alpha+1)(d-1)}, \tag{45}$$

where

$$K_{d,\alpha}:=2(2^{\alpha+2}B)^d(4d\log 3)^\alpha\bigg(\frac{2^{d-1}}{(d-1)!}\bigg)^{\alpha+2} \tag{46}$$

is decreasing and $h(n-1)\leqslant 2^{-\alpha n/2}$ for all $n\geqslant n_0$. We put $n_1=\lfloor(8d\log 3)2^{m_0}\binom{m_0+d-1}{d-1}\rfloor+1$ and define $\varepsilon_0=\min\{h(n_0),h(n_1),1/2\}$. For $\varepsilon\in(0,\varepsilon_0)$ we choose $n\in\mathbb{N}$, $n\geqslant\max\{n_0,n_1\}$, such that $h(n)\leqslant\varepsilon/2<h(n-1)$ and then m such that

$$(\log d)3^{2^{m+1}\binom{m+d-1}{d-1}}2^m\binom{m+d-1}{d-1}m\leqslant 2^{\frac{n}{d}}<(\log d)3^{2^{m+2}\binom{m+d}{d-1}}2^{m+1}\binom{m+d}{d-1}(m+1). \tag{47}$$

These choices imply

$$3^{2^{m+1}\binom{m+d-1}{d-1}}\leqslant 2^{\frac{n}{d}}<3^{2^{m+3}\binom{m+d}{d-1}}$$

and

$$2^{m+1}\binom{m+d-1}{d-1}\log 3<\frac{n}{d}<(8\log 3)2^m\binom{m+d}{d-1}\quad\text{and}\quad m\leqslant\log n. \tag{48}$$

Since $n\geqslant(8d\log 3)2^{m_0}\binom{m_0+d}{d-1}$ we get $m\geqslant m_0\geqslant d$ and

$$\begin{aligned}A_{n,m}&\leqslant 2^{1-\alpha}(2^{\alpha+2}B)^d2^{-n\alpha}\bigg[dn^{-1}(8\log 3)\binom{m+d}{d-1}\bigg]^\alpha\binom{m+d}{d-1}\binom{n+d-1}{d-1}\\ &\leqslant 2(2^{\alpha+2}B)^d(4d\log 3)^\alpha\bigg(\frac{2^{d-1}}{(d-1)!}\bigg)^{\alpha+2}2^{-\alpha n}n^{d-1-\alpha}m^{(\alpha+1)(d-1)}\\ &=h(n)\leqslant\frac{\varepsilon}{2}.\end{aligned}$$

Step 3. Estimating the size and depth of $\Phi_{\varepsilon'}(S_{\boldsymbol{\ell},m}(f))$. From $n+1-d\leqslant|\boldsymbol{\ell}|_1\leqslant n+1$ we have

$$|\boldsymbol{\ell}|_1 - |\boldsymbol{\ell}|_\infty \leqslant n + 1 - \frac{n+1-d}{d} \leqslant n - \frac{n}{d} + 2$$

which by (39) leads to

$$W\big(\Phi_{\varepsilon'}(S_{\boldsymbol{\ell},m}(f))\big) \leqslant Cd\bigg(2^n + (\log d)2^{n-\frac{n}{d}+m}\binom{m+d-1}{d-1}\log(dB^d\varepsilon'^{-1})N_d(m)\bigg).$$

Note that by the choice of ε' we get

$$\log(dB^d\varepsilon'^{-1}) \leqslant \log\left(d2^{\alpha m}\binom{m+d}{d-1}^{-1}\right) \leqslant \alpha m.$$

It yields from (47)

$$(\log d)2^m\binom{m+d-1}{d-1}mN_d(m) \leqslant 3^{2^{m+1}\binom{m+d-1}{d-1}}2^m\binom{m+d-1}{d-1}m \leqslant 2^{\frac{n}{d}}.$$

Consequently

$$W\big(\Phi_{\varepsilon'}(S_{\boldsymbol{\ell},m}(f))\big) \leqslant C_\alpha d2^n. \tag{49}$$

Similarly, we have

$$L\big(\Phi_{\varepsilon'}(S_{\boldsymbol{\ell},m}(f))\big) \leqslant C\log d\, N_d(m)2^m\log(dB^d\varepsilon'^{-1}) \leqslant C_\alpha(\log d)3^{2^{m+1}\binom{m+d-1}{d-1}}2^m m \leqslant C_\alpha 2^{\frac{n}{d}}.$$

Step 4. Estimation of the size and depth of $\Phi_\varepsilon(f)$. We recall that $\Phi_{\varepsilon/2}(f - R_n(f))$ is the network obtained by parallelization of $\Phi_{\varepsilon'}(S_{\boldsymbol{\ell},m}(f))$ with $\boldsymbol{\ell}$ in the multi-set

$$\Lambda = \big\{\boldsymbol{\ell} \in \Lambda(\boldsymbol{k}_j),\ j = 0,\dots,d-1,\ |\boldsymbol{k}_j|_1 \leqslant n\big\}$$

and has the output equal to the double sum on the right side of (44). We have

$$|\Lambda| \leqslant \sum_{j=0}^{d-1}\sum_{|\boldsymbol{k}_j|_1\leqslant n} 2^j = \sum_{j=0}^{d-1} 2^j\binom{n+j}{j} \leqslant 2^d\binom{n+d-1}{d-1}.$$

The network $\Phi_\varepsilon(f)$ is a parallelization of $\Phi_{\varepsilon/2}(R_n(f))$ and $\Phi_{\varepsilon/2}(f - R_n(f))$. Therefore, by Lemma 1 and the construction of $\Phi_{\varepsilon/2}(R_n(f))$ and $\Phi_{\varepsilon/2}(f - R_n(f))$ we obtain

$$\begin{aligned} W(\Phi_\varepsilon(f)) &\leqslant C\max\big\{W(\Phi_{\varepsilon/2}(R_n(f))), W(\Phi_{\varepsilon/2}(f - R_n(f))\big\} \\ &\leqslant C\max\left\{W(\Phi_{\varepsilon/2}(R_n(f)))\,, 2^d\binom{n+d-1}{d-1}\max_{\boldsymbol{\ell}\in\Lambda} W(\Phi_{\varepsilon'}(S_{\boldsymbol{\ell},m}(f)))\right\}. \end{aligned}$$

From Lemma 10 and (49) we deduce that

$$W(\Phi_\varepsilon(f)) \leqslant C_\alpha\max\left\{d2^n\log(dB^d2\varepsilon^{-1})\binom{n+d-1}{d-1}, d2^d2^n\binom{n+d-1}{d-1}\right\}.$$

Since $h(n) \leqslant \varepsilon/2$, from (45) by simple calculation we get

$$\log(dB^d\varepsilon^{-1}) \leqslant C_\alpha(d+n) \leqslant C_\alpha dn$$

which implies

$$W(\Phi_\varepsilon(f)) \leqslant C_\alpha d2^d2^n n^d\frac{2^{d-1}}{(d-1)!}\,.$$

From $\varepsilon/2 \leqslant h(n-1) \leqslant 2^{-\alpha n/2}$ we deduce $n \leqslant 2\alpha^{-1}\log(2\varepsilon^{-1})$ and by (45)

$$2^n \leqslant C_\alpha\Big(K_{d,\alpha}2\varepsilon^{-1}n^{d-1-\alpha}(\log n)^{(\alpha+1)(d-1)}\Big)^{\frac{1}{\alpha}}. \tag{50}$$

Consequently

$$W(\Phi_\varepsilon(f)) \leqslant C_\alpha d 2^d \frac{2^{d-1}}{(d-1)!}\Big(K_{d,\alpha}\varepsilon^{-1}(n\log n)^{(\alpha+1)(d-1)}\Big)^{\frac{1}{\alpha}}$$
$$\leqslant C_\alpha d 2^d \frac{2^{d-1}}{(d-1)!}(K_{d,\alpha})^{\frac{1}{\alpha}}\varepsilon^{-\frac{1}{\alpha}}\Big(\log\big(2\alpha^{-1}\log(2\varepsilon^{-1})\big)2\alpha^{-1}\log(2\varepsilon^{-1})\Big)^{(1+\frac{1}{\alpha})(d-1)}.$$

We use the inequalities with $p := (1+\frac{1}{\alpha})(d-1) \geqslant 1$, $\varepsilon \in (0,1/2)$

$$\begin{aligned}\big[\log\big(2\alpha^{-1}\log(2\varepsilon^{-1})\big)\big]^p &= \big[\log(2\alpha^{-1}) + \log\log(2\varepsilon^{-1})\big]^p \\ &\leqslant \big[2\log\big(2\alpha^{-1}\big)\big(\log\log(2\varepsilon^{-1})\big)\big]^p\end{aligned} \tag{51}$$

to obtain

$$W(\Phi_\varepsilon(f)) \leqslant C_\alpha d 2^d \frac{2^{d-1}}{(d-1)!}(K_{d,\alpha})^{\frac{1}{\alpha}}\varepsilon^{-\frac{1}{\alpha}}\Big(4\alpha^{-1}\log(2\alpha^{-1})\log\log(2\varepsilon^{-1})\big)\log(2\varepsilon^{-1})\Big)^{(1+\frac{1}{\alpha})(d-1)}.$$

Replacing $K_{d,\alpha}$ by the right-hand side of (46), we prove (21).

Now we estimate the depth of $\Phi_\varepsilon(f)$. By applying Lemmata 2, 10 and by the construction of $\Phi_{\varepsilon/2}(f - R_n(f))$ we have that

$$\begin{aligned}L(\Phi_\varepsilon(f)) &= C\max\big\{L\big(\Phi_{\varepsilon/2}\big(R_n(f)\big)\big), L\big(\Phi_{\varepsilon/2}(f-R_n(f))\big)\big\} \\ &\leqslant C\max\big\{\log d\log(dB^d(\varepsilon/2)^{-1}), \max_{\boldsymbol{\ell}\in\Lambda} L\big(\Phi_{\varepsilon'}(S_{\boldsymbol{\ell},m}(f))\big)\big\} \\ &\leqslant C_\alpha\max\big\{\log d\log(dB^d\varepsilon^{-1}), 2^{\frac{n}{d}}\big\} \\ &\leqslant C_\alpha\max\big\{(d\log d)n, 2^{\frac{n}{d}}\big\}.\end{aligned}$$

By (47) and (48) it is easily seen that $2^{\frac{n}{d}}$ dominates $(d\log d)n$. From (50), the inequality $n \leqslant 2\alpha^{-1}\log(2\varepsilon^{-1})$, and (51) we get

$$\begin{aligned}2^{\frac{n}{d}} &\leqslant C_\alpha\Big(K_{d,\alpha}\varepsilon^{-1}n^{d-1-\alpha}(\log n)^{(\alpha+1)(d-1)}\Big)^{1/(d\alpha)} \\ &\leqslant C_\alpha\Big(K_{d,\alpha}\varepsilon^{-1}(2\alpha^{-1}\log(2\varepsilon^{-1}))^{d-1-\alpha}(\log(2\alpha^{-1}\log(2\varepsilon^{-1})))^{(\alpha+1)(d-1)}\Big)^{1/(d\alpha)} \\ &\leqslant C_\alpha\Big(K_{d,\alpha}\varepsilon^{-1}(2\alpha^{-1}\log(2\varepsilon^{-1}))^{d-1-\alpha}(2\log(2\alpha^{-1})\log\log(2\varepsilon^{-1})))^{(\alpha+1)(d-1)}\Big)^{1/(d\alpha)}.\end{aligned}$$

In view of (46), we find that

$$\Big(K_{d,\alpha}(2\alpha^{-1})^{d-1-\alpha}(2\log(2\alpha^{-1}))^{(\alpha+1)(d-1)}\Big)^{1/(d\alpha)}$$

is bounded by a constant depending only on α. Consequently

$$L(\Phi_\varepsilon(f)) \leqslant C_\alpha\varepsilon^{-\frac{1}{d\alpha}}(\log(2\varepsilon^{-1}))^{\frac{d-1-\alpha}{d\alpha}}(\log\log(2\varepsilon^{-1}))^{\frac{(\alpha+1)(d-1)}{d\alpha}}$$

which proves (22). ∎

6. AN APPLICATION TO NUMERICAL SOLVING PDES

In this section, we apply the results on approximation by deep ReLU neural networks in Sections 4 and 5 to numerical approximation to the solution of elliptic PDEs. Before going

into detail, let us analyze the differences between approximation by deep ReLU neural networks and finite element methods. It is well-known that every deep ReLU neural network in $\mathbb{R}^d$ represents a continuous piecewise linear function defined on a number of polyhedral subdomains and conversely every continuous piecewise linear function in $\mathbb{R}^d$ can be represented by a deep ReLU. This shows that there exists some deep ReLU neural network which is at least as good as adaptive finite element for solutions of PDEs. However, the approximation to solutions of PDEs in high dimensions by finite element methods is burdened the curse of dimensionality. For some PDEs, it has been shown that deep neural networks are capable of representing solutions without incurring the curse of dimensionality, see, for instance, [15, 20, 21]. For further discussion about relationship between deep ReLU neural networks and finite element methods in numerical solving PDEs, we refer the reader to [17, 28, 30].

Consider a modeled diffusion elliptic equation

$$-\mathrm{div}(a(\boldsymbol{x})\nabla u(\boldsymbol{x})) \;=\; f(\boldsymbol{x}) \quad \text{in} \quad \mathbb{I}^d,\; u|_{\partial\mathbb{I}^d} \;=\; 0,$$

with a function f and a diffusion coefficient a having sufficient regularity. Denote by $V := \mathring{W}_2^1(\mathbb{I}^d)$ the energy space. If a satisfies the ellipticity assumption

$$0 < a_{\min} \leqslant a(\boldsymbol{x}) \leqslant a_{\max} < \infty,\; \forall \boldsymbol{x} \in \mathbb{I}^d,$$

by the well-known Lax-Milgram lemma, there exists a unique solution $u \in V$ in weak form which satisfies the variational equation

$$\int_{\mathbb{I}^d} a(\boldsymbol{x})\nabla u(\boldsymbol{x}) \cdot \nabla v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \;=\; \int_{\mathbb{I}^d} f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\,,\; \forall v \in V.$$

We want to approximate the solution u by deep ReLU neural networks. The approximation error is measured in the norm of $L_\infty(\mathbb{I}^d)$. Assume for the modeled case that a and f have Hölder-Nikol'skii mixed smoothness 1, i.e., $a, f \in H_\infty^1(\mathbb{I}^d)$. Then, the solution u has at least mixed derivatives $\partial^{\boldsymbol{\alpha}} u$ with $\boldsymbol{\alpha} \in \mathbb{N}_0^d$, $\max_{j=1,\ldots,d} \alpha_j \leqslant 1$, belonging to $L_2(\mathbb{I}^d)$ [12], and therefore, by embedding for function spaces of mixed smoothness, see [32, Theorem 2.4.1], u belongs to $\mathring{H}_\infty^{1/2}(\mathbb{I}^d)$. For simplicity we assume that $u \in \mathring{U}_\infty^{1/2}$.

For the nonadaptive approximation, according to Theorem 7, for any $\varepsilon > 0$ sufficient small one can explicitly construct a deep neural network architecture $\mathbb{A}_\varepsilon$ independent of f and a, and a deep ReLU neural network $\Phi_\varepsilon(u)$ having the architecture $\mathbb{A}_\varepsilon$ such that

$$\|u - \Phi_\varepsilon(u)\|_\infty \leqslant \varepsilon,$$

$$W(\mathbb{A}_\varepsilon) \leqslant Cd\left(\frac{K_1^d}{(d-1)!}\right)^3 \varepsilon^{-2} \log(2\varepsilon^{-1})^{3(d-1)+1},$$

and

$$L(\mathbb{A}_\varepsilon) \leqslant C \log d \log(2\varepsilon^{-1}),$$

where $K_1 := 8(\sqrt{2}+1)^{3/2}$.

For the adaptive approximation, according to Theorem 11, for any $\varepsilon > 0$ sufficient small one can explicitly construct an adaptive deep ReLU neural network $\Phi_\varepsilon(u)$ so that

$$\|u - \Phi_\varepsilon(u)\|_\infty \leqslant \varepsilon,$$

$$W(\Phi_\varepsilon(u)) \leqslant Cd^2\left(\frac{K_2^d}{(d-1)!}\right)^6 \varepsilon^{-2}\big(\log(2\varepsilon^{-1})\log\log(2\varepsilon^{-1})\big)^{3(d-1)},$$

and

$$L(\Phi_\varepsilon(u)) \leqslant C'\varepsilon^{-\frac{2}{d}}(\log(2\varepsilon^{-1}))^{\frac{2d-3}{d}}(\log\log(2\varepsilon^{-1}))^{\frac{3(d-1)}{d}},$$

where $K_2 := 16((2+\sqrt{2}))^{1/3}$.

7. CONCLUSIONS

We have presented both nonadaptive and adaptive methods for explicit construction of deep ReLU neural network $\Phi_\varepsilon(f)$ having an output that approximates functions f in the Hölder-Nikol'skii spaces with an arbitrary prescribed accuracy ε in the L_∞-norm. Nonadaptivity means that the architecture of approximating deep ReLU neural networks is the same for all functions in $\mathring{U}_\infty^{\alpha,d}$. For nonadaptive approximation, by using truncation of Faber series as a intermediate approximation, we have established a dimension-dependent estimate for the computation complexity characterized by the size $W(\Phi_\varepsilon(f))$ estimated by

$$W(\Phi_\varepsilon(f)) \leqslant C_\alpha d \left(\frac{K_1^d}{(d-1)!}\right)^{\frac{1}{\alpha}+1} \varepsilon^{-\frac{1}{\alpha}} \log(2\varepsilon^{-1})^{(d-1)(\frac{1}{\alpha}+1)+1},$$

where $K_1 = B^{1/(\alpha+1)} 4\alpha^{-1}$ with $B = (2^\alpha - 1)^{-1}$.

Concerning adaptive method, for any $f \in \mathring{U}_\infty^{\alpha,d}$, we explicitly construct a deep ReLU neural network $\Phi_\varepsilon(f)$ of adaptive architecture having the output that approximates f in the $L_\infty(\mathbb{I}^d)$-norm with a prescribed accuracy ε and having the size estimated by

$$W(\Phi_\varepsilon(f)) \leqslant C_\alpha d^2 \left(\frac{K_2^d}{(d-1)!}\right)^{\frac{2}{\alpha}+2} \varepsilon^{-\frac{1}{\alpha}} \big(\log(2\varepsilon^{-1}) \log\log(2\varepsilon^{-1})\big)^{(1+\frac{1}{\alpha})(d-1)},$$

where $K_2 = 4(2^{\alpha+3}B)^{\frac{1}{2\alpha+2}} (\alpha^{-1}\log(2\alpha^{-1}))^{1/2}$.

Construction of deep ReLU neural networks in the adaptive method is more involved but improves $\log(2\varepsilon^{-1})$ in the computation complexity of the approximating deep ReLU neural networks compared to the nonadaptive one.

Our theory is illustrated by an application to numerical approximation to the solution of elliptic PDEs.

ACKNOWLEDGMENT

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under Grant number 102.01-2020.03. A part of this work was done when Dinh Dũng and Van Kien Nguyen were working at the Vietnam Institute for Advanced Study in Mathematics (VIASM). They would like to thank the VIASM for providing a fruitful research environment and working condition.

REFERENCES

[1] M. Ali and A. Nouy, "Approximation of smoothness classes by deep ReLU networks," *arXiv:2007.15645*, 2020.

[2] D. Dũng, "B-spline quasi-interpolant representations and sampling recovery of functions with mixed smoothness," *J. Complexity*, vol. 27, pp. 541–567, 2011.

[3] D. Dũng and V. K. Nguyen, "Deep ReLU neural networks in high-dimensional approximation," *Neural Netw.*, vol. 142, pp. 619–635, 2021.

[4] ——, "High-dimensional nonlinear approximation by parametric manifolds in Hölder-Nikol'skii spaces of mixed smoothness," *arXiv:2102.04370*, 2021.

[5] D. Dũng, V. N. Temlyakov, and T. Ullrich, *Hyperbolic Cross Approximation.* Advanced Courses in Mathematics - CRM Barcelona, Birkhäuser/Springer, 2018.

[6] D. Dũng and M. X. Thao, "Dimension-dependent error estimates for sampling recovery on Smolyak grids based on B-spline quasi-interpolation," *J. Approx. Theory*, vol. 250, pp. 185–205, 2020.

[7] I. Daubechies, R. DeVore, S. Foucart, B. Hanin, and G. Petrova, "Nonlinear approximation and (Deep) ReLU networks," *arXiv:1905.02199*, 2019.

[8] W. E and Q. Wang, "Exponential convergence of the deep neural network approximation for analytic functions," *Sci. China Math.*, vol. 61, pp. 1733–1740, 2018.

[9] M. Geist, P. Petersen, M. Raslan, R. Schneider, and G. Kutyniok, "Numerical solution of the parametric diffusion equation by deep neural networks," *arXiv:2004.12131*, 2020.

[10] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," *In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Ft. Lauderdale, FL, USA*, pp. 315–323, 2011.

[11] R. Gribonval, Kutyniok, M. Nielsen, and F. Voigtlaender, "Approximation spaces of deep neural networks," *arXiv:1905.01208*, 2019.

[12] M. Griebel and S. Knapek, "Optimized general sparse grid approximation spaces for operator equations," *Math. Comp.*, vol. 78, pp. 2223–2257, 2009.

[13] A. Griewank, F. Y. Kuo, H. Leövey, and I. H. Sloan, "High dimensional integration of kinks and jumps – smoothing by preintegration," *J. Comput. Appl. Math.*, vol. 344, pp. 259–274, 2018.

[14] P. Grohs, D. Perekrestenko, D. Elbrachter, and H. Bolcskei, "Deep neural network approximation theory," *arXiv: 1901.02220*, 2019.

[15] P. Grohs, A. Jentzen, and D. Salimova, "Deep neural network approximations for Monte Carlo algorithms," *arXiv:1908.10828*, 2019.

[16] I. Gühring, G. Kutyniok, and P. Petersen, "Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms," *Anal. Appl. (Singap.)*, vol. 18, pp. 803–859, 2020.

[17] J. He, L. Li, J. Xu, and C. Zheng, "Relu deep neural networks and linear finite elements," *J. Comput. Math*, vol. 38, pp. 502–529, 2020.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *2015 IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

[19] D. Hebb, *The Organization of Behavior: A Neuropsychological Theory.* Wiley, 1949.

[20] M. Hutzenthaler, A. Jentzen, T. Kruse, and T. A. Nguyen, "A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations," *arXiv:1901.10854*, 2019.

[21] A. Jentzen, D. Salimova, and T. Welti, "A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients," *arXiv:1809.07321*, 2018.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *NeurIPS*, pp. 1106–1114, 2012.

[23] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[24] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," *Proceedings of the 30 th International Conference on Machine Learning, Atlanta, Georgia, USA*, pp. 315–323, 2014.

[25] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, pp. 115–133, 1943.

[26] H. N. Mhaskar, "Neural networks for optimal approximation of smooth and analytic functions," *Neural Comput.*, vol. 8, pp. 164–177, 1996.

[27] H. Montanelli and Q. Du, "New error bounds for deep ReLU networks using sparse grids," *SIAM J. Math. Data Sci.*, vol. 1, pp. 78–92, 2019.

[28] J. A. A. Opschoor, P. C. Petersen, and C. Schwab, "Deep ReLU networks and high-order finite element methods," *Anal. Appl. (Singap.)*, vol. 18, pp. 715–770, 2020.

[29] P. Petersen and F. Voigtlaender, "Optimal approximation of piecewise smooth functions using deep ReLU neural networks," *Neural Netw.*, vol. 108, pp. 296–330, 2018.

[30] P. C. Petersen, "Neural network theory," *Preprint*, 2020.

[31] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, pp. 386–408, 1958.

[32] H. Schmeisser and H. Triebel, *Topics in Fourier Analysis and Function Spaces.* Chichester; New York : Wiley, 1987.

[33] C. Schwab and J. Zech, "Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in UQ," *Anal. Appl. (Singap.)*, vol. 17, pp. 19–55, 2019.

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

[35] L. Tóth, "Phone recognition with deep sparse rectifier neural networks," *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6985–6989, 2013.

[36] H. Triebel, *Bases in Function Spaces, Sampling, Discrepancy, Numerical Integration.* European Math. Soc. Publishing House, Zürich, 2010.

[37] ——, *Hybrid Function Spaces, Heat and Navier-Stokes Equations.* European Mathematical Society, 2015.

[38] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, and M. Norouzi, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv: 1609.08144*, 2016.

[39] D. Yarotsky, "Error bounds for approximations with deep ReLU networks," *Neural Netw.*, vol. 94, pp. 103–114, 2017.

[40] ——, "Quantified advantage of discontinuous weight selection in approximations with deep neural networks," *arXiv: 1705.01365*, 2017.

[41] H. Yserentant, *Regularity and Approximability of Electronic Wave Functions.* Lecture Notes in Mathematics, Springer, 2010.

Received on March 2, 2021
Accepted on August 15, 2021