Adv Comput Math (2022) 48:84
https://doi.org/10.1007/s10444-022-09991-x

Approximation of functions from Korobov spaces by deep convolutional neural networks

Tong Mao[1] · Ding-Xuan Zhou[2]

Received: 14 March 2022 / Accepted: 13 October 2022 / Published online: 7 December 2022

Abstract

The efficiency of deep convolutional neural networks (DCNNs) has been demonstrated empirically in many practical applications. In this paper, we establish a theory for approximating functions from Korobov spaces by DCNNs. It verifies rigorously the efficiency of DCNNs in approximating functions of many variables with some variable structures and their abilities in overcoming the curse of dimensionality.

Keywords Machine learning · Deep convolutional neural networks · Curse of dimensionality · Korobov spaces

Mathematics Subject Classification (2010) 68T07 · 41A25

1 Introduction

Deep neural networks (DNNs) demonstrate excellent performances in many fields of science and technology these days. In particular, for functions with special properties or structures, DNNs can often take advantage of these properties or structures to improve the learning and approximation abilities of many classical tools remarkably and break the "curse of dimensionality" (e.g., [3, 4, 10, 11, 13, 16, 18–20]).

Deep convolutional neural networks (DCNNs), as an important class of structured deep neural networks, are very efficient for tasks in many areas [7, 11] such as speech recognition and computer vision. Compared with their practical success,

Communicated by: Rachel Ward

✉ Ding-Xuan Zhou
dingxuan.zhou@sydney.edu.au

Tong Mao
tongmao2-c@my.cityu.edu.hk

[1] School of Data Science, City University of Hong Kong, Kowloon, Hong Kong

[2] School of Mathematics and Statistics, University of Sydney, Sydney NSW 2006, Australia

the theory of DCNNs is far behind. Recently the universality of DCNNs is proved in [21, 23] asserting that any continuous function on any compact subset of a Euclidean space of the input data variable can be approximated to an arbitrary accuracy by a DCNN with zero padding when the number of layers is large enough. The rates of uniformly approximating functions from Sobolev spaces $W^{r,2}$ with $r > \frac{d}{2} + 2$ are obtained. It is further shown in [22] that every fully connected neural network (FNN) can be realized by a downsampled DCNN with the same order of free parameters. Inspired by this, we may expect that downsampled DCNNs can also make use of special structures to improve rates of function approximation. In fact, it is true for additive ridge functions [5] and radial functions [14].

All the above results for DNNs and CNNs present rates of type $\mathcal{O}(\mathcal{N}^{-\frac{r}{d}})$ for approximating functions of smoothness index $r > 0$ on subsets of the Euclidian space $\mathbb{R}^d$ by neural networks with $\mathcal{N}$ free parameters. When d is large, the convergence is rather slow. This is due to the isotropic nature of the function smoothness measured with respect to all the variables.

The great success in practical applications dealing with data from spaces of large dimensions d motivates us to expect faster convergence of deep learning algorithms when the target function has some special structures involving the variables $x_1, \ldots, x_d$. One such variable structure considered in [16] for DNNs is measured by Korobov spaces defined below in terms of mixed derivatives.

The purpose of this paper is to show that DCNNs perform excellently for approximating in L^p $(1 \leq p \leq \infty)$ functions from Korobov spaces involving mixed derivatives of order 2.

In this paper, we use the ReLU activation function $\sigma : \mathbb{R} \to \mathbb{R}$ defined as

$$\sigma(x) = \max\{0, x\}, \qquad x \in \mathbb{R}.$$

For vectors, it acts componentwise.

Given a sequence $a = (a_k)_{k\in\mathbb{Z}}$ supported in $\{n_1, \ldots, m_1\}$ and another $b = (b_k)_{k\in\mathbb{Z}}$ supported in $\{n_2, \ldots, m_2\}$, the convolution of a and b is a sequence supported in $\{n_1+n_2, \ldots, m_1+m_2\}$ given by $(a*b)_i = \sum_{k\in\mathbb{Z}} a_{i-k}b_k = \sum_{k=n_2}^{m_2} a_{i-k}b_k$ for $i \in \mathbb{Z}$.

Consider a sequence $w = (w_k)_{k\in\mathbb{Z}}$ supported in $\{0, 1, \ldots, s\}$ and $x = (x_k)_{k\in\mathbb{Z}}$ supported in $\{1, 2, \ldots, D\}$ with $s, D \in \mathbb{N}$. The convolution of w and x is a sequence supported in $\{1, 2, \ldots, D+s\}$, which can be expressed alternatively with possibly nonzero terms by $[(w * x)_i]_{i=1}^{D+s} = T^w [x_i]_{i=1}^{D}$, where

$$T^w := [w_{i-j}]_{\substack{i=1,\ldots,D+s,\\ j=1,\ldots,D}} = \begin{bmatrix} w_0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ w_1 & w_0 & 0 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ w_s & w_{s-1} & \ldots & w_0 & \ldots & 0 & 0 \\ 0 & w_s & \ldots & w_1 & \ddots & \vdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \ldots & \ldots & 0 & w_s & \ldots & w_1 & w_0 \\ \ldots & \ldots & \ldots & 0 & w_s & \ldots & w_1 \\ \vdots & \ldots & \ldots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & \ldots & \ldots & \ldots & 0 & w_s \end{bmatrix}. \tag{1.1}$$

Here the $(D+s)\times D$ Toeplitz matrix T^w is called a **convolutional matrix**. In DCNNs, this is the connection matrix between layers.

For notational simplicity, for a sequence $a=(a_k)_{k\in\mathbb{Z}}$, we use the notation $[a]_n^m$ to denote the vector $[a_n,\ldots,a_m]^T\in\mathbb{R}^{m-n+1}$ in the rest of this paper (instead of $[a_k]_{k=n}^m$). We also say a sequence $a=(a_k)_{k=-\infty}^{\infty}$ is **represented by** $[a]_n^m=[\alpha_1,\ldots,\alpha_{m-n+1}]^T$ if

$$a_k=\begin{cases}\alpha_{k-n+1}, & k\in\{n,\ldots,m\},\\ 0, & \text{otherwise.}\end{cases} \tag{1.2}$$

Now we state deep convolutional neural networks and Korobov spaces of functions vanishing on the boundaries.

Definition 1 Let $x=(x_1,\ldots,x_d)\in\mathbb{R}^d$ be the input data vector, $s, J\in\mathbb{N}$, $\{d_j\}_{j=1}^J$ given by $d_0=d$,

$$d_j=d_{j-1}+s, \qquad j\in\{1,\ldots,J\}.$$

The **DCNN** $\{h^{(j)}:\mathbb{R}^d\to\mathbb{R}^{d_j}\}_{j=1}^J$ with widths $\{d_j\}_{j=1}^J$, filters $\mathbf{w}:=\{w^{(j)}\}_{j=1}^J$ supported in $\{0,1,\ldots,s\}$ and biases $\{b^{(j)}\in\mathbb{R}^{d_j}\}_{j=1}^J$ is defined by the following composition

$$h^{(j)}(x)=\mathcal{A}_j\circ\ldots\circ\mathcal{A}_1(x), \quad j\in\{1,\ldots,J\}, \tag{1.3}$$

where for $j=1,\ldots,J$, $\mathcal{A}_j:\mathbb{R}^{d_{j-1}}\to\mathbb{R}^{d_j}$ is a map given by

$$\mathcal{A}_j(v)=\sigma(T^{w^{(j)}}v-b^{(j)}), \qquad v\in\mathbb{R}^{d_{j-1}}.$$

The classical DNNs have the same expression as (1.3) except that the connection matrix $T^{w^{(j)}}$ is replaced by a $d_j\times d_{j-1}$ full matrix. As we can see, the free parameters in the connection matrix $T^{w^{(j)}}$ come from the filters $\{w_k\}_{k=0}^s$. While the number of free parameters in the connection matrix is $(d_{j-1}+s)d_{j-1}$ in a fully connected layer, the number in the convolutional matrix $T^{w^{(j)}}$ is only $s+1$. This great reduction allows DCNNs to have large depths.

For the deep CNNs of depth J, the hypothesis space is a set of functions defined by

$$\mathcal{H}_J^{\mathbf{w},\mathbf{b}}=\left\{\sum_{k=1}^{d_J}c_k h_k^{(J)}(x):\ c\in\mathbb{R}^{d_J}\right\}. \tag{1.4}$$

For $k\in\mathbb{Z}_+^d$, denote $D^k f=\frac{\partial^{\|k\|_1}f}{\partial x_1^{k_1}\ldots\partial x_d^{k_d}}$ with $\|k\|_1=\sum\limits_{j=1}^d k_j$ and $\|k\|_\infty=\max\limits_{1\le j\le d}k_j$. For any $r\in\mathbb{N}$ and $1\le p\le\infty$, the norm of a classical Sobolev space $W^{r,p}([0,1]^d)$ is defined as

$$\|f\|_{W^{r,p}([0,1]^d)}=\max_{\|k\|_1=r}\left\|D^k f\right\|_p+\|f\|_p. \tag{1.5}$$

Definition 2 For $1\le p\le\infty$, the **Korobov space** $X^{2,p}([0,1]^d)$ consists of functions $f\in L^p([0,1]^d)$ which vanish on the boundary of $[0,1]^d$ and satisfy

$D^k f \in L^p([0,1]^d)$ for any $k \in \mathbb{Z}_+^d$ with $|k|_\infty \leq 2$. The norm is given in terms of the L^p-norm $\|f\|_p := \left(\int_{[0,1]^d} |f(x)|^p dx\right)^{1/p}$ by

$$\|f\|_{\mathbf{2},p} = \left\| \frac{\partial^{2d} f}{\partial x_1^2 \ldots \partial x_d^2} \right\|_p + \|f\|_p. \tag{1.6}$$

Remark 1 The property of vanishing on the boundary satisfied by functions from the Korobov space $X^{2,p}([0,1]^d)$ was required in the sparse grid method [1] for numerical analysis to handle boundary elements. The DCNNs represented in terms of the convolutional matrix (1.1) are ones with zero padding [24], meaning that we fill the entries of x outside $\{1, \ldots, D\}$ by 0. This corresponds to the condition of vanishing on the boundary for the approximated functions from the Korobov space. In our approximation analysis, we also need a function expansion (6.1) in terms of a basis of hat functions which naturally vanish on the boundary.

2 Main results

The following theorem to be proved in Section 6 is our first main result. The theorem gives rates for approximating functions from $X^{2,p}([0,1]^d)$ by deep convolutional neural networks.

Theorem 1 *Let $d \in \mathbb{N}$, $1 \leq p \leq \infty$ and f be a function in $X^{2,p}([0,1]^d)$ that satisfies $\|f\|_{\mathbf{2},p} \leq 1$. For any $N \geq 2^{16}$, there exists a deep neural network of depth*

$$J \leq \left(\frac{168}{s-1} + 2\right)\left(\log_2 d\right) d^2 (\log_2 N) N$$

constructed in Definition 1 associated with a filter sequence $\mathbf{w} = \{w_j\}_{j=1}^J$ and a bias sequence $\mathbf{b} = \{b_j\}_{j=1}^J$ such that

$$\inf\left\{\|f_J^{\mathbf{w},\mathbf{b}} - f\|_p : \ f_J^{\mathbf{w},\mathbf{b}} \in \mathcal{H}_J^{\mathbf{w},\mathbf{b}}\right\} \leq \left(\left(\log_2 N\right)^{\left(3-\frac{1}{p}\right)(d-1)} + 1\right) N^{-\left(2-\frac{1}{p}\right)}. \tag{2.1}$$

The number of free parameters of the CNN is bounded as

$$\mathcal{N} \leq 13385 d^2 (\log_2 d)^2 (\log_2 N)^2 N. \tag{2.2}$$

What is nice about the bounds for the depth and number of free parameters is the slow growth of their dependence on the data dimension d.

The following complexity analysis is an immediate consequence of Theorem 1 with $\frac{p}{2p-1} = \frac{1}{2}$ for $p = \infty$. When approximating functions from $X^{2,\infty}([0,1]^d)$, DCNNs perform as well as the DNN constructed in [16] (up to a multiplication by $|\log \epsilon|$).

Corollary 1 *Let $d \in \mathbb{N}$, $1 \le p \le \infty$ and f be a function in $X^{2,p}([0,1]^d)$ that satisfies $\|f\|_{\mathbf{2},p} \le 1$. For any $\epsilon > 0$, there exists a DCNN of depth*

$$J = \mathcal{O}\left(\epsilon^{-\frac{p}{2p-1}} |\log_2 \epsilon|^{\left(\frac{p}{2p-1}+1\right)(d-1)+1}\right)$$

constructed in Definition 1 associated with a filter sequence $\mathbf{w} = \{w_j\}_{j=1}^J$ and a bias sequence $\mathbf{b} = \{b_j\}_{j=1}^J$ such that

$$\inf\left\{\|f_J^{\mathbf{w},\mathbf{b}} - f\|_p : f_J^{\mathbf{w},\mathbf{b}} \in \mathcal{H}_J^{\mathbf{w},\mathbf{b}}\right\} \le \epsilon. \tag{2.3}$$

The number of free parameters of the DCNN satisfies

$$\mathcal{N} = \mathcal{O}\left(\epsilon^{-\frac{p}{2p-1}} |\log_2 \epsilon|^{\left(\frac{p}{2p-1}+1\right)(d-1)+2}\right).$$

Proof Choosing $N = \left\lceil 2^{3(d-1)} \epsilon^{-\frac{p}{2p-1}} |\log_2 \epsilon|^{\frac{p}{2p-1}+1} \right\rceil$ in Theorem 1, we know

$$\begin{aligned}\|f_J^{\mathbf{w},\mathbf{b}} - f\|_p &\le 2^{-3(d-1)} \epsilon |\log_2 \epsilon|^{-\left(3-\frac{1}{p}\right)(d-1)} |2\log_2 \epsilon|^{\left(3-\frac{1}{p}\right)(d-1)} \\ &\le \epsilon.\end{aligned}$$

We also see that the depth can be bounded as

$$\begin{aligned}J &\le \left(\frac{168}{s-1} + 2\right) \left(\log_2 d\right) d^2 (\log_2 N) N \\ &\le \left(\frac{168}{s-1} + 2\right) \left(\log_2 d\right) d^2 (6d-3) 2^{3(d-1)} \left[\epsilon^{-\frac{p}{2p-1}} |\log_2 \epsilon|^{\left(\frac{p}{2p-1}+1\right)(d-1)+1}\right].\end{aligned}$$

The number of free parameters can be bounded as

$$\begin{aligned}\mathcal{N} &\le 13385 d^2 (\log_2 d)^2 (\log_2 N)^2 N \\ &\le 13385 d^2 (\log_2 d)^2 (6d-3)^2 2^{3(d-1)} \left[\epsilon^{-\frac{p}{2p-1}} |\log_2 \epsilon|^{\left(\frac{p}{2p-1}+1\right)(d-1)+2}\right].\end{aligned}$$

This proves the desired bounds. □

Let us demonstrate the role of Korobov spaces in measuring smoothness by the following example.

Example 1 Let g be a piecewise quadratic polynomial on $\mathbb{R}$ given by

$$g(x) = \begin{cases} x^2, & \text{if } x \in [0, \frac{1}{3}], \\ -2(x-\frac{1}{2})^2 + \frac{1}{6}, & \text{if } x \in (\frac{1}{3}, \frac{2}{3}), \\ (1-x)^2, & \text{if } x \in [\frac{2}{3}, 1], \\ 0, & \text{if } x \notin [0, 1]. \end{cases} \tag{2.4}$$

Then $g \in C^1(\mathbb{R})$ and g'' exists almost everywhere as

$$g''(x) = \begin{cases} 2, & \text{if } x \in (0, \frac{1}{3}) \bigcup (\frac{2}{3}, 1), \\ -4, & \text{if } x \in (\frac{1}{3}, \frac{2}{3}), \\ 0, & \text{if } x \notin (-\infty, 0) \bigcup (1, \infty). \end{cases}$$

Hence $g \in X^{2,\infty}([0,1])$. For $1 \le p \le \infty$ and $0 < t < \frac{1}{3}$,

$$g''(x+t) - g''(x) = \begin{cases} 2, & \text{if } x \in (-t, 0), \\ -6, & \text{if } x \in (\frac{1}{3} - t, \frac{1}{3}), \\ 6, & \text{if } x \in (\frac{2}{3} - t, \frac{2}{3}), \\ -2, & \text{if } x \in (1-t, 1), \\ 0, & \text{if } x \in (-\infty, -t) \bigcup (0, \frac{1}{3} - t) \\ & \qquad \bigcup (\frac{1}{3}, \frac{2}{3} - t) \bigcup (\frac{2}{3}, 1-t) \bigcup (1, \infty). \end{cases}$$

It follows that $\|g''(\cdot + t) - g''\|_{L^p(\mathbb{R})} = 2^{\frac{1}{p}}(2^p + 6^p)^{\frac{1}{p}} t^{\frac{1}{p}}$. Then $g \in W^{2+\frac{1}{p},p}(\mathbb{R})$ and $g \notin W^{r,p}(\mathbb{R})$ for any $r > 2 + \frac{1}{p}$.

For $d \in \mathbb{N}$, we define f_1 and f_2 on $[0,1]^d$ by

$$f_1(x) = \sum_{j=1}^{d} g(x_j), \quad f_2(x) = \prod_{j=1}^{d} g(x_j), \qquad x \in [0,1]^d.$$

We see that $D^k f_1 \in L^p([0,1]^d)$, $D^k f_2 \in L^p([0,1]^d)$ for $1 \le p \le \infty$ and $|k|_\infty \le 2$. Hence $f_1, \ f_2 \in X^{2,p}([0,1]^d)$. However, $f_1 \notin W^{r,p}([0,1]^d)$ and $f_2 \notin W^{r,p}([0,1]^d)$ for $r > 2 + \frac{1}{p}$.

3 Comparisons and discussion

Although there is a large classical literature on approximation by shallow networks [9, 15, 17], the recent success of deep learning gives strong reasons to use deep neural networks instead of shallow ones [4, 6, 12, 18, 19, 25]. The most important reason is the curse of dimensionality: deep neural networks often break the curse of dimensionality since they can make use of special structures or properties of the function classes, while shallow neural networks usually cannot. Our result can be regarded as evidence of breaking the curse of dimensionality in approximation by deep neural networks. For functions from Hölder spaces $W^{r,\infty}([0,1]^d)$, it is known that the optimal rate of approximation by neural networks is $\mathcal{O}(\mathcal{N}^{-\frac{r}{d}})$, where $\mathcal{N}$ is the number of free parameters. Then a rate $\mathcal{O}(\mathcal{N}^{-2})$ is possible only when $r \ge 2d$, whereas for functions from the Korobov space $X^{2,\infty}([0,1]^d)$ the approximation rate is $\mathcal{O}(\mathcal{N}^{-2})$. As we can see from Example 1, the restriction on the smoothness of functions from Korobov spaces is much weaker: Hölder spaces require the essential boundedness of all derivatives of order $2d$:

$$D^k f \in L^\infty([0,1]^d), \quad \forall k = (k_1, \ldots, k_d) \in \mathbb{Z}_+^d \text{ satisfying } \sum_{j=1}^{d} k_j \le 2d,$$

while the Korobov space $X^{2,\infty}([0,1]^d)$ only requires that of

$$\frac{\partial^{2d} f}{\partial x_1^2 \ldots \partial x_d^2}.$$

In the literature, theory of deep CNNs has been established for various problems. It was shown in [21] functions with Fourier transform $\hat{f}$ satisfying $\int_{\mathbb{R}^d} |\hat{f}(\omega)||\omega^2|d\omega < \infty$ can be approximated by CNNs of depth J in the rate $\mathcal{O}(J^{-\frac{1}{2}-\frac{1}{d}})$. Then it was found that deep CNNs approximate optimally functions with some special structures, such as ridge functions [5], radial functions, and functions with polynomial features [14], which is much faster than FNNs. In this paper, we consider a function class in another perspective: the Korobov space defined by the regularity in terms of mixed derivatives instead of some special composite structures or properties.

It was also shown in [22] that a downsampled DCNN with at most 8 times free parameters can realize the same output of an FNN. However, when constructing deep neural networks to approximate functions, the networks are often sparse, and it is often possible that the neurons share common weights in most of the layers [16, 20] (which is called parameter sharing in [8]). From our construction, we can see that in some cases DCNNs do not need to be designed with specified sparsity. They can automatically make use of the sparsity and reduce the number of free parameters remarkably. One reason for this phenomenon is that DCNNs benefit from the orderliness of DNNs to carry out the sparsity. Another reason is that convolutions naturally share weights, so we do not need to treat each layer as a fully connected one and produce a large number of weights.

More work can be done about DCNNs to see how they make use of different structures. Since the work [2] relies heavily on locally Taylor polynomials, one may expect DCNNs to also perform efficiently for learning spatially sparse functions.

4 Two basic blocks of CNNs

In this section, we construct two groups of deep CNNs, which represent two basic blocks in the construction of $f_J^{\mathbf{w},\mathbf{b}}$. Throughout the paper, we use notations $\boldsymbol{a}_k = [a, \ldots, a]$ or $\boldsymbol{a}_k = [a, \ldots, a]^T$ for vectors of k identical components $a \in \mathbb{R}$.

Before introducing these deep CNNs, we specify the following fact: zeros at the beginning or end will make no difference to the result of the convolution. Mathematically, let $\alpha,\ \beta$ be sequences supported in $\{1, \ldots, l_1\}$ and $\{0, \ldots, l_2\}$, and

$$a = \left(\alpha_{k-n_1}\right)_{k\in\mathbb{Z}},\ b = \left(\beta_{k-n_2}\right)_{k\in\mathbb{Z}}$$

with $n_1, n_2 \in \mathbb{Z}_+$. Then with $L = l_1 + l_2 + n_1 + n_2 + m_1 + m_2$ for $m_1, m_2 \in \mathbb{Z}_+$, there holds

$$[a * b]_1^L = [\mathbf{0}_{n_1+n_2}, (\alpha * \beta)_1, \ldots, (\alpha * \beta)_{l_1+l_2}, \mathbf{0}_{m_1+m_2}]. \tag{4.1}$$

This reflects the shift-invariance of convolutions, which is believed to ensure the super efficiency of DCNNs.

4.1 Representing shallow networks by DCNNs

The following lemma was proved in [21]. We apply (4.1) and give the following version for convenience.

Lemma 1 *Let $s, m, n \in \mathbb{N}$ and $M > 0$. For any sequence W supported in $\{0, \ldots, n\}$, B supported in $\{1, \ldots, m+n\}$, there exist $J \le \left\lceil \frac{n}{s-1} \right\rceil$, filters $\mathbf{w} = \{w^{(j)}\}_{j=1}^{J}$, each supported in $\{0, 1, \ldots, s\}$, and biases $\mathbf{b} = \{b^{(j)} \in \mathbb{R}^{d_j}\}_{j=1}^{J}$ with $b^{(j)}$ of the form*

$$b^{(j)} = [b_1^{(j)}, \ldots, b_{s-1}^{(j)}, \underbrace{b_s^{(j)}, \ldots, b_s^{(j)}}_{d_j - 2s}, b_{d_j - s+2}^{(j)}, \ldots, b_{d_j}^{(j)}]^T, \qquad j = 1, \ldots, J-1, \tag{4.2}$$

such that

$$\left[w^{(1)} * \cdots * w^{(J)}\right]_0^{Js} = \begin{bmatrix} W_0 \\ \vdots \\ W_n \\ \mathbf{0}_{Js-n} \end{bmatrix}$$

and for any input

$$\hat{z} = \left[\mathbf{0}_{L_1}, \check{z}^T, \mathbf{0}_{L_2}\right]^T \in [-M, M]^{L_1+m+L_2},$$

the last layer of the deep CNN with filters $\mathbf{w}$ and biases $\mathbf{b}$ is

$$h^{(J)}(\hat{z}) = \sigma\left(\begin{bmatrix} \mathbf{0}_{L_1} \\ [z * W - B]_1^{m+n} \\ \mathbf{0}_{L_2+Js-n} \end{bmatrix}\right), \tag{4.3}$$

where z is the sequence supported and identical to $\check{z}$ on $\{1, \ldots, m\}$.

The number of free parameters in this network is bounded by

$$(4s+1)J + L_1 + L_2 + m. \tag{4.4}$$

4.2 Approximating quadratic polynomials by DCNNs

Definition 3 Let $u,\ L \in \mathbb{N}$. The **hat function** and its iterations, **tooth functions**, are defined as

$$S(x) = 2\sigma(x) - 4\sigma\left(x - \frac{\mathbf{1}}{\mathbf{2}}\right) + 2\sigma(x - \mathbf{1}),$$

$$S_u(x) = \underbrace{S \circ \cdots \circ S}_{u \text{ folds}}(x), \qquad x \in \mathbb{R}^L.$$

We denote

$$T_u = 2^{-u} S_u$$

and the sum of tooth functions as

$$R_u(x) := \sum_{j=1}^{u} T_j(x).$$

For convenience, we also define

$$T_0(x) = x, \qquad R_0(x) \equiv \mathbf{0}_L.$$

Using the functions above, Yarotsky [20] proved that the univariable quadratic polynomial $f(x) = x^2$ with $L = 1$ can be approximated with accuracy 2^{-V} for $V \in$

$\mathbb{N}$ by a deep fully connected network with $\mathcal{O}(V)$ layers and $\mathcal{O}(V)$ free parameters. The following lemma shows that this process can be replaced by a deep CNN with $\mathcal{O}(V)$ layers and $\mathcal{O}(V^2)$ free parameters.

Lemma 2 *Let $L_1,\ L,\ L_2 \in \mathbb{N}$. For any $V \in \mathbb{N}$ there exists a deep CNN $\left\{h^{(j)} : \mathbb{R}^{L_1+L+L_2} \to \mathbb{R}^{d_j}\right\}_{j=1}^{K}$ such that for any input $\hat{y} = [\mathbf{0}_{L_1}, y^T, \mathbf{0}_{L_2}]^T \in [0,1]^{L_1+L+L_2}$, the last layer is*

$$h^{(K)}(\hat{y}) = \left[\mathbf{0}_{L_V}, y^T, \mathbf{0}_{7L}, y^T - (R_V(y))^T, \mathbf{0}_{L'_V}\right]^T, \tag{4.5}$$

where $y = [y_1, \ldots, y_L]^T$, the zero vectors are set consistent, and K is bounded as

$$K \le \frac{(7V + 15)L}{s-1} + 3V + 2. \tag{4.6}$$

Furthermore, only $3V+2$ bias vectors do not satisfy the restriction (4.2). The number of free parameters in this network is bounded in terms of dimension $\dim\left(\hat{y}\right) = L + L_1 + L_2$ of $\hat{y}$ by

$$\mathcal{N} \le (6V + 10)(7V + 15)L + (3V + 2)(3V + 5)s + (3V + 2)\dim\left(\hat{y}\right). \tag{4.7}$$

The proof of this lemma is given in Appendix.

In many cases, when [20, Proposition 2] or its method can be applied, Lemma 2 also works. As an example, we can use Lemma 2 to prove DCNNs (without downsamplings) of depth J can approximate almost optimally (up to a logarithmic term) functions from the Sobolev space $W^{r,\infty}$ with rate $\mathcal{O}\left(J^{-r/d}(\log J)^2\right)$.

5 Constructing deep CNNs for approximation

In this section, we introduce the standard *nodal point basis* of the Korobov space [1] to deduce our approximation scheme. This basis consists of functions of the form $\prod_{j=1}^{d} \phi_{i_j,l_j}(x_j)$, where $\left\{\phi_{i_j,l_j}\right\}$ are hat functions.

To realize such a basis function by deep CNNs, we first use $\mathcal{O}(N)$ convolutional layers to construct univariable basis functions, then we introduce $\mathcal{O}(N\log N)$ layers to compute the approximations of the products $\prod_{j=1}^{d} \phi_{i_j,l_j}(x_j)$. Finally, a linear combination gives an approximation of f, which gives our construction.

5.1 Generating univariable basis functions by deep CNNs

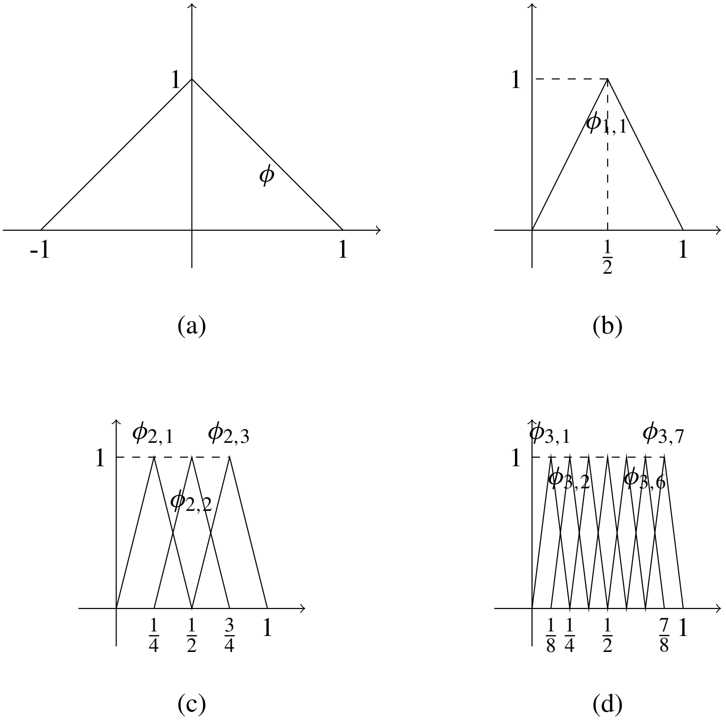
We first introduce a standard basis in [1]. Let $i,\ l,\ n \in \mathbb{N}$ and

$$\phi(x) = \begin{cases} 1 - |x|, & \text{if } x \in [-1, 1], \\ 0, & \text{otherwise.} \end{cases}$$

Define

$$\phi_{l,i}(x) = \phi\left(\frac{x - x_{l,i}}{h_l}\right), \qquad 1 \le i \le 2^l - 1,$$

where $h_l = 2^{-l}$ and $x_{l,i} = ih_l$.



(a) (b)

(c) (d)

Figure (a) for ϕ, Figures (b), (c), (d) for $\phi_{l,i}$.

For any $\boldsymbol{i},\ \boldsymbol{l} \in \mathbb{N}^d$, let $2^{\boldsymbol{l}} := (2^{l_1}, \ldots, 2^{l_d}) \in \mathbb{N}^d$,

$$\boldsymbol{I_l} := \prod_{j=1}^{d} \left\{1, 3, 5, \ldots, 2^{l_j} - 1\right\}$$

be the set of integer vectors with positive odd entries, and

$$\phi_{\boldsymbol{l},\boldsymbol{i}}(x) := \prod_{j=1}^{d} \phi_{l_j,i_j}(x_j), \qquad \boldsymbol{i} = (i_1, \ldots, i_d) \in \boldsymbol{I_l}.$$

Now we choose $n = n_N := \max\left\{n' \in \mathbb{N} : \sum_{|\boldsymbol{l}|_1 \le n'+d-1} \#\boldsymbol{I_l} \le N\right\}$ for $N \in \mathbb{N}$ where $\#\boldsymbol{I_l}$ denotes the number of elements of the set $\boldsymbol{I_l}$. By [1, Lemma 3.6], n satisfies

$$\log_2\left(\frac{N}{(\log_2 N)^{d-1}}\right) \le n \le \log_2 N. \tag{5.1}$$

For notational simplicity, let

$$N' = N'_n := \#\{\phi_{\boldsymbol{l},\boldsymbol{i}}(x) :\ |\boldsymbol{l}|_1 \le n + d - 1,\ \boldsymbol{i} \in \boldsymbol{I_l}\},$$

then $2^{15} \leq N/2 \leq N' \leq N$. We also denote a bijection

$$\mu : \{1, \dots, N'\} \to \{\phi_{\boldsymbol{l},\boldsymbol{i}}(x) : \ |\boldsymbol{l}|_1 \leq n + d - 1, \ \boldsymbol{i} \in \boldsymbol{I_l}\}.$$

For each $k \in \{1, \dots, N'\}$, we also use a notation $\mu(k)_j$ to denote the index (l_j, i_j), where $(\boldsymbol{l}, \boldsymbol{i})$ is the image $\mu(k)$.

We first construct the univariable hat functions

$$\{\phi_{l_j,i_j}(x_j) : \ |\boldsymbol{l}|_1 \leq n + d - 1, \ \boldsymbol{i} \in \boldsymbol{I_l}, \ 1 \leq j \leq d\}$$

by deep CNNs.

Now we introduce the first group of deep CNN layers. Let $P = \lceil \log_2 d \rceil \in \mathbb{N}$, then $2^{P-1} < d \leq 2^P$. Set

$$\phi_{l_j,i_j}(x_j) \equiv 1, \qquad \forall |\boldsymbol{l}|_1 \leq n + d - 1, \ \boldsymbol{i} \in \boldsymbol{I_l}, \ d + 1 \leq j \leq 2^P,$$

then

$$\phi_{\boldsymbol{l},\boldsymbol{i}}(x) = \prod_{j=1}^{2^P} \phi_{l_j,i_j}(x_j), \qquad \forall x \in [0, 1]^d.$$

Notice that for $1 \leq j \leq d$,

$$\phi_{l_j,i_j}(x_j) = \sigma\left(1 - \sigma\left(\frac{x_j - x_{l_j,i_j}}{h_{l_j}}\right) - \sigma\left(\frac{x_{l_j,i_j} - x_j}{h_{l_j}}\right)\right).$$

For each $\boldsymbol{l}$ satisfying $|\boldsymbol{l}| \leq n + d - 1$ and $\boldsymbol{i} \in \boldsymbol{I_l}$, we take 4 vectors in $\mathbb{R}^{d2^P}$ as

$$W_{0,\boldsymbol{l},\boldsymbol{i}} = \begin{bmatrix} W_{0,\boldsymbol{l},\boldsymbol{i},1} \\ \vdots \\ W_{0,\boldsymbol{l},\boldsymbol{i},2^P} \end{bmatrix}, \ \hat{W}_{0,\boldsymbol{l},\boldsymbol{i}} = \begin{bmatrix} \hat{W}_{0,\boldsymbol{l},\boldsymbol{i},1} \\ \vdots \\ \hat{W}_{0,\boldsymbol{l},\boldsymbol{i},2^P} \end{bmatrix},$$

$$B_{0,\boldsymbol{l},\boldsymbol{i}} = \begin{bmatrix} B_{0,\boldsymbol{l},\boldsymbol{i},1} \\ \vdots \\ B_{0,\boldsymbol{l},\boldsymbol{i},2^P} \end{bmatrix}, \ \hat{B}_{0,\boldsymbol{l},\boldsymbol{i}} = \begin{bmatrix} \hat{B}_{0,\boldsymbol{l},\boldsymbol{i},1} \\ \vdots \\ \hat{B}_{0,\boldsymbol{l},\boldsymbol{i},2^P} \end{bmatrix}$$

where for $j = 1, \dots, d$,

$$W_{0,\boldsymbol{l},\boldsymbol{i},j} = \begin{bmatrix} \mathbf{0}_{d-j} \\ \frac{1}{h_{l_j}} \\ \mathbf{0}_{j-1} \end{bmatrix}, \ \hat{W}_{0,\boldsymbol{l},\boldsymbol{i},j} = \begin{bmatrix} \mathbf{0}_{d-j} \\ -\frac{1}{h_{l_j}} \\ \mathbf{0}_{j-1} \end{bmatrix},$$

and

$$B_{0,\boldsymbol{l},\boldsymbol{i},j} = \begin{bmatrix} 2^{n+d}\mathbf{1}_{d-1} \\ \frac{x_{l_j,i_j}}{h_{l_j}} \end{bmatrix}, \ \hat{B}_{0,\boldsymbol{l},\boldsymbol{i},j} = \begin{bmatrix} 2^{n+d}\mathbf{1}_{d-1} \\ -\frac{x_{l_j,i_j}}{h_{l_j}} \end{bmatrix},$$

while for $j = d + 1, \dots, 2^P$, the vectors are $\mathbf{0}_d$.

Now we take $L_1 = L_2 = 0, \ m = d, \ M = 1, W$ be represented by

$$[W]_0^{2d2^PN'-1} = [W_{0,\mu(1)}^T, \dots, W_{0,\mu(N')}^T, \hat{W}_{0,\mu(1)}^T, \dots, \hat{W}_{0,\mu(N')}^T]^T,$$

B by

$$[B]_1^{2d2^PN'+d-1} = [B_{0,\mu(1)}^T, \dots, B_{0,\mu(N')}^T, \hat{B}_{0,\mu(1)}^T, \dots, \hat{B}_{0,\mu(N')}^T, \mathbf{0}_{d-1}]^T$$

and $\check{z} = x$ according to Lemma 1, we know that there exist $J_0 \leq \left\lceil \frac{2d2^P N'-1}{s-1} \right\rceil$, filters $\{w^{(j)}\}_{j=1}^{J_0}$ and biases $\{b^{(j)}\}_{j=1}^{J_0}$ satisfying (4.2) such that

$$\begin{aligned} h^{(J_0)}(x) = &\ \sigma\left(\begin{bmatrix} [z * W - B]_1^{2d2^P N'+d-1} \\ \mathbf{0}_{J_0 s+1-2d2^P N'} \end{bmatrix}\right) \\ = &\ \left[H_{0,\mu(1)}, \ldots, H_{0,\mu(N')}, \hat{H}_{0,\mu(1)}, \ldots, \hat{H}_{0,\mu(N')}, \mathbf{0}_{J_0 s+d-2d2^P N'}\right]^T, \end{aligned} \tag{5.2}$$

where $H_{0,\mu(k)} = H_{0,\boldsymbol{l},\boldsymbol{i}}$ is given by means of the bijection μ as

$$H_{0,\boldsymbol{l},\boldsymbol{i}}^T = \left[H_{0,\boldsymbol{l},\boldsymbol{i},1}^T, H_{0,\boldsymbol{l},\boldsymbol{i},2}^T, \ldots, H_{0,\boldsymbol{l},\boldsymbol{i},2^P}^T\right], \quad \hat{H}_{0,\boldsymbol{l},\boldsymbol{i}}^T = \left[\hat{H}_{0,\boldsymbol{l},\boldsymbol{i},1}^T, \hat{H}_{0,\boldsymbol{l},\boldsymbol{i},2}^T, \ldots, \hat{H}_{0,\boldsymbol{l},\boldsymbol{i},2^P}^T\right],$$

with

$$\begin{aligned} H_{0,\boldsymbol{l},\boldsymbol{i},j} &= \begin{cases} \left[\mathbf{0}_{d-1}, \sigma\left(\frac{x_j - x_{l_j,i_j}}{h_{l_j}}\right)\right]^T, & \text{if } 1 \leq j \leq d, \\ \mathbf{0}_d^T, & \text{if } d+1 \leq j \leq 2^P, \end{cases} \\ \hat{H}_{0,\boldsymbol{l},\boldsymbol{i},j} &= \begin{cases} \left[\mathbf{0}_{d-1}, \sigma\left(\frac{x_{l_j,i_j} - x_j}{h_{l_j}}\right)\right]^T, & \text{if } 1 \leq j \leq d, \\ \mathbf{0}_d^T, & \text{if } d+1 \leq j \leq 2^P. \end{cases} \end{aligned}$$

The number of free parameters from the input layer x to the J_0-th layer is bounded by (4.4) as

$$\mathcal{N}_1 \leq (4s+1)J_0 + d \leq 18d2^P N' + d + 4s - 8. \tag{5.3}$$

The explicit expressions of $H_{0,\boldsymbol{l},\boldsymbol{i},j}$ and $\hat{H}_{0,\boldsymbol{l},\boldsymbol{i},j}$ follow from direct computations, using the facts $|x_j/h_{l_i}| \leq 2^l \leq 2^{d+n-1}$ and $\sigma(t) = 0$ for $t \leq 0$. This gives the first group of J_0 layers.

For the second group of convolutional layers, we take $L_1 = 0$, $m = 2d2^P N'$, $L_2 = J_0(s-2) + d - 2d2^P N'$, $M = 2^{d+n-1}$, W be represented by

$$[W]_0^{d2^P N'} = [-1, \mathbf{0}_{d2^P N'-1}, -1]^T,$$

B by

$$[B]_1^{3d2^P N'} = [\mathbf{0}_{d2^P N'}, \mathbf{-1}_{d2^P N'}, \mathbf{0}_{d2^P N'}]^T,$$

and

$$\check{z} = \left[H_{0,\mu(1)}, \ldots, H_{0,\mu(N')}, \hat{H}_{0,\mu(1)}, \ldots, \hat{H}_{0,\mu(N')}\right]^T,$$

in compliance with Lemma 1, and know that there exist $J_1 \leq J_0 + \left\lceil \frac{d2^P N'}{s-1} \right\rceil$, filters $\{w^{(j)}\}_{j=J_0+1}^{J_1}$ and biases $\{b^{(j)}\}_{j=J_0+1}^{J_1}$ satisfying (4.2) such that

$$\begin{aligned} h^{(J_1)}(x) &= \sigma\left(\begin{bmatrix} \left[W * \eta^{(J_0)}(x) - B\right]_1^{3d2^P N'} \\ \mathbf{0}_{J_1 s+d-3d2^P N'} \end{bmatrix}\right) \\ &= [\mathbf{0}_{d2^P N'}, H_{1,\mu(1)}, \ldots, H_{1,\mu(N')}, \mathbf{0}_{J_1 s+d-2d2^P N'}]^T, \end{aligned} \tag{5.4}$$

where each $H_{1,\boldsymbol{l},\boldsymbol{i}} \in [0,1]^{d\times 2^P}$ and the jd-th component satisfies

$$\left(H_{1,\boldsymbol{l},\boldsymbol{i}}\right)_{jd} = \begin{cases} \sigma\left(-\sigma\left(\frac{x_j - x_{l_j,i_j}}{h_{l_j}}\right) - \sigma\left(\frac{x_{l_j,i_j} - x_j}{h_{l_j}}\right) + 1\right) = \phi_{l_j,i_j}(x_j), & \text{if } 1 \le j \le d, \\ 1 = \phi_{l_j,i_j}(x_j), & \text{if } d+1 \le j \le 2^P. \end{cases}$$

Using (4.4), we see that the number of free parameters for this second group of $J_1 - J_0$ convolutional layers is bounded by

$$\mathcal{N}_2 \le (4s+1)(J_1 - J_0) + J_0 s + d \le 13d2^P N' + d + 5s - 1. \tag{5.5}$$

5.2 Approximating products exponentially by deep CNNs

In this subsection, we use Lemma 2 to construct next groups of convolutional layers by induction to realize an approximation $\widetilde{\times}$ of the product function $(t_1, t_2) \mapsto t_1 t_2$ satisfying $\left|\widetilde{\times}(t_1, t_2) - t_1 t_2\right| = \mathcal{O}\left(\frac{1}{2^U}\right)$.

Definition 4 Let $\widetilde{\times} = \widetilde{\times}(U)$ be a map $\widetilde{\times} : \mathbb{R}^2 \to \mathbb{R}$ defined by

$$\widetilde{\times}(x, y) = (\mathrm{id} - R_U)\left(\frac{x+y}{2}\right) - \frac{1}{4}(\mathrm{id} - R_U)(x) - \frac{1}{4}(\mathrm{id} - R_U)(y), \tag{5.6}$$

where id is the identity function, and

$$U = \left\lceil 3 + \log_2 d + 2\log_2 N' \right\rceil. \tag{5.7}$$

For vectors $x,\ y$ with equal dimension, $\widetilde{\times}$ acts componentwise.

For any $k \in \{1, \ldots, N'\}$ and $r \in \{1, \ldots, 2^P\}$, define

$$\Lambda(k, 1, r; x) = \phi_{\mu(k)_r}(x_r) = \phi_{l_r, i_r}(x_r),$$

where $(\boldsymbol{l}, \boldsymbol{i}) = (l_1, \ldots, l_d, i_1, \ldots, i_d)$ is the image $\mu(k)$.

For $Q \in \{2, \ldots, P+1\}$ and $r \in \{1, \ldots, 2^{P-Q+1}\}$, define

$$\Lambda(k, Q, r; x) = \widetilde{\times}\left(\Lambda(k, Q, 2r-1; x), \Lambda(k, Q, 2r; x)\right).$$

Now we are ready to construct CNNs realizing the approximate products $\Lambda(k, Q, r; x)$ by induction on $Q = 2, 3, \ldots, P$ based on the hat functions $\phi_{\ell_r, i_r}(x)$ with $Q = 1$. Suppose that for some $Q \le P$, we have

$$h^{(J_Q)}(x)^T = [\mathbf{0}_{L_Q}^T, H_{Q,\mu(1)}^T, \ldots, H_{Q,\mu(N')}^T, \mathbf{0}_{L'_Q}^T], \tag{5.8}$$

where $L_Q\ L'_Q \in \mathbb{Z}_+$, each $H_{Q,\mu(k)} \in [0,1]^{d\times 2^P}$ and its $d \times 2^{Q-1} \times r$-th component is

$$(H_{Q,\mu(k)})_{d\times 2^{Q-1}\times r} = \Lambda(k, Q, r; x), \qquad r = 1, 2, \ldots, 2^{P-Q+1}.$$

We show how to construct the convolutional layers for realizing the approximate products for $Q+1$. First applying Lemma 1 to $L_1 = L_Q,\ m = d2^P N',\ L_2 = L'_Q,\ M = 1,\ W$ represented by

$$[W]_0^{d2^P N' + d2^{Q-1}} = [1, \mathbf{0}_{d2^P N' - 1}, \frac{1}{2}, \mathbf{0}_{d2^{Q-1}-1}, \frac{1}{2}]^T,$$

B by

$$[B]_1^{2d2^P N'+d2^{Q-1}} = \left[\mathbf{0}_{2d2^P N'}, \mathbf{2}_{d2^{Q-1}}\right],$$

and

$$\check{z} = \left[H^T_{Q,\mu(1)}, \ldots, H^T_{Q,\mu(N')}\right]^T,$$

we know there exist $J_{Q+1,1} \leq J_Q + \left\lceil \frac{d2^P N'+d2^{Q-1}}{s-1} \right\rceil$, $\{w^{(j)}\}_{j=J_Q+1}^{J_{Q+1,1}}$ and $\{b^{(j)}\}_{j=J_Q+1}^{J_{Q+1,1}}$ such that

$$\begin{aligned} h^{(J_{Q+1,1})}(x) &= \sigma\left(\begin{bmatrix} \mathbf{0}_{L_Q} \\ [z * W - B]_1^{2d2^P N'+d2^{Q-1}} \\ \mathbf{0}_{L'_Q+(J_{Q+1,1}-J_Q)s-(2d2^P N'+d2^{Q-1})} \end{bmatrix}\right). \\ &= \left[\mathbf{0}^T_{L_Q}, H^T_{Q,\mu(1)}, \ldots, H^T_{Q,\mu(N')}, \hat{H}^T_{Q,\mu(1)}, \ldots, \hat{H}^T_{Q,\mu(N')}, \mathbf{0}^T_{L'_{Q+1,1}}\right]^T, \end{aligned}$$

where $L'_{Q+1,1} = L'_Q + \left(J_{Q+1,1} - J_Q\right)s - 2d2^P N'$, $\hat{H}_{Q,\mu(k)} \in [0,2]^{d\times 2^P}$ and for $r \in \{1, \ldots, 2^{P-Q}\}$ the $d \times 2^{Q-1} \times 2r$-th component of $\hat{H}_{Q,\mu(k)}$ is the half sum $(H_{Q,\mu(k)})_{d\times 2^{Q-1}\times 2r} + (H_{Q,\mu(k)})_{d\times 2^{Q-1}\times(2r-1)}$ given by

$$(\hat{H}_{Q,\mu(k)})_{d\times 2^{Q-1}\times 2r} = \frac{1}{2}\left(\Lambda(k, Q, 2r; x) + \Lambda(k, Q, 2r-1; x)\right).$$

Again by (4.4), the number of free parameters for these $J_{Q+1,1} - J_Q$ convolutional layers is bounded in terms of the width $\dim\left(h^{(J_Q)}(x)\right)$ by

$$\begin{aligned} \mathcal{N}_3(Q &\leq (4s+1)(J_{Q+1,1} - J_Q) + \dim\left(h^{(J_Q)}(x)\right) \\ &\leq 9(d2^P N' + d2^{Q-1}) + \dim\left(h^{(J_Q)}(x)\right) + 4s + 1. \end{aligned} \tag{5.9}$$

Then we take $L_1 = L_Q$, $L = 2d2^P N'$, $L_2 = L'_{Q+1,1}$, $V = U$ in Lemma 2, and we can see there exist $L_U = L_U(Q)$, $L'_U = L'_U(Q) \in \mathbb{Z}_+$, $\{w^{(j)}\}_{j=K_Q+1}^{J_{Q+1,2}}$ and $\{b^{(j)}\}_{j=K_Q+1}^{J_{Q+1,2}}$ such that

$$\begin{aligned} h^{(J_{Q+1,2})}(x) = \Big[&\mathbf{0}_{L_U}, H^T_{Q,\mu(1)}, \ldots, H^T_{Q,\mu(N')}, \hat{H}^T_{Q,\mu(1)}, \ldots, \hat{H}_{Q,\mu(N')}, \mathbf{0}_{14d2^P N'}, \\ &\left((\mathrm{id} - R_U)\,(H_{Q,\mu(1)})\right)^T, \ldots, \left((\mathrm{id} - R_U)\,(H_{Q,\mu(N')})\right)^T, \\ &\left((\mathrm{id} - R_U)\,(\hat{H}_{Q,\mu(1)})\right)^T, \ldots, \left((\mathrm{id} - R_U)\,(\hat{H}_{Q,\mu(1)})\right)^T, \mathbf{0}_{L'_U}\Big]^T, \end{aligned}$$

where

$$J_{Q+1,2} - J_{Q+1,1} \leq \frac{2(7U+15)d2^P N'}{s-1} + 3U + 2$$

and the number of free parameters of these convolutional layers is bounded by (4.7) as

$$\mathcal{N}_4(Q) \leq 2(6U+10)(7U+15)d2^P N' + (3U+2)(3U+5)s + (3U+2)\dim\left(h^{(J_Q)}(x)\right). \tag{5.10}$$

Finally, another application of Lemma 1 with $L_1 = L_U(Q)$, $m = 18d2^P N'$, $L_2 = L'_U(Q)$, $M = 1$, W represented by

$$[W]_0^{d2^P N' + d2^{Q-1}} = [1, \mathbf{0}_{d2^P N'-1}, -\frac{1}{4}, \mathbf{0}_{2^{Q-1}-1}, -\frac{1}{4}]^T,$$

B by

$$[B]_1^{19d2^P N' + d2^{Q-1}} = [\mathbf{2}_{17d2^P N'}, \mathbf{0}_{2d2^P N' + 2^{Q-1}}]^T,$$

and

$$\begin{aligned}\check{z} = \Big[\; & H^T_{Q,\mu(1)}, \ldots, H^T_{Q,\mu(N')}, \hat{H}^T_{Q,\mu(1)}, \ldots, \hat{H}_{Q,\mu(N')}, \mathbf{0}_{14d2^P N'}, \\ & \left((\mathrm{id} - R_U)\left(H_{Q,\mu(1)}\right)\right)^T, \ldots, \left((\mathrm{id} - R_U)\left(H_{Q,\mu(N')}\right)\right)^T, \\ & \left((\mathrm{id} - R_U)\left(\hat{H}_{Q,\mu(1)}\right)\right)^T, \ldots, \left((\mathrm{id} - R_U)\left(\hat{H}_{Q,\mu(1)}\right)\right)^T \Big]^T\end{aligned}$$

tells us that there exist $J_{Q+1} \leq J_{Q+1,2} + \left\lceil \frac{d2^P N' + 2^{Q-1}}{s-1} \right\rceil$, $\{w^{(j)}\}_{j=J_{Q+1,2}+1}^{J_{Q+1}}$ and $\{b^{(j)}\}_{j=J_{Q+1,2}+1}^{J_{Q+1}}$ such that

$$\begin{aligned} h^{(J_{Q+1})}(x) &= \sigma\left(\begin{bmatrix} \mathbf{0}_{L_U} \\ [z * W - B]_1^{19d2^P N' + d2^{Q-1}} \\ \mathbf{0}_{L'_U + (J_{Q+1} - J_{Q+1,2})s - d2^P N' - 2^{Q-1}} \end{bmatrix}\right) \\ &= \left[\mathbf{0}_{L_U + 16d2^P N' + d2^P N'}, H^T_{Q+1,\mu(1)}, \ldots, H^T_{Q+1,\mu(N')}, \mathbf{0}_{L'_U + (J_{Q+1} - J_{Q+1,2})s}\right]^T \end{aligned} \tag{5.11}$$

where $H_{Q+1,\mu(k)} \in [0,1]^{d \times 2^P}$ and by direct computation the $d \times 2^{Q-1} \times 2r$-th component of $H_{Q+1,\mu(k)}$ is

$$\begin{aligned} & \left((\mathrm{id} - R_U)\left(\hat{H}_{Q,\mu(k)}\right)\right)_{d \times 2^{Q-1} \times 2r} - \left((\mathrm{id} - R_U)\left(H_{Q,\mu(k)}\right)\right)_{d \times 2^{Q-1} \times 2r} \\ & - \left((\mathrm{id} - R_U)\left(H_{Q,\mu(k)}\right)\right)_{d \times 2^{Q-1} \times (2r-1)} \\ = {} & \tfrac{1}{2}(\mathrm{id} - R_U)\left(\tfrac{1}{2}\left(\Lambda(k, Q, 2r; x) + \Lambda(k, Q, 2r-1; x)\right)\right) \\ & -\tfrac{1}{4}\left[(\mathrm{id} - R_U)\left(\Lambda(k, Q, 2r; x)\right)\right] - \tfrac{1}{4}\left[(\mathrm{id} - R_U)\left(\Lambda(k, Q, 2r-1; x)\right)\right] \\ = {} & \widetilde{\times}\left(\Lambda(k, Q, 2r; x), \Lambda(k, Q, 2r-1; x)\right) \\ = {} & \Lambda(k, Q+1, r; x). \end{aligned}$$

The number of free parameters of these layers is bounded by

$$\begin{aligned} \mathcal{N}_5(Q) &\leq (4s+1)(J_{Q+1} - J_{Q+1,2}) + \dim\left(h^{(J_{Q+1,2})}(x)\right) \\ &\leq 9(d2^P N' + d2^{Q-1}) + \dim\left(h^{(J_{Q+1,2})}(x)\right) + 4s + 1. \end{aligned} \tag{5.12}$$

Hence $h^{(J_{Q+1})}$ satisfies (5.8) with Q replaced by $Q+1$, $L_{Q+1} = L_U(Q) + 16d2^P N' + d2^P N'$ and $L'_{Q+1} = L'_U(q) + \left(J_{Q+1} - J_{Q+1,2}\right)s$. This completes the induction procedure. The final step of the procedure with $Q+1 = P+1$ gives

$$h^{(J_{P+1})}(x)^T = [\mathbf{0}^T_{L_{P+1}}, H^T_{P+1,\mu(1)}, \ldots, H^T_{P+1,\mu(N')}, \mathbf{0}^T_{L'_{P+1}}], \tag{5.13}$$

where for each $\mu(k)$, the $d2^P$-th component of $H_{P+1,\mu(k)}$ is

$$\left(H_{P+1,\mu(k)}\right)_{d2^P} = \Lambda(k, P+1, 1; x).$$

Let $J = J_{P+1}$, and $c \in \mathbb{R}^{d_J}$ be the vector given by

$$[\mathbf{0}_{L_{P+1}}, \mathbf{0}_{d2^P-1}, v_{\mu(1)}, \mathbf{0}_{d2^P-1}, v_{\mu(2)}, \ldots, \mathbf{0}_{d2^P-1}, v_{\mu(N'-1)}, \mathbf{0}_{d2^P-1}, v_{\mu(N')}, \mathbf{0}_{L'_{P+1}}]^T,$$

where

$$v_{l,i} = \int_{[0,1]^d} \prod_{j=1}^{d} \left(-2^{l_j+1} \phi_{l_j,i_j}(x_j)\right) \frac{\partial^{2d} f}{\partial x_1^2 \ldots \partial x_d^2}(x) dx. \tag{5.14}$$

Then

$$f_J^{\mathbf{w},\mathbf{b}}(x) = \sum_{i=1}^{d_J} c_i h_i^{(J)}(x) = \sum_{k=1}^{N'} v_{\mu(k)} \Lambda(k, P+1, 1; x) \tag{5.15}$$

is the desired output function constructed by our deep CNN network.

5.3 Complexity analysis

We analyze the complexity of our CNN network by counting its depth and number of free parameters. The depth is bounded as

$$\begin{aligned} J &= J_0 + (J_1 - J_0) + \sum_{Q=1}^{P} \left[(J_{Q+1} - J_{Q+1,2}) + (J_{Q+1,2} - J_{Q+1,1}) + (J_{Q+1,1} - J_Q)\right] \\ &\le \tfrac{(7UP+16P+4)}{s-1} d2^P N' + (3U+4)P + 2. \end{aligned}$$

By means of the fact that $P \le 1 + \log_2 d$ and $U \le 3 \log_2 N'$, we have

$$J \le \left(\frac{168}{s-1} + 2\right) \left(\log_2 d\right) d^2 (\log_2 N) N.$$

Given the upper bound of the depth J, for any $j \le J$, the width of the layer $h^{(j)}(x)$ can be bounded as

$$\dim \left(h^{(j)}(x)\right) \le d + Js \le \left(\frac{168}{s-1} + 2\right) \left(\log_2 d\right) d^2 (\log_2 N) Ns + d.$$

The total number of free parameters in our network can be bounded as

$$\begin{aligned} \mathcal{N} &= \mathcal{N}_1 + \mathcal{N}_2 + \sum_{Q=1}^{P} \left[\mathcal{N}_3(Q) + \mathcal{N}_4(Q) + \mathcal{N}_5(Q)\right] + \dim\left(h^{(J)}(x)\right) \\ &\le [2(6U+10)(7U+15)P + 18P + 32]\, d2^P N' + [(3U+2)(3U+5)P + 4P + 9]\, s \\ &\quad + [(3U+4)P + 1] \dim\left(h^{(J)}(x)\right) + P - 9. \end{aligned}$$

Since $N \ge 2^{16}$, we have $N' \ge 2^{15}$, using the previous upper bounds and the fact that $s < d$ we can conclude

$$\mathcal{N} \le 13385 d^2 (\log_2 d)^2 (\log_2 N)^2 N. \tag{5.16}$$

6 Estimating the approximation error

Now we carry out our error estimates. Take an intermediate function $f_n^{(1)}$ defined on $[0, 1]^d$ by

$$f_n^{(1)} = \sum_{|\boldsymbol{l}|_1 \le n+d-1} \sum_{\boldsymbol{i} \in \boldsymbol{I}_{\boldsymbol{l}}} v_{\boldsymbol{l},\boldsymbol{i}} \phi_{\boldsymbol{l},\boldsymbol{i}} \tag{6.1}$$

The basis of hat functions provides nice bounds [1] for the error term $\| f_n^{(1)} - f \|_p$ when f is from the Korobov space $X^{2,p}\left([0, 1]^d\right)$. Then we make use of the so-called *0-in-0-out* property of the map $\widetilde{\times}$ applied in [16] in the case $p = \infty$ to bound $\| f_n^{(1)} - f_J^{\mathbf{w},\mathbf{b}} \|_p$.

Proof of Theorem 1 A series expansion of $f \in W^{2,p}\left([0, 1]^d\right)$ in terms of the basis $\left\{\phi_{\boldsymbol{l},\boldsymbol{i}}\right\}$ found in [1] (3.19) and (3.24) provides an expansion of the error function $f - f_n^{(1)}$ in L^p as

$$f - f_n^{(1)} = \sum_{|\boldsymbol{l}|_1 > n+d-1} \sum_{\boldsymbol{i} \in \boldsymbol{I}_{\boldsymbol{l}}} v_{\boldsymbol{l},\boldsymbol{i}} \phi_{\boldsymbol{l},\boldsymbol{i}}. \tag{6.2}$$

Observe that $\operatorname{supp}(\phi_{\boldsymbol{l},\boldsymbol{i}}) \bigcap \operatorname{supp}(\phi_{\boldsymbol{l},\boldsymbol{i}'}) = \varnothing$ for $\boldsymbol{i} \neq \boldsymbol{i}'$. We first estimate in the case $p = \infty$. By (6.1),

$$\begin{aligned} \| f - f_n^{(1)} \|_\infty &\le \sum_{|\boldsymbol{l}|_1 > n+d-1} \max_{\boldsymbol{i} \in \boldsymbol{I}_{\boldsymbol{l}}} |v_{\boldsymbol{l},\boldsymbol{i}}| \le \sum_{k > n+d-1} 2^{-2k} k^{d-1} \le 2 \times 2^{-2n} n^{d-1} \\ &\le \left(\log_2 N\right)^{3(d-1)} N^{-2}, \end{aligned} \tag{6.3}$$

where the second inequation follows from [1, Lemma 3.3].

It remains to estimate the distance between $f_J^{\mathbf{w},\mathbf{b}}$ and $f_n^{(1)}$. We claim that for $Q \in \{1, \ldots, P + 1\}$, the functions $\{\Lambda(k, Q, r; \cdot)\}_{r=1}^{2^{P-Q+1}}$ satisfy

$$\left| \Lambda(k, Q, r; x) - \prod_{j=2^{Q-1} \times (r-1)+1}^{2^{Q-1} \times r} \phi_{\mu(k)_j}(x_j) \right| \le \frac{2^{Q-1} - 1}{4dN^2}, \quad x \in [0, 1]^d,$$

$\Lambda(k, Q, r; x) = 0$ whenever $\prod_{j=2^{Q-1} \times (r-1)+1}^{2^{Q-1} \times r} \phi_{\mu(k)_j}(x_j) = 0$, and $\Lambda(k, Q, r; x) \in [0, 1]$ for all $x \in [0, 1]^d$.

We prove our claim by induction. The case $Q = 1$ is trivial since

$$\Lambda(k, 1, r; x) = \phi_{\mu(k)_r}(x_r), \qquad r = 1, \ldots, 2^P.$$

Suppose that the claim is true for some $Q \in \{1, \ldots, P\}$. Consider $\Lambda(k, Q + 1, r; x)$ with some $r \in \{1, \ldots, 2^{P-Q}\}$. By our construction,

$$\Lambda(k, Q, r; x) = \widetilde{\times} \left(\Lambda(k, Q, 2r - 1; x), \Lambda(k, Q, 2r; x)\right).$$

A direct computation shows that, on the domain $[0, 1]$, the function $\mathrm{id} - R_U$ is exactly the linear interpolation of the univariable quadratic polynomial t^2 at the points

$\left\{\left(\frac{k}{2^U}, (\frac{k}{2^U})^2\right)\right\}_{k=0}^{2^U}$ (see also [20]). Then for any $t_1,\ t_2 \in [0, 1]$, there holds

$$\widetilde{\times}\,(t_1, t_2) = 0 \quad \text{if } t_1 t_2 = 0 \tag{6.4}$$

and

$$\begin{aligned}&\left|\widetilde{\times}\,(t_1, t_2) - t_1 t_2\right|\\ \leq\ &\left|(\mathrm{id} - R_U)\,(\tfrac{t_1+t_2}{2}) - (\tfrac{t_1+t_2}{2})^2\right| + \tfrac{1}{4}|(\mathrm{id} - R_U)\,(t_1) - t_1^2| + \tfrac{1}{4}|(\mathrm{id} - R_U)\,(t_2) - t_2^2|\\ \leq\ &\tfrac{1}{2^U} + \tfrac{1}{4}\tfrac{1}{2^U} + \tfrac{1}{4}\tfrac{1}{2^U} \leq \tfrac{1}{4dN^2}.\end{aligned}$$

Hence for any $x \in [0, 1]^d$,

$$\begin{aligned}&\left|\Lambda(k, Q+1, r; x) - \prod_{j=2^Q\times(r-1)+1}^{2^Q\times r} \phi_{\mu(k)_j}(x_j)\right|\\ \leq\ &\left|\widetilde{\times}\,(\Lambda(k, Q, 2r; x), \Lambda(k, Q, 2r-1; x)) - \Lambda(k, Q, 2r; x) \times \Lambda(k, Q, 2r-1; x)\right|\\ &+ \left|\Lambda(k, Q, 2r; x)\Lambda(k, Q, 2r-1; x) - \prod_{j=2^{Q-1}\times(2r-2)+1}^{2^{Q-1}\times 2r} \phi_{\mu(k)_j}(x_j)\right|\\ \leq\ &\tfrac{1}{4dN^2} + 2 \times \tfrac{2^{Q-1}-1}{4dN^2} = \tfrac{2^Q-1}{4dN^2}.\end{aligned}$$

Furthermore, if $\prod_{j=2^Q\times(r-1)+1}^{2^Q\times r} \phi_{\mu(k)_j}(x_j) = 0$, then $\prod_{j=2^{Q-1}\times(2r-2)+1}^{2^{Q-1}\times(2r-1)} \phi_{\mu(k)_j}(x_j) = 0$ or $\prod_{j=2^{Q-1}\times(2r-1)+1}^{2^{Q-1}\times 2r} \phi_{\mu(k)_j}(x_j) = 0$. By the induction hypothesis, this implies $\Lambda(k, Q, 2r; x)\Lambda(k, Q, 2r-1; x)$ vanishes and thereby

$$\Lambda(k, Q+1, r; x) = \widetilde{\times}\,(\Lambda(k, Q, 2r; x), \Lambda(k, Q, 2r-1; x)) = 0.$$

Finally, it is easy to verify $\Lambda(k, Q+1, r; x) \in [0, 1]$, hence we complete the induction and verify our claim. From the proved claim, we know that

$$\left|\Lambda(k, P+1, 1; x) - \prod_{j=1}^{2^P} \phi_{\mu(k)_j}(x_j)\right| \leq \frac{2^P - 1}{4dN^2} \leq \frac{1}{N^2}$$

and $\mathrm{supp}\,(\Lambda(k, P+1, 1; \cdot)) \subseteq \mathrm{supp}(\phi_{\mu(k)})$.

Since $\mathrm{supp}(\phi_{\boldsymbol{l},\boldsymbol{i}}) \bigcap \mathrm{supp}(\phi_{\boldsymbol{l},\boldsymbol{i}'}) = \varnothing$, we have

$$|f_n^{(1)}(x) - f_J^{\mathbf{w},\mathbf{b}}(x)| \leq \frac{1}{N^2} \sum_{\boldsymbol{l}\in\mathbb{N}^d} \max_{\boldsymbol{i}\in\boldsymbol{I}_{\boldsymbol{l}}} |v_{\boldsymbol{l},\boldsymbol{i}}| \leq \frac{1}{N^2} \sum_{k=1}^{\infty} 2^{-2k} k^{d-1} \leq \frac{1}{N^2}. \tag{6.5}$$

Together with (6.3), $f_J^{\mathbf{w},\mathbf{b}}$ is a function constructed by a deep neural networks which satisfies

$$\|f - f_J^{\mathbf{w},\mathbf{b}}\|_\infty \leq \left(\left(\log_2 N\right)^{3(d-1)} + 1\right) N^{-2}. \tag{6.6}$$

This completes the estimate in the case $p = \infty$.

Then, we turn to the case $1 \leq p < \infty$. Notice that

$$\|f\|_{\mathbf{2},p} = \left\| \frac{\partial^{2d} f}{\partial x_1^2 \dots \partial x_d^2} \right\|_p \leq 1.$$

By (6.2) we have

$$\|f - f_n^{(1)}\|_p \leq \sum_{|\boldsymbol{l}|_1 > n+d-1} \left\| \sum_{\boldsymbol{i} \in \boldsymbol{I_l}} v_{\boldsymbol{i},\boldsymbol{l}} \phi_{\boldsymbol{i},\boldsymbol{l}} \right\|_p .$$

Since $\operatorname{supp}(\phi_{\boldsymbol{l},\boldsymbol{i}}) \bigcap \operatorname{supp}(\phi_{\boldsymbol{l},\boldsymbol{i}'}) = \varnothing$ for $\boldsymbol{i} \neq \boldsymbol{i}'$, we have

$$\begin{aligned} &\int_{[0,1]^d} \left| \sum_{\boldsymbol{j} \in \boldsymbol{I_l}} v_{\boldsymbol{j},\boldsymbol{l}} \phi_{\boldsymbol{j},\boldsymbol{l}} \right|^p dx = \sum_{\boldsymbol{i} \in \boldsymbol{I_l}} \int_{supp(\phi_{\boldsymbol{l},\boldsymbol{i}})} \left| \sum_{\boldsymbol{j} \in \boldsymbol{I_l}} v_{\boldsymbol{j},\boldsymbol{l}} \phi_{\boldsymbol{j},\boldsymbol{l}} \right|^p dx \\ &= \sum_{\boldsymbol{i} \in \boldsymbol{I_l}} \int_{supp(\phi_{\boldsymbol{l},\boldsymbol{i}})} \left| v_{\boldsymbol{i},\boldsymbol{l}} \phi_{\boldsymbol{i},\boldsymbol{l}} \right|^p dx \\ &\leq \left(\tfrac{2}{p+1} \right)^d 2^{-|\boldsymbol{l}|_1} \sum_{\boldsymbol{i} \in \boldsymbol{I_l}} |v_{\boldsymbol{i},\boldsymbol{l}}|^p, \end{aligned}$$

where the last inequation is a consequence of [1, Lemma 3.1].

By the explicit expression (5.14) for the expansion coefficients $\{v_{\boldsymbol{l},\boldsymbol{i}}\}$ and [1, Lemma 3.1], for any $\boldsymbol{l} \in \mathbb{N}^d$ and $\boldsymbol{i} \in \boldsymbol{I_l}$, we have

$$\begin{aligned} |v_{\boldsymbol{l},\boldsymbol{i}}| &= 2^{-|\boldsymbol{l}|_1 - d} \left| \int_{[0,1]^d} \phi_{\boldsymbol{l},\boldsymbol{i}}(x) \tfrac{\partial^{2d} f}{\partial x_1^2 \dots \partial x_d^2}(x) dx \right| \leq 2^{-|\boldsymbol{l}|_1 - d} \|\phi_{\boldsymbol{l},\boldsymbol{i}}\|_q \left\| \tfrac{\partial^{2d} f}{\partial x_1^2 \dots \partial x_d^2} \right\|_p \\ &\leq 2^{-|\boldsymbol{l}|_1 - d} \left(\tfrac{2}{q+1} \right)^{\frac{d}{q}} 2^{-\frac{|\boldsymbol{l}|_1}{q}}, \end{aligned}$$

where q is the dual number of p given by $q = \frac{p}{p-1}$ if $p > 1$ and $q = \infty$ if $p = 1$.

Therefore,

$$\begin{aligned} \left\| \sum_{\boldsymbol{i} \in \boldsymbol{I_l}} v_{\boldsymbol{i},\boldsymbol{l}} \phi_{\boldsymbol{i},\boldsymbol{l}} \right\|_p &\leq \left\{ \left(\tfrac{2}{p+1} \right)^d 2^{-|\boldsymbol{l}|_1} \sum_{\boldsymbol{i} \in \boldsymbol{I_l}} \left[2^{-|\boldsymbol{l}|_1 - d} \left(\tfrac{2}{q+1} \right)^{\frac{d}{q}} 2^{-\frac{|\boldsymbol{l}|_1}{q}} \right]^p \right\}^{\frac{1}{p}} \\ &\leq \left(\tfrac{2}{q+1} \right)^{\frac{d}{q}} \left(\tfrac{2}{p+1} \right)^{\frac{d}{p}} 2^{-|\boldsymbol{l}|_1 \left(1 + \frac{1}{q}\right) - d} \leq 2^{-\left(2 - \frac{1}{p}\right)|\boldsymbol{l}|_1} \end{aligned} \tag{6.7}$$

and

$$\begin{aligned} \|f - f_n^{(1)}\|_p &\leq \sum_{|\boldsymbol{l}|_1 > n+d-1} 2^{-\left(2 - \frac{1}{p}\right)|\boldsymbol{l}|_1} \leq \sum_{k > n+d-1} 2^{-\left(2 - \frac{1}{p}\right)k} k^{d-1} \\ &\leq 2 \times 2^{-\left(2 - \frac{1}{p}\right)n} n^{d-1} \leq \left(\log_2 N\right)^{\left(3 - \frac{1}{p}\right)(d-1)} N^{-\left(2 - \frac{1}{p}\right)}. \end{aligned} \tag{6.8}$$

On the other hand,

$$\|f_n^{(1)} - f_J^{\mathbf{w},\mathbf{b}}\|_p \leq \|f_n^{(1)} - f_J^{\mathbf{w},\mathbf{b}}\|_\infty \leq \frac{1}{N^2}. \tag{6.9}$$

Hence

$$\|f - f_J^{\mathbf{w},\mathbf{b}}\|_p \leq \left(\left(\log_2 N\right)^{\left(3 - \frac{1}{p}\right)(d-1)} + 1 \right) N^{-\left(2 - \frac{1}{p}\right)}. \tag{6.10}$$

This verifies the desired error bound (2.1).

The bounds for J and $\mathcal{N}$ were proved in Section 5.3 . The proof of Theorem 1 is complete. □

Appendix: Proof of Lemma 2

In this appendix, we prove Lemma 2.

Proof of Lemma 2 We show how the iterations of tooth functions can be realized by DCNNs.

For the 1st step, we take in Lemma 1 that $L_1 = L_1$, $m = L$, $L_2 = L_2$, $M = 1$, W represented by

$$[W]_0^{7L} = [1, \mathbf{0}_{4L-1}, 1, \mathbf{0}_{3L}]^T,$$

B by $[B]_1^{8L} = \mathbf{0}_{8L}$, and $\check{z} = y$. Then we conclude there exist filters $\{w^{(j)}\}_{j=1}^{K_0}$ and biases $\{b^{(j)}\}_{j=1}^{K_0}$ satisfying (4.2) such that

$$h^{(K_0)}(\hat{y}) = \begin{bmatrix} \mathbf{0}_{L_1} \\ [z * W]_1^{8L} \\ \mathbf{0}_{n_0} \end{bmatrix} = \left[\mathbf{0}_{L_1}, y^T, \mathbf{0}_{3L}, y^T, \mathbf{0}_{3L}, \mathbf{0}_{n_0}\right]^T,$$

where $K_0 \le \left\lceil \frac{7L}{s-1} \right\rceil$ and $n_0 = L_2 + K_0 s - 7L$.

For the $u+1$-th step, we assume the K_u-th layer has the form

$$h^{(K_u)}(\hat{y}) = \left[\mathbf{0}_{L_{1,u}}, y^T, \mathbf{0}_{3L}, T_u(y)^T, \mathbf{0}_{2L}, R_u(y)^T, \mathbf{0}_{L_{2,u}}\right]. \tag{6.1}$$

Notice $h^{(K_0)}$ already has this form (see Definition 3).

Now following from Lemma 1 by letting $L_1 = L_{1,u}$, $m = 8L$, $L_2 = L_{2,u}$, $M = 1$, W represented by

$$[W]_0^{2L} = [2, \mathbf{0}_{L-1}, 4, \mathbf{0}_{L-1}, 2]^T,$$

B by

$$[B]_1^{10L} = [\mathbf{4}_{2L}, \mathbf{0}_{3L}, \left(\mathbf{2^{-u+1}}\right)_{2L}, \mathbf{4}_{2L}, \mathbf{0}_L]^T,$$

and

$$\check{z} = \left[y^T, \mathbf{0}_{3L}, T_u(y)^T, \mathbf{0}_{2L}, R_u(y)^T\right],$$

we find there exist filters $\{w^{(j)}\}_{j=K_u+1}^{K_{u+1,1}}$ and biases $\{b^{(j)}\}_{j=K_u+1}^{K_{u+1,1}}$ satisfying the restriction (4.2) such that

$$\begin{aligned} h^{(K_{u+1,1})}(\hat{y}) &= \sigma\left(\begin{bmatrix} \mathbf{0}_{L_{1,u}} \\ [z * W - B]_1^{10L} \\ \mathbf{0}_{L_{2,u}+(K_{u+1,1}-K_u)s-2L} \end{bmatrix}\right) \\ &= \sigma\left(\begin{bmatrix} \mathbf{0}_{L_{1,u}} \\ 2y \\ 4y \\ 2y \\ \mathbf{0}_L \\ 2T_u(y) \\ 4T_u(y) \\ 2T_u(y) \\ 2R_u(y) \\ 4R_u(y) \\ 2R_u(y) \\ \mathbf{0}_{n_{u+1,1}} \end{bmatrix} - \begin{bmatrix} \mathbf{0}_{L_{1,u}} \\ \mathbf{4}_L \\ \mathbf{4}_L \\ \mathbf{0}_L \\ \mathbf{0}_L \\ \mathbf{0}_L \\ \left(\mathbf{2^{-u+1}}\right)_L \\ \left(\mathbf{2^{-u+1}}\right)_L \\ \mathbf{4}_L \\ \mathbf{4}_L \\ \mathbf{0}_L \\ \mathbf{0}_{n_{u+1,1}} \end{bmatrix}\right) = \begin{bmatrix} \mathbf{0}_{L_{1,u}} \\ \mathbf{0}_{2L} \\ 2y \\ \mathbf{0}_L \\ 2^{-u}\sigma(2S_u(y)) \\ 2^{-u}\sigma(4S_u(y) - \mathbf{2}_L) \\ 2^{-u}\sigma(2S_u(y) - \mathbf{2}_L) \\ \mathbf{0}_{2L} \\ 2R_u(y) \\ \mathbf{0}_{n_{u+1,1}} \end{bmatrix}, \end{aligned}$$

where $n_{u+1,1} = L_{2,u} + \left(K_{u+1,1} - K_u\right)s - 2L$ and the number of layers $K_{u+1,1} - K_u$ is bounded by $\left\lceil \frac{2L}{s-1} \right\rceil$.

Again, appealing Lemma 1 by letting $L_1 = L_{1,u} + 2L$, $m = 8L$, $L_2 = n_{u+1,1}$, $M = 2$, W represented by

$$[W]_0^{2L} = [1, \mathbf{0}_{L-1}, -1, \mathbf{0}_{L-1}, 1]^T,$$

B by

$$[B]_1^{10L} = [\mathbf{0}_L, \mathbf{4}_{3L}, \mathbf{0}_L, \mathbf{4}_{2L}, \mathbf{0}_L, \mathbf{4}_{2L}]^T,$$

and

$$\check{z} = \Big[\, 2y^T, \mathbf{0}_L, 2^{-u}\sigma(2S_u(y))^T, 2^{-u}\sigma(4S_u(y) - \mathbf{2}_L)^T, \\ 2^{-u}\sigma(2S_u(y) - \mathbf{2}_L)^T, \mathbf{0}_{2L}, 2R_u(y)^T \Big],$$

we find there exist filters $\{w^{(j)}\}_{j=K_{u+1,1}+1}^{K_{u+1,2}}$, biases $\{b^{(j)}\}_{j=K_{u+1,1}+1}^{K_{u+1,2}}$ satisfying the restriction (4.2) such that

$$
\begin{aligned}
h^{(K_{u+1,2})}(\hat{y}) &= \sigma\left(\begin{bmatrix} \mathbf{0}_{L_{1,u}+2L} \\ [z * W - B]_1^{10L} \\ \mathbf{0}_{n_{u+1,1}+(K_{u+1,2}-K_{u+1,1})s-2L} \end{bmatrix}\right) \\
&= \sigma\left(\begin{bmatrix} \mathbf{0}_{L_{1,u}+2L} \\ 2y \\ -2y \\ 2y + 2^{-u}\sigma(2S_u(y)) \\ -2^{-u}\sigma(2S_u(y)) + 2^{-u}\sigma(4S_u(y) - \mathbf{2}_L) \\ 2^{-u}\sigma(2S_u(y)) - 2^{-u}\sigma(4S_u(y) - \mathbf{2}_L) + 2^{-u}\sigma(2S_u(y) - \mathbf{2}_L) \\ 2^{-u}\sigma(4S_u(y) - \mathbf{2}_L) - 2^{-u}\sigma(2S_u(y) - \mathbf{2}_L) \\ 2^{-u}\sigma(2S_u(y) - \mathbf{2}_L) \\ 2R_u(y) \\ -2R_u(y) \\ 2R_u(y) \\ \mathbf{0}_{n_{u+1,2}} \end{bmatrix} - \begin{bmatrix} \mathbf{0}_{L_{1,u}+2L} \\ \mathbf{0}_L \\ \mathbf{4}_L \\ \mathbf{4}_L \\ \mathbf{4}_L \\ \mathbf{0}_L \\ \mathbf{4}_L \\ \mathbf{4}_L \\ \mathbf{0}_L \\ \mathbf{4}_L \\ \mathbf{4}_L \\ \mathbf{0}_{n_{u+1,2}} \end{bmatrix}\right) \\
&= \left[\mathbf{0}_{L_{1,u}+2L}, 2y^T, \mathbf{0}_{3L}, 2T_{u+1}(y)^T, \mathbf{0}_{2L}, 2R_u(y)^T, \mathbf{0}_{n_{u+1,2}+2L}\right]^T,
\end{aligned}
$$

where $n_{u+1,2} = n_{u+1,1} + \left(K_{u+1,2} - K_{u+1,1}\right)s - 2L$ and the number of layers $K_{u+1,2} - K_{u+1,1}$ is bounded by $\left\lceil \frac{2L}{s-1} \right\rceil$.

Again we can deduce from putting $L_1 = L_{1,u} + 2L$, $m = 8L$, $L_2 = n_{u+1,2} + 2L$, $M = 2$, W represented by

$$[W]_0^{3L} = [\frac{1}{2}, \mathbf{0}_{3L-1}, \frac{1}{2}]^T,$$

B by

$$[B]_1^{11L} = [\mathbf{0}_{3L}, \mathbf{1}_L, \mathbf{0}_{6L}, \mathbf{1}_L]^T,$$

and

$$\check{z} = \left[2y^T, \mathbf{0}_{3L}, 2T_{u+1}(y)^T, \mathbf{0}_{2L}, 2R_u(y)^T\right]^T$$

in Lemma 1 that there exist filters $\{w^{(j)}\}_{j=K_{u+1,1}+1}^{K_{u+1,2}}$, biases $\{b^{(j)}\}_{j=K_{u+1,1}+1}^{K_{u+1,2}}$ satisfying the restriction (4.2) such that

$$\begin{aligned}
h^{(K_{u+1,3})}(y) &= \sigma\left(\begin{bmatrix} \mathbf{0}_{L_{1,u}+2L} \\ [z * W - B]_1^{11L} \\ \mathbf{0}_{n_{u+1,2}+(K_{u+1,3}-K_{u+1,2})s-3L} \end{bmatrix}\right) \\
&= \sigma\left(\begin{bmatrix} \mathbf{0}_{L_{1,u}+2L} \\ y \\ \mathbf{0}_{2L} \\ y \\ T_{u+1}(y) \\ \mathbf{0}_{2L} \\ R_u(y) + T_{u+1}(y) \\ \mathbf{0}_{2L} \\ R_u(y) \\ \mathbf{0}_{n_{u+1,3}} \end{bmatrix} - \begin{bmatrix} \mathbf{0}_{L_{1,u}+2L} \\ \mathbf{0}_L \\ \mathbf{0}_{2L} \\ \mathbf{1}_L \\ \mathbf{0}_L \\ \mathbf{0}_{2L} \\ \mathbf{0}_L \\ \mathbf{0}_{2L} \\ \mathbf{1}_L \\ \mathbf{0}_{n_{u+1,3}} \end{bmatrix}\right) \\
&= \left[\mathbf{0}_{L_{1,u}+2L}, y^T, \mathbf{0}_{3L}, T_{u+1}(y)^T, \mathbf{0}_{2L}, R_{u+1}(y)^T, \mathbf{0}_{n_{u+1,3}+3L}\right]^T,
\end{aligned}$$

where $n_{u+1,3} = n_{u+1,2} + \left(K_{u+1,3} - K_{u+1,2}\right)s - 3L$ and the number of layers $K_{u+1,3} - K_{u+1,2}$ is bounded by $\left\lceil \frac{3L}{s-1} \right\rceil$.

Let $K_{u+1} = K_{u+1,3}$, $L_{1,u+1} = L_{1,u} + 2L$ and $L_{2,u+1} = n_{u+1,3} + 3L$. This is exactly the form (6.1). By repeating this process V times, from the input y we obtain

$$h^{(K_V)}(y) = \left[\mathbf{0}_{L_{1,V}}, y^T, \mathbf{0}_{3L}, T_V(y)^T, \mathbf{0}_{2L}, R_V(y)^T, \mathbf{0}_{L_{2,V}}\right]^T.$$

To realize (4.5), we only need to construct $y - R_V(y)$ by a deep CNN. Applying Lemma 1 to $L_1 = L_{1,V}$, $m = 8L$, $L_2 = L_{2,V}$, $M = 1$, W represented by

$$[W]_0^{8L} = [1, \mathbf{0}_{L-1}, -1, \mathbf{0}_{7L-1}, 1]^T,$$

B by

$$[B]_1^{16L} = [\mathbf{0}_{4L}, \mathbf{1}_L, \mathbf{0}_{10L}, \mathbf{1}_L]^T,$$

and

$$\check{z} = \left[y^T, \mathbf{0}_{3L}, T_V(y)^T, \mathbf{0}_{2L}, R_V(y)^T\right]^T,$$

we see that there exist filters $\{w^{(j)}\}_{j=K_V+1}^{K}$, biases $\{b^{(j)}\}_{j=K_V+1}^{K}$ satisfying the restriction (4.2) such that

$$\begin{aligned} h^{(K)}(y) &= \sigma\left(\begin{bmatrix} \mathbf{0}_{L_{1,V}} \\ [z * W - B]_1^{16L} \\ \mathbf{0}_{L_{2,V}+(K-K_{V+1})s-8L} \end{bmatrix}\right) \\ &= \sigma\left(\begin{bmatrix} \mathbf{0}_{L_{1,V}} \\ y \\ -y \\ \mathbf{0}_{2L} \\ T_V(y) \\ -T_V(y) \\ \mathbf{0}_L \\ R_V(y) \\ y - R_V(y) \\ \mathbf{0}_{6L} \\ R_V(y) \\ \mathbf{0}_{n_{V+1}} \end{bmatrix} - \begin{bmatrix} \mathbf{0}_{L_{1,V}} \\ \mathbf{0}_L \\ \mathbf{0}_L \\ \mathbf{0}_{2L} \\ \mathbf{1}_L \\ \mathbf{0}_L \\ \mathbf{0}_L \\ \mathbf{0}_L \\ \mathbf{0}_L \\ \mathbf{0}_{6L} \\ \mathbf{1}_L \\ \mathbf{0}_{n_{V+1}} \end{bmatrix}\right) \\ &= \left[\mathbf{0}_{L_{1,V}}, y^T, \mathbf{0}_{7L}, y^T - R_V(y)^T, \mathbf{0}_{7L}, \mathbf{0}_{n_{V+1}}\right]^T, \end{aligned}$$

where $n_{V+1} = L_{2,V} + (K - K_V)s - 8L$ and the number of layers $K - K_V$ is bounded by $\left\lceil \frac{8L}{s-1} \right\rceil$. This is exactly (4.5) with $L_V = L_{1,V}$ and $L'_V = n_{V+1}$.

We finally count the depth K and the number of free parameters $\mathcal{N}$. At the first step, $K_0 \le \left\lceil \frac{7L}{s-1} \right\rceil + 1$. At the $u+1$-th step, $K_{u+1,1} - K_u \le \left\lceil \frac{2L}{s-1} \right\rceil$, $K_{u+1,2} - K_{u+1,1} \le \left\lceil \frac{2L}{s-1} \right\rceil$, $K_{u+1,3} - K_{u+1,2} \le \left\lceil \frac{3L}{s-1} \right\rceil$. At the last step, $K - K_V \le \left\lceil \frac{8L}{s-1} \right\rceil$. Therefore,

$$K \le \frac{(7V + 15)L}{s - 1} + 3V + 2.$$

The dimension of each bias $b^{(j)}$ are bounded by $\dim(b^{(j)}) \le d_K \le L + Ks$. Then the number of free parameters in these biases $b^{(K_0)}$, $b^{(K)}$, $b^{(K_{u+1,1})}$, $b^{(K_{u+1,2})}$, $b^{(K_{u+1,3})}$, $u = 1, \ldots, p$ satisfies:

$$\mathcal{N}_6 \le (3V + 2)[\dim\left(\hat{y}\right) + Ks].$$

Together with the number of free parameters in the other layers, we have

$$\begin{aligned} \mathcal{N}_7 &\le (3V + 2)[\dim\left(\hat{y}\right) + Ks] + 3Ks \\ &\le (6V + 10)(7V + 15)L + (3V + 2)(3V + 5)s + (3V + 2)\dim\left(\hat{y}\right). \end{aligned}$$

This completes the proof of Lemma 2. □

Acknowledgements The work described in this paper is supported partially by the Laboratory for AI-Powered Financial Technologies, the Research Grants Council of Hong Kong (Projects # C1013-21GF and #11308121), the Germany/Hong Kong Joint Research Scheme (Project No. G-CityU101/20), NSFC/RGC Joint Research Scheme (RGC Project No. N_CityU102/20 and NSFC Project No. 12061160462), and Hong Kong Institute for Data Science.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Declarations

The authors have no other relevant financial or non-financial interests to disclose, except the sponsorships stated in the section of Acknowledgements.

References

1. Bungartz, H.-J., Griebel, M.: Sparse grids. Acta Numer. **13**, 147–269 (2004)
2. Chui, C.K., Lin, S.B., Zhang, B., Zhou, D.X.: Realization of spatial sparseness by deep reLU nets with massive data. IEEE Trans. Neural Netw. Learn. Syst. **33**, 229–243 (2022)
3. Chui, C.K., Lin, S.B., Zhou, D.X.: Deep neural networks for rotation-invariance approximation and learning. Anal. Appl. **17**, 737–772 (2019)
4. Eldan, R., Shamir, O.: The power of depth for feedforward neural networks. In: 29th Annual Conference on Learning Theory, PMLR, vol. 49, pp. 907–940 (2016)
5. Fang, Z., Feng, H., Huang, S., Zhou, D.X.: Theory of deep convolutional neural networks II: spherical analysis. Neural Netw. **131**, 154–162 (2020)
6. Feng, H., Hou, S.Z., Wei, L.Y., Zhou, D.X.: CNN models for readability of Chinese texts. Math. Found. Comp. **5**, 351–362 (2022)
7. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**, 1527–1554 (2006)
8. Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., Peste, A.: Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. J. Mach. Learn. Res. **22**, 1–124 (2021)
9. Klusowski, J.M., Barron, A.R.: Approximation by combinations of reLU and squared reLU ridge functions with ℓ^1 and ℓ^0 controls. IEEE Trans. Inf. Theory **64**, 7649–7656 (2018)
10. Kohler, M., Krzyżak, A.: Nonparametric regression based on hierarchical interaction models. IEEE Trans. Inf. Theory **63**, 1620–1630 (2016)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Commun. ACM **60**, 84–90 (2012)
12. Liang, S., Srikant, R.: Why deep neural networks for function approximation? In: Proceedings of international conference on learning representations (2017)
13. Lin, S.B.: Generalization and expressivity for deep nets. IEEE Trans. Neural Netw. Learn Syst. **30**, 1392–1406 (2019)
14. Mao, T., Shi, Z.J., Zhou, D.X.: Theory of deep convolutional neural networks III: Approximating radial functions. Neural Netw. **144**, 778–790 (2021)
15. Mhaskar, H.N.: Approximation properties of a multilayered feedforward artificial neural network. Adv. Comput. Math. **1**, 61–80 (1993)

16. Montanelli, H., Du, Q.: New error bounds for deep reLU networks using sparse grids. SIAM Journal on Mathematics of Data Science **1**, 78–92 (2019)
17. Pinkus, A.: Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Netw. **6**, 861–867 (1993)
18. Poggio, T., Mhaskar, H.N., Rosasco, L., Miranda, B., Liao, Q.: Why and when can deep—but not shallow—networks avoid the curse of dimensionality: a review. Internat. J. Automation Comput. **14**, 503–519 (2017)
19. Telgarsky, M.: Benefits of depth in neural networks. In: 29th Annual Conference on Learning Theory, PMLR, vol. 49, pp. 1517–1539 (2016)
20. Yarotsky, D.: Error bounds for approximations with deep reLU networks. Neural Netw. **94**, 103–114 (2017)
21. Zhou, D.X.: Universality of deep convolutional neural networks. Appl. Comput. Harmon. Anal. **48**, 787–794 (2020)
22. Zhou, D.X.: Theory of deep convolutional neural networks: Downsampling. Neural Netw. **124**, 319–327 (2020)
23. Zhou, D.X.: Deep distributed convolutional neural networks: universality. Anal. Appl. **16**, 895–919 (2018)
24. Zhou, D.X. In: Webster, J. (ed.): Deep Convolutional Neural Networks. Wiley Encyclopedia of Electrical and Electronics Engineering, Hoboken (2021). https://doi.org/10.1002/047134608X.W8424
25. Zhu, X.N., Li, Z.Y., Sun, J.: Expression recognition method combining convolutional features and Transformer, Math. Found. Comp., online first