Deep Convolutional Neural Networks

Ding-Xuan Zhou
School of Data Science, Department of Mathematics
Liu Bie Ju Centre for Mathematical Sciences
City University of Hong Kong, Kowloon, Hong Kong
Email: mazhou@cityu.edu.hk

Abstract

Deep learning has been very successful in dealing with big data from various fields of science and engineering. It has brought breakthroughs by using various deep neural network architectures and structures according to different learning tasks. An important family of deep neural networks are deep convolutional neural networks. We give a survey for deep convolutional neural networks induced by 1-D or 2-D convolutions. We demonstrate how these networks are derived from convolutional structures, and how they can be used to approximate functions efficiently. In particular, we illustrate with explicit rates of approximation that in general, deep convolutional neural networks perform at least as well as fully connected shallow networks, and they can outperform fully connected shallow networks in approximating radial functions when the dimension of data is large.

Keywords: deep learning, convolutional neural networks, convolutional filters, approximation, radial functions, generalization ability, zero-padding, parallel channels

1 Classical Fully Connected Neural Networks

Deep learning has become a very powerful tool for processing big data from computer vision, signal analysis, natural language processing, and many other fields of science and technology [10]. Its great success is based on structured deep neural networks by imposing special structures to the classical fully connected neural networks according to various purposes and applications [20, 13, 19].

Classical neural networks aim at modelling human brains by neural network architectures. A basic structure is motivated by observations from neural sciences, to use linear maps followed by simple nonlinearities induced by nonlinear activation functions. Commonly used nonlinear activation functions include the tanh function

$$\tanh(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}, \qquad u \in \mathbb{R},$$

the sigmoid function

$$s(u) = \frac{1}{1+e^{-u}}, \qquad u \in \mathbb{R},$$

and the rectified linear unit (ReLU)

$$\sigma(u) = \max\{0, u\} = \begin{cases} u, & \text{if } u \geq 0, \\ 0, & \text{if } u < 0. \end{cases} \tag{1.1}$$

The classical shallow neural network to represent functions or process data on $\mathbb{R}^d$ has an output function of the form

$$f_N(x) = \sum_{k=1}^{N} c_k \sigma([w]_k \cdot x - b_k). \tag{1.2}$$

Here $x := (x_1, x_2, \ldots, x_d)^T \in \mathbb{R}^d$ is the vector of input variables, $\sigma : \mathbb{R} \to \mathbb{R}$ is an activation function, N is the number of hidden neurons, and $\{[w]_k = (w_{k1}, w_{k2}, \ldots, w_{kd})^T \in \mathbb{R}^d, b_k \in \mathbb{R}, c_k \in \mathbb{R}\}$ are parameters corresponding to connection weights, biases (or thresholds), and coefficients, with $[w]_k \cdot x$ being the dot product in $\mathbb{R}^d$ representing the stimulation of the input data to the k-th hidden neuron. The shallow network (1.2) can be expressed in a matrix form as

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \xrightarrow{\{[w]_k, b_k\}} \begin{bmatrix} \sigma([w]_1 \cdot x - b_1) \\ \sigma([w]_2 \cdot x - b_2) \\ \vdots \\ \sigma([w]_N \cdot x - b_N) \end{bmatrix} = \sigma(Fx - b) \xrightarrow{\{c_k\}_{k=1}^N} c \cdot \sigma(Fx - b),$$

where $b = (b_k)_{k=1}^N \in \mathbb{R}^N$, σ acts componentwise on vectors, and

$$F = \begin{bmatrix} [w]_1^T \\ [w]_2^T \\ \vdots \\ [w]_N^T \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1d} \\ w_{21} & w_{22} & \cdots & w_{2d} \\ \vdots & \ddots & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{Nd} \end{bmatrix} \tag{1.3}$$

is an $N \times d$ matrix, called **connection matrix**, with rows given by the connection vectors. Such a network is **fully connected** since the connection matrix F is full with all the entries being free parameters to be trained. There are $N(d+2)$ free parameters contained in $\{[w]_k \in \mathbb{R}^d, b_k, c_k \in \mathbb{R}\}$. This number of parameters to be trained is huge when the input data dimension d is large and $N >> d$. For example, in image processing, one starts with a digital image $X : \{1, \ldots, m\} \times \{1, \ldots, \ell\} \to \mathbb{R}$ with the side widths m, ℓ in the order of hundreds. Applying vectorization to X leads to the input data of size $d = m\ell$. Then the number of parameters to be trained in (1.2) with $N >> d$ would be at least in the order of tens of billions, which is too huge to be implemented.

Shallow neural networks have been well developed. They have been extended to a form with more hidden layers $\{H^{(j)} : \mathbb{R}^d \to \mathbb{R}^{n_j}\}_{j=1}^J$ with widths $\{n_j\}$ defined iteratively with the input layer $H^{(0)}(x) = x$ of width $n_0 = d$ as

$$H^{(j)}(x) = \sigma\left(F^{(j)}H^{(j-1)}(x) - \hat{b}^{(j)}\right), \qquad j = 1, 2, \ldots, J \tag{1.4}$$

with connection matrices $F^{(j)} \in \mathbb{R}^{n_j \times n_{j-1}}$ and bias vectors $\hat{b}^{(j)} \in \mathbb{R}^{n_j}$. The output function is $c \cdot H^{(J)}(x)$ with $c \in \mathbb{R}^{n_J}$. It can be displayed as

$$x \overset{F^{(1)},\hat{b}^{(1)}}{\longrightarrow} H^{(1)}(x) \overset{F^{(2)},\hat{b}^{(2)}}{\longrightarrow} H^{(2)}(x) \to \cdots \to H^{(J-1)}(x) \overset{F^{(J)},\hat{b}^{(J)}}{\longrightarrow} H^{(J)}(x) \overset{c}{\to} c \cdot H^{(J)}(x).$$

Again, an essential property of the above multi-layer neural network is the fully-connected nature of (1.4) where $F^{(j)} \in \mathbb{R}^{n_j \times n_{j-1}}$ is a full matrix consisting of $n_j n_{j-1}$ free parameters. So the difficulty with large numbers of free parameters also exists.

Representation and approximation of functions on subsets of $\mathbb{R}^d$ by the shallow neural networks (1.2) or multi-layer neural networks (1.4) was studied well in a large classical literature [7, 14, 1, 27, 3] in the late 1980s. See the survey [31] for details and references. A particular research problem called universality of approximation is to consider when a neural network of the form (1.2) or (1.4) can approximate any continuous function on any compact subset of $\mathbb{R}^d$ to an arbitrary accuracy when N or $\sum_{j=1}^J n_j$ is large enough, see [7, 14, 1] and references therein. A key point to ensure the universality of the neural networks (1.2) or (1.4) is the complete freedom in taking the weights $\{[w]_k\}$ in (1.2) or $\{F^{(j)}\}$ in (1.4), and these neural networks are called fully connected because of this feature. As we have seen for the shallow neural network (1.2), from the fully connectedness one can easily calculate the number of free parameters of weights to be $\sum_{j=1}^J n_j n_{j-1}$ for the multi-layer neural network (1.4), very large when the input data dimension d is high, which makes these neural networks hard to implement for dealing with big data in huge dimensions.

2 Deep CNNs Induced by 1-D Convolutions

A basic idea of deep learning is to reduce the computational complexity of the multi-layer neural networks (1.4) involving too many free parameters to a lower level at each single layer by imposing structures and network architectures while increasing the learning abilities by allowing more layers and channels [20]. This has led to great progress on artificial intelligence. The imposed structures and architectures are crucial, which makes deep learning essentially different from classical learning schemes based on fully connected neural networks. The earliest and most important family of deep neural network structures are generated by convolutions. The produced deep network architectures are called **deep convolutional neural networks** (CNNs).

These structured networks were introduced in [38] as time-delay neural networks and in [41] shift invariant neural networks, and have been proved to be very efficient for speech recognition, image classification, language translation, and many other practical tasks [13, 19, 10]. By applying error-correction tuning methods in graphics processing units such as backpropagation and stochastic gradient descent, they provide scalable deep learning algorithms and computationally feasible and satisfactory solutions to many practical problems for capturing hierarchical data features efficiently and modelling deep abstractions from big data. It has been believed that the convolutional structures in CNNs enable the induced deep learning algorithms to capture local shift-invariance properties of natural image and speech data, leading to super efficiency in implementing many learning tasks. We provide some theoretical hints in this review article.

2.1 1-D convolution and 1-D CNNs

Now we describe how the special structure imposed on 1-D CNNs is induced by convolutions. The 1-D convolution is computed for sequences $w = (w_i)_{i\in\mathbb{Z}}$ and $x = (x_i)_{i\in\mathbb{Z}}$ on the 1-D lattice $\mathbb{Z}$ and is defined as another sequence $w*x$ on $\mathbb{Z}$ by

$$(w*x)_i = \sum_{j=-\infty}^{\infty} w_{i-j}x_j, \qquad i \in \mathbb{Z}.$$

In deep learning, short supported filters w are often used to attract local features. We assume that w is a filter $w = (w_j)_{j\in\mathbb{Z}}$ supported in $\{0, 1, \ldots, s\}$ for some filter length $s \in \mathbb{N}$ to control the locality and sparsity, meaning that $w_j = 0$ for $j \notin [0, s]$. When x is a digital signal supported in $\{1, 2, \ldots, D\}$ for some $D \in \mathbb{N}$, we have

$$(w*x)_i = \sum_{k\in\mathbb{Z}} w_{i-k}x_k = \sum_{k=1}^{D} w_{i-k}x_k, \qquad i \in \mathbb{Z}. \tag{2.1}$$

In processing 1-D digital signal x, we believe its energy to be finite and regard its restriction to a finite set as a good approximation, leading to the assumption by shifting that x is supported in $\{1, 2, \ldots, D\}$ for some $D \in \mathbb{N}$. The identity (2.1) becomes valid by setting entries outside $\{1, 2, \ldots, D\}$ to be zero. This approach is called **zero-padding**. By this approach, the convoluted sequence $w*x$ defined by (2.1) is supported in $\{1, 2, \ldots, D + s\}$. By restricting the index i onto this set, we know that the possibly nonzero entries of the convoluted sequence $w*x$ can be

expressed in a vector form as

$$\begin{bmatrix} (w*x)_1 \\ (w*x)_2 \\ \vdots \\ (w*x)_D \\ \vdots \\ (w*x)_{D+s} \end{bmatrix} = T^w \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}$$

with

$$T^w := \begin{bmatrix} w_0 & 0 & 0 & 0 & \dots & 0 & 0 \\ w_1 & w_0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ w_s & w_{s-1} & \dots & w_0 & \dots & 0 & 0 \\ 0 & w_s & \dots & w_1 & \ddots & \vdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \dots & \dots & 0 & w_s & \dots & w_1 & w_0 \\ \dots & \dots & \dots & 0 & w_s & \dots & w_1 \\ \vdots & \dots & \dots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & \dots & 0 & w_s \end{bmatrix} \in \mathbb{R}^{(D+s)\times D}. \tag{2.2}$$

Here the Toeplitz type sparse matrix T^w, with "T" standing for "Toeplitz", is induced by the 1-D convolution and is called a **convolutional matrix**. The number of parameters $\{w_k\}_{k=0}^s$ contained in this structured connection matrix is $s+1$, much smaller than the number of entries $D(D+s)$ of a full connection matrix of the same size. This great reduction of parameter numbers at one layer allows CNNs to have large depths. A 1-D CNN of depth J with zero-padding is induced by a sequence of convolutional filters $\mathbf{w} = \{w^{(j)}\}_j$, each supported in $\{0, 1, \dots, s\}$.

Definition 1. *A 1-D CNN of depth J with convolutional filters $\mathbf{w} := \{w^{(j)}\}_{j=1}^J$ and linearly increasing widths $\{d_j = d + js\}_{j=1}^J$ is a sequence of J vectors $h^{(j)}(x)$ of functions on $\mathbb{R}^d$ given iteratively by $h^{(0)}(x) = x \in \mathbb{R}^d$ and*

$$h^{(j)}(x) = \sigma\left(T^{(j)} h^{(j-1)}(x) - b^{(j)}\right), \qquad j = 1, 2, \dots, J, \tag{2.3}$$

where $T^{(j)} = \left(w^{(j)}_{i-k}\right)$ is a $d_j \times d_{j-1}$ convolutional matrix and $b^{(j)}$ a bias vector.

Observe that the convolutional matrix $T^{(w)}$ has identical sums of the rows in the middle. So to reduce the number of free parameters, except the last iteration, we take $b^{(j)}$ of the form

$$[b_1 \ \dots b_s \ b_{s+1} \ b_{s+1} \ \dots \ b_{s+1} \ b_{d_j - s+1} \ \dots b_{d_j}]^T \tag{2.4}$$

with the $d_j - 2s$ repeated components in the middle. The iteration relation (2.3) for the CNN is the same as (1.4) for fully connected networks, but the full connection matrix $F^{(j)}$ is replaced by the sparse convolutional matrix $T^{(j)}$. The sparsity of $T^{(j)}$ and the special form (2.4) of $b^{(j)}$ tell us that the j-th iteration of the 1-D CNN involves $3s + 2$ free parameters only. So in addition to the $2d_J + s + 1$ free parameters for $b^{(J)}, c \in \mathbb{R}^{d_J}, w^{(J)}$, the total number of free parameters in the CNN is $(5s + 2)J + 2d - 2s - 1$, much smaller than that in the classical fully connected multi-layer neural network (1.4) with full connection matrices $T^{(j)}$ involving $d_j d_{j-1}$ free parameters. It demonstrates the computational efficiency of CNNs.

2.2 Approximation of functions by 1-D CNNs

To measure the modelling, representing, or approximating abilities of 1-D CNNs, we take linear combinations of the components of the last layer $h^{(J)}$ to form the **hypothesis space** for learning or approximation:

$$\mathcal{H}_J^{\mathbf{w},\mathbf{b}} = \left\{ c \cdot h^{(J)}(x) = \sum_{k=1}^{d_J} c_k h_k^{(J)}(x) : c \in \mathbb{R}^{d_J} \right\}. \tag{2.5}$$

Observe that each function in the hypothesis space $\mathcal{H}_J^{\mathbf{w},\mathbf{b}}$ is a continuous piecewise linear function (linear spline) on any compact subset Ω of $\mathbb{R}^d$. This hypothesis space and its approximation ability depend completely on the sequence of convolutional filters $\mathbf{w} = \{w^{(j)}\}_{j=1}^J$ and the sequence of bias vectors $\mathbf{b} := \{b^{(j)}\}_{j=1}^J$. As the depth J becomes larger, the dimension $d + Js$ of $\mathcal{H}_J^{\mathbf{w},\mathbf{b}}$ increases linearly and the corresponding 1-D CNN can represent functions of richer structures. The following theorem [43] on universality of 1-D CNNs asserts that any function $f \in C(\Omega)$, the space of continuous functions on Ω with norm $\|f\|_{C(\Omega)} = \sup_{x\in\Omega} |f(x)|$, can be approximated by $\mathcal{H}_J^{\mathbf{w},\mathbf{b}}$ to an arbitrary accuracy when the depth J is large enough.

Theorem 1. *Let $2 \le s \le d$. For any compact subset Ω of $\mathbb{R}^d$ and any $f \in C(\Omega)$, there exist sequences $\mathbf{w}$ of filters, $\mathbf{b}$ of bias vectors and $f_J^{\mathbf{w},\mathbf{b}} \in \mathcal{H}_J^{\mathbf{w},\mathbf{b}}$ such that*

$$\lim_{J\to\infty} \|f - f_J^{\mathbf{w},\mathbf{b}}\|_{C(\Omega)} = 0.$$

To understand the modelling ability of CNNs quantitatively, rates of approximation are desired. The following theorem [43] presents rates of approximating functions in the Sobolev space $W_2^r(\Omega)$ with an integer index $r > 2 + d/2$. Such a function f is the restriction to Ω of a function F from the Sobolev space $W_2^r(\mathbb{R}^d)$ on $\mathbb{R}^d$ meaning that F and all its partial derivatives up to order r are square integrable on $\mathbb{R}^d$.

Theorem 2. *Let $2 \le s \le d$ and $\Omega \subseteq [-1,1]^d$. If $J \ge 2d/(s-1)$ and $f = F|_\Omega$ with $F \in W_2^r(\mathbb{R}^d)$ and an integer index $r > 2+d/2$, then there exist $\mathbf{w}$, $\mathbf{b}$ and $f_J^{\mathbf{w},\mathbf{b}} \in \mathcal{H}_J^{\mathbf{w},\mathbf{b}}$ such that*

$$\|f - f_J^{\mathbf{w},\mathbf{b}}\|_{C(\Omega)} \le c\,\|F\|_{W_2^r}\,\sqrt{\log J}\,(1/J)^{\frac{1}{2}+\frac{1}{d}}\,, \tag{2.6}$$

where c is an absolute constant and $\|F\|_{W_2^r}$ denotes the Sobolev space norm of $F \in W_2^r(\mathbb{R}^d)$.

Take $s = \lceil 1 + d^\tau/2 \rceil$ and $J = \lceil 4d^{1-\tau} \rceil L$ with $0 \le \tau \le 1$ and $L \in \mathbb{N}$ in Theorem 2, where $\lceil u \rceil$ denotes the smallest integer not smaller than u. Then we have

$$\|f - f_J^{\mathbf{w},\mathbf{b}}\|_{C(\Omega)} \le c\,\|F\|_{W_2^r}\,\sqrt{\frac{(1-\tau)\log d + \log L + \log 5}{4d^{1-\tau}L}},$$

while the widths of the CNN are bounded by $12Ld$ and the total number of free parameters by

$$(5s+2)J + 2d - 2s - 1 \le (73L+2)d.$$

We can even take $L = 1$ and $\tau = 1/2$ to get a bound for the relative error

$$\frac{\|f - f_J^{\mathbf{w},\mathbf{b}}\|_{C(\Omega)}}{\|F\|_{W_2^r}} \le \frac{c}{2} d^{-\frac{1}{4}} \sqrt{\log(5\sqrt{d})}$$

achieved by a 1-D CNN of depth $\lceil 4\sqrt{d} \rceil$ and at most $75d$ free parameters, which decreases as the dimension d increases. This interesting observation is new for CNNs, and does not exist in the literature of fully connected neural networks. It may explain the strong modelling ability of CNNs.

The regularity index $r > 2 + d/2$ required in Theorem 2 is large when d increases. It is needed in the analysis due to the function regularity on the whole Euclidean space $\mathbb{R}^d$. The issue of approximating non-smooth functions by fully connected neural networks was considered in [35, 15]. Moreover, the Sobolev space $W_2^r(\mathbb{R}^d)$ requires derivatives of various orders to belong to the L_2 space, while the error is measured in the L_∞ norm in Theorem 2. The difficulty in the large regularity index and the inconsistency of L_2 and L_∞ norms was overcome in [9] for the setting with data from the unit sphere $\mathbb{S}^{d-1}$ of $\mathbb{R}^d$. The approximation of functions in the Sobolv space $W_\infty^r(\Omega)$ with $r > 0$ is measured in the $C(\Omega)$ norm and the analysis there is conducted with spherical harmonic expansions.

3 Superiority of 1-D CNNs

CNNs have demonstrate their super performances in many learning tasks in practice and are better than fully connected neural networks in some applications. It

is believed and desired to prove theoretically that CNNs can attract some special features faster than fully connected neural networks. We describe some evidence on the superiority of 1-D CNNs in this section.

3.1 Realizing fully connected networks by 1-D CNNs

CCNs stated in Definition 1 have linearly increasing widths $\{d_j = d + js\}$, which restricts the depth and computational complexity of the network. In deep learning, there are various techniques to control network widths such as dropout and pooling [10]. One operation motivated by the literature of wavelets [8, 24] is **downsampling** which enables rigorous analysis of 1-D CNNs. Denote the integer part of $u \in \mathbb{R}_+$ as $[u]$. Define an activated affine mapping $\mathcal{A}_{F,b} : \mathbb{R}^{d_{j-1}} \to \mathbb{R}^{d_{j-1}+s}$ associated with a $(d_{j-1} + s) \times d_{j-1}$ matrix F and a bias vector $b \in \mathbb{R}^{d_{j-1}+s}$ as

$$\mathcal{A}_{F,b}(u) = \sigma\left(Fu - b\right), \qquad u \in \mathbb{R}^{d_{j-1}}.$$

Definition 2. *The downsampling operator $\mathcal{D}_m : \mathbb{R}^D \to \mathbb{R}^{[D/m]}$ with a scaling parameter $m \le D$ is defined by*

$$\mathcal{D}_m(v) = (v_{im})_{i=1}^{[D/m]}, \qquad v \in \mathbb{R}^D. \tag{3.1}$$

A **downsampled 1-D CNN** *with ℓ downsamplings at layers $\mathcal{J} := \{J_k\}_{k=1}^{\ell}$ with $1 < J_1 \le J_2 \le \ldots \le J_\ell = J$ has widths $\{d_j\}_{j=0}^{J}$ defined iteratively by $d_0 = d$ and for $k = 1, \ldots, \ell$,*

$$d_j = \begin{cases} d_{j-1} + s, & \text{if } J_{k-1} < j < J_k, \\ \left[(d_{j-1} + s)/d_{J_{k-1}}\right], & \text{if } j = J_k, \end{cases} \tag{3.2}$$

and is a sequence of function vectors $\left\{h^{(j)}(x) : \mathbb{R}^d \to \mathbb{R}^{d_j}\right\}_{j=1}^{J}$ defined iteratively by $h^{(0)}(x) = x$ and for $k = 1, \ldots, \ell$,

$$h^{(j)}(x) = \begin{cases} \mathcal{A}_{T^{(j)},b^{(j)}}\left(h^{(j-1)}(x)\right), & \text{if } J_{k-1} < j < J_k, \\ \mathcal{D}_{d_{J_{k-1}}} \circ \mathcal{A}_{T^{(j)},b^{(j)}}\left(h^{(j-1)}(x)\right), & \text{if } j = J_k. \end{cases} \tag{3.3}$$

The following theorem [44] proves rigorously that output functions produced by any deep fully connected neural network (1.4) associated with ReLU can be realized by a downsampled 1-D CNN. It confirms as done for periodized CNNs in [30] that the representation and approximation ability of CNNs is at least as good as that of fully connected networks.

Theorem 3. *Let Ω be a compact subset of $\mathbb{R}^d$ and $\{H^{(k)} : \mathbb{R}^d \to \mathbb{R}^{n_k}\}_{k=1}^{\ell}$ be an ℓ-layer fully connected neural network satisfying (1.4) with connection matrices $F^{(k)}$, bias vector $\hat{b}^{(k)}$ such that $n_k n_{k-1} > 1$ for each $k \in \{1, \ldots, \ell\}$. If $2 \le s \le n_k n_{k-1}$ for each k, then there is a downsampled 1-D CNN $\left\{h^{(j)}(x) : \mathbb{R}^d \to \mathbb{R}^{d_j}\right\}_{j=1}^{J}$ with ℓ*

downsamplings at layers $\{J_k = \sum_{j=1}^{k} \Delta_j\}$ *with* $\Delta_j \leq \lceil \frac{n_j n_{j-1}-1}{s-1} \rceil$ *for each* j *together with bias vectors* $b^{(j)} \in \mathbb{R}^{d_{j-1}+s}$ *satisfying (2.4) for* $j \notin \mathcal{J}$ *such that*

$$h^{(J_k)}(x) = H^{(k)}(x), \qquad \forall k \in \{1, \ldots, \ell\},\ x \in \Omega. \tag{3.4}$$

The total number of free parameters in the above net is at most $8 \sum_{k=1}^{\ell} (n_k n_{k-1})$ *and is at most* 8 *times of that of the fully connected network.*

The conclusion of Theorem 3 is drawn by means of a fundamental result [42, 43] on convolutional factorization that an arbitrary pre-assigned sequence $W = (W_k)_{-\infty}^{\infty}$ supported in $\{0, \ldots, \mathcal{M}\}$ can be factorized into convolutions of a filter sequence $\{w^{(j)}\}_{j=1}^{J}$ supported in $\{0, \ldots, s\}$ with $J < \frac{\mathcal{M}}{s-1} + 1$. It demonstrates the role of convolutions in 1-D CNNs.

Theorem 4. *Let* $s \geq 2$ *and* $W = (W_k)_{-\infty}^{\infty}$ *be a sequence supported in* $\{0, \ldots, \mathcal{M}\}$ *with* $\mathcal{M} \geq 0$. *Then there exists a finite sequence of filters* $\{w^{(j)}\}_{j=1}^{J}$ *supported in* $\{0, \ldots, s\}$ *with* $J < \frac{\mathcal{M}}{s-1} + 1$ *such that the convolutional factorization*

$$W = w^{(J)} * \ldots * w^{(2)} * w^{(1)} \tag{3.5}$$

holds true.

Let us illustrate how the convolutional factorization (3.5) in Theorem 4 leads to the construction of 1-D CNNs in Theorem 3 by showing the case $\ell = 1$. For each $j \in \{1, \ldots, J\}$, denote $T^{(j)}$ to be the $(d + js) \times (d + (j-1)s)$ convolutional matrix. Then for the sequence W satisfying (3.5), we have $T^{(J)} \ldots T^{(1)} = T_d^W$, where T_d^W is the $(d + Js) \times d$ Toeplitz type matrix given by

$$T_d^W = (W_{i-k})_{i=1,\ldots,d+Js,k=1,\ldots,d} = \begin{bmatrix} W_0 & 0 & \cdots 0 \\ \vdots & \ddots & \ddots \vdots \\ W_{d-1} & \cdots & W_0 \\ \vdots & \ddots & \ddots \vdots \\ W_{2d-1} & \cdots & W_d \\ \vdots & \ddots & \ddots \vdots \\ W_{Nd-1} & \cdots & W_{(N-1)d} \\ \vdots & \ddots & \ddots \vdots \\ 0 & \cdots & 0\ W_{Js} \end{bmatrix}.$$

For the full connection matrix $F^{(1)}$ defining the layer $H^{(1)}(x)$ in (1.4) given by (1.3), we set a sequence W supported in $\{0, \ldots, \mathcal{M}\}$ with $\mathcal{M} = Nd - 1$ by

$$[W_{Nd-1}\ W_{Nd-2} \ldots W_0] = [w_{N1}\ w_{N2}\ \ldots w_{Nd} \ldots w_{21}\ \ldots w_{2d}\ w_{11}\ w_{12}\ \ldots w_{1d}].$$

By Theorem 4, there exists a finite sequence of filters $\{w^{(j)}\}_{j=1}^{J}$ supported in $\{0,\dots,s\}$ with $J=\left\lceil \frac{Nd-1}{s-1}\right\rceil$ such that the convolutional factorization (3.5) holds. Define a 1-D CNN of depth J with convolutional filters $\mathbf{w}:=\{w^{(j)}\}_{j=1}^{J}$ and linearly increasing widths $\{d_j=d+js\}_{j=1}^{J}$ by (2.3). By choosing the biases to be small enough satisfying (2.4), we can make the iterations (2.3) linear and obtain for $j=1,\dots,J-1$,

$$h^{(j)}(x)=T^{(j)}\cdots T^{(1)}x+B\left(\Pi_{p=1}^{j}\|w^{(p)}\|_1\right)\mathbf{1}_{d_j}, \tag{3.6}$$

where B is a constant depending on Ω and $\mathbf{1}_{d_j}$ is the constant 1 vector in $\mathbb{R}^{d_j}$. At the end, we choose the bias vector $b^{(J)}$ according to $\hat{b}^{(1)}$ in the expression of $H^{(1)}$ and get

$$h^{(J)}(x)=\sigma\left(\mathcal{D}_d\left(T^{(J)}\cdots T^{(1)}x\right)-\hat{b}^{(1)}\right). \tag{3.7}$$

But $d+Js=d+\left\lceil\frac{Nd-1}{s-1}\right\rceil s\geq d+\frac{Nd-1}{s-1}s>d+Nd-1$, So $d+Js\geq(N+1)d$, and for $i=1,\dots,N$, the id-th row of the matrix $T^{(J)}\cdots T^{(1)}=T_d^W$ equals

$$\left[W_{id-1}\ W_{id-2}\dots W_{(i-1)d}\right]=\left[w_{i1}\ w_{i2}\dots w_{id}\right],$$

which together with (3.7) implies the realization of $H^{(1)}$ by the 1-D CNN as

$$h^{(J)}(x)=\sigma\left(F^{(1)}x-\hat{b}^{(1)}\right)=H^{(1)}.$$

3.2 Superiority in approximating radial functions

Theorem 3 shows that in general, the representation and approximation ability of CNNs is at least as good as that of fully connected networks. A natural question is whether CNNs actually preform better in approximating some important classes of functions with special features. This question is answered for the class of radial functions in [25].

A function on $\Omega\subseteq\mathbb{R}^d$ is called **radial** if it takes the form $f(|x|^2)$ with $|x|=\sqrt{x_1^2+\ldots+x_d^2}$ being the norm of the input vector $x=(x_1,\dots,x_d)\in\Omega\subseteq\mathbb{R}^d$. Such functions arise naturally in statistical physics, early warning of earthquakes, 3-D point-cloud segmentation, and image rendering, and their learning by fully connected neural networks was studied in [26, 5].

The class of radial functions considered here are defined on the unit ball $\Omega=\mathbb{B}:=\{x\in\mathbb{R}^d:|x|\leq 1\}$ of $\mathbb{R}^d$, and consists of radial functions in the unit ball of the space $C^{0,1}(\mathbb{B})$ of Lipschitz functions on $\mathbb{B}$ defined by

$$\mathcal{B}\left(C_{|\cdot|}^{0,1}\right):=\left\{f(|\cdot|^2):\|f(|\cdot|^2)\|_{C^{0,1}(\mathbb{B})}\leq 1\right\}, \tag{3.8}$$

where $\|f(|\cdot|^2)\|_{C^{0,1}(\mathbb{B})}$ is the Lipschitz-1 norm of the function $f(|\cdot|^2)$ defined for functions g on $\mathbb{B}$ by

$$\|g\|_{C^{0,1}(\mathbb{B})}=\sup_{x\neq y\in\mathbb{B}}\frac{|g(x)-g(y)|}{|x-y|}+\sup_{x\in\mathbb{B}}|g(x)|. \tag{3.9}$$

The set of functions we compare outputs of CNNs with is the span of N ridge functions given as in [18] by

$$\mathcal{S}_N = \left\{ \sum_{k=1}^{N} c_k \sigma_k(a_k \cdot x - b_k) : \ \sigma_k \in C(\mathbb{R}),\ a_k \in \mathbb{R}^d,\ c_k, b_k \in \mathbb{R} \right\}. \tag{3.10}$$

Recall the hypothesis space generated by a shallow neural network is a subset of $\mathcal{S}_N$ consisting of functions with $\sigma_1 = \ldots = \sigma_N$ being an activation function.

The efficiency of a neural network generating a hypothesis space V in approximating a set U of functions on $\Omega = \mathbb{B}$ uniformly is measured by the quantity

$$\operatorname{dist}(U, V) := \sup_{f \in U} \inf_{g \in V} \|f - g\|_{L_\infty(\mathbb{B})} \tag{3.11}$$

which is the deviation of U from V in $L_\infty(\mathbb{B})$.

Theorem 5. *Let $2 \le s \le d$. We have*

$$dist\left(\mathcal{B}\left(C^{0,1}_{|\cdot|}\right), \mathcal{S}_N\right) \ge c_d N^{-\frac{1}{d-1}}, \qquad \forall N \in \mathbb{N} \tag{3.12}$$

with a constant c_d independent of N. For the 1-D CNN $\{h^{(j)} : \mathbb{R}^d \to \mathbb{R}^{d+js}\}_{j=1}^{J}$ of depth $J := \left\lceil \frac{(2N+3)d}{s-1} \right\rceil$ defined in Definition 1 with convolutional filters $\mathbf{w} := \{w^{(j)}\}_{j=1}^{J}$ and bias vectors $\mathbf{b} := \{b^{(j)}\}_{j=1}^{J}$ satisfying (2.4), followed by a fully connected layer of widths $2N+3$ with a connection matrix $F^{[J+1]}$ of identical rows and a bias vector $b^{(J+1)}$, the hypothesis space

$$\mathcal{H}_N = \left\{ c \cdot h^{(J+1)}(x) : c \in \mathbb{R}^{2N+3},\ \mathbf{w},\ \mathbf{b},\ F^{[J+1]},\ b^{(J+1)} \right\}$$

with the last layer given by

$$h^{(J+1)}(x) = \mathcal{A}_{F^{[J+1]}, b^{(J+1)}} \circ \mathcal{A}_{T^{(J)}, b^{(J)}} \circ \ldots \circ \mathcal{A}_{T^{(1)}, b^{(1)}}(x)$$

satisfies

$$dist\left(\mathcal{B}\left(C^{0,1}_{|\cdot|}\right), \mathcal{H}_N\right) \le 3\sqrt{1+4d} N^{-\frac{1}{2}}, \qquad \forall N \in \mathbb{N}. \tag{3.13}$$

The total number $\mathcal{N}$ of free parameters in this network can be bounded as $\mathcal{N} \le (14d+2)N + 22d + s + 1$.

3.3 Survey on computational complexity

In this subsection we give a short survey on computational complexity of various families of neural networks in the literature. Comparisons on approximation abilities are made by means of the total number of free parameters $\mathcal{N}$ and the total number

of computation units $\mathcal{W}$ (widths, or hidden units) required for achieving the same approximation accuracy $\epsilon > 0$.

A classical literature on approximation by shallow or multi-layer fully connected nets was developed around 1990. Besides those results [7, 14, 22] on universality of approximation with non-polynomial locally bounded and piecewise continuous activation functions, rates of approximation were obtained in [14, 1, 27, 3, 23] and references therein. When a C^∞ activation function σ satisfies $\lim_{u\to-\infty}\sigma(u) = 0, \lim_{u\to\infty}\sigma(u) = 1$ (sigmoid function) and $f = F|_{[-1,1]^d}$ for some $F \in L^2(\mathbb{R}^d)$ with the Fourier transform $\hat{F}$ satisfying $|w|\hat{F}(w) \in L^1(\mathbb{R}^d)$, it was shown in [1] that for the fully connected shallow network (1.2) and an arbitrary probability measure μ, there holds $\|f_N - f\|_{L^2_\mu([-1,1]^d)} = O(1/\sqrt{N})$. This result was extended to the ReLU activation function recently in [16]. Most results in the classical literature about rates of approximation by fully connected networks were stated for continuous activation functions σ with two special assumptions: one is that for some $b \in \mathbb{R}$,

$$\sigma^{(i)}(b) \neq 0, \qquad \forall i \in \mathbb{Z}_+ \tag{3.14}$$

and the other is that for some integer $q \neq 1$, there holds

$$\lim_{u\to-\infty} \sigma(u)/|u|^q = 0 \quad \text{and} \quad \lim_{u\to\infty} \sigma(u)/u^q = 1. \tag{3.15}$$

Such a result was presented in [27] for shallow networks (1.2) as

$$\|f_N - f\|_{C([-1,1]^d)} \leq c_{f,d,r} N^{-r/d}, \qquad \forall N \in \mathbb{N} \tag{3.16}$$

with a constant $c_{f,d,r}$, under the regularity assumption that the approximated function f lies in the Sobolev space $W^r_\infty([-1,1]^d)$ with $r \in \mathbb{N}$. For the approximation accuracy $\|f_N - f\|_{C([-1,1]^d)} \leq \epsilon$, one needs

$$\mathcal{W} = N \geq \left(\frac{c_{f,d,r}}{\epsilon}\right)^{d/r}, \quad \mathcal{N} \geq (d+2)\left(\frac{c_{f,d,r}}{\epsilon}\right)^{d/r}. \tag{3.17}$$

The ReLU activation function σ used in the recent deep learning literature does not satisfy the two special assumptions (3.14), (3.15). Explicit rates of approximation by fully connected ReLU networks were obtained recently in [16] for shallow networks, in [33] for networks with 3 hidden layers, and in [37, 40, 2, 29] for networks with more layers, based on an interesting observation that iterates of the hat function on the interval $[0,1]$ can be used to produce linear interpolating approximations of the quadratic function u^2. In particular, Theorem 1 of [40] asserts that $f \in W^r_\infty([0,1]^d)$ can be approximated within an accuracy $\epsilon \in (0,1)$ by a ReLU deep networks with at most $c(\log(1/\epsilon)+1)$ layers and $\mathcal{W} \leq c\epsilon^{-d/r}(\log(1/\epsilon)+1)$ with a constant $c = c(d,r)$. But this constant may increase very fast as d becomes large, as shown in [43]. To

be more specific, the approach in [40] is to first approximate f by a localized Taylor polynomial

$$f_1(x) = \sum_{m\in\{0,1,\ldots,N\}^d} \sum_{\|\alpha\|_1<r} \frac{D^\alpha f(m/N)}{\alpha!}\phi_m(x)(x-m/N)^\alpha, \tag{3.18}$$

where the localization at scale $1/N$ with $N \in \mathbb{N}$ is made by means of trapezoid functions $\phi_m(x) = \Pi_{i=1}^d \varphi(3Nx_i - m_i)$ supported on $m/N + [-2/N, 2/N]^d$ defined with a univariate trapezoid function $\varphi(u) = \sigma(u+2) - \sigma(u+1) - \sigma(u-1) + \sigma(u-2)$. Then for each basis function $\phi_m(x)(x-m/N)^\alpha$ in (3.18), a ReLU network of depth at least $c_1(d + \|\alpha\|_1)\log(1/\delta)$ was constructed in [40] to achieve an approximation accuracy $(d+r)\delta$ for $\delta \in (0,1)$ where $c_1 = c_1(d,r)$ is a constant. Thus, to have an accuracy $\epsilon \in (0,1)$ for approximating f by a ReLU deep network, one takes $N = \left\lceil \left(\frac{2^{d+1}d^r}{\epsilon r!}\right)^{1/r} \right\rceil$ and $\delta = \frac{\epsilon}{2^{d+1}d^r(d+r)}$ as in [40] and the depth of the network is at least $C_0 d(\log(1/\epsilon) + d + r\log d)$ with an absolute constant $C_0 > 0$ while the total number of free parameters $\mathcal{N}$ for the approximation and the number of computation units are more than the number of coefficients $\frac{D^\alpha f(m/N)}{\alpha!}$:

$$(N+1)^d \begin{pmatrix} d+r-1 \\ d \end{pmatrix} > \left(\frac{2^{d+1}d^r}{\epsilon r!}\right)^{d/r} \frac{d^{r-1}}{(r-1)!} > \epsilon^{-d/r} \left(\frac{2^{\frac{d+1}{r}}d}{r}\right)^d \frac{d^{r-1}}{(r-1)!}. \tag{3.19}$$

This shows that for a fixed r, the constant $c(d,r)$ in Theorem 1 of [40] increases very fast as d becomes large. It was shown in [2, 29, 28] that rates of approximation of some function classes by multi-layer fully connected neural networks (1.4) may be achieved by networks with sparse connection matrices $F^{(j)}$, but the locations of the sparse connections are unknown. This sparsity of unknown pattern is different from that of CNNs, the latter enables computing methods like stochastic gradient descent to learn values of the free parameters efficiently.

For the CNN network constructed in Theorem 5 of [25], the total number of free parameters $\mathcal{N}$ for achieving an accuracy $\epsilon > 0$ in approximating functions from the class $\mathcal{B}\left(C^{0,1}_{|\cdot|}\right)$ is $\mathcal{N} = O(\epsilon^{-2})$ while that of a fully connected shallow network is $O(\epsilon^{-(d-1)})$. This shows that deep neural networks are much more efficient than shallow networks in approximating radial functions when the dimension $d > 3$ is large. Further analysis on approximating functions of the form $f \circ Q$ induced by a polynomial Q on $\mathbb{R}^d$ and a univariate function f can be found in [25].

Based on the rates of approximation, we may get generalization error bounds for CNN-based learning algorithms, as done for kernel-based learning algorithms [6] and fully connected neural networks in [17, 32, 5]. The main technical difficulty for CNNs arises from the hypothesis space (2.5) which depends on the filters $\mathbf{w}$ and bias

vectors $\mathbf{b}$, and is different from a reproducing kernel Hilbert space used in kernel methods. Detailed generalization error bounds are presented in [25]. In particular, it shows that the generalization error bound decreases with the network depth to a minimum and then increases, verifying theoretically a trade-off phenomenon observed for network depths in many practical applications [4, 11].

Benchmarked applications of CNN models include face recognition, pose estimation, and activity recognition in computer vision, search query retrieval and sentence modeling in natural language processing, and various learning tasks in speech recognition.

Training deep learning models is key to practical applications, which can be done with modern libraries such as Caffe, TensorFlow, Theano, and Torch. Implementing CNN models with guaranteed approximation properties can be carried out by the algorithm of stochastic configuration networks [39] and some other randomized methods surveyed in [21] and references therein.

4 Deep CNNs Induced by 2-D Convolutions

CNNs used in most practical applications of deep learning for image processing and other related applications are induced by 2-D convolutions **without zero-padding**. For simplicity, we consider a digital image which can be represented as a $(d+1) \times (d+1)$ matrix $X : \{0, 1, \ldots, d\}^2 \to \mathbb{R}$ of size width $d \in \mathbb{N}$ with the entry X_α, $\alpha \in \{0, 1, \ldots, d\}^2$, representing the grey level of the image at pixel α. A possible essential difference between images and 1-D signals is that the image is only part of a broader view, meaning that setting X_α to be zero for $\alpha \notin \{0, 1, \ldots, d\}^2$ might be unreasonable. This leads to the 2-D CNNs without zero-padding.

Let $W = (W_\alpha)_{\alpha \in \mathbb{Z}^2}$ be a 2-D filter supported on $\{0, 1, \ldots, s\}^2$. The 2-D convolution of a filter W and $X = (X_\alpha)_{\alpha \in \mathbb{Z}^2}$ is another sequence on $\mathbb{Z}^2$ given by

$$(W * X)_\alpha = \sum_{\beta \in \mathbb{Z}^2} W_{\alpha - \beta} X_\beta, \qquad \alpha \in \mathbb{Z}^2.$$

When no zero-padding is applied, meaning that X is not extended to the lattice $\mathbb{Z}^2$ by assigning zero values outside the available domain $\{0, 1, \ldots, d\}^2$, we only take those entries $(W * X)_\alpha$ whose computations do not involve entries of X outside $\{0, 1, \ldots, d\}^2$, that is, $\alpha \in \{s, s+1, \ldots, d\}^2$. This defines a linear map $\mathcal{T}^W$ mapping a digital image $X : \{0, 1, \ldots, d\}^2 \to \mathbb{R}$ to another $\mathcal{T}^W X : \{s, s+1, \ldots, d\}^2 \to \mathbb{R}$ given by

$$\left(\mathcal{T}^W X\right)_\alpha = \sum_{\beta \in [0,d]^2} W_{\alpha - \beta} X_\beta = (W * X)_\alpha, \qquad \alpha \in \{s, \ldots, d\}^2. \tag{4.1}$$

In general, for $n \in \{0, \ldots, d\}$, we can define a linear map $\mathcal{T}^W$ mapping $X : \{n, n+1, \ldots, d\}^2 \to \mathbb{R}$ to $\mathcal{T}^W X : \{n+s, n+s+1, \ldots, d\}^2 \to \mathbb{R}$ in a similar way as (4.1) by replacing the index range $[0, d]^2$ in the summation with $[n, d]^2$ and the domain $\{s, \ldots, d\}^2$ with $\{n+s, \ldots, d\}^2$. Then a 2-D CNN can be stated as follows.

Definition 3. *Given the input side width $d \in \mathbb{N}$ and filter size $s \in [2, d]$, a 2-D CNN of depth $J \leq d/s$ is a neural network $\{h^{(j)} : \mathbb{R}^{\{0,\ldots,d\}^2} \to \mathbb{R}^{\{js,\ldots,d\}^2}\}_{j=1}^J$ defined iteratively by $h^{(0)}(X) = X \in \mathbb{R}^{\{0,\ldots,d\}^2}$ and*

$$h^{(j)}(X) = \sigma\left(\mathcal{T}^{(j)} h^{(j-1)}(X) + B^{(j)}\right), \qquad j = 1, \ldots, J, \tag{4.2}$$

where $\mathcal{T}^{(j)}$ is the linear operator $\mathcal{T}^{(j)} := \mathcal{T}^{W^{(j)}}$ defined by (4.1) with $n = (j-1)s$, a 2-D filter $W^{(j)} : \mathbb{Z}^2 \to \mathbb{R}$ supported on $\{0, \ldots, s\}^2$, and $B^{(j)} \in \mathbb{R}^{\{js,\ldots,d\}^2}$ is a bias matrix.

The linear operator $\mathcal{T}^W$ defined by (4.1) is a tensor of type $(2, 2)$ called convolutional tensor. It may have a matrix representation if we vectorize the matrices X and $\mathcal{T}^W X$ in (4.1). But the vectorization leads to large-size vectors, as commented before, and the matrix representation of $\mathcal{T}^W$ is a Toeplitz block matrix with Toeplitz blocks which is not as nice as a simple Toeplitz matrix in (2.2) for the 1-D CNNs.

CNNs in 2-D defined in Definition 3 have decreasing sizes $\{(d - js + 1) \times (d - js + 1)\}$, so their representing and modelling ability is limited, especially when j increases. A core idea in the deep learning practice is to use many parallel **channels** with different filters for improving the power of CNNs in processing images. As a filter has its special function in extracting information from data, applying more channels with various filters can extract richer information and help processing the data efficiently. This parallel mechanism is commonly used, which is demonstrated by winners in the ImageNet Large Scale Visual Recognition Challenge listed in the following examples. All the deep neural networks consist of quite a few CNNs with many channels and possibly a couple of fully connected networks.

Example 1. *AlexNet (2012 winner)[19]: a 2-D CNN of* 5 *layers with* $s = 10, 4, 2, 2, 2$ *uses* $96, 256, 384, 384, 256$ *filter channels respectively.*

Example 2. *ZFNet (2013 winner)[47] : a 2-D CNN of* 3 *layers uses* $512, 1024, 512$ *filter channels respectively.*

Example 3. *VGG16 net (2014 winner in localization and second in classification performance) [34]: a 2-D CNN of* 12 *layers, all with* $s = 2$, *uses* $64, 128, 256$ *(all twice),* 512 *(*6 *times) filter channels respectively.*

Example 4. *GoogLeNet (2014 winner)[36]: a 2-D CNN of* 22 *layers is used, some involving* 256 *channels of inception modules with* $128, 192, 96, 64$ *filters respectively.*

Example 5. *ResNet (2015 winner)[12]: a deep neural network of* 152 *layers uses residual connections based on 2-D CNNs with* $s = 2$.

The trend observed from the above architectures is to use more channels and filters of smaller sizes, followed by a few fully connected networks. As the number of channels increases, the computational complexity becomes larger but the learning ability of the deep network tends stronger.

CNNs in 2-D are difficult to analyze due to the lack of a convolutional factorization for 2-D sequences as done for 1-D sequences in Theorem 4. When the 2-D filters $W = (W_\alpha)_{\alpha\in\mathbb{Z}^2}$ take a rank-1 tensor product form $W = u \otimes v$ with 1-D filters $u = (u_k)_{k\in\mathbb{Z}}$ and $v = (v_k)_{k\in\mathbb{Z}}$, some mathematical analysis has been conducted in [46] to show that such 2-D CNNs can realize approximately data relationships induced by target functions depending on linear features $U^T XV$ with $U, V \in \mathbb{R}^{d+1}$. The analysis as done for 1-D CNNs in [45] is based on Toeplitz type convolutional matrices induced by 1-D convolution without zero-padding

$$[w_{i-k}]_{s\le i\le d,\ 0\le k\le d} = \begin{bmatrix} w_s & \cdots & \cdots & w_0 & 0\cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & 0\cdots & w_s & \cdots & & w_0 \end{bmatrix}.$$

Acknowledgments

The author is supported partially by the Research Grants Council of Hong Kong [Project # CityU 11307319] and by Hong Kong Institute for Data Science. He would like to thank Tian-yi Zhou for preparing some material on 2-D CNNs for image classification and the referees for their constructive comments. Part of the material in the paper is from [43, 44, 25, 46].

References

[1] A. R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, IEEE Trans. Inform. Theory **39** (1993), 930–945.

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

[3] C. K. Chui, X. Li, H. N. Mhaskar, Limitations of the approximation capabilities of neural networks with one hidden layer, Adv. Comput. Math. **5** (1996), 233-243.

[4] C. K. Chui, S. B. Lin, B. Zhang, and D. X. Zhou, Realization of spatial sparseness by deep ReLU nets with massive data, IEEE Transactions on Neural Networks and Learning Systems, in press.

[5] C. K. Chui, S. B. Lin, D. X. Zhou, Deep neural networks for rotation-invariance approximation and learning, Analysis and Applications, Vol. 17, No. 05, pp. 737-772 (2019).

[6] F. Cucker and D. X. Zhou. Learning Theory: An Approximation Theory Viewpoint, Cambridge University Press, Cambridge, MA, 2007.

[7] G. Cybenko, Approximations by superpositions of sigmoidal functions, Math. Control, Signals, and Systems **2** (1989), 303–314.

[8] I. Daubechies, Ten Lectures on Wavelets, SIAM, 1992.

[9] Z. Y. Fang, H. Feng, S. Huang, and D. X. Zhou, Theory of deep convolutional neural networks II: Spherical analysis, Neural Networks **131** (2020), 154-162.

[10] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016.

[11] Z. Han, S. Q. Yu, S. B. Lin, and D. X. Zhou, Depth selection for deep ReLU nets in feature extraction and generalization, IEEE Transactions on Pattern Analysis and Machine Intelligence, in press.

[12] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, In Proceedings of the IEEE conference on Computer Vision and pattern recognition, pp. 770-778, 2016.

[13] G. E. Hinton, S. Osindero, Y. W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. **18** (2006), 1527-1554.

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

[15] M. Imaizumi and K. Fukumizu, Deep neural networks learn non-smooth functions effectively, in Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS), 2019.

[16] J. Klusowski and A. Barron, Approximation by combinations of ReLU and squared ReLU ridge functions with ℓ^1 and ℓ^0 controls, IEEE Transactions on Information Theory **64** (2018), 7649–7656.

[17] M. Kohler and A. Krzyzak, Nonparametric regression based on hierarchical interaction models, IEEE Trans. Inform. Theory **63** (2017), 1620-1630.

[18] V. N. Konovalov, D. Leviatan, and V. E. Maiorov, Approximation of Sobolev classes by polynomials and ridge functions, J. Approx. Theory **159** (2009), 97–108.

[19] A. Krizhevsky, I. Sutskever, and G. Hinton G, Imagenet classification with deep convolutional neural networks, *NIPS* (2012): 1097-1105.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE **86** (1998), 2278-2324.

[21] M. Li and D. Wang, Insights into randomized algorithms for neural networks: practical issues and common pitfalls, Information Sciences **382-383** (2017), 170-178.

[22] Y. V. Lin and A. Pinkus, Fundamentality of ridge functions, J. Approx. Theory **75** (1993), 295–311.

[23] V. E. Maiorov, On best approximation by ridge functions, J. Approx. Theory **99** (1999), 68–94.

[24] S. Mallat, Understanding deep convolutional networks, Phil. Trans. Royal Soc. A **374**:20150203.

[25] T. Mao, Z. J. Shi, and D. X Zhou, Theory of deep convolutional neural networks III: Approximating radial functions, preprint, 2020.

[26] B. McCane and L. Szymanski, Efficiency of deep networks for radially symmetric functions, Neurocomputing **313** (2018), 119–124.

[27] H. N. Mhaskar, Approximation properties of a multilayered feedforward artificial neural network, Adv. Comput. Math. **1** (1993), 61-80.

[28] H. N. Mhaskar, T. Poggio, Deep vs. shallow networks: An approximation theory perspective, Anal. Appl., 2016 (14), 829-848.

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

[30] P. Petersen and F. Voigtlaender, Equivalence of approximation by convolutional neural networks and fully-connected networks, Proceedings of the American Mathematical Society **148** (2020), 1567-1581.

[31] A. Pinkus, Approximation theory of the MLP model in neural networks, Acta Numerica **8** (1999), 143–195.

[32] J. Schmidt-Hieber, Nonparametric regression using deep neural networks with ReLU activation function, Ann. Stat., to appear. arXiv preprint arXiv: 1708.06633, 2017.

[33] U. Shaham, A. Cloninger, and R. Coifman, Provable approximation properties for deep neural networks, Appl. Comput. Harmonic Anal. **44** (2018), 537–557.

[34] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, ICLR (2015).

[35] T. Suzuki, Adaptivity of deep ReLU network for learning in Besov and mixed smooth Besov spaces: optimal rate and curse of dimensionality, in Proceedings of the International Conference on Learning Representations (ICLR), 2019.

[36] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, Going deeper with convolutions, In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 1-9, 2015.

[37] Telgarsky M (2016) Benefits of depth in neural networks. *29th Annual Conference on Learning Theory* PMLR 49:1517–1539.

[38] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, Phoneme recognition using time-delay neural networks, IEEE Transactions on Acoustics, Speech, and Signal Processing **37** (1989), 328-339.

[39] D. Wang and M. Li, Stochastic configuration networks: Fundamentals and algorithms, IEEE Transactions on Cybernetics **47** (2017), 3466-3479.

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

[41] W. Zhang, K. Itoh, J. Tanida, and Y. Ichioka, Parallel distributed processing model with local space-invariant interconnections and its optical architecture, Applied Optics **29** (1990), 4790-4797.

[42] D. X. Zhou, Deep distributed convolutional neural networks: universality, Anal. Appl. **16** (2018), 895–919.

[43] D. X. Zhou, Universality of deep convolutional neural networks, Appl. Comput. Harmonic Anal. **48** (2020), 787-794.

[44] D. X. Zhou, Theory of deep convolutional neural networks: Downsampling, Neural Networks **124** (2020), 319-327.

[45] D. X. Zhou, Distributed approximation with deep convolutional neural networks, preprint, 2020.

[46] T. Y. Zhou and D. X. Zhou, Theory of deep CNNs induced by 2D convolutions, preprint, 2020.

[47] M. D. Zeiler and R. Fergus, Visualizing and understanding convolutional networks, European Conference on Computer Vision, Springer, Cham, 2014.