Hindawi
Journal of Applied Mathematics
Volume 2023, Article ID 8208424, 12 pages
https://doi.org/10.1155/2023/8208424

Research Article

Error Bounds for Approximations Using Multichannel Deep Convolutional Neural Networks with Downsampling

Xinling Liu and Jingyao Hou

Key Laboratory of Optimization Theory and Applications at China West Normal University of Sichuan Province, School of Mathematics and Information, China West Normal University, Nanchong 637009, China

Correspondence should be addressed to Xinling Liu; fsliuxl@163.com

Received 30 December 2022; Revised 1 May 2023; Accepted 2 May 2023; Published 18 May 2023

Academic Editor: Zhihua Zhang

Deep learning with specific network topologies has been successfully applied in many fields. However, what is primarily called into question by people is its lack of theoretical foundation investigations, especially for structured neural networks. This paper theoretically studies the multichannel deep convolutional neural networks equipped with the downsampling operator, which is frequently used in applications. The results show that the proposed networks have outstanding approximation and generalization ability of functions from ridge class and Sobolev space. Not only does it answer an open and crucial question of why multichannel deep convolutional neural networks are universal in learning theory, but it also reveals the convergence rates.

1. Introduction

Deep learning [1] has made remarkable achievements in many fields. Essentially, it is based on structured neural networks similar to the biological nervous system to extract data features for realizing specific learning goals. In these structured neural networks, a particularly important one called deep convolutional neural networks (DCNNs) has achieved state-of-the-art performance in many domains [2–4]. Normally, multichannel convolution is used, and the resulting multichannel deep convolutional neural networks (MDCNNs) have also achieved excellent performances in classification [5, 6], natural language processing [7], biological [8–10], and many other domains [11–13].

However, compared with the successful applications of MDCNNs, the theoretical basis is incomplete, which is the main reason why it is widely criticized. In this paper, we present some approximation theories of functions for downsampled MDCNNs where the downsampling operator plays the role of pooling, which reduces the width of deep neural networks. Before giving the main results of downsampled MDCNNs, we first briefly look back at the basic concepts of fully connected neural networks (FNNs) and DCNNs.

An FNN with input vector $\mathbf{x} \in \mathbb{R}^d$ and L hidden layers of neurons $\{h^{(j)} : \mathbb{R}^d \longrightarrow \mathbb{R}^{d_j}\}$ with widths $d_j \in \mathbb{N}_+$ is defined iteratively by

$$h^{(j)}(\mathbf{x}) = \sigma\left(\boldsymbol{W}^{(j)} h^{(j-1)}(\mathbf{x}) - \mathbf{b}^{(j)}\right), \quad j = 1, 2, \cdots, L, \tag{1}$$

where $\sigma : \mathbb{R} \longrightarrow \mathbb{R}$ is an univariate activation function acting componentwise on vectors, $\boldsymbol{W}^{(j)} \in \mathbb{R}^{d_j \times d_{j-1}}$ is a weight matrix, $\mathbf{b}^{(j)} \in \mathbb{R}^{d_j}$ is a bias vector in layer j, and $h^{(0)}(\mathbf{x}) = \mathbf{x}$ with the width $d_0 = d$. Now, the form of (1) used to approximate functions is

$$\sum_{i=1}^{d_L} c_i \left(h^{(L)}(\mathbf{x})\right)_i \in \operatorname{span}\left\{\left(h^{(L)}(\mathbf{x})\right)_i\right\}_{i=1}^{d_L} \stackrel{\text{def}}{=} \mathscr{F}\left\{\mathbf{W}^{(j)}, \mathbf{b}^{(j)}, 1 \le j \le L\right\}. \tag{2}$$

Note that if $L = 1$, the FNNs defined by (1) degenerate into the well-known classical shallow neural networks. The most important part of (2) to learning functions is the free parameters of weights and bias. It is easy to find that the

form (2) involves free parameters of weights $\sum_{i=1}^{L} d_i d_{i-1}$ and bias $\sum_{i=1}^{L} d_i$ to be trained, leading to huge computational complexity when d_i is large.

For DCNNs, we use the definition from [14]. Let d, d_j, and L be the positive integers. The convolution of $\mathbf{w} \in \mathbb{R}^K$ and $\mathbf{x} \in \mathbb{R}^d$ is mathematically defined as $\mathbf{w} * \mathbf{x} \in \mathbb{R}^{d+K-1}$, where $(\mathbf{w} * \mathbf{x})_i = \sum_{j=1}^{d} w_{i-j+1} x_j$, $i \in [d+K-1]$ ($[d+K-1]$ denotes the set $\{1, 2, \cdots, d+K-1\}$) which can be equivalently rewritten as

$$\mathbf{w} * \mathbf{x} = \mathbf{T}^{\mathbf{w}} \mathbf{x}, \tag{3}$$

where $\mathbf{T}^{\mathbf{w}} \in \mathbb{R}^{(d+K-1)\times d}$ is a Toeplitz-type matrix given by

$$\mathbf{T}^{\mathbf{w}} = \begin{bmatrix} w_1 & 0 & 0 & 0 & \cdots & \cdots & 0 \\ w_2 & w_1 & 0 & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ w_K & w_{K-1} & \cdots & w_1 & 0 & \cdots & 0 \\ 0 & w_K & \cdots & w_2 & w_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & w_K & \cdots & \cdots & w_1 \\ 0 & \cdots & 0 & \cdots & w_K & \cdots & w_2 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & w_K \end{bmatrix} \in \mathbb{R}^{(d+K-1)\times d}. \tag{4}$$

With the above notations, a DCNN with input vector $\mathbf{x} \in \mathbb{R}^d$ and L hidden layers of neurons $\{h^{(j)} : \mathbb{R}^d \longrightarrow \mathbb{R}^{d_j}\}_{j=0}^{L}$ is defined iteratively by

$$h^{(j)}(\mathbf{x}) = \sigma\left(\mathbf{T}^{\left(\mathbf{w}^{(j)}\right)} h^{(j-1)}(\mathbf{x}) - \mathbf{b}^{(j)}\right), \quad j = 1, 2, \cdots, L, \tag{5}$$

where $\sigma : \mathbb{R} \longrightarrow \mathbb{R}$ is an univariate activation function as before, $\mathbf{w}^{(j)} \in \mathbb{R}^K$ denotes a filter supported on $[K]$, and $\mathbf{b}^{(j)} \in \mathbb{R}^{d_j}$ is a bias vector in layer j, $h^{(0)}(\mathbf{x}) = \mathbf{x}$. The form of (5) to learning functions is

$$\begin{aligned} \sum_{i=1}^{d_L} c_i \left(h^{(L)}(\mathbf{x})\right)_i &\in \operatorname{span}\left\{\left(h^{(L)}(\mathbf{x})\right)_i\right\}_{i=1}^{d_L} \\ &\stackrel{\text{def}}{=} \mathscr{C}\left\{\mathbf{w}^{(j)}, \mathbf{b}^{(j)}, 1 \le j \le L\right\}. \end{aligned} \tag{6}$$

Compared with FNNs, DCNNs defined by (5) involve a sparse matrix $\mathbf{T}^{\mathbf{w}^{(j)}}$ in the j-th layer, each row of which has no more than K nonzero elements. The number of weights and biases is KL and $\sum_{i=1}^{L} d_i$, respectively, which is a large reduction of parameters.

However, this kind of DCNNs results in width increasing; that is, for input signal $\mathbf{x} \in \mathbb{R}^d$, we have $\mathbf{T}^{\mathbf{w}}\mathbf{x} \in \mathbb{R}^{d+K-1}$ which is rarely used in practice. To improve this unusual structure, downsampling also known as pooling operators is applied in the DCNNs to reduce width formally [15, 16]. The key role of downsampling is reducing the dimension of features and retaining effective information. To describe it mathematically, we adopt a general version given below.

Definition 1 (downsampling [15]). Let $\mathbf{x} \in \mathbb{R}^d$, a downsampling set $S \subset [d]$, be an index set. D_S is called a downsampling operator indexed by S if $D_S(\mathbf{x}) = \mathbf{x}_S$, where $\mathbf{x}_S$ denotes the vector indexed by S.

Factually, except for downsampling, in real applications, multiple filters are usually utilized in each layer of DCNNs to obtain multichannel outputs. Each output is made up of channel combinations that provide the flexibility needed to avoid variance issues and loss of information [17], and different channels will play the role of extracting multiple features of the input data [16]. Specifically, as pointed out in [18], convolution from the current layer to the next in the multichannel case is often organized as follows: inputs of each input channel first convolute with all related filters to compose the convoluted inputs, and then the convoluted outputs are composed of linear combinations of the convoluted inputs, and finally, an activation function (usually ReLU) is acted on each convoluted outputs componentwise. Along with this fact in mind, the key to the MDCNNs considered in this paper is multichannel convolution which is mathematically defined as follows.

Definition 2 (multichannel convolution). Let C, C', and $K \in \mathbb{N}_+$ be input channel size, output channel size, and filter size, respectively. Filters $\mathscr{W} = (\mathscr{W}_{n,j,i})_{n\in[K], j\in[C'], i\in[C]}$ are defined as a three order tensor. Let $\boldsymbol{X} \in \mathbb{R}^{d\times C}$ be the input data with C channels and the output of channel $j \in [C']$ named $\boldsymbol{Y}_{:,j}$ without bias, and activation function is defined as the sum of convoluted input data, i.e.,

$$\mathbf{Y}_{:,j}(\mathbf{X}) = \sum_i \mathbf{T}^{\mathscr{W}_{:,j,i}} \mathbf{X}_{:,i} \in \mathbb{R}^{d+K-1}, \tag{7}$$

where the Toeplitz-type matrix $\mathbf{T}^{\mathscr{W}_{:,j,i}}$ is defined as (3). Further, let $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_{C'}] \in \mathbb{R}^{(d+K-1)\times C'}$ be a bias matrix and σ be the activation function; the multichannel convolution $\mathrm{Conv}^{\sigma}_{\mathscr{W},\mathbf{B}} : \mathbb{R}^{d\times C} \longrightarrow \mathbb{R}^{(d+K-1)\times C'}$ is given by

$$\mathrm{Conv}^{\sigma}_{\mathscr{W},\mathbf{B}}(\mathbf{X}) = \sigma(\mathbf{Y} - \mathbf{B}) = \sigma\left(\left[\mathbf{Y}_{:,1} - b_1, \mathbf{Y}_{:,2} - b_2, \cdots, \mathbf{Y}_{:,C'} - b_{C'}\right]\right). \tag{8}$$

The whole multichannel convolution structure is shown in Figure 1.

Remark 3. Here, we remark that Definition 2 implies that there are $C \times C'$ filters in total, and each filter has the same size K. For convenience, we assume that the input data have the same size d for all channels such that the corresponding

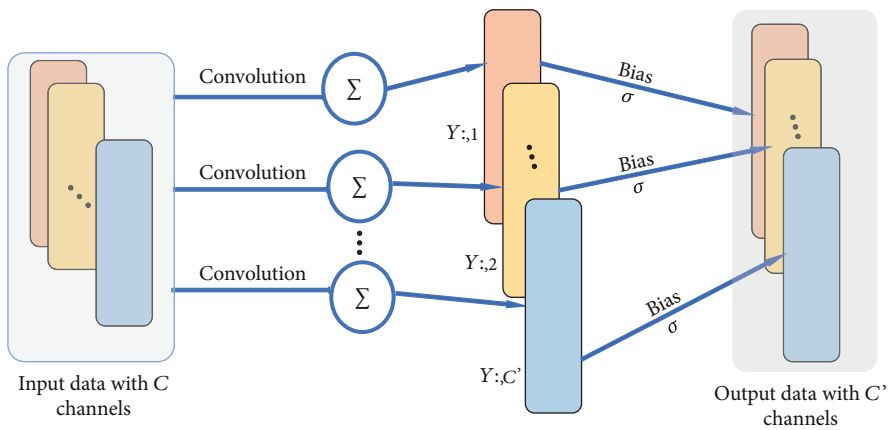

FIGURE 1: Illustration of multichannel convolution.

outputs also have the same size $d+K-1$. If $C=C'=1$, equation (7) degenerates into equation (3).

The multichannel convolution from the current layer to the next provides the main ingredient of MDCNNs. Combined with the downsampling operator given by Definition 1, MDCNNs with downsampling are given below.

Definition 4 (MDCNNs with downsampling). Let $C^{(l)}, K^{(l)} \in \mathbb{N}_+$ be the channel size and filter size in layer $l(1 \le l \le L)$, the set $A=\{l_1, l_2, \cdots, l_n\} \subset [L]$ satisfying $1 \le l_1 < l_2 < \cdots < l_n \le L$ is used to introduce the downsamplings, and $A_{l_j} \subset [d_j](1 \le j \le n)$ is the downsampling sets. A MDCNN with downsampling operators D_{A_j} s and input data $\mathbf{X} \in \mathbb{R}^{d \times C}$ having widths $\{d_i\}_{i=0}^{L}$ is defined iteratively by $d_0 = d$ and for $j=2,3,\cdots,n$

$$d_i = \begin{cases} d_{i-1} + K^{(i)} - 1, & \text{if} \quad l_{j-1} < l < l_j \\ \text{card}\left(A_j\right), & \text{if} \quad i = l_j \text{ or } i = l_{j-1} \end{cases} \tag{9}$$

is a sequence of function vectors $\{h^{(i)}(\mathbf{X}): \mathbb{R}^{d \times C} \longrightarrow \mathbb{R}^{d_i \times C^{(i)}}\}_{i=1}^{L}$ defined iteratively by

$$h^{(i)}(\mathbf{X}) = \begin{cases} \text{Con}v^{\sigma}_{\mathscr{W}^{(i)}, \mathbf{B}^{(i)}}\left(h^{(i-1)}(\mathbf{X})\right), & \text{if} \quad l_{j-1} < i < l_j, \\ D_{A_i} \circ \text{Con}v^{\sigma}_{\mathscr{W}^{(i)}, \mathbf{B}^{(i)}}\left(h^{(i-1)}(\mathbf{X})\right), & \text{if} \quad i = l_j. \end{cases} \tag{10}$$

Here, all channels in the same layer have equal size, the downsampling operators D_{A_j}s act on each channel of the layer l_j, card (A_j) denotes the cardinal number of A_j, and the tensor $\mathscr{W}^{(i)}$ denotes filters between layers i and $i-1$. Finally, the form of MDCNNs used to approximate functions is

$$\sum_{i=1}^{C^{(L)}} \sum_{j=1}^{d_L} c_{j,i} \left(h^{(L)}(\mathbf{X})\right)_{j,i}, \tag{11}$$

where $c_{j,i} \in \mathbb{R}$ are coefficients. The structure of MDCNNs is shown in Figure 2.

Remark 5. The form (11) indicates that the objective form has three important ingredients $\{\mathscr{W}^{(i)}, \mathbf{B}^{(i)}, C^{(i)}\}$ corresponding to filters, bias, and channel size. From another perspective, it belongs to

$$\text{span}\left\{\left(h^{(L)}(\mathbf{X})\right)_{j,i}\right\}_{i=1,j=1}^{C^{(L)},d_L} \stackrel{\text{def}}{=} \mathscr{M}\left\{\mathscr{W}^{(i)}, \mathbf{B}^{(i)}, C^{(i)}, 1 \le i \le L\right\}. \tag{12}$$

If all layers have only one channel, MDCNNs will degenerate into DCNNs. We say that the MDCNNs with downsampling have uniform filter lengths if all channels in every layer have the same size. Under this circumstance, we call the MDCNNs with downsampling uniform. All MDCNNs with downsampling considered in our main results are uniform.

However, the existing theoretical studies cannot be applied to MDCNNs. For example, Zhou [14, 15, 19] only considers single-channel DCNNs whose widths are increasing to depth. The multichannel convolution was also used in a recent network Butterfly-Net [20, 21], which is based on butterfly algorithm. However, the multichannel convolution is only part of its network structure, and the structure of our MDCNNs relying on multichannel convolution solely is different from that of Butterfly-Net. Moreover, they study the approximation of Fourier representation of input data, which is also different with ours. To investigate the approximation ability of MDCNNs, we study its behavior on ridge functions and functions from Sobolev space $H^r(\mathbb{R}^d)$. MDCNNs considered in this paper have finite width d, finite filter size $K+1(<d)$, and finite channels in each layer. In addition, the activation function is the popular rectified linear unit (ReLU) defined as a univariate function given by $\sigma(u) = (u)_+ = \max\{0, u\}, u \in \mathbb{R}$, which is often utilized to guarantee the nonlinear properties of the neural networks. As pointed out by [19, 22], linear combinations of ReLU units can express the objective functions with arbitrary

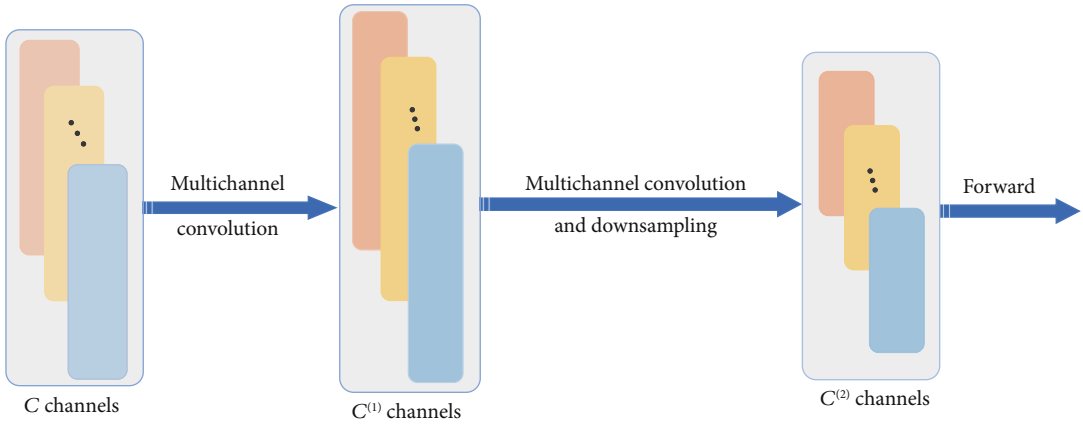

Figure 2: Illustration of MDCNNs with downsampling: the input data has C channels; then, the multichannel convolution is acted on the input data with $C^{(1)} \times C$ filters, and the output of the first layer contains $C^{(1)}$ channels; next, the multichannel convolution is acted on the outputs of the first layer with $C^{(2)} \times C^{(1)}$ filters following by a downsampling operator, and the output of the second layer contains $C^{(2)}$ channels; and so on.

accuracy. Hence, the main proof techniques of our theorems are constructing the structured MDCNNs to obtain the ReLU approximations of the objective functions. In addition, we emphasize the benefit of multiple channels: different channels from some fixed layers can extract transformed data features from the previous layer. Concretely, we utilize channels to store the ReLU units, obtain new ReLU units, and deposit initial data. In this way, our proposed MDCNNs can achieve better results in approximating functions than the structure from DCNNs and FNNs. In summary, we make the following contributions to the approximation theory of MDCNNs:

(i) To construct MDCNNs by introducing the multichannel convolution so that different channels are used to extract different data features. To introduce the downsampling operator into the MDCNNs so that the width-increasing nature can be avoided from layer to layer

(ii) To present a theorem for approximating ridge functions by MDCNNs of the form $g(\boldsymbol{\xi} \cdot \mathbf{x})$ with $\boldsymbol{\xi} \in \mathbb{R}^d$ and $g : \mathbb{R} \longrightarrow \mathbb{R}$ which demonstrates that for this widely used simple but important function family, MDCNNs have better approximation abilities than FNNs and DCNNs

(iii) To prove a theorem for approximating functions in Sobolev space $H^r(\mathbb{R}^d)$ which shows the universality of MDCNNs and the benefit of depth. In addition, it also reveals better approximation performances than FNNs and DCNNs

The structure of this article is organized as follows: in Section 2, we present the main results for approximating functions from ridge class and Sobolev space and further compare them with some related work. Proofs of our main results are given in Section 3. Finally, we summarize the research of this paper in Section 4.

2. Main Results

Complicated functions can often be approximated by simple families [23], such as polynomials, splines, wavelets, radial basis functions, and ridge functions. Specifically, many approximation results are based on the combination of ridge functions [24, 25]. Our first main result of downsampled MDCNNs shows its good performance of approximation ability for ridge class. After that, we further provide the approximation ability of MDCNNs of functions from Sobolev's space. The two approximations constitute our main results. The main techniques of our proofs are constructing the approximations of objective functions by linear combinations of ReLU units at first and then specifying the networks' parameters such that the constructed MDCNNs' outputs match the linear approximations of ReLU units.

2.1. Approximation on Ridge Function. Mathematically, ridge functions are any multivariate real-valued function $f : \Omega \longrightarrow \mathbb{R}(\Omega \subset \mathbb{R}^d)$ of the form

$$f(\mathbf{x}) = g(\boldsymbol{\xi} \cdot \mathbf{x}), \mathbf{x} \in \Omega, \tag{13}$$

induced by an unknown eigenvector $\boldsymbol{\xi} \in \mathbb{R}^d$ and an unknown univariate external function $g : \mathbb{R} \longrightarrow \mathbb{R}$. Further, let K_α be the class of univariate Lipschitz-α functions defined in $[-1, 1]$ with constant $C_\alpha < \infty (0 < \alpha \leq 1)$; that is, for any $x, y \in [-1, 1]$,

$$|g(x) - g(y)| \leq C_\alpha |x - y|^\alpha. \tag{14}$$

Our first result shows the approximation ability of MDCNNs for ridge functions with the external function $g \in K_\alpha$ and $\mathbf{x} \in \mathbb{B}^d$, where $\mathbb{B}^d = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2 \leq 1\}$ represents the unit ball. Denote $\|f\|_\Omega = \max_{x \in \Omega} |f(x)|$. Throughout this paper, we will use $\{\mathscr{U}, \mathscr{P}\}$ to represent the number of computation units (widths or hidden units [15]) and free

parameters, where computation units can be calculated by counting the hidden units of MDCNNs.

Theorem 6. *Let $\boldsymbol{\xi} \in \mathbb{B}^d$, $K+1(K \in [d-1])$ be the uniform filter size, $L_0 = \lceil d/K \rceil$ ($\lceil \cdot \rceil$ is the ceil function), and $m, T \in \mathbb{N}_+$. If $g \in K_\alpha$, then there exists a downsampled MDCNN with at most 3 channels in each layer, the width of each channel is no more than d, and $L = L_0 + T + 2$ layers satisfy*

$$\left\| \sum_{j=1}^{3} c_j \left(h^{(L)}(\mathbf{x}) \right)_j - g(\boldsymbol{\xi} \cdot \mathbf{x}) \right\|_{[-1,1]^d} \leq \frac{2C_\alpha 2^\alpha}{T^\alpha}, \tag{15}$$

where $c_j \in \mathbb{R}$. The number of computation units is $\mathcal{U} = 3L + 2(L_0 - 1)d - 3L_0$, and free parameters are $\mathcal{P} = 3(L - L_0) + d$.

Remark 7. The constructed MDCNNs have finite channels, finite width, and finite filter sizes, and the convergence rate denoted by (15) is not only dimension-free but also reveals the benefit of depth. Given arbitrary approximation accuracy $\varepsilon \in (0,1)$, Theorem 6 shows that we need at least $T \geq 2(2C_\alpha/\varepsilon)^{1/\alpha}$. Taking $T = \lceil 2(2C_\alpha/\varepsilon)^{1/\alpha} \rceil$, it needs $L = \lceil 2(2C_\alpha/\varepsilon)^{1/\alpha} \rceil + \lceil d/K \rceil + 2$ layers, computation units $\mathcal{U} = 3\lceil 2(2C_\alpha/\varepsilon)^{1/\alpha} \rceil + 2(\lceil d/K \rceil - 1)d + 6$, and free parameters $\mathcal{P} = 3\lceil 2(2C_\alpha/\varepsilon)^{1/\alpha} \rceil + d + 6$ to get (15).

Remark 8. A concrete example is as follows: let $\boldsymbol{\xi} \in \mathbb{B}^5$, $d = 5$, $K = 3$, $L_0 = 2$, $g(\mathbf{x}) = \sin(\mathbf{x})$ satisfying $g(\mathbf{x})$ belong to Lipschitz-1 class. By Theorem 6, we can construct an MDCNN with at most 3 channels, and the width of each channel is no more than 5, and $L = T + 4$ layers such that

$$\left\| \sum_{j=1}^{3} c_j \left(h^{(L)}(\mathbf{x}) \right)_j - \sin(\boldsymbol{\xi} \cdot \mathbf{x}) \right\|_{[-1,1]^5} \leq \frac{4}{T}. \tag{16}$$

2.2. Approximation on Function from Sobolev Space. How do MDCNNs behave for smooth functions? Our second theorem shows that functions in Sobolev space of order r can be well approximated by a downsampled MDCNN with at most 4 channels.

Theorem 9. *Let $K+1(K \in [d-1])$ be the uniform filter size, $L_0 = \lceil d/K \rceil$, $G \in H^r(\mathbb{R}^d)$, and $\Omega \subset [-1,1]^d$. If $L \geq 3(L_0+1)$, then, for any $f = G|_\Omega$ and an integer $r > 2 + (d/2)$, there exists a downsampled MDCNN with finite width and at most 4 channels such that*

$$\left\| f(\mathbf{x}) - h^{(L)}(\mathbf{x}) \right\|_\Omega \leq C\|G\| \frac{(\ln L)^{1/2}}{L^{1/2+1/d}}, \tag{17}$$

where $C > 0$ is an universal constant and $\|G\|$ denotes the Sobolev norm of $G \in H^r(\mathbb{R}^d)$ given by $\|G\| = \|(1+|\mathbf{w}|^2)^{r/2} \mathcal{F}(G)(\mathbf{w})\|_{L^2}$ with $\mathcal{F}(G)$ being the Fourier transform of G. The number of computation units is $\mathcal{U} \leq 4Ld$, free parameters are $\mathcal{P} \leq ((d+2)/(L_0+1))L + 5$.

Remark 10. In fact, Theorem 9 demonstrates the universality of MDCNNs; that is, for any compact subset $\Omega \subset \mathbb{R}^d$, any function in $C(\Omega)$ can be approximated by MDCNNs to an arbitrary accuracy when the depth L is large enough. The reason is that the set $H^r(\Omega)$ is dense in $C(\Omega)$ when we consider the Sobolev spaces that can be embedded into the space of continuous functions on Ω. Moreover, the proof of this theorem shows that our constructed MDCNNs have at most 4 channels in each layer and the width of each layer equals d. Given arbitrary $\varepsilon \in (0,1)$, we requires at least $L \geq (C\|G\|)^2/\varepsilon^2$. Taking $L = \max\{3(L_0+1), \lceil (C\|G\|)^2/\varepsilon^2 \rceil\}$, we have $L = \lceil (C\|G\|)^2/\varepsilon^2 \rceil$ for small ε. Thus, the number of computation units is $\mathcal{U} \leq 4d\lceil (C\|G\|)^2/\varepsilon^2 \rceil$, and free parameters are $\mathcal{P} \leq K\lceil (C\|G\|)^2/\varepsilon^2 \rceil + 5$.

Both of the two main results reveal the benefit of depth in terms of approximations of functions from ridge class and Sobolev space, which indicate that MDCNNs can approximate the two types of functions to arbitrary accuracy if the depth $L \longrightarrow \infty$. Moreover, the constructed MDCNNs have finite channels, finite width, and finite filter sizes, wich is more close to real-world scenes compared with [14, 15, 19].

2.3. Comparison and Discussion. Most studies on the approximation theory of neural networks focus on two aspects: the first is obtained in the late 1980s about universality [26–28] meaning that any continuous functions can be approximated by (2) to arbitrary accuracy; in other words, the space $\mathcal{F}\{\mathbf{W}^{(j)}, \mathbf{b}^{(j)}, 1 \leq j \leq l\}$ is dense in the objective function space; the second is obtained about convergence rates of functions [24, 25, 29–31] in the view of neurons, parameters, or depth. For fairness, in this part, we aim to compare our main results with other theoretical investigations of networks existing in the literature under approximation error $\varepsilon \in (0,1)$. Specifically, we shall do our comparisons in terms of width d_L, filter size K, depth L, the number of computation units $\mathcal{U}$, and free parameters $\mathcal{P}$.

Let R_n denote the set of combination of ridge functions with cardinal number no larger than n; it had been proven in [24] that any function from the Sobolev space $W_p^{r,d}$ in the space L_q with $2 \leq q \leq p \leq \infty$ behaves asymptotically of the order $n^{-r/(d-1)}$ by FNNs. The superiority of Theorem 6 over [24] is the dimension-free property of the convergence rate given by (15) which demonstrates the good performance of MDCNNs in approximating ridge functions. Besides, let $D_A(\mathbf{x}) = D_m(\mathbf{x}) = (x_{im})_{i=1}^{\lfloor d/m \rfloor}$ ($\lfloor \cdot \rfloor$ is the floor function), where $m \leq d$ is a scaling parameter. Paper [15] constructed a DCNN with filter size $4N+6$ in the last layer and finite depth $\lceil ((d-1)/(k-2)) + 1 \rceil$. It obtained a convergence rate of $\mathcal{O}(1/N^\alpha)$ for ridge functions with external function $g \in K_\alpha$, where one needs computation units at most $3d(d-1)/K - 1 + 2(2C_\alpha/\varepsilon)^{1/\alpha} + 8$ and free parameters at most $2(2C_\alpha/\varepsilon)^{1/\alpha} + 8d$. However, the filter size is often no larger than the input dimension in practice, meaning that this structure is not frequently used. By comparison, even though our Remark 7 indicates that the computation units and free parameters of MDCNNs constructed from Theorem

6 have the same order of [15], it is easy to find that our constructed network may be closer to real-world applications, and the convergence rate from (15) reveals the benefit of depth.

Let $f \in W_{\infty}^{r}([0,1]^d)$, based on Taylor expansion; paper [32] had shown that one needs ReLU neural networks with length at most $c(d,r)(\ln 1/\varepsilon + 1)$ and free parameters along with computation units at most $c(d,r)\varepsilon^{-d/r}(\ln 1/\varepsilon + 1)$ to ensure the approximation accuracy ε. As a comparison, Remark 10 states that our MDCNNs only need $L = \lceil (C\|G\|)^2/\varepsilon^2 \rceil$ layers which are dimension-free, the number of computation units is $\mathscr{U} \le 4d\lceil (C\|G\|)^2/\varepsilon^2 \rceil$, and free parameters are $\mathscr{P} \le K\lceil (C\|G\|)^2/\varepsilon^2 \rceil + 5$ to get the approximation accuracy ε. Our Theorem 9 has huge advantages over [32] since the discussions from [14, 15] indicate that $c(d,r)$ has a 2^d factor which may be very large when the input dimension d is large. In addition, paper [14] had considered the DCNNs without downsampling operators, and it leads to the networks' widths being linearly increasing. In such a situation, the number of computation units and free parameters of the DCNNs with $L = \lceil (C\|G\|)^2/\varepsilon^2 \rceil$ is $Ld + (L(L-1)/2)(K-1) = \mathcal{O}((1/\varepsilon^4) + (d/\varepsilon^2))$ and $(5K+3)L + 2d - 2K + 1 = \mathcal{O}((1/\varepsilon^2) + d)$, respectively. Compared with that work, the MDCNNs from Theorem 9 have finite width, and the computation units are at most $\mathscr{U} \le 4d\lceil (C\|G\|)^2/\varepsilon^2 \rceil = \mathcal{O}(d/\varepsilon^2)$, and free parameters are at most $\mathscr{P} \le K\lceil (C\|G\|)^2/\varepsilon^2 \rceil + 5 = \mathcal{O}(1/\varepsilon^2)$ which demonstrates better performances to [14].

3. Proofs of Main Results

There are two kinds of downsampling operators D_{A_1} and D_{A_2} acting on each layer in our constructed MDCNNs to ensure the finite width property, where $A_1 = [K+1 : K+d]$ and $A_2 = [1 : d]$ ($[a : b]$ denotes all integers belong to $[a, b]$). Thereby, for any $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{w} \in \mathbb{R}^{K+1}$, we can write the downsampled convolution as $D_{A_1} \circ (\mathbf{w} * \mathbf{x}) = \mathbf{T}_1^{\mathbf{w}}\mathbf{x}$ and $D_{A_2} \circ (\mathbf{w} * \mathbf{x}) = \mathbf{T}_2^{\mathbf{w}}\mathbf{x}$, where $\mathbf{T}_1^{\mathbf{w}}$ and $\mathbf{T}_2^{\mathbf{w}} \in \mathbb{R}^{d\times d}$ are square matrices consisting of rows from $\mathbf{T}^{\mathbf{w}}$ given by

$$\mathbf{T}_1^{\mathbf{w}} = \begin{bmatrix} w_{K+1} & w_K & \cdots & w_1 & \cdots & 0 \\ 0 & w_{K+1} & \cdots & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & w_{K+1} & \cdots & w_1 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & w_{K+1} \end{bmatrix},$$
$$\mathbf{T}_2^{\mathbf{w}} = \begin{bmatrix} w_1 & 0 & 0 & \cdots & \cdots & 0 \\ w_2 & w_1 & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ w_{K+1} & \cdots & w_2 & w_1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & w_{K+1} & \cdots & w_2 & w_1 \end{bmatrix}. \tag{18}$$

That is, convolution of $\mathbf{w}$ and $\mathbf{x}$ with downsampling operators D_{A_1} and D_{A_2} is equivalent to special matrix-vector multiplication. Furthermore, the two kinds of convolution have the property that for input signal $\mathbf{x} \in \mathbb{R}^d$, we have $\mathbf{T}_1^{\mathbf{w}}\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{T}_2^{\mathbf{w}}\mathbf{x} \in \mathbb{R}^d$; i.e., the input data and output data are equal widths.

We first introduce some lemmas in Subsection 3.1 that will be used later to prove our main results. Detailed proofs of our main results will be shown in Subsection 3.2.

3.1. Auxiliary Lemmas

Lemma 11. *Let $\boldsymbol{\alpha} \in \mathbb{R}^d$, $\mathbf{x} \in [-1, 1]^d$, $t \in \mathbb{R}$, $K+1 (K \in [d-1])$ be the uniform filter size, $A_{\boldsymbol{\alpha}} = \sum_{i=1}^{d}|\alpha_i|$, the constant $B \ge 1$ is an arbitrary upper bound of $\max_{1\le i\le d}|\mathbf{x}_i|$, and $L_0 = \lceil d/K \rceil$. Then, there exists an MDCNN with 3 channels having downsampling set $[L_0]$ and*

$$\mathscr{W}^{(l)} \in \begin{cases} \mathbb{R}^{(K+1)\times 3\times 1}, & \text{if} \quad l = 1, \\ \mathbb{R}^{(K+1)\times 3\times 3}, & \text{if} \quad l = 2, \cdots, L_0 - 1, \\ \mathbb{R}^{(K+1)\times 2\times 3}, & \text{if } l = L_0, \end{cases} \tag{19}$$

such that $h^{(L_0)}(x) \in \mathbb{R}^{d\times 2}$ with $(h^{(L_0)}(x))_{1,1} = \boldsymbol{\alpha}\cdot x + A_{\boldsymbol{\alpha}}B$ and $(h^{(L_0)}(\mathbf{x}))_{:,2} = (x_i + B)_{i=1}^{d}$. Moreover, $\max_{1\le l\le L_0}\|\mathscr{W}^{(l)}\|_\infty = \max\{1, \|\alpha\|_\infty\}$, computation units $\mathscr{U} = 3L_0 d - d$, and free parameters $\mathscr{P} = d + 3$.

Remark 12. The proof procedure of this lemma suggests that by changing the bias in layer L_0 to be $\mathbf{b}^{(L_0)} = [(2A_{\boldsymbol{\alpha}}B + t + \sum_{i=K+1}^{d}\alpha_i)\mathbf{1}_{d+K}, 0] \otimes \mathbf{1}_{K+d}$ ($\otimes$ denotes the Kronecker product), we have $(h^{(L_0)}(\mathbf{x}))_{1,1} = \sigma(\boldsymbol{\alpha}\cdot\mathbf{x} - t)$. Besides, it is not difficult to get that, if we abandon the last channel of each layer; i.e., the channel used to store the input data x is deleted in the process of its proof; after L_0 layers of convolution, we have $h^{(L_0)}(\mathbf{x}) = \boldsymbol{\alpha}\cdot\mathbf{x} + B$ (here, in layer L_0, we choose the downsampling set $[K+1 : K+d]$), and $\max_{1\le l\le L_0}\|w^{(l)}\|_\infty = \max\{1, \|\alpha\|_\infty\}$, the computation units are $2L_0 d - d$, and the free parameters $\mathscr{P} = d + 3$.

Remark 13. The proof of this lemma tells us that our constructed MDCNN has three characteristics: first, only 3 channels are used in all layers; second, all channels have the same width d; at last, it has finite layers L_0. It can be seen from these characteristics that MDCNNs have great superiority in the view of computation units and free parameters.

Proof. Our MDCNN contains 3 channels in layer l $(1 \le l \le L_0 - 1)$, the first channel is used to get the target output, the second channel to shift the input data by K units, and the third channel to store the input data; the last layer contains 2 channels. By using convolution computation through L_0 layers, we get the desired result. We first construct filters and bias in the first layer, choosing $\mathscr{W}_{:,1,1}^{(1)} =$

$[0, \alpha_K, \alpha_{K-1}, \cdots, \alpha_1]^T \in \mathbb{R}^{K+1}$, $\mathscr{W}^{(1)}_{:,2,1} = [1, 0, \cdots, 0]^T \in \mathbb{R}^{K+1}$, $\mathscr{W}^{(1)}_{:,3,1} = [0, \cdots, 0, 1]^T \in \mathbb{R}^{K+1}$, and $b^{(1)} = [-A_{\boldsymbol{\alpha}}B, -B, -B] \otimes \mathbf{1}_{d+K}$, where $\mathbf{1}_{d+K}$ denotes the vector in $\mathbb{R}^{d+K}$ whose entries equal to 1; we have

$$\mathrm{Conv}^{\sigma}_{\mathscr{W}^{(1)},\mathbf{b}^{(1)}}(\mathbf{x}) = \begin{bmatrix} A_{\boldsymbol{\alpha}}B & x_1 + B & B \\ \alpha_K x_1 + A_{\boldsymbol{\alpha}}B & x_2 + B & B \\ \vdots & \vdots & \vdots \\ \sum_{i=1}^{K} \alpha_i x_i + A_{\boldsymbol{\alpha}}B & x_{K+1} + B & x_1 + B \\ \vdots & \vdots & \\ \alpha_1 x_d + A_{\boldsymbol{\alpha}}B & B & x_d + B \end{bmatrix} \in \mathbb{R}^{(K+d)\times 3}. \tag{20}$$

By taking the downsampling set $A_1 = [K+1 : K+d]$, we get

$$h^{(1)}(x) = D_{A_1} \circ \mathrm{Conv}^{\sigma}_{\mathscr{W}^{(1)},\mathbf{b}^{(1)}}(\mathbf{x}) = \begin{bmatrix} \sum_{i=1}^{K} \alpha_i x_i + A_{\boldsymbol{\alpha}}B & x_{K+1} + B & x_1 + B \\ \vdots & \vdots & \vdots \\ \alpha_1 x_d + A_{\boldsymbol{\alpha}}B & B & x_d + B \end{bmatrix} \in \mathbb{R}^{d\times 3}. \tag{21}$$

It is noteworthy that, for $j \in [3]$, $(h^{(1)}(x))_{:,j} = \sigma(\mathbf{T}_1^{\mathscr{W}^{(1)}_{:,j,1}} h^{(0)}(\mathbf{x}) - D_{A_1}(\mathbf{b}^{(1)}_{:,j}))$ with $\mathbf{T}_1^{\mathscr{W}^{(1)}_{:,j,1}}$ having the form of (18). For $2 \le l \le L_0 - 1$, $\mathscr{W}^{(l)}_{:,j,:} \in \mathbb{R}^{(K+1)\times 3}$ has the following form:

$$\mathscr{W}^{(l)}_{:,1,:} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \alpha_{lK} & 0 \\ \vdots & \vdots & \vdots \\ 0 & \alpha_{(l-1)K+2} & 0 \\ 1 & \alpha_{(l-1)K+1} & 0 \end{bmatrix},$$
$$\mathscr{W}^{(l)}_{:,2,:} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix}, \tag{22}$$
$$\mathscr{W}^{(l)}_{:,3,:} = \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

and $\mathbf{b}^{(2)} = [-A_{\boldsymbol{\alpha}}B, 0, 0] \otimes \mathbf{1}_{d+K}$, for $3 \le l \le L_0 - 1$, $\mathbf{b}^{(l)} = [0, 0, 0] \otimes \mathbf{1}_{d+K}$. By choosing the downsampling set A_1, we have

$$h^{(l)}(\mathbf{x}) = D_{A_1} \circ \mathrm{Conv}^{\sigma}_{\mathscr{W}^{(l)},\mathbf{b}^{(l)}}(\mathbf{x}) = \begin{bmatrix} \sum_{i=1}^{lK} \alpha_i x_i + 2A_{\boldsymbol{\alpha}}B + B \sum_{i=K+1}^{lK} \alpha_i & x_{lK+1} + B & x_1 + B \\ \vdots & \vdots & \vdots \\ \alpha_1 x_d + \sum_{i=1}^{l-1} \alpha_{iK+1} B & B & x_d + B \end{bmatrix} \in \mathbb{R}^{d\times 3}. \tag{23}$$

Similarly, for $j \in [3]$, $(h^{(l)}(\mathbf{x}))_{:,j} = \sigma(\sum_{i=1}^{3} \mathbf{T}_1^{\mathscr{W}^{(l)}_{:,j,i}} (h^{(l-1)}(\mathbf{x}))_{:,i} - D_{A_1}(\mathbf{b}^{(l)}_{:,j}))$ with $\mathbf{T}_1^{\mathscr{W}^{(l)}_{:,j,i}}$ having the form of (18). For $l = L_0$, by choosing

$$\mathscr{W}^{(L_0)}_{:,1,:} = \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ 0 & \alpha_d & 0 \\ \vdots & \vdots & \vdots \\ 1 & \alpha_{(L_0-1)K+1} & 0 \end{bmatrix} \in \mathbb{R}^{(K+1)\times 3},$$
$$\mathscr{W}^{(l)}_{:,2,:} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \vdots & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{(K+1)\times 3}, \tag{24}$$

$\mathbf{b}^{(L_0)} = [(A_{\boldsymbol{\alpha}}B + \sum_{i=K+1}^{d} \alpha_i)\mathbf{1}_{d+K}, 0] \otimes \mathbf{1}_{K+d}$, and the downsampling set A_1, we have

$$h^{(L_0)}(\mathbf{x}) = D_{A_1} \circ \mathrm{Conv}^{\sigma}_{\mathscr{W}^{(L_0)},\mathbf{b}^{(L_0)}}(\mathbf{x}) = \begin{bmatrix} \boldsymbol{\alpha} \cdot x + A_{\boldsymbol{\alpha}}B & x_1 + B \\ \vdots & \vdots \\ \alpha_1 x_d + \sum_{i=1}^{L_0-1} \alpha_{iK+1} B & x_d + B \end{bmatrix} \in \mathbb{R}^{d\times 2}. \tag{25}$$

In the same way, for $j \in [2]$, $(h^{(L_0)}(\mathbf{x}))_{:,j} = \sigma(\sum_{i=1}^{3} \mathbf{T}_1^{\mathscr{W}^{(L_0)}_{:,j,i}} (h^{(L_0-1)}(\mathbf{x}))_{:,i} - D_{A_2} \circ \mathbf{b}^{(L_0)}_{:,j})$ with $\mathbf{T}_1^{\mathscr{W}^{(L_0)}_{:,j,i}}$ having the form of (18). In the representation of (18), we can easily find that outputs of the whole constructed convolutional network in each channel have equal width d. Thus, the computation units of the network are $(3L_0 - 1)d$, and free parameters $\mathscr{P} = d + 3$. □

Our next goal is aimed at giving the convergence rates of functions coming from the Lipschitz-$\alpha(0<\alpha\le 1)$ class. Before that, we introduce one more lemma inspired from [33, 34].

Lemma 14. *Let $g\in K_\alpha$ and $m,T\in\mathbb{N}_+$; there exists a piecewise linear function $\widetilde{g}_1(x)$ with breakpoints $\{-1+(2r/m)\}_{r=0}^{m}$ such that*

$$\|\widetilde{g}_1-g\|_{[-1,1]}\le\frac{2C_\alpha 2^\alpha}{T^\alpha},\tag{26}$$

where

$$\widetilde{g}_1(\mathbf{x})=\sum_{j=0}^{T-1}w_j\sigma\left(x-\frac{2j}{T}+1\right)+g(-1).\tag{27}$$

Remark 15. There are two differences between Lemma 14 and [33, 34]; on the one hand, Lemma 7.3 of [34] gives a similar result of this lemma, but it does not give the concrete expression of g used to approximate functions; on the other hand, [33] gives a concrete ReLU expression of functions used to approximate functions, but it is only for functions coming from the Lipschitz-1 class. Besides, both [33, 34] are acquired under the assumption that input data $\mathbf{x}\in[0,1]$.

Proof. Our proof will be divided into two parts. Firstly, we shall give an upper bound for $\|g-\widetilde{g}_1\|_{[-1,1]}$ based on modulus of continuity of g. By Lemma 6 of [19], let $[t_2,t_{N-1}]=[-1,1]$, $N=T+3$, and $t_i=-1+2(i-2)/T$, $i=1,2,\cdots,N$; the piecewise linear function $\widetilde{g}_1$ used to approximate functions in $[t_2,t_{N-1}]=[-1,1]$ has the form $\widetilde{g}_1(x)=\sum_{j=2}^{N-1}g(t_j)\delta_j(x)$, where δ_j is a univariate function with $j\in\{2,3,\cdots,N-1\}$ given by $\delta_j(x)=T/2(\sigma(x-t_{j-1})-2\sigma(x-t_j)+\sigma(x-t_{j+1}))$. By reordering, let

$$h_2(\mathbf{x})=\begin{cases}w_0\sigma(x-t_2)+g(-1), & x\in[\mathrm{t}_2,\mathrm{t}_3]\\ 0, & \text{otherwise},\end{cases},\tag{28}$$

and for $3\le j\le T+1$,

$$h_j(\mathbf{x})=w_{j-2}\sigma(x-t_j),\mathbf{x}\in[t_j,t_{j+1}]0,\text{otherwise},\tag{29}$$

where $w_0=T/2(g(t_3)-g(t_2))$ and $w_j=T/2(g(t_{j+3})-2g(t_{j+2})+g(t_{j+1}))(1\le j\le T-1)$, we have

$$\widetilde{g}_1(\mathbf{x})=\sum_{j=0}^{T-1}h_{j+2}(\mathbf{x})=\sum_{j=0}^{T-1}w_j\sigma(\mathbf{x}-t_{j+2})+g(-1).\tag{30}$$

Then, by Lemma 6 of [19], for any $g\in K_\alpha$, we have

$$\|g-\widetilde{g}_1\|_{[-1,1]}\le\frac{2C_\alpha 2^\alpha}{T^\alpha},\tag{31}$$

which gives (26). □

Equation (27) indicates that any function $g\in K_\alpha$ can be excellently approximated by the linear combinations of ReLU units. Thereby, the forms of linear combinations by ReLU units inspire us to construct MDCNNs with downsampling to approximate functions. Our next lemma provides specific skills on how to embed (27) into downsampled MDCNNs.

Lemma 16. *Let $g\in K_\alpha$, $x\in[-1,1]$, and $m,T\in\mathbb{N}_+$; there exists a downsampled MDCNN having $L=T+2(L\ge 3)$ layers all of which have only 3 channels, such that*

$$\left\|\sum_{j=1}^{3}c_j\left(h^{(L)}(\mathbf{x})\right)_j-g(x)\right\|_{[-1,1]}\le\frac{2C_\alpha 2^\alpha}{T^\alpha},\tag{32}$$

with $c_j\in\mathbb{R}$, $(j\in[3])$. The computation units are $\mathscr{U}=3T+6$, and the free parameters are weights $\|W\|_0=3T+4$, bias $\|b\|_0=2T+3$, and $\mathscr{P}=3T+3$.

Proof. The main techniques are embedding the ReLU expression from Lemma 14 into some specific MDCNNs. Different channels will play the role of storing input data, shifting input data, and storing σ units.

Choosing $T=L-2$, we have $L=T+2\ge 3$. For the first layer, $\mathscr{W}^{(1)}\in\mathbb{R}^{1\times3\times1}$ with $\mathscr{W}^{(1)}_{1,:,1}=[1,1,0]$, $\mathbf{B}^{(1)}=[-1,-1,0]$, we have $\mathrm{Conv}^{\sigma}_{\mathscr{W}^{(1)},\mathbf{B}^{(1)}}(x)=[x+1,\sigma(x+1),0]$.

For $2\le l\le T+1$, $\mathscr{W}^{(l)}\in\mathbb{R}^{1\times3\times3}$,

$$\mathscr{W}^{(l)}_{1:,:}=\begin{bmatrix}1&1&0\\0&0&w_{l-2}\\0&0&0\end{bmatrix},\qquad \mathbf{B}^{(l)}=\left[0,\frac{2(l-1)}{T},-B_{l-1}\right],\tag{33}$$

where $B_1=2|w_0|$, $B_l=2|w_{l-1}|$, and $M_l=\sum_{i=1}^{l}B_i$. In this matrix, columns represent filters for different output channels and filters in different rows are corresponding to the corresponding input channels with the index of rows and columns corresponding to the index of channels. Then, $\mathrm{Conv}^{\sigma}_{\mathscr{W}^{(l)},\mathbf{B}^{(l)}}(x)=[x+1,\sigma(x-(2(l-1)/T)+1),\sum_{t=0}^{l-2}w_t\sigma(x-(2t/T)+1)+M_{l-1}]$. By induction, we have

$$\mathrm{Conv}^{\sigma}_{\mathscr{W}^{(T+1)},\mathbf{B}^{(T+1)}}(\mathbf{x})=\left[\mathbf{x}+1,0,\sum_{t=0}^{T-1}w_t\sigma\left(\mathbf{x}-\frac{2t}{T}+1\right)+M_T\right].\tag{34}$$

Here, when $l=T+1$, we change the bias of the second output channel to be 1 such that it has zero output. The third element contains the linear ReLU units from $\widetilde{g}_1(x)$.

For $l=L=T+2$, $\mathscr{W}^{(L)}\in\mathbb{R}^{1\times3\times3}$ with all elements equal to zero except for top left and bottom right which is 1, and $\mathbf{B}^{(L)}=[0,-1,0]$, we get $\mathrm{Conv}^{\sigma}_{\mathscr{W}^{(L)},\mathbf{B}^{(L)}}(x)=[x+1,1,\sum_{t=0}^{T-1}$

$w_t\sigma(x-(2t/T)+1)+M_T]$. Choosing $c_1=0$, $c_2=-(M_T-g(-1))$, and $c_3=1$, we have

$$\sum_{j=1}^{3} c_j\left(h^{(L)}(\mathbf{x})\right)_j=\sum_{t=0}^{T-1} w_t\sigma\left(x-\frac{2t}{T}+1\right)+g(-1)=\widetilde{g}_1(\mathbf{x}). \tag{35}$$

Thus, by (26), we have

$$\left\|\sum_{j=1}^{3} c_j\left(h^{(L)}(\mathbf{x})\right)_j-g(x)\right\|_{[-1,1]}\le\frac{2C_\alpha 2^\alpha}{T^\alpha}, \tag{36}$$

with computation units $3T+6$ and free parameters $\mathscr{P}=3T+3$. □

3.2. Proofs of Main Theorems

Proof of Theorem 6. Since $\boldsymbol{\xi}\in\mathbb{B}^d$, $\mathbf{x}\in[-1,1]^d$, we have $\boldsymbol{\xi}\cdot\mathbf{x}\in[-1,1]$. By Remark 12, if we take an upper bound $B=h=1$, then there exists an MDCNN with at most 2 channels such that $h^{(L_0)}(\mathbf{x})=\boldsymbol{\xi}\cdot\mathbf{x}+1$. By changing $\mathbf{B}^{(1)}$ to be zero in the proof of Lemma 16, we have $\mathrm{Conv}^{\sigma}_{\mathscr{W}^{(1)},\mathbf{B}^{(1)}}(\mathbf{x})=[\boldsymbol{\xi}\cdot\mathbf{x}+1,\sigma(\boldsymbol{\xi}\cdot\mathbf{x}+1),0]$. In the sequel, with L replaced by $L-L_0$ in Lemma 16, we get the desired results. □

Proof of Theorem 9. Let $m\in\mathbb{R}_+$; the approximation of f is based on $f_m(\mathbf{x})$ from [22] having the following form:

$$f_m(\mathbf{x})=b_0+\boldsymbol{\alpha}^{(0)}\cdot\mathbf{x}+\frac{\nu}{m}\sum_{k=1}^{m} b_k\sigma\left(\boldsymbol{\alpha}^{(k)}\cdot\mathbf{x}-t_k\right), \tag{37}$$

with $b_k\in[-1,1]$, $\|\boldsymbol{\alpha}^{(k)}\|_1=1$, $0\le t_k\le 1$, $b_0=f(0)$, $\boldsymbol{\alpha}^{(0)}=\nabla f(0)$, $\nu\le 2\nu_{f,2}$, and $\nu_{f,2}=\int_{\mathbb{R}^d}\|\mathbf{w}\|_1^2|\mathscr{F}(f)(\mathbf{w})|d\mathbf{w}<\infty$. By Theorem 2 of [22], we have

$$\sup_{\mathbf{x}\in[-1,1]^d}|f(\mathbf{x})-f_m(\mathbf{x})|\le c\nu_{f,2}\sqrt{d+\ln m}\,m^{-1/2-1/d}, \tag{38}$$

for some universal constant $c>0$. We will embed $f_m(\mathbf{x})$ into a downsampled MDCNN with at most 4 channels to get the target approximation. The core of our main method is using different channels to store a variety of data features. Next, we will prove the theorem by induction. For the first L_0 layers, by Lemma 11, there exists a downsampled MDCNN with 3 channels having L_0 layers such that $h^{(L_0)}(\mathbf{x})\in\mathbb{R}^{d\times 2}$ with $(h^{(L_0)}(\mathbf{x}))_{1,1}=\sigma(\boldsymbol{\alpha}^{(1)}\cdot\mathbf{x}-t_1)$ and $(h^{(L_0)}(\mathbf{x}))_{:,2}=(x_i+B)_{i=1}^d(B\ge\max_{1\le i\le d}|x_i|)$, where the coefficient vector $\boldsymbol{\alpha}$ in Lemma 11 is replaced by $\boldsymbol{\alpha}^{(1)}$. Moreover, the first output channel stores the linear rectifier units, and the second channel stores the input data at layer L_0. For $l=L_0+1$, by choosing

$$\mathscr{W}^{(l)}_{:,1,:}=\begin{bmatrix}\frac{\nu}{m}b_1 & 0\\ 0 & 0\\ \vdots & \vdots\\ 0 & 0\end{bmatrix}\in\mathbb{R}^{(K+1)\times 2},\qquad \mathscr{W}^{(l)}_{:,2,:}=\begin{bmatrix}0 & 1\\ 0 & 0\\ \vdots & \vdots\\ 0 & 0\end{bmatrix}\in\mathbb{R}^{(K+1)\times 2}, \tag{39}$$

and $\mathbf{B}^{(l)}=[-B_1,0]\otimes\mathbf{1}_{d+K}$, where $B_k=\sum_{i=1}^k|(\nu/m)b_i|(B\sum_{j=1}^d|\boldsymbol{\alpha}_j^{(i)}|+|t_i|)(1\le k\le m)$, and the downsampling set A_2, we have $h^{(l)}(\mathbf{x})\in\mathbb{R}^{d\times 2}$ with $(h^{(l)}(\mathbf{x}))_{1,1}=\nu/mb_1\sigma(\boldsymbol{\alpha}^{(1)}\cdot\mathbf{x}-t_1)+B_1$ and $(h^{(l)}(\mathbf{x}))_{:,2}=(x_i+B)_{i=1}^d$.

Next, the MDCNN we constructed will contain at most 4 channels: the first channel is used to store linear combinations of linear rectification units, the second channel is used to store the next linear rectification unit, the third channel is used to shift the input data by K steps, and the fourth channel is used to store raw input information. Suppose for $l=k(L_0+1)$, $h^{(l)}(\mathbf{x})\in\mathbb{R}^{d\times 2}$ with $(h^{(l)}(\mathbf{x}))_{1,1}=\sum_{i=1}^k\nu/mb_i\sigma(\boldsymbol{\alpha}^{(i)}\cdot\mathbf{x}-t_i)+B_k$ and $(h^{(l)}(\mathbf{x}))_{:,2}=(x_i+B)_{i=1}^d$; then, for $l=k(L_0+1)+1$, we choose $\mathscr{W}^{(l)}\in\mathbb{R}^{(K+1)\times 4\times 2}$ given by

$$\begin{aligned}
\mathscr{W}^{(l)}_{:,1,:}&=\begin{bmatrix}0 & 0\\ \vdots & \vdots\\ 0 & 0\\ 1 & 0\end{bmatrix},\\
\mathscr{W}^{(l)}_{:,2,:}&=\begin{bmatrix}0 & 0\\ 0 & \boldsymbol{\alpha}_K^{(k+1)}\\ \vdots & \vdots\\ 0 & \boldsymbol{\alpha}_1^{(k+1)}\end{bmatrix},\\
\mathscr{W}^{(l)}_{:,3,:}&=\begin{bmatrix}0 & 1\\ 0 & 0\\ \vdots & \vdots\\ 0 & 0\end{bmatrix},\\
\mathscr{W}^{(l)}_{:,4,:}&=\begin{bmatrix}0 & 0\\ \vdots & \vdots\\ 0 & 0\\ 0 & 1\end{bmatrix}.
\end{aligned} \tag{40}$$

Here, the column label of each matrix represents the

input channel index, and the column vectors represent the corresponding filters. Further, by choosing $\mathbf{B}^{(l)} = [0,-2B_{\max}, 0, 0] \otimes \mathbf{1}_{d+K}$ with $B_{\max} = \max_{1\le i\le m}\sum_{j=1}^{d}|\boldsymbol{\alpha}_j^{(i)}|B$ and the downsampling set A_1, we have $h^{(l)}(\mathbf{x}) \in \mathbb{R}^{d\times 4}$ with $(h^{(l)}(\mathbf{x}))_{1,1} = \nu/m\sum_{i=1}^{k} b_i\sigma(\boldsymbol{\alpha}^{(i)}\cdot\mathbf{x} - t_1) + B_k$, $(h^{(l)}(\mathbf{x}))_{1,2} = \sum_{i=1}^{K}\boldsymbol{\alpha}_i^{(k+1)}x_i + B\sum_{i=1}^{K}\boldsymbol{\alpha}_i^{(k+1)} + 2B_{\max}$, $(h^{(l)}(\mathbf{x}))_{1:d-K,3} = (x_i + B)_{i=K+1}^{d}$, $(h^{(l)}(\mathbf{x}))_{d-K+1:d,3} = \mathbf{0}_K$, and $(h^{(l)}(\mathbf{x}))_{:,4} = (x_i + B)_{i=1}^{d}$.

For $l = k(L_0+1) + s(2 \le s \le L_0 - 1)$, $\mathscr{W}^{(l)} \in \mathbb{R}^{(K+1)\times 4\times 4}$ and $\mathscr{W}_{:,1,:}^{(l)}, \mathscr{W}_{:,2,:}^{(l)}, \mathscr{W}_{:,3,:}^{(l)}, \mathscr{W}_{:,4,:}^{(l)}$ are given in turn by

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \boldsymbol{\alpha}_{sK}^{(k+1)} & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & \boldsymbol{\alpha}_{(s-1)K+1}^{(k+1)} & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{41}$$

By choosing $\mathbf{b}^{(l)} = [0,0,0,0] \otimes \mathbf{1}_{d+K}$ and the downsampling set A_1, we have $h^{(l)}(\mathbf{x}) \in \mathbb{R}^{d\times 4}$ with $(h^{(l)}(\mathbf{x}))_{1,1} = \nu/m\sum_{i=1}^{k} b_i\sigma(\boldsymbol{\alpha}^{(i)}\cdot\mathbf{x} - t_1) + B_k$, $(h^{(l)}(\mathbf{x}))_{1,2} = \sum_{i=1}^{Ks}\boldsymbol{\alpha}_i^{(k+1)}x_i + B\sum_{i=1}^{Ks}\boldsymbol{\alpha}_i^{(k+1)} + 2B_{\max}$, $(h^{(l)}(\mathbf{x}))_{1:d-Ks,3} = (x_i + B)_{i=Ks+1}^{d}$, $(h^{(l)}(\mathbf{x}))_{d-Ks+1:d,3} = \mathbf{0}_{Ks}$, and $(h^{(l)}(\mathbf{x}))_{:,4} = (x_i + B)_{i=1}^{d}$.

For $l = k(L_0+1) + L_0$, $\mathscr{W}^{(l)} \in \mathbb{R}^{(K+1)\times 4\times 4}$ with $\mathscr{W}_{:,1,:}^{(l)} = \mathscr{W}_{:,1,:}^{(l-1)}$, $\mathscr{W}_{:,3,:}^{(l)} = \mathscr{W}_{:,3,:}^{(l-1)}$, $\mathscr{W}_{:,4,:}^{(l)} = \mathscr{W}_{:,4,:}^{(l-1)}$, $\mathscr{W}_{:,2,2}^{(l)} = [0,\cdots,0,1]^T$, and $\mathscr{W}_{:,2,3}^{(l)} = [0,\cdots,0,\boldsymbol{\alpha}_d^{(k+1)},\cdots,\boldsymbol{\alpha}_{(L_0-1)k+1}]^T$; otherwise, the elements in $\mathscr{W}_{:,2:}^{(l)}$ are equal 0; by choosing $\mathbf{B}^{(l)} = [0, t_{k+1} + 2B_{\max} + \sum_{i=1}^{d}\boldsymbol{\alpha}_i B, 0, 0] \otimes \mathbf{1}_{d+K}$ and the downsampling set A_1, we have $h^{(l)}(\mathbf{x}) \in \mathbb{R}^{d\times 4\times 4}$ with $(h^{(l)}(\mathbf{x}))_{1,1} = \nu/m\sum_{i=1}^{k} b_i\sigma(\boldsymbol{\alpha}^{(i)}\cdot\mathbf{x} - t_1) + B_k$, $(h^{(l)}(\mathbf{x}))_{1,2} = \sigma(\boldsymbol{\alpha}^{(k+1)}\cdot\mathbf{x} - t_{k+1})$, $(h^{(l)}(\mathbf{x}))_{:,3} = \mathbf{0}_d$, and $(h^{(l)}(\mathbf{x}))_{:,4} = (x_i + B)_{i=1}^{d}$.

For $l = (k+1)(L_0+1)$, $\mathscr{W}^{(l)} \in \mathbb{R}^{(K+1)\times 2\times 4}$, and in the first channel, $\mathscr{W}_{1,1,1}^{(l)} = 1$ and $\mathscr{W}_{1,1,2}^{(l)} = \nu/mb_{k+1}$, and the elements are 0 otherwise; in the second channel, $\mathscr{W}_{1,2,4}^{(l)} = 1$, and the elements are 0 otherwise. By choosing $\mathbf{B}^{(l)} = [-|\nu/mb_i|(B\sum_{j=1}^{d}|\boldsymbol{\alpha}_j^{(k+1)}| + |t_{k+1}|), 0] \otimes \mathbf{1}_{d+K}$ and the downsampling set A_2, we have $h^{(l)}(\mathbf{x}) \in \mathbb{R}^{d\times 2\times 4}$ with $(h^{(l)}(\mathbf{x}))_{1,1} = \nu/m\sum_{i=1}^{k+1} b_i\sigma(\boldsymbol{\alpha}^{(i)}\cdot\mathbf{x} - t_i) + B_{k+1}$ and $(h^{(l)}(\mathbf{x}))_{:,2} = (x_i + B)_{i=1}^{d}$.

By induction, for $l = m(L_0+1)$, we have $h^{(l)}(\mathbf{x}) \in \mathbb{R}^{d\times 2}$, and $(h^{(l)}(\mathbf{x}))_{1,1} = \nu/m\sum_{i=1}^{m} b_i\sigma(\boldsymbol{\alpha}^{(i)}\cdot\mathbf{x} - t_i) + B_m$, $(h^{(l)}(\mathbf{x}))_{:,2} = (x_i + B)_{i=1}^{d}$.

Next, for $m(L_0+1) < l \le m(L_0+1) + L_0$, we will use the first channel to store the linear combination of ReLU units, and there is no need to store the input data. By Lemma 11, we have for $l = m(L_0+1) + L_0$, $h^{(l)}(\mathbf{x}) \in \mathbb{R}^{d\times 2}$ with $(h^{(l)}(\mathbf{x}))_{1,1} = \nu/m\sum_{i=1}^{m} b_i\sigma(\boldsymbol{\alpha}^{(i)}\cdot\mathbf{x} - t_i) + B_m$ and $(h^{(l)}(\mathbf{x}))_{1,2} = \boldsymbol{\alpha}^{(0)}\cdot\mathbf{x} + B_0$ where $B_0 = \sum_{i=1}^{d}|\boldsymbol{\alpha}^{(0)}|$.

For $l = (m+1)(L_0+1)$, $\mathscr{W}^{(l)} \in \mathbb{R}^{(K+1)\times 2}$ with $\mathscr{W}_{1,1,1}^{(l)} = \mathscr{W}_{1,1,2}^{(l)} = 1$; the elements are 0 otherwise, $\mathscr{W}_{:,2,:}^{(l)} = \mathbf{0}_{(K+1)\times 2}$. By choosing $\mathbf{B}^{(l)} = [0,-1] \otimes \mathbf{1}_{d+K}$ and the downsampling set A_2, we have $h^{(l)}(\mathbf{x}) \in \mathbb{R}^{d\times 2}$ with $(h^{(l)}(\mathbf{x}))_{1,1} = \nu/m\sum_{i=1}^{m} b_i\sigma(\boldsymbol{\alpha}^{(i)}\cdot\mathbf{x} - t_i) + B_m + \boldsymbol{\alpha}^{(0)}\cdot\mathbf{x} + B_{\max}B$ and $(h^{(l)}(\mathbf{x}))_{1,2} = 1$.

Thus, by choosing $c_{1,1} = 1$, $c_{1,2} = -(B_m + A_0B) + b_0$ and $c_{j,i} = 0$ otherwise, we have

$$\sum_{i=1}^{2}\sum_{j=1}^{d} c_{j,i}\left(h^{((m+1)(L_0+1))}(\mathbf{x})\right)_{j,i} = f_m(\mathbf{x}). \tag{42}$$

At last, by choosing $m = \lfloor (L/L_0+1) - 1\rfloor$ leading to $(L_0+1)(m+1) \le L < (L_0+1)(m+2)$, it is inevitable to appear $L > (L_0+1)(m+1)$. However, we need not worry about it since by using the identity map similar to that in Lemma 16, we can always have

$$\sum_{i=1}^{2}\sum_{j=1}^{d} c_{j,i}\left(h^{(L)}(\mathbf{x})\right)_{j,i} = f_m(\mathbf{x}). \tag{43}$$

In addition, through its concrete form of m, we have $\ln m \le \ln L$, and since $K \in [d-1]$, we further have

$$\begin{aligned} \frac{1}{m} &\le \frac{L_0+1}{L - 2(L_0+1)} \le \frac{(1/K + 2/d)d}{(1 - 2(L_0+1)/L)L} \\ &\overset{(a)}{\le} \frac{(1/K + 2/d)d}{1/3L} \le 6\frac{d}{L}, \end{aligned} \tag{44}$$

where we use $L \ge 3(L_0+1)$ in (a). Thus, we obtain $m^{-1/2-1/d} \le 6(d/L)^{1/2+1/d}$. Putting these into (38), we have

$$\sup_{\mathbf{x}\in[-1,1]^d} |f(\mathbf{x}) - f_m(\mathbf{x})| \le 6c\nu_{f,2}d^{1+1/d}\frac{(\ln L)^{1/2}}{L^{1/2+1/d}}. \tag{45}$$

In a similar way of [14], by using the Cauchy-Buniakowsky-Schwarz inequality, we have

$$\begin{aligned} 6c\nu_{f,2}d^{1+1/d} &= 6cd^{1+1/d}\int_{\mathbb{R}^d}\|w\|_1^2|\mathscr{F}(f)(\mathbf{w})|d\mathbf{w} \\ &\le 6c\|G\|d^{1+1/d}\int_{\mathbb{R}^d}\|w\|_1^2\left(1 + \|w\|_2^2\right)^{-r/2}dw \\ &\le 6c\|G\|\sqrt{\frac{d^6\pi^{d/2}}{\Gamma((d/2)+1)}}\left(1 + \frac{1}{\sqrt{2r-d-4}}\right) \\ &\le 12cc'\|G\|, \end{aligned} \tag{46}$$

where $c' = \max_{l \in \mathbb{N}} \sqrt{l^6 \pi^{l/2} / \Gamma((l/2)+1)}$ is an absolute constant. By taking $C = 12cc'$, we get the inequality (17). The computation units are $\mathscr{U} = (4dL_0 + 2d)m + dL_0 + d \leq 4Ld$ and free parameters are $\mathscr{P} = (d+2)(m-1) + 1 + 2d + 8 = (d+3)(m+1) + 5 \leq ((d+3)/(L_0+1))L + 5$. This completes the Proof of Theorem 9. □

4. Conclusion and Future Work

This paper studies the approximations of structured MDCNNs with downsampling. The results show that for functions from ridge class and Sobolev's space $H^r(\mathbb{R}^d)$, our proposed MDCNNs have better function approximation performances over other relevant studies, explaining the reason why MDCNNs are successful in applications to some extent. But, note that our MDCNNs only consider the signal input and convolution of vectors; therefore, how does the matrix or higher order of convolution work? Is there some relationship between MDCNNs and FNNs? How does MDCNNs behave in terms of their generalization and expressivity? These are interesting questions we left them as future work.

Data Availability

Data sharing is not applicable to this article as no datasets were generated or analysed during the current study.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This research was funded by the Fundamental Research Funds of China West Normal University (CN) (Grant No. 20B001), in part by the Sichuan Science and Technology Program (Grant No. 2023NSFSCO060), and in part by the Initiative Projects for Ph.D. in China West Normal University (Grant No. 22kE030).

References

[1] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015, https://arxiv.org/abs/1409.1556.

[3] Y. Wu, M. Schuster, Z. Chen et al., "Google's neural machine translation system: bridging the gap between human and machine translation," 2016, https://arxiv.org/abs/1609.08144.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[5] M. Kerzel, M. Ali, H. G. Ng, and S. Wermter, "Haptic material classification with a multi-channel neural network," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 439–446, Anchorage, AK, USA, 2017.

[6] S. Opałka, B. Stasiak, D. Szajerman, and A. Wojciechowski, "Multi-channel convolutional neural networks architecture feeding for effective EEG mental tasks classification," *Sensors*, vol. 18, no. 10, p. 3451, 2018.

[7] C. Xu, W. R. Huang, H. W. Wang, G. Wang, and T. Y. Liu, "Modeling local dependence in natural language with multi-channel recurrent neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 1, pp. 5525–5532, 2019.

[8] C. Q. Quan, L. Hua, X. Sun, and W. J. Bai, "Multichannel convolutional neural network for biological relation extraction," *BioMed Research International*, vol. 2016, Article ID 1850404, 10 pages, 2016.

[9] Z. W. Zheng, N. Q. K. Le, and M. C. H. Chua, "MaskDNA-PGD: an innovative deep learning model for detecting DNA methylation by integrating mask sequences and adversarial PGD training as a data augmentation method," *Chemometrics and Intelligent Laboratory Systems*, vol. 232, article 104715, 2023.

[10] J. N. Sua, S. Y. Lim, M. H. Yulius et al., "Incorporating convolutional neural networks and sequence graph transform for identifying multilabel protein lysine PTM sites," *Chemometrics and Intelligent Laboratory Systems*, vol. 206, article 104171, 2020.

[11] S. Ohtani, Y. Kato, N. Kuroki, T. Hirose, and M. Numa, "Multi-channel convolutional neural networks for image super-resolution," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. -E100.A, no. 2, pp. 572–580, 2017.

[12] H. B. Wang, W. Cui, and B. Ye, "Lane Departure assistance coordinated control system for in-wheel motor drive vehicles based on dynamic extension boundary decision," *Journal of Systems Science and Complexity*, vol. 33, no. 4, pp. 1040–1063, 2020.

[13] J. P. Zhu, Q. Meng, W. Chen, and Z. M. Ma, "Interpreting the basis path set in neural networks," *Journal of Systems Science and Complexity*, vol. 34, no. 6, pp. 2155–2167, 2021.

[14] D. X. Zhou, "Universality of deep convolutional neural networks," *Applied and Computational Harmonic Analysis*, vol. 48, no. 2, pp. 787–794, 2020.

[15] D. X. Zhou, "Theory of deep convolutional neural networks: downsampling," *Neural Networks*, vol. 124, pp. 319–327, 2020.

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT press, 2016.

[17] S. Mallat, "Understanding deep convolutional networks," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, article 20150203, 2016.

[18] P. Petersen and F. Voigtlaender, "Equivalence of approximation by convolutional neural networks and fully-connected networks," *Proceedings of the American Mathematical Society*, vol. 148, no. 4, pp. 1567–1581, 2020.

[19] D. X. Zhou, "Deep distributed convolutional neural networks: universality," *Analysis and Applications*, vol. 16, no. 6, pp. 895–919, 2018.

[20] Y. Li, X. Cheng, and J. Lu, "Butterfly-Net: optimal function representation based on convolutional neural networks," 2020, https://arxiv.org/abs/1805.07451.

[21] M. Li, L. Demanet, and L. Zepeda-Núñez, "Wide-band butterfly network: stable and efficient inversion via multi-frequency neural networks," *Multiscale Modeling and Simulation*, vol. 20, no. 4, pp. 1191–1227, 2022.

[22] J. M. Klusowski and A. R. Barron, "Approximation by combinations of ReLU and squared ReLU ridge functions with ℓ^1 and ℓ^0 controls," *Transactions on Information Theory*, vol. 64, no. 12, pp. 7649–7656, 2018.

[23] F. Cucker and D. X. Zhou, *Learning Theory: An Approximation Theory Viewpoint*, Cambridge University Press, 2010.

[24] Y. Gordon, V. Maiorov, M. Meyer, and S. Reisner, "On the best approximation by ridge functions in the uniform norm," *Constructive Approximation*, vol. 18, no. 1, pp. 61–85, 2001.

[25] P. P. Petrushev, "Approximation by ridge functions and neural networks," *SIAM Journal on Mathematical Analysis*, vol. 30, no. 1, pp. 155–189, 1998.

[26] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 930–945, 1993.

[27] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[28] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[29] A. Pinkus, "Approximation theory of the MLP model in neural networks," *Acta Numerica*, vol. 8, pp. 143–195, 1999.

[30] V. E. Maiorov and R. Meir, "On the near optimality of the stochastic approximation of smooth functions by neural networks," *Advances in Computational Mathematics*, vol. 13, no. 1, pp. 79–103, 2000.

[31] J. J. Wang and Z. B. Xu, "Neural networks and the best trigomometric approximation," *Journal of Systems Science and Complexity*, vol. 24, no. 2, pp. 401–412, 2011.

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

[33] D. Yarotsky, "Quantified advantage of discontinuous weight selection in approximations with deep neural networks," 2017, https://arxiv.org/abs/1705.01365.

[34] I. Daubechies, R. DeVore, S. Foucart, B. Hanin, and G. Petrova, "Nonlinear approximation and (deep) ReLU networks," *Constructive Approximation*, vol. 55, no. 1, pp. 127–172, 2022.