Expressivity of deep neural networks

Jianfei Li[1] and Gitta Kutyniok[2]

[1] Ludwig-Maximilians-Universität München, Munich, Germany,
lijianfei@math.lmu.de,
[2] Ludwig-Maximilians-Universität München, Munich, Germany,
Munich Center for Machine Learning (MCML), Munich, Germany,
University of Tromsø, Tromsø, Norway,
DRL-German Aerospace Center, Germany,
kutyniok@math.lmu.de

Abstract. This chapter focuses on the approximation theory of deep ReLU neural networks, analyzing their ability to approximate various target functions with different network architectures. We begin by introducing the universal approximation theory of deep neural networks, stating that given enough neurons, neural networks can approximate general functions. We then delve into the fundamental properties of ReLU neural networks and explore the role of width and depth of neural networks, highlighting that increasing layers could be more effective than increasing width in improving approximation accuracy. Next, we discuss the approximation rates for Sobolev functions using fully connected and convolutional neural networks. To alleviate the curse of dimensionality, we further consider Korobov functions. Finally, we focus on the approximation properties of self-attention and transformers, which have become increasingly important in modern deep learning. These results shed light on the expressivity and reliability of deep learning models, providing valuable insights into networks' behavior and performance.

Keywords: deep learning, neural networks, universality, approximation error

1 Neural networks and their expressivity

1.1 Introduction

In the era of artificial intelligence (AI), deep learning has become a cornerstone of modern science and technology, playing a pivotal role in image processing and natural language processing [90, 45, 34, 71, 11, 31, 2]. Beyond these traditional machine learning tasks, deep learning is rapidly evolving and has achieved significant breakthroughs in various fields, including autonomous driving [113, 56], chatbots and conversational AI [3, 54, 51], drug discovery [88, 16, 91], and many more.

The origins of deep learning can be traced back decades, with research conducted in classical machine learning. Inspired by the human brain, deep learning

aims to mimic its functionality [5]. Artificial neurons, developed to model their biological counterparts, are mutually connected to form neural network models. As the number of these artificial neurons grows, the complexity of the network increases, enabling it to process big data. However, on the other hand, neural network architectures have become increasingly sophisticated, presenting great challenges, such as safety issues involving robots [100], security concerns in AI systems [49], and privacy violations related to health data [79]. These problems underscore the need to improve the reliability of AI technology. From a mathematical point of view, exploring the mathematical foundations of deep learning is a promising direction for improving our understanding of deep learning and, hopefully, helping to address the aforementioned problems.

One way to understand the learning ability of neural networks is to consider their approximation properties. Approximation theory is a fundamental mathematical tool for the quantitative analysis of how closely neural networks can approach a certain target function. By studying approximation errors, we can gain insight into the complexity and limitations of neural networks.

There are at least three distinct benefits that approximation theory brings to the field of deep learning. Firstly, approximation theory provides a mathematical framework for comparing different neural network architectures, such as fully connected neural networks, convolutional neural networks, and transformers. Secondly, it enhances our understanding of neural network design, for example, offering insights into why a deep network instead of a wide network is preferred in practice. Third, we can explore new architectures by leveraging approximation theory, which could make the proposed models both effective and efficient.

This chapter is organized as follows. In Sect. 2, we will review the foundational principles of neural networks and describe the most widely used architectures. In Sect. 3, we will study the universal approximation properties of deep neural networks, focusing on those with a bounded depth or width. Before analyzing complex target functions, we will present the approximation properties for some simple yet crucial target functions in Sect. 4, highlighting the significance of depth over width in neural networks. In Sect. 5, we will focus on the approximation properties for various smooth functions and different architectures.

2 Foundational principles of neural networks

A neural network is composed of two key ingredients: activation functions and its architecture. The activation function defines how a neuron processes incoming signals based on a specific predetermined function. The architecture of a neural network refers to the way neurons are connected, as illustrated in Fig. 1. For example, in fully connected neural networks, neurons are arranged in layers, with the outputs of one layer being fed to the subsequent layer as inputs. This compositional structure allows neural networks with many layers to become highly complex and to be able to learn hierarchical structures that enhance the ability of neural networks to generalize well to new data.

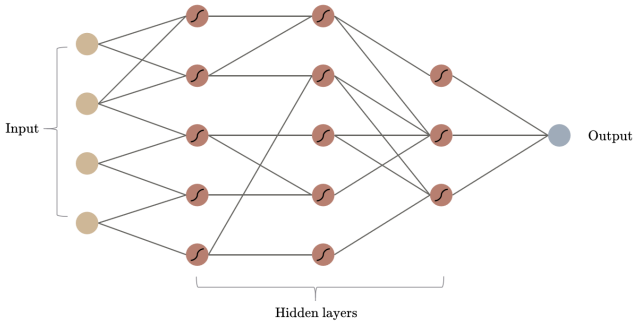

Fig. 1. Visualization of a neural network featuring a 4-dimensional input layer, 3 hidden layers, and a 1-dimensional output layer.

2.1 Activation functions

For decades, sigmoidal functions have been among the most popular activation functions, with their universality well studied in the literature [23, 9, 48, 86]. One of the most well-known sigmoidal functions is the Sigmoid function, while another commonly used activation function is Tanh, which has a similar shape but is symmetric around the origin, defined as

$$\mathrm{Sigmoid}(x) = \frac{1}{1+e^{-x}}, \quad \mathrm{Tanh}(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}.$$

These functions are closely related up to a transformation

$$\mathrm{Tanh}(x) = 2 \times \mathrm{Sigmoid}(2x) - 1.$$

However, both Sigmoid and Tanh suffer from the vanishing gradient problem and lead to a low training speed [13, 10, 46, 47, 12].

The ReLU activation is one of the simplest and most efficient activation functions, defined as

$$\mathrm{ReLU}(x) = \max(x, 0).$$

Due to its simplicity, it is now widely popular in deep learning. However, the main drawback of ReLU is that its negative part has a gradient zero, leading to dead neurons during training. To address these limitations, numerous activation functions have been proposed, including LeakyReLU [67], Softplus [36], Exponential Linear Unit (ELU) [26], Scaled Exponential Linear Unit (SELU) [52], Swish [87], and Mish [75], aiming at enhancing the performance of neural networks. Notably, these activation functions share a similar shape to ReLU, demonstrating the significance of ReLU-type activation functions in neural networks. Figure 2 illustrates the graphs of Sigmoid, Tanh, and ReLU, while the survey [35] provides a detailed summary and experimental comparison of the most popular activation functions across different network architectures and datasets.

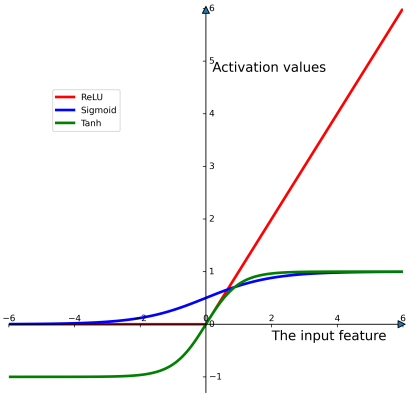

Fig. 2. Visualization of ReLU, Sigmoid, and Tanh.

2.2 Fully connected neural networks

Neural networks are structured through affine transformations. The first network architecture that we will introduce is the fully connected neural network, which has connections between all neurons across layers.

Let $\mathbb{R}$ represent the set of all real numbers, $\mathbb{N}$ represent the set of natural numbers, and $\mathbb{N}_+$ represent the set of non-zero natural numbers. We use boldface lowercase letters to denote vectors, such as $\boldsymbol{x} := (x_1, x_2, \ldots, x_d)^\top \in \mathbb{R}^d$. Additionally, for any $N \in \mathbb{N}_+$, we define $[N] := \{1, \ldots, N\}$ and $[N]_0 := \{0, 1, \ldots, N\}$. In the following, we will use $\sigma : \mathbb{R} \to \mathbb{R}$ to denote the activation function, which acts component-wise on vectors, matrices, and tensors.

Definition 1 (Fully connected neural networks). *A fully connected neural network ϕ with L layers is defined iteratively by*

$$\begin{aligned}
\phi^{(1)}(\boldsymbol{x}) &= \sigma\left(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right), \\
\phi^{(\ell)}(\boldsymbol{x}) &= \sigma\left(\boldsymbol{W}^{(\ell)}\phi^{(\ell-1)}(\boldsymbol{x}) + \boldsymbol{b}^{(\ell)}\right), \\
\phi(\boldsymbol{x}) &= \boldsymbol{W}^{(L)}\phi^{(L-1)}(\boldsymbol{x}) + \boldsymbol{b}^{(L)},
\end{aligned}$$

for some weight matrices $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and bias vectors $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{N_\ell}$.

In the mathematical analysis of neural networks, we typically focus on the maximum width $W := \max\{N_1, N_2, \ldots, N_{L-1}\}$ of a network instead of the number of neurons in each layer. The total number of neurons in a neural network ϕ is denoted as $N := \sum_{\ell=1}^{L-1} N_\ell$. In approximation theory, the depth L, width W, and total number of neurons N are the most frequently used quantities to characterize the approximation ability of neural networks.

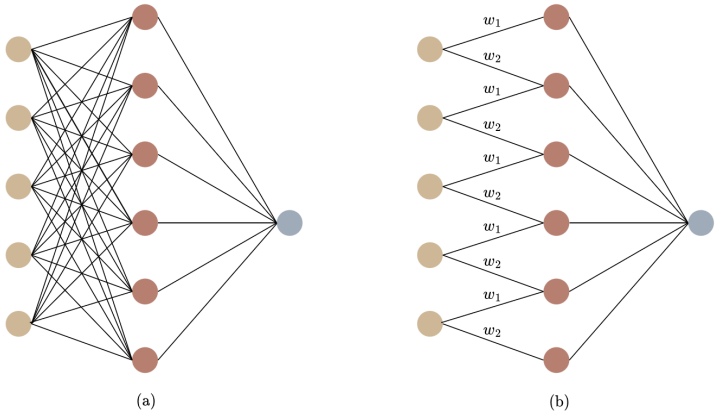

Fig. 3. Visualization of (**a**) a fully connected neural network and (**b**) a convolutional neural network. Both of them have a single hidden layer.

Example 1. Let $\phi : \mathbb{R}^3 \to \mathbb{R}^2$ be a neural network with a depth of $L = 3$ layers, a width of at most $W = 3$ neurons per layer, and a total of $N = 5$ neurons across all layers, along with the following weight matrices

$$\boldsymbol{W}^{(1)} = \begin{bmatrix} W_{11}^{(1)} & 0 & 0 \\ W_{21}^{(1)} & 0 & 0 \\ 0 & W_{32}^{(1)} & W_{33}^{(1)} \end{bmatrix}, \quad \boldsymbol{W}^{(2)} = \begin{bmatrix} W_{11}^{(2)} & 0 & W_{13}^{(2)} \\ 0 & W_{22}^{(2)} & 0 \end{bmatrix},$$
$$\boldsymbol{W}^{(3)} = \begin{bmatrix} W_{11}^{(3)} & 0 \\ W_{21}^{(3)} & W_{22}^{(3)} \end{bmatrix},$$

and bias vectors $\boldsymbol{b}^{(1)}$, $\boldsymbol{b}^{(2)}$, and $\boldsymbol{b}^{(3)}$. The neural network expression is then given by

$$\phi(\boldsymbol{x}) = \boldsymbol{W}^{(3)}\sigma\left(\boldsymbol{W}^{(2)}\sigma\left(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) + \boldsymbol{b}^{(2)}\right) + \boldsymbol{b}^{(3)},$$

where σ denotes the activation function. Figure 4 provides an illustration of this neural network. When there are no connections between neurons, the corresponding entries in weight matrices are set to zero. The specific arrangements of zero elements determine the structure of fully connected neural networks and contribute to different properties.

2.3 Convolutional neural networks

Convolutional neural networks are among the most popular and efficient architectures and can be considered as a specialized version of fully connected neural networks. Let $\boldsymbol{w} = (w_k)_{k=-\infty}^{\infty}$ be a convolutional filter mask and assume that it

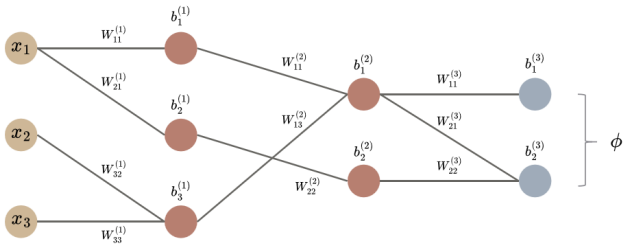

Fig. 4. Visualization of the neural network in Example 1.

has support on $[K]$, that is, $w_k = 0$ for any $k \notin [K]$, where K is the so-called kernel size. The 1D convolution is computed by $(\boldsymbol{w} * \boldsymbol{x})_i = \sum_{k=1}^{d} w_{i-k+1} x_k$, which is associated with a Toeplitz type matrix $\boldsymbol{T}^{\boldsymbol{w}} \in \mathbb{R}^{(d+K-1)\times d}$:

$$\boldsymbol{T}^{\boldsymbol{w}} := \begin{bmatrix} w_1 & 0 & 0 & \cdots & \cdots & 0 \\ w_2 & w_1 & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \cdots & 0 \\ w_K & w_{K-1} & \cdots & w_1 & \cdots & \vdots \\ 0 & w_K & w_{K-1} & \cdots & w_1 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \cdots & \vdots \\ 0 & \cdots & \ddots & w_K & w_{K-1} & w_{K-2} \\ 0 & \cdots & \ddots & \ddots & w_K & w_{K-1} \\ 0 & \cdots & \cdots & \cdots & 0 & w_K \end{bmatrix}. \tag{1}$$

We can see that the Toeplitz matrix is very sparse, with weights shared between neurons, leading to substantially fewer parameters than the corresponding dense matrix used in fully connected neural networks. For example, when the convolutional filter mask has support $\{1, 2\}$, then the corresponding Toeplitz type matrix is given by

$$\boldsymbol{T}^{\boldsymbol{w}} := \begin{bmatrix} w_1 & 0 & 0 & 0 & 0 \\ w_2 & w_1 & 0 & 0 & 0 \\ 0 & w_2 & w_1 & 0 & 0 \\ 0 & 0 & w_2 & w_1 & 0 \\ 0 & 0 & 0 & w_2 & w_1 \\ 0 & 0 & 0 & 0 & w_2 \end{bmatrix},$$

which depends solely on w_1, w_2. If we use $\boldsymbol{T}^{\boldsymbol{w}}$ instead of a full matrix for a neural network with only a single hidden layer, then the connectivity (the number of nonzero elements in the weight matrices) of convolutional neural networks is decreased to 10, governed by just 2 free parameters, in contrast to $5 \times 6 = 30$ parameters for a fully connected neural network. For further visualization, please refer to Fig. 3 (b).

Definition 2 (Convolutional neural networks (CNNs)). *A deep convolutional neural network ϕ is defined iteratively by*

$$\begin{aligned}\phi^{(1)}(\boldsymbol{x}) &= \sigma\left(\boldsymbol{T}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right),\\ \phi^{(\ell)}(\boldsymbol{x}) &= \sigma\left(\boldsymbol{T}^{(\ell)}\phi^{(\ell-1)}(\boldsymbol{x}) + \boldsymbol{b}^{(\ell)}\right), \quad \ell \in [L-1],\\ \phi(\boldsymbol{x}) &= \sigma\left(\boldsymbol{T}^{(L)}\phi^{(L-1)}(\boldsymbol{x}) + \boldsymbol{b}^{(L)}\right),\end{aligned}$$

where each $\boldsymbol{T}^{(\ell)} := \boldsymbol{T}^{\boldsymbol{w}^{(\ell)}}$ is a Toeplize type matrix associated with a filter mask $\boldsymbol{w}^{(\ell)}$ and $\boldsymbol{b}^{(\ell)}$ is a bias vector.

In deep learning, the above definition of convolution is commonly referred to as the convolution with zero padding. Let us denote $\tilde{\boldsymbol{x}}$ as the zero-padded version of $\boldsymbol{x}$, where $K-1$ zero elements are added on both sides, i.e.,

$$\tilde{\boldsymbol{x}} = (\underbrace{0,\ldots,0}_{K-1}, \boldsymbol{x}^\top, \underbrace{0,\ldots,0}_{K-1})^\top,$$

We also define $\tilde{\boldsymbol{w}}$ as the flipped version of the convolutional filter mask $\boldsymbol{w}$, i.e., $\tilde{\boldsymbol{w}} := (w_K, w_{K-1}, \ldots, w_1)^T$. Then the convolution characterized in (1) can be represented as

$$(\boldsymbol{w} * \boldsymbol{x})_i = \sum_{k=1}^{K} \tilde{w}_k \tilde{x}_{i+k-1}. \tag{2}$$

This formula describes the commonly used notation for convolution with zero padding and a stride of one in deep learning implementations. In particular, it will be useful for our later analysis. For a matrix input $\boldsymbol{X} \in \mathbb{R}^{d\times d}$ and a convolutional kernel $\boldsymbol{W} \in \mathbb{R}^{K\times K}$, the 2D convolution at position (i,j) is computed as

$$(\boldsymbol{W} * \boldsymbol{X})_{i,j} = \sum_{m,n=1}^{K} W_{mn}\tilde{X}_{i+m-1,j+n-1},$$

where $\tilde{\boldsymbol{X}}$ is the zero-padded version of $\boldsymbol{X}$. Zero-padding helps adjust the output dimensions of convolutional layers, which is frequently used in practical programming.

To extend 2D convolutions for processing, for example, color images, which have three or more channels, we need to consider higher-dimensional tensors. Given a tensor $\boldsymbol{X} \in \mathbb{R}^{d\times d\times c_{\text{in}}}$ with c_{in} input channels and a collection of kernels $\{\boldsymbol{W}_c \in \mathbb{R}^{K\times K\times c_{\text{in}}}\}_{c=1}^{c_{\text{out}}}$, the convolution between $\boldsymbol{X}$ and each kernel $\boldsymbol{W}_c$ involves applying a 2D convolution to each channel and summing the results.

$$(\boldsymbol{W}_c * \boldsymbol{X})_{i,j} = \sum_{s=1}^{c_{\text{in}}} \sum_{m,n=1}^{K} (\boldsymbol{W}_c)_{m,n,s}\tilde{X}_{i+m-1,j+n-1,s}.$$

The multi-channel convolution $\boldsymbol{W} * \boldsymbol{X} : \mathbb{R}^{d\times d\times c_{\text{in}}} \to \mathbb{R}^{d\times d\times c_{\text{out}}}$ is then defined as

$$(\boldsymbol{W} * \boldsymbol{X})_{i,j,c} = (\boldsymbol{W}_c * \boldsymbol{X})_{i,j}, \quad c = 1, \dots, c_{\text{out}},$$

where $\boldsymbol{W} \in \mathbb{R}^{K\times K\times c_{\text{in}}\times c_{\text{out}}}$ represents the concatenation of kernels $\boldsymbol{W}_c$ along the output channel dimension, i.e., $\boldsymbol{W}_{:,:,:,c} = \boldsymbol{W}_c$.

Here and in the following, we use the notation $\boldsymbol{A}_{:,i}$ to represent the i-th column of matrix $\boldsymbol{A}$ and $\boldsymbol{A}_{j,:}$ to represent the j-th row of matrix $\boldsymbol{A}$. For a higher-dimensional tensor $\boldsymbol{A}$, we extend the notation: $\boldsymbol{A}_{i,j,:}$ denotes the vector slice corresponding to the first two dimension indices i, j. Similarly, $\boldsymbol{A}_{i,j,k,:}$ denotes the vector slice corresponding to the indices (i, j, k) along the first three dimensions (assuming a four-dimensional tensor). We treat all the aforementioned vectors as column vectors for convenience.

Based on the above definition and notations, we can rewrite the multi-channel convolution in a more compact form by

$$\begin{aligned}
(\boldsymbol{W} * \boldsymbol{X})_{i,j,:}^{\top} &= \left[\sum_{m,n=1}^{K}\sum_{s=1}^{c_{\text{in}}} (\boldsymbol{W}_c)_{m,n,s}\tilde{X}_{i+m-1,j+n-1,s}\right]_{c=1}^{c_{\text{out}}} \\
&= \left[\sum_{m,n=1}^{K}\sum_{s=1}^{c_{\text{in}}} (\boldsymbol{W})_{m,n,s,c}\tilde{X}_{i+m-1,j+n-1,s}\right]_{c=1}^{c_{\text{out}}} \\
&= \left[\sum_{m,n=1}^{K} \left\langle (\boldsymbol{W})_{m,n,:,c}, \tilde{\boldsymbol{X}}_{i+m-1,j+n-1,:}\right\rangle\right]_{c=1}^{c_{\text{out}}} \\
&= \sum_{m,n=1}^{K} \tilde{\boldsymbol{X}}_{i+m-1,j+n-1,:}^{\top}(\boldsymbol{W})_{m,n,:,:}.
\end{aligned} \tag{3}$$

This unveils that multi-channel convolution is equivalent to applying linear transformations channel-wise. All of the aforementioned convolutions utilize a stride of 1. Similarly, we can extend the formulation presented in, for instance, (2), to accommodate convolutions with a stride of s by

$$(\boldsymbol{w} *_s \boldsymbol{x})_i = \sum_{k=1}^{K} \tilde{w}_k \tilde{x}_{(i-1)s+k},$$

with zero-padding adjusted accordingly.

2.4 Self-attention and Transformer

A more powerful network architecture is the attention mechanism, which originally mimics human behavior by focusing on parts of information rather than processing the whole [81]. Among the various attention mechanisms, we shall introduce the self-attention mechanism, one of the most important building blocks

of Transformer and Vision Transformer (ViT), which have achieved great success in many applications like natural language processing and computer vision [106, 34]. In self-attention, the core components are the query, key, and value. The query represents the current element and interacts with the keys to assess similarity scores. The key acts as a sequence of identifiers that align with the query to calculate these similarity scores. The value provides contextual data about the elements and, when combined with similarity scores, produces the final output.

To define self-attention, let us introduce the matrices $\boldsymbol{W}_{\mathrm{qry}} \in \mathbb{R}^{c_{\mathrm{in}} \times t}$, $\boldsymbol{W}_{\mathrm{key}} \in \mathbb{R}^{c_{\mathrm{in}} \times t}$, $\boldsymbol{W}_{\mathrm{val}} \in \mathbb{R}^{c_{\mathrm{in}} \times c_{\mathrm{out}}}$. Attention score $\boldsymbol{A} \in \mathbb{R}^{d \times d \times d \times d}$ is calculated as

$$A_{(i,j),(q,k)} = \left\langle \boldsymbol{X}_{i,j,:}^{\top} \boldsymbol{W}_{\mathrm{qry}} \,,\ \boldsymbol{X}_{q,k,:}^{\top} \boldsymbol{W}_{\mathrm{key}} \right\rangle, \tag{4}$$

and the attention probability $\boldsymbol{P} \in \mathbb{R}^{d \times d \times d \times d}$ is obtained by the normalization

$$P_{(i,j),(q,k)} = \exp(A_{(i,j),(q,k)}) / \sum_{q,k=1}^{d} \exp(A_{(i,j),(q,k)}),$$

Attention probability is used to measure the similarity between features at the locations (i, j) and (q, k). The output of the self attention $\mathcal{SA} : \mathbb{R}^{d \times d \times c_{\mathrm{in}}} \to \mathbb{R}^{d \times d \times c_{\mathrm{out}}}$ is given by

$$(\mathcal{SA}(\boldsymbol{X})_{i,j,:})^{\top} = \sum_{q,k=1}^{d} P_{(i,j),(q,k)} \boldsymbol{X}_{q,k,:}^{\top} \boldsymbol{W}_{\mathrm{val}}.$$

Multi-head self-attention further generalizes this concept by concatenating several self-attentions with additional projection matrices $\boldsymbol{W}_h$

$$(\mathcal{SA}_{\mathrm{mul}}(\boldsymbol{X})_{i,j,:})^{\top} = \sum_{h=1}^{N_h} \sum_{q,k=1}^{d} P_{(i,j),(q,k)} \boldsymbol{X}_{q,k,:}^{\top} \boldsymbol{W}_{\mathrm{val}}^{(h)} \boldsymbol{W}_h. \tag{5}$$

From expressions, we can see that multi-head self-attention (5) combines features $\boldsymbol{X}_{q,k,:}$, $(q, k) \in [d]^2$ according to similarities and when $P_{(i,j),(q,k)}$ approaches $\delta_{(i,j),(q,k)}$, multi-head self attentions get close to multi-channel convolutions (3), and it is the main idea for the discussion on the universality of the self-attention and Transformer [28]. Figure 5 (a) provides an illustration of the multi-head self-attention.

To formally define Transformers, we present the following mappings:

$$T_{d,c} : \mathbb{R}^{cd^2} \to \mathbb{R}^{d \times d \times c}, \quad M_{d,c} : \mathbb{R}^{d \times d \times c} \to \mathbb{R}^{cd^2},$$

which satisfy

$$M_{d,c} \circ T_{d,c}(\mathbb{R}^{cd^2}) = \mathbb{R}^{cd^2}, \quad T_{d,c} \circ M_{d,c}(\mathbb{R}^{d \times d \times c}) = \mathbb{R}^{d \times d \times c}.$$

In this context, the reshape mapping $T_{d,c}$ reconfigures a vector of dimension cd^2 into a tensor of dimension $d \times d \times c$, while the flatten mapping $M_{d,c}$ reduces a tensor of dimension $d \times d \times c$ into a vector of dimension cd^2. Consequently, the Transformer model can be formulated as follows.

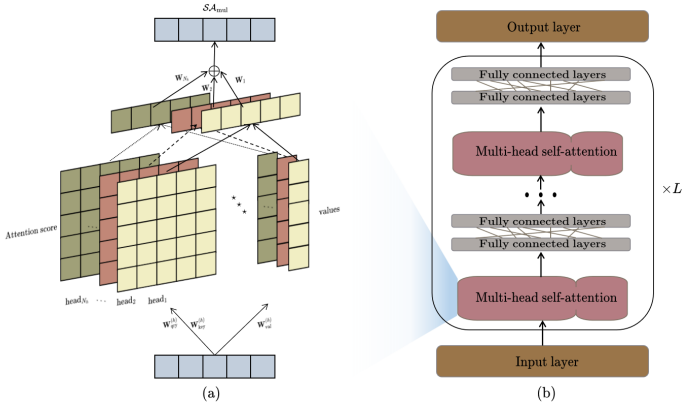

Fig. 5. Visualization of (**a**) a multi-head self-attention and (**b**) a Transformer.

Definition 3 (Transformers). *Let* $d, L \in \mathbb{N}$, $\{c_\ell\}_{\ell=0}^{L} \subset \mathbb{N}$, *and* $\boldsymbol{X} \in \mathbb{R}^{d\times d\times c_0}$. *Let*

$$\{\psi^\ell : \mathbb{R}^{c_{\ell-1}d^2} \to \mathbb{R}^{c_\ell d^2}\}_{\ell=1}^{L},$$

be a sequence of fully connected neural networks, and

$$\{\mathcal{SA}_{mul}^{(\ell)} : \mathbb{R}^{d\times d\times c_{\ell-1}} \to \mathbb{R}^{d\times d\times c_{\ell-1}}\}_{\ell=1}^{L},$$

be a sequence of multi-head self-attention operations. Then a Transformer model ϕ *is defined iteratively by*

$$\begin{aligned}
\phi^{(1)}(\boldsymbol{X}) &= T_{d,c_1} \circ \psi^{(1)} \circ M_{d,c_0} \circ \mathcal{SA}_{mul}^{(1)}(\boldsymbol{X}),\\
\phi^{(\ell)}(\boldsymbol{X}) &= T_{d,c_\ell} \circ \psi^{(\ell)} \circ M_{d,c_{\ell-1}} \circ \mathcal{SA}_{mul}^{(\ell)}\left(\phi^{(\ell-1)}(\boldsymbol{X})\right), \quad \ell \in [L-1],\\
\phi(\boldsymbol{X}) &= T_{d,c_L} \circ \psi^{(L)} \circ M_{d,c_{L-1}} \circ \mathcal{SA}_{mul}^{(L)}\left(\phi^{(L-1)}(\boldsymbol{X})\right).
\end{aligned}$$

Figure 5 (b) provides an illustration of the Transformer. The authors of [106] originally implemented position-wise feed-forward networks in their Transformer architecture instead of fully connected layers. It is noteworthy that Definition 3 aligns with [106] if the weight matrices in $\psi^{(\ell)}$ take the form of block diagonal matrices under particular conditions as described in T_{d,c_ℓ} and M_{d,c_ℓ}. Moreover, the first two dimensions of input $\boldsymbol{X}$ as well as the output dimension of multi-head self-attention can be adapted to a broader context.

2.5 Application of approximation to generalization

In machine learning, our goal is to learn a proper model $\phi(\boldsymbol{x};\boldsymbol{w})$ that approximates an unknown target function $f(\boldsymbol{x})$ over a dataset $\{(\boldsymbol{x}_i, f(\boldsymbol{x}_i)) \in \mathbb{R}^d \times \mathbb{R}\}_{i=1}^{n}$ [63, 114]. The model is parameterized by a weight $\boldsymbol{w}$ and for some learning algorithms we usually only consider selecting it in a parameter space Θ, e.g.,

$\Theta = \{\boldsymbol{w} : \|\boldsymbol{w}\|_\infty \leq B\}$ for some constant $B > 0$. Given a loss function $\mathrm{Loss}(\cdot, \cdot)$, we estimate $\boldsymbol{w}$ by minimizing the empirical risk:

$$\mathcal{E}_n(\boldsymbol{w}) := \frac{1}{n}\sum_{i=1}^{n} \mathrm{Loss}\left(\phi(\boldsymbol{x}_i; \boldsymbol{w}), f(\boldsymbol{x}_i)\right),$$

and we expect that the weight $\boldsymbol{w}$ can be close enough to the minimizer $\hat{\boldsymbol{w}}$, given by

$$\hat{\boldsymbol{w}} := \arg\min_{\boldsymbol{w}\in\Theta} \mathcal{E}_n(\boldsymbol{w}).$$

Assuming that the data points $\boldsymbol{x}_i$ are independently and identically distributed (i.i.d.) samples from a probability measure μ, we want our model to generalize well to new data. To do so, our aim becomes to minimize the expected loss

$$\mathcal{E}(\boldsymbol{w}) := \mathbb{E}_\mu\left(\mathrm{Loss}\left(\phi(\boldsymbol{x}; \boldsymbol{w}), f(\boldsymbol{x})\right)\right).$$

Next, we denote $\boldsymbol{w}^*$ as its minimizer, i.e.,

$$\boldsymbol{w}^* := \arg\min_{\boldsymbol{w}\in\Theta} \mathcal{E}(\boldsymbol{w}).$$

To estimate the performance of our learned model $\phi(\boldsymbol{x}; \boldsymbol{w})$, we decompose the expected loss of parameter $\boldsymbol{w}$ into three parts [63, 114]

$$\begin{aligned}
&\mathcal{E}(\boldsymbol{w}) \\
&= \mathcal{E}(\boldsymbol{w}) - \mathcal{E}_n(\boldsymbol{w}) + \mathcal{E}_n(\boldsymbol{w}) - \mathcal{E}_n(\hat{\boldsymbol{w}}) + \underbrace{(\mathcal{E}_n(\hat{\boldsymbol{w}}) - \mathcal{E}_n(\boldsymbol{w}^*))}_{\leq 0} + \mathcal{E}_n(\boldsymbol{w}^*) - \mathcal{E}(\boldsymbol{w}^*) + \mathcal{E}(\boldsymbol{w}^*) \\
&\leq \underbrace{\mathcal{E}(\boldsymbol{w}^*)}_{\text{(I)}} + \underbrace{2\sup_{\boldsymbol{w}\in\Theta} |\mathcal{E}(\boldsymbol{w}) - \mathcal{E}_n(\boldsymbol{w})|}_{\text{(II)}} + \underbrace{\mathcal{E}_n(\boldsymbol{w}) - \mathcal{E}_n(\hat{\boldsymbol{w}})}_{\text{(III)}}.
\end{aligned}$$

Here, the first term (I) represents the approximation error, which is the best possible performance of our model. The second term (II) accounts for the variance due to the finite sample size. The last term (III) represents the optimization error, which is the difference between the empirical risk of our estimated weights and the minimum possible empirical risk. The first term (I), which constitutes part of the upper bound, highlights the importance of studying the approximation properties of deep neural networks [63, 114, 14]. For a more detailed discussion, the reader is encouraged to refer to [85].

2.6 Construction properties of neural networks

Estimating approximation error bounds of deep ReLU neural networks involves progressing from simple to complex functions. It is similar to constructing a structure using basic blocks, ultimately achieving a more intricate and diverse function space. The simplicity and near-linearity of the ReLU activation function

enable three efficient operations: composition, summation, and concatenation. These operations simplify our discussions in the later sections. In the statement of the following theorem and proof, for simplicity, we abuse the notation $(\boldsymbol{v}_1, \boldsymbol{v}_2)$ to represent the vertical concatenation (or stacking) of any two column vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, which is formally equivalent to $(\boldsymbol{v}_1^\top, \boldsymbol{v}_2^\top)^\top$.

Lemma 1 ([84, 42]). *Given two ReLU neural networks $\phi_1 : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1}$ and $\phi_2 : \mathbb{R}^{d_2} \to \mathbb{R}^{d_3}$. Assume that ϕ_1 is characterized by (L_1, W_1, N_1) and ϕ_2 is characterized by (L_2, W_2, N_2), where L_i is the number of layers, W_i is the width, and N_i is the total number of neurons. Then their operations:*

(a) composition $\phi_2 \circ \phi_1$ (when $d_1 = d_2$),
(b) concatenation (ϕ_1, ϕ_2),
(c) summation $\phi_1 + \phi_2$ (when $d_1 = d_3$),

can be realized by a single ReLU neural network. More specifically, the resulting ReLU neural networks are characterized by the following parameters

(a) $(L_1 + L_2 - 1, \max\{W_1, W_2\}, N_1 + N_2)$,
(b) $(\max\{L_1, L_2\}, 2(W_1 + W_2), 2(N_1 + N_2))$,
(c) $(\max\{L_1, L_2\}, 2(W_1 + W_2), 2(N_1 + N_2))$.

Proof. (a). Let $\{(\boldsymbol{W}_i^{(\ell)}, \boldsymbol{b}_i^{(\ell)})\}_{\ell=1}^{L_i}$ denote the weight matrices and bias vectors of ϕ_i. The composition $\phi_2 \circ \phi_1$ can be realized by a neural network ϕ defined as follows:

$$\begin{aligned}
\phi^{(\ell)}(\boldsymbol{x}) &:= \phi_1^{(\ell)}(\boldsymbol{x}), \quad \ell = 1, \dots, L_1 - 1, \\
\phi^{(L_1)}(\boldsymbol{x}) &:= \sigma\left(\boldsymbol{W}_2^{(1)} \boldsymbol{W}_1^{(L_1)} \phi^{(L_1-1)}(\boldsymbol{x}) + \boldsymbol{W}_2^{(1)} \boldsymbol{b}_1^{(L_1)} + \boldsymbol{b}_2^{(1)}\right), \\
\phi^{(\ell)}(\boldsymbol{x}) &:= \phi_2^{(\ell - L_1 + 1)}(\boldsymbol{x}), \quad \ell = L_1 + 1, \dots, L_1 + L_2 - 1.
\end{aligned}$$

In the second step, we use the result of the composition of $\phi_2^{(1)}$ and $\phi_1^{(L_1)}$

$$\begin{aligned}
\phi_2^{(1)} \circ \phi_1^{(L_1)}(\boldsymbol{x}) &= \sigma\left(\boldsymbol{W}_2^{(1)} \phi_1^{(L_1)}(\boldsymbol{x}) + \boldsymbol{b}_2^{(1)}\right) \\
&= \sigma\left(\boldsymbol{W}_2^{(1)} \boldsymbol{W}_1^{(L_1)} \phi_1^{(L_1-1)}(\boldsymbol{x}) + \boldsymbol{W}_2^{(1)} \boldsymbol{b}_1^{(L_1)} + \boldsymbol{b}_2^{(1)}\right).
\end{aligned}$$

This neural network has $L_1 + L_2 - 1$ layers, width $\max\{W_1, W_2\}$, and no more than $N_1 + N_2$ neurons.

(b). For concatenation (ϕ_1, ϕ_2), assume without loss of generality that $L_1 > L_2$. Let $\boldsymbol{x} \in \mathbb{R}^{d_0}$ and $\boldsymbol{y} \in \mathbb{R}^{d_2}$. Define a neural network ϕ with input dimension $d_0 + d_2$ as follows:

$$\boldsymbol{W}^{(\ell)} := \begin{bmatrix} \boldsymbol{W}_1^{(\ell)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}_2^{(\ell)} \end{bmatrix}, \quad \boldsymbol{b}^{(\ell)} := \begin{bmatrix} \boldsymbol{b}_1^{(\ell)} \\ \boldsymbol{b}_2^{(\ell)} \end{bmatrix},$$

for $\ell = 1, \ldots, L_2 - 1$,

$$\boldsymbol{W}^{(L_2)} := \begin{bmatrix} \boldsymbol{W}_1^{(L_2)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}_2^{(L_2)} \\ \boldsymbol{0} & -\boldsymbol{W}_2^{(L_2)} \end{bmatrix}, \quad \boldsymbol{b}^{(L_2)} := \begin{bmatrix} \boldsymbol{b}_1^{(L_2)} \\ \boldsymbol{b}_2^{(L_2)} \\ -\boldsymbol{b}_2^{(L_2)} \end{bmatrix},$$

for $\ell = L_2$,

$$\boldsymbol{W}^{(\ell)} := \begin{bmatrix} \boldsymbol{W}_1^{(\ell)} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}, \quad \boldsymbol{b}^{(\ell)} := \begin{bmatrix} \boldsymbol{b}_1^{(\ell)} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix},$$

for $\ell = L_2 + 1, \ldots, L_1 - 1$, and for the last layer, we define

$$\boldsymbol{W}^{(L_1)} := \begin{bmatrix} \boldsymbol{W}_1^{(L_1)} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & -\boldsymbol{I} \end{bmatrix}, \quad \boldsymbol{b}^{(L_1)} := \begin{bmatrix} \boldsymbol{b}_1^{(L_1)} \\ \boldsymbol{0} \end{bmatrix}.$$

Here $\boldsymbol{I}$ and $\boldsymbol{0}$ denote the identity matrix and zero matrix of appropriate dimensions. Since $\boldsymbol{W}^{(\ell)}$ is a block diagonal matrix for $\ell = 1, \ldots, L_2 - 1$, it is obvious that we have $\phi^{(\ell)} = (\phi_1^{(\ell)}, \phi_2^{(\ell)})$. When $\ell = L_2$, we split ϕ_2 into positive and negative parts and get

$$\phi^{(L_2)}(\boldsymbol{x}, \boldsymbol{y}) = \begin{bmatrix} \phi_1^{(L_2)}(\boldsymbol{x}) \\ \sigma\left(\phi_2^{(L_2)}(\boldsymbol{y})\right) \\ \sigma\left(-\phi_2^{(L_2)}(\boldsymbol{y})\right) \end{bmatrix}, \quad \ell = 1, \ldots, L_2.$$

Notice that $\sigma \circ \sigma(c) = \sigma(c)$ for any $c \in \mathbb{R}$. Hence, for $\ell = L_2 + 1, \ldots, L_1 - 1$, the first block of $\phi^{(\ell)}$ is equal to $\phi_1^{(\ell)}$ while the second and third blocks are kept the same, leading to

$$\phi^{(\ell)}(\boldsymbol{x}, \boldsymbol{y}) = \begin{bmatrix} \phi_1^{(\ell)}(\boldsymbol{x}) \\ \sigma\left(\phi_2^{(\ell)}(\boldsymbol{y})\right) \\ \sigma\left(-\phi_2^{(\ell)}(\boldsymbol{y})\right) \end{bmatrix}, \quad \ell = L_2 + 1, \ldots, L_1 - 1.$$

In the last layer, using the fact that $\sigma(c) - \sigma(-c) = c$ for any $c \in \mathbb{R}$, we obtain that $\phi = (\phi_1, \phi_2)$. According to the definition of weight matrices, width of ϕ is upper-bounded by $\max\{W_1 + W_2, W_1 + 2W_2\} \leq 2(W_1 + W_2)$ and similarly number of neurons is bounded by $2(N_1 + N_2)$. When $\boldsymbol{x} = \boldsymbol{y}$, we only need to revise the first layer with the following parameters

$$\boldsymbol{W}^{(1)} := \begin{bmatrix} \boldsymbol{W}_1^{(1)} \\ \boldsymbol{W}_2^{(1)} \end{bmatrix}, \quad \boldsymbol{b}^{(1)} := \begin{bmatrix} \boldsymbol{b}_1^{(1)} \\ \boldsymbol{b}_2^{(1)} \end{bmatrix},$$

and finally can get $\phi(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}))$.

(c). For summation $\phi_1 + \phi_2$, we follow similar steps as in (b) but adjust the final layer to combine the outputs:

$$\boldsymbol{W}^{(L_1)} := \left[\boldsymbol{W}_1^{(L_1)}\ \boldsymbol{I}\ -\boldsymbol{I}\right], \quad \boldsymbol{b}^{(L_1)} := \boldsymbol{b}_1^{(L_1)}, \text{if } L_1 > L_2.$$

where $L = \max\{L_1, L_2\}$. This network has $\max\{L_1, L_2\}$ layers, width $2(W_1 + W_2)$, and no more than $2(N_1 + N_2)$ neurons.

The proof of Lemma 1 highlights two crucial properties of ReLU neural networks, $\sigma \circ \sigma = \sigma$ and $\sigma(\cdot) - \sigma(-\cdot) = \sigma(\cdot)$. These properties facilitate the straightforward calculation of the structure of the resulting neural networks when applying these operations. Notably, the depth, width, and number of neurons of the resulting networks increase linearly. In subsequent sections, we will leverage this lemma frequently without further mentioning it.

3 Universality of neural networks

The universal approximation property serves as the theoretical foundation of the expressivity of neural networks, explaining their ability to approximate general functions within any error tolerance when the number of neurons increases. Understanding universality is important, as the target mapping in many real-world problems is highly complex. The richness of neural networks allows researchers and engineers to design a neural network for various tasks as long as sufficient neurons are employed.

In this section, we shall introduce the universality of two types of neural networks: those with bounded depth and those with bounded width. These results demonstrate that, provided a neural network is wide enough or deep enough, it would be able to approximate general functions to arbitrary accuracy.

3.1 Universality of shallow neural networks

We will first consider shallow neural networks, as shown in Fig. 3, which have only one hidden layer and are a linear combination of several neurons.

Definition 4 (Shallow neural networks). *A shallow neural network is a neural network with only one hidden layer (i.e., $L = 2$), defined as*

$$\phi(\boldsymbol{x}) = \sum_{i=1}^{N} \beta_i \sigma\left(\langle \boldsymbol{w}_i, \boldsymbol{x}\rangle + b_i\right), \quad \boldsymbol{x} \in \mathbb{R}^d, \tag{6}$$

for some $\boldsymbol{w}_i \in \mathbb{R}^d$ and $b_i, \beta_i \in \mathbb{R}$.

One of the remarkable universality results of shallow neural networks was developed in [55]. The density considered in this subsection is in the topology of uniform convergence on compact sets. Let $C^\infty(\mathbb{R})$ be the collection of infinitely differentiable functions. The subsequent theorem outlines the condition for an activation function $\sigma \in C^\infty(\mathbb{R})$ to guarantee the validity of the universal approximation theorem.

Theorem 1 ([55]). *If $\sigma \in C^\infty(\mathbb{R})$ is not a polynomial, then shallow neural networks are dense in $C(\mathbb{R}^d)$.*

Proof. It is easy to see that when σ is a polynomial, then the corresponding shallow neural networks are polynomials with the same degrees, which cannot be a universal approximator.

When σ is not a polynomial, we observe that linear functions can be arbitrarily well-approximated by shallow neural networks

$$\frac{\sigma\left((w+h)x+b\right)-\sigma\left(wx+b\right)}{h} \xrightarrow{h\to 0} x\cdot\sigma'(wx+b) \xrightarrow{w\to 0} x\cdot\sigma'(b),$$

where we use the definition of the derivative. Similarly, we can generalize this idea to polynomials. That is, any polynomials $x^k\cdot\sigma^{(k)}(b_k)$ where $\sigma^{(k)}$ denotes the k-th derivative of σ can be well approximated. Since $\sigma\in C^\infty$, one can always find a b_k such that $\sigma^{(k)}(b_k)\neq 0$ for any $k\in\mathbb{N}$. Hence, any polynomial can be arbitrarily well approximated by shallow neural networks. Using Stone–Weierstrass theorem, we conclude that shallow neural networks are dense in $C(\mathbb{R})$.

To extend the density result from $\mathbb{R}$ to $\mathbb{R}^d$, we can use the universality of ridge functions, which is well studied, e.g., in [22, 60, 107], stating that $\operatorname{span}\{g(\boldsymbol{w}\cdot\boldsymbol{x}) : \boldsymbol{w}\in\mathbb{R}^d, g\in C(\mathbb{R})\}$ is dense in $\mathbb{R}^d$. With the above result that $g\in C(\mathbb{R})$ can be well approximated by shallow neural networks, we conclude the universality of shallow neural networks in $C(\mathbb{R}^d)$.

Then we wonder whether we can extend the above result to more general activation functions like ReLU. We can utilize the properties of mollifications to achieve this goal. Subsequently, we consider activation functions σ that are locally essentially bounded on any compact subset of $\mathbb{R}$, i.e., $\sigma\in L^\infty_{\mathrm{loc}}(\mathbb{R})$. Additionally, we require that σ is continuous on some interval U_i which forms a finite partition of $\mathbb{R}$ (their measures are positive). We denote the set M as the collection of all activation functions that satisfy the above properties.

Let $C_0^\infty(\Omega)$ be the space of continuous functions that have compact support in $\Omega\subset\mathbb{R}$. The following theorem shows that the convolution of σ and $\omega\in C_0^\infty(\mathbb{R})$ is smooth.

Theorem 2 ([4, 25]). *Given $\sigma\in M$. Then $\sigma*\omega\in C^\infty(\mathbb{R})$ for any $\omega\in C_0^\infty(\mathbb{R})$.*

Theorem 2 tells us that we can always refine σ through convolutions, and any continuous activation functions like ReLU are included. However, there are still two crucial aspects we need to verify. Firstly, we have to determine whether the convolution $\sigma*\omega$ is not a polynomial. Secondly, we need to ensure that the convolution $\sigma*\omega$ can be approximated by shallow neural networks. If both are met, we can then conclude the universality of σ-activated shallow neural networks. Theorem 2 is a classical result in functional analysis, with complete proofs available in [4, 25].

Lemma 2 ([55]). *Let $\sigma\in M$. If $\sigma*\omega$ is a polynomial for all $\omega\in C_0^\infty(\mathbb{R})$, then there exists an $m\in\mathbb{N}_+$ such that $\sigma*\omega$ is a polynomial of degree at most m for all $\omega\in C_0^\infty(\mathbb{R})$.*

Proof. It is sufficient to prove the claim for any $\omega \in C_0^\infty([a,b])$ for some interval $[a,b]$. To see this, consider $\omega \in C_0^\infty(\mathbb{R})$ with a general support $[c,d]$. Then there exists a decomposition of $\omega = \sum_{i=1}^k \omega_i$ such that $\omega_i \in C_0^\infty([c_i,d_i])$ with supports satisfying $[c,d] \subset \cup_i [c_i,d_i]$ with $d_i = b + t_i$, $c_i = a + t_i$ for some $t_i \in \mathbb{R}$. Thus we obtain

$$\begin{aligned}\sum_{i=1}^k \sigma * \omega_i(x) &= \sum_{i=1}^k \int \sigma(x-y)\omega_i(y)dy \\ &= \sum_{i=1}^k \int \sigma(x-(y+t_i))\omega_i(y+t_i)dy \\ &= \sum_{i=1}^k [\sigma * \omega_i(\cdot + t_i)](x-t_i),\end{aligned}$$

where $\omega_i(\cdot + t_i)$ has the support $[a,b]$. This implies that there exists some universal constant m such that $\sigma * \omega$ is a polynomial with degree $\leq m$ for any $\omega \in C_0^\infty(\mathbb{R})$ if the statement degree$(\sigma * \omega) \leq m$ holds for any $\omega \in C_0^\infty([a,b])$.

Now let us show the result for $C_0^\infty([a,b])$. For this, define the metric

$$\varrho(\omega_1,\omega_2) := \sum_{n=0}^\infty 2^{-n} \frac{\|\omega_1 - \omega_2\|_n}{1 + \|\omega_1 - \omega_2\|_n},$$

where $\|\omega\|_n := \sum_{j=0}^n \sup_{x\in[a,b]} |\omega^{(j)}(x)|$. Then $C_0^\infty([a,b])$ becomes a complete metric space. Define $V_k = \{\omega \in C_0^\infty([a,b]) : \text{degree}(\sigma * \omega) \leq k\}$. According to the definition and assumption, V_k has the following properties

(a) V_k is a closed subspace,
(b) $V_k \subset V_{k+1}$,
(c) $\cup_{k=0}^\infty V_k = C_0^\infty([a,b])$.

By Baire's category theorem, there exists m such that there is an open set contained in V_m. Hence $V_m = C_0^\infty([a,b])$. The proof is completed.

The theory of distribution and convolution indicates that σ is a polynomial under the conditions of Lemma 2 [40, 25]. It implies that once σ is not a polynomial, then there always exists at least one $\omega \in C_0^\infty(\mathbb{R})$ such that $\sigma * \omega$ is not a polynomial.

Now, the last result to show the universality is to prove $\sigma * \omega$ can be well-approximated by shallow neural networks.

Lemma 3 ([55]). *Let $\sigma \in M$. If $\omega \in C_0^\infty(\mathbb{R})$, then*

$$\sigma * \omega \in \overline{span\ \{\sigma(wx+b) : w,b \in \mathbb{R}\}}.$$

Proof. Assume that the support of ω is within $[a,b]$. Let us define

$$y_i := a + i\Delta y_i, \quad \Delta y_i := \frac{b-a}{m}, \quad i = 0,\dots,m,$$

and $I_i := [y_i, y_{i+1}]$. Then we can decompose the error between $\sigma * \omega = \int \sigma(x-y)\omega(y)dy$ and a shallow neural network $\sum_{i=1}^m \sigma(x-y_i)\omega(y_i)\Delta y_i$ (which is the discretization of the integral) into two parts over each interval I_i, namely

$$\begin{aligned}
&\left|\int_{I_i} \sigma(x-y)\omega(y)dy - \sigma(x-y_i)\omega(y_i)\Delta y_i\right| \\
&\leq \left|\int_{I_i} \sigma(x-y)\omega(y)dy - \int_{I_i} \sigma(x-y_i)\omega(y)dy\right| + \left|\int_{I_i} \sigma(x-y_i)\omega(y)dy - \sigma(x-y_i)\omega(y_i)\Delta y_i\right| \\
&\leq \underbrace{\int_{I_i} |\sigma(x-y)-\sigma(x-y_i)|\,|\omega(y)|dy}_{\text{(I)}} + \underbrace{\int_{I_i} |\sigma(x-y_i)|\,|\omega(y)-\omega(y_i)|\,dy}_{\text{(II)}}.
\end{aligned}$$

Let us consider $x \in [-c, c]$ for some $c > 0$. Notice that σ is locally bounded and $\omega \in C_0^\infty(\mathbb{R})$. Then we can find some $C < \infty$ such that $\max_{|x|\leq b+c} |\sigma(x)| \leq C$ and $\max_{|x|\leq b} |\omega(x)| \leq C$. For any $\varepsilon > 0$, we choose a large enough m such that $|\omega(y) - \omega(y_i)| \leq \varepsilon$. Hence, the quantity (II) can be upper-bounded by

$$\text{(II)} \leq C\Delta y_i \varepsilon.$$

According to the definition of σ, there exist finitely many discontinuous points of σ over the interval $[a-c, b+c]$. Let us denote U as the collection of some small enough open intervals with each interval containing one discontinuous point of σ on $[a-c, b+c]$. The interval U can be chosen to have a measure arbitrarily small, e.g. $|U| \leq \varepsilon/m$. Then we have

$$\begin{aligned}
\text{(I)} &\leq \int_{I_i \cap U} |\sigma(x-y)-\sigma(x-y_i)|\,|\omega(y)|dy + \int_{I_i \cap U^c} |\sigma(x-y)-\sigma(x-y_i)|\,|\omega(y)|dy \\
&\leq 2C^2\varepsilon/m + C\Delta y_i \varepsilon,
\end{aligned}$$

where we use the fact that σ is uniformly continuous on $I_i \cap U^c$ (set m large enough) and the boundedness of σ and ω. Summing the above bound with respect to $i = 1, \ldots m$, we have

$$|\sigma * \omega(x) - \sum_{i=1}^m \sigma(x-y_i)\omega(y_i)\Delta y_i| \leq 2C(b-a)\varepsilon + 2C^2\varepsilon \leq C'\varepsilon,$$

where C' is independent of x and ε. The proof is completed

Combining the above result with Lemma 1 and Theorem 2, we derive the following well-known universality result.

Theorem 3 ([55]). *Let $\sigma \in M$. Then the family of shallow neural networks is dense in $C(\mathbb{R}^d)$ if and only if σ is not a polynomial.*

3.2 Universality of deep ReLU neural networks with bounded width

We now analyze the constrained-width regime to determine if deep neural networks have universal approximation capabilities in $L_1(\mathbb{R}^d)$.

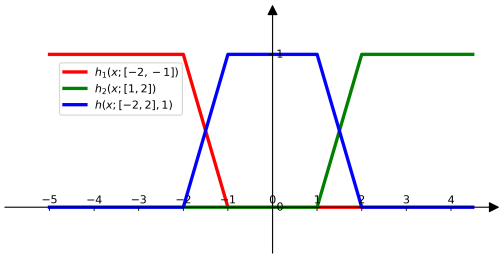

Fig. 6. Visualization of $h_1(x;[-2,-1])$ (red line), $h_2(x;[1,2])$ (green line), and $h(x;[-2,2],1)$ (blue line).

The idea is to utilize the ReLU activation function to approximate simple functions on cubes.

Intuitively, any characteristic function $\chi_{[a,b]}$ given by

$$\chi_{[a,b]}(x) := \begin{cases} 1, & x \in [a,b], \\ 0, & x \neq [a,b], \end{cases}$$

can be well approximated by hat functions. Let us construct the clipped version of ReLU with two neurons, as shown in Fig. 6, which is defined as

$$\begin{aligned} h_1(x;[a,a+\delta]) &= -\frac{1}{\delta}\left(\sigma(-x+a)-\sigma(-x+a+\delta)\right) \\ &= \begin{cases} 1, & x < a, \\ \text{linear}, & x \in [a,a+\delta], \\ 0, & x > a+\delta, \end{cases} \end{aligned}$$

and

$$\begin{aligned} h_2(x;[b-\delta,b]) &= \frac{1}{\delta}\left(\sigma(x-b+\delta)-\sigma(x-b)\right) \\ &= \begin{cases} 0, & x < b-\delta, \\ \text{linear}, & x \in [b-\delta,b], \\ 1, & x > b. \end{cases} \end{aligned}$$

When the linear part is not crucial for analysis, both here and subsequently, we will exclude its explicit expression. Now we can define the following "hat function"

$$\begin{aligned} h(x;[a,b],\delta) &:= 1-\left(h_1(x;[a,a+\delta])+h_2(x;[b-\delta,b])\right) \\ &= \begin{cases} 1, & x \in [a+\delta,b-\delta], \\ \text{linear}, & x \in (a,a+\delta)\cup(b-\delta,b), \\ 0, & x \notin [a,b]. \end{cases} \end{aligned} \tag{7}$$

In Fig. 6, we show an example of $h(x;[-2,2],1)$ which has support $[-2,2]$. We can see that compared with $\chi_{[-1,1]}$, the difference only appears at the linear part of h. It is similar when we consider an error analysis between a general $\chi_{[a,b]}$ and $h(x;[a,b],\delta)$ and difference only over the interval $(a,a+\delta)\cup(b-\delta,b)$ matters. We have $\left\|h(x;[a,b],\delta)-\chi_{[a,b]}\right\|_{L_1(\mathbb{R})}\to 0$ as $\delta\to 0$. It implies the density of shallow ReLU neural networks in $L_1(\mathbb{R})$. In the following theorem, we extend this simple example to higher dimensions and show the universality of deep ReLU neural networks with bounded width.

Theorem 4 ([65]). *The family of deep ReLU neural networks with a width no more than $O(d)$ is dense in $L_1(\mathbb{R}^d)$.*

Proof. Let us consider the following iteration

$$\begin{aligned} L_1 &:= \sigma\left(1-h_1(x_1;[a_1,a_1+\delta])-h_2(x_1;[b_1-\delta,b_1])\right),\\ L_2 &:= \sigma\left(L_1-h_1(x_2;[a_2,a_2+\delta])-h_2(x_2;[b_2-\delta,b_2])\right),\\ L_k &:= \sigma\left(L_{k-1}-h_1(x_k;[a_k,a_k+\delta])-h_2(x_k;[b_k-\delta,b_k])\right), \end{aligned} \tag{8}$$

which aims to approximate $\chi_{\prod_{i=1}^d[a_i,b_i]}$.

The function L_1 is equal to $h(x_1;[a_1,b_1],\delta)$. When $x_1\in[a_1+\delta,b_1-\delta]$, i.e. $L_1=1$, then L_2 is equal to $h(x_2;[a_2,b_2],\delta)$. Since h_1 and h_2 are nonnegative, we have $L_1=0$ and hence $L_2=0$ for any $x_1\notin[a_1,b_1]$. Utilizing these results we obtain the expression of L_2, which approximates 2D characteristic function $\chi_{[a_1,b_1]\times[a_2,b_2]}$,

$$L_2=\begin{cases}1, & (x_1,x_2)\in[a_1+\delta,b_1-\delta]\times[a_2+\delta,b_2-\delta],\\ 0, & (x_1,x_2)\notin[a_1,b_1]\times[a_2,b_2],\\ \text{linear}, & \text{otherwise.}\end{cases}$$

We assume that L_k is equal to 1 on $\prod_{i=1}^k[a_i+\delta,b_i-\delta]$ and equal to 0 outside the square $\prod_{i=1}^k[a_i,b_i]$.

Then for $k+1$, if

$$(x_1,\dots,x_k)\in\prod_{i=1}^k[a_i+\delta,b_i-\delta],\quad x_{k+1}\in[a_{k+1}+\delta,b_{k+1}-\delta],$$

then we get $L_k=1$ and hence have

$$L_{k+1}=\sigma(h(x_{k+1};[a_{k+1},b_{k+1}],\delta))=1.$$

If we have

$$(x_1,\dots,x_k)\notin\prod_{i=1}^k[a_i,b_i],\quad x_{k+1}\notin[a_{k+1},b_{k+1}],$$

then $L_k=0$ and hence

$$L_{k+1}=\sigma(-h_1(x_{k+1};[a_{k+1},a_{k+1}+\delta])-h_2(x_{k+1};[b_{k+1}-\delta,b_{k+1}]))=0.$$

By induction, we prove that L_d is equal to 1 on $\prod_{i=1}^{d}[a_i+\delta, b_i-\delta]$ and equal to 0 outside the square $\prod_{i=1}^{d}[a_i, b_i]$. We can similarly show the quantity L_{k+1} has the upper bound 1

$$|L_{k+1}| \leq |L_k - h_1 - h_2| \leq |1 - h_1 - h_2| \leq 1,$$

where in the first step we use the fact that $|\sigma(x)| \leq |x|$, $x \in \mathbb{R}$ and apply (7) to the last equality.

For each iteration of L_k, the formula can be represented by a shallow neural network with input L_{k-1}, $h_1(x_k; [a_k, a_k+\delta])$ and $h_2(x_k; [b_k-\delta, b_k])$. For $h_1(x_k; [a_k, a_k+\delta])$, we need two neurons $\sigma(-x_k + a_k)$ and $\sigma(-x_k + a_k + \delta)$. Hence, we have to keep L_k and x_i, $i = 1, \ldots, d$ via $\sigma(x_i) - \sigma(-x_i)$ from the input layer to subsequent layers and it is easy to see the depth of this block is finite and width is no more than $O(d)$.

Now we shall construct a deep neural network to approximate a given $g \in L_1(\mathbb{R}^d)$. Notice that the family of simple functions is dense in $L_1(\mathbb{R}^d)$. For any $\varepsilon > 0$, one can always find a sequence of hyperrectangles Ω_i and constants $a_i \in \mathbb{R}$, $i = 1, \ldots, m$, such that $\|\sum_{i=1}^{m} a_i \chi_{\Omega_i} - g\|_{L_1(\mathbb{R}^d)} \leq \varepsilon$. According to (8), we need m such iterations to produce each χ_{Ω_i}. For the summation between χ_{Ω_i}, we only need to add two more neurons $\sigma(x) - \sigma(-x) = x$ to keep it accumulating until the output layer. Together with the above discussion of the neural network $L_d(\boldsymbol{x}; \Omega_i)$ which has the property $\|L_d - \chi_{\prod_{\Omega_i}}\|_{L_1(\mathbb{R}^d)} \to 0$, $\delta \to 0$. This completes the proof.

The "hat function" h in this subsection simulates the characteristic functions on hyperrectangles in a compositional manner (8). We will show later that deep ReLU neural networks can efficiently interpolate polynomials and approximate local Taylor expansions.

3.3 Related works

In the 20th century, the universal approximation properties of neural networks garnered a lot of attention, particularly for those equipped with sigmoidal functions [23, 9, 48], which satisfy $\lim_{x\to+\infty} \sigma(x) = 1$ and $\lim_{x\to-\infty} \sigma(x) = 0$ in general. On compact sets, sigmoidal shallow neural networks can approximate any continuous function to any prescribed error as measured by the supremum norm [23]. Furthermore, universality results for measurable functions were discussed in [48]. In [72], multilayer neural networks activated by certain sigmoidal functions were shown to be capable of approximating any continuous function and achieving the Jackson rate. For characteristic functions and functions possessing smoothness properties, the work [72] also explored related approximation algorithms. When the target function f has Fourier transform $\hat{f}$ and satisfies $\|\boldsymbol{w}\|_2 \hat{f}(\boldsymbol{w})$ being integrable, error analysis was given in [9], with the error at a rate of $O(N^{-1/2})$ when given N neurons. The survey paper [86] comprehensively reviewed the related results.

As research progressed, attention to the approximation theory of neural networks shifted to ReLU neural networks. Error bounds of shallow ReLU neural networks were investigated in [53]. Given a target function f with $\|\boldsymbol{w}\|_1^2 \hat{f}(\boldsymbol{w}) \in L_1(\mathbb{R}^d)$, they achieved an error rate of $O(\sqrt{\log N} N^{-1/2-1/d})$. However, for approximating Hölder functions with regularity $r > 2 + d/2$, the results in [53] are not applicable. This limitation was addressed in [70] and the provided bounds are close to optimal when d is large. Korobov functions were considered in [62], which belong to an important function space for understanding the curse of dimensionality, and achieved a rate of $O(\sqrt{N} N^{-(2d+4)/(5d)})$. By constructing wavelet frame with ReLU networks, in [93], the depth-4 ReLU neural networks were studied, giving optimal rates of approximation $O(N^{-2/d})$ for C^2 functions on some manifolds. The discussion on shallow neural networks with ReLU^k activation functions ($\mathrm{ReLU}^k(x) = (\mathrm{ReLU}(x))^k$) has been provided in [53, 109, 98, 7, 73]. These works have extended our understanding of expressivity of neural networks with different activation functions. The studies [55, 48, 29] focused on the networks with general activation functions and established the foundation for widespread applications.

4 Approximation properties of ReLU neural networks

This section details the convergence rates of deep ReLU neural networks for some basic functions and operations. While deeper or wider neural networks ensure the universality of neural networks, it remains unclear which configuration is superior and how to choose an optimal neural network for real-world applications. The findings of this study illuminate a fundamental question arising from observations in practice: why are deep neural networks more important? We will present some foundational results that demonstrate that increasing depth could be more efficient than increasing width regarding the expressivity of neural networks.

4.1 Efficient approximation of polynomials

First, we observe that neural networks can effectively approximate x^2 by utilizing sawtooth functions. Given that ReLU neural networks are piecewise linear functions, it is natural to employ linear interpolation to derive error bounds. Intuitively, a more refined approximation—with tighter error bounds—can be achieved if ReLU neural networks can represent a larger number of 'pieces'. Thus, our primary challenge lies in determining how to efficiently increase the number of these pieces.

We call a function $f : \mathbb{R} \to \mathbb{R}$ *t-sawtooth* if it is piecewise affine with t piecewise, i.e. there exists a partition $\{I_i : I_i \text{ are intervals}, i \in [t]\}$ of $\mathbb{R}$ such that $f(x) = a_i x + b_i$, $x \in I_i$ for some $a_i, b_i \in \mathbb{R}$.

Given a 2^n-sawtooth function, it can be easily implemented by a ReLU shallow neural network with 2^n neurons. In [104], it was found with a compositional

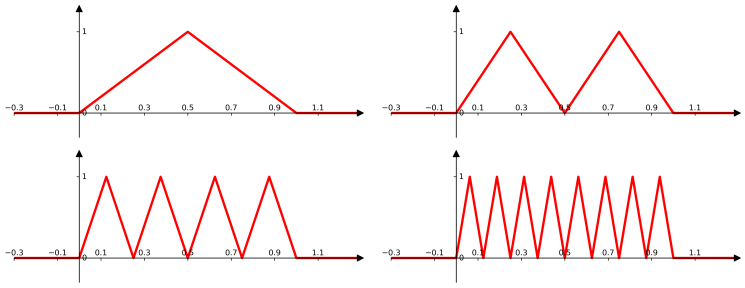

Fig. 7. Visualization of sawtooth functions $h^{(k)}$ (9), $k = 1, 2, 3, 4$.

structure, a deep ReLU neural network can efficiently produce functions with more segments. Consider the function $h(x) := h(x; [0, 1], \frac{1}{2})$, given by

$$h(x) = \begin{cases} 2x, & x \in [0, \frac{1}{2}], \\ 2(\frac{1}{2} - x) + 1, & x \in (\frac{1}{2}, 1]. \end{cases}$$

Obviously, $h(x)$ can be represented by a neural network with three neurons and itself is a 4-sawtooth function. The k-times composition of $h(x)$ results in a $(2^k + 2)$-sawtooth function, which can be realized with only $O(k)$ neurons [104].

Lemma 4 ([110]). *Let $f(x) = x^2$. For any $\varepsilon > 0$, there exists a ReLU neural network ϕ with bounded width and at most $O(|\ln \varepsilon|)$ layers and neurons such that $\|f - \phi\|_{L_\infty([0,1])} \leq \varepsilon$ and $\|\phi\|_{L_\infty([0,1])} \leq 1$. Besides, $\phi(x) = 0$ when $x = 0$.*

Proof. The proof leverages the efficient compositional structure of the sawtooth function. We shall first show that $h^{(k)}(x) = \underbrace{h \circ h \circ \cdots \circ h}_{k}$ has the following form:

$$h^{(k)}(x) = \begin{cases} 2^k \left(x - \frac{2i}{2^k}\right), & x \in \left[\frac{2i}{2^k}, \frac{2i+1}{2^k}\right], i = 0, 1, \ldots, 2^{k-1} - 1, \\ 2^k \left(\frac{2i+1}{2^k} - x\right) + 1, & x \in \left[\frac{2i+1}{2^k}, \frac{2i+2}{2^k}\right], i = 0, 1, \ldots, 2^{k-1} - 1, \end{cases} \tag{9}$$

where $h^{(k)}$ takes the value 1 at points $\frac{2i+1}{2^k}$ and 0 at points $\frac{2i}{2^k}$, $i = 0, 1, \ldots, 2^{k-1} - 1$. In other words, $h^{(k)}$ is linear over intervals $\left[\frac{2i+j}{2^k}, \frac{2i+j+1}{2^k}\right]$, $j = 0, 1$. Figure 7 provides an illustration of this function.

The basic case for $k = 2$ is straightforward. For $k > 2$, assume that the claim holds for k. Consider $x \in \left[\frac{2i}{2^k}, \frac{2i+1}{2^k}\right]$ for $k + 1$. Notice that we can rewrite h as

$$h(x) = 2x \cdot \chi_{[0,1/2]}(x) + 2(1 - x) \cdot \chi_{(1/2,1]}(x).$$

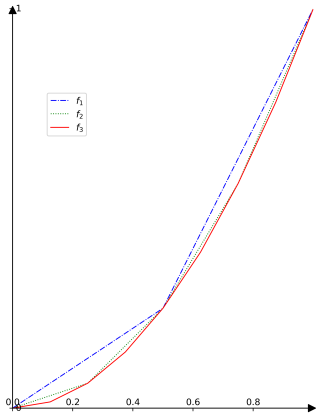

Fig. 8. Visualization of linear interpolation f_n, $n = 1, 2, 3$ of x^2.

Thus,

$$h^{(k+1)} = h \circ h^{(k)} = 2h^{(k)}\chi_{[0,\frac{1}{2}]}\left(h^{(k)}\right) + 2(1-h^{(k)})\chi_{(\frac{1}{2},1]}\left(h^{(k)}\right)$$
$$= \begin{cases} 2h^{(k)}, & x \in \left[\frac{2i}{2^k}, \frac{2i}{2^k} + \frac{1}{2^{k+1}}\right], \\ 2(1-h^{(k)}), & x \in \left[\frac{2i}{2^k} + \frac{1}{2^{k+1}}, \frac{2i+1}{2^k}\right], \\ 2h^{(k)}, & x \in \left[\frac{2i+1}{2^k}, \frac{2i+1}{2^k} + \frac{1}{2^{k+1}}\right], \\ 2(1-h^{(k)}), & x \in \left[\frac{2i+1}{2^k} + \frac{1}{2^{k+1}}, \frac{2i+2}{2^k}\right], \end{cases} \tag{10}$$

where on each interval $\left[\frac{2i}{2^k} + \frac{j}{2^{k+1}}, \frac{2i}{2^k} + \frac{j+1}{2^{k+1}}\right]$, $j = 0, \ldots, 3$, $h^{(k)}$ is linear and takes value $\{0, 1/2\}$ or $\{1/2, 1\}$ at endpoints. Direct calculation shows that $h^{(k+1)}$ is linear and takes values $\{0, 1\}$ at the endpoints of these intervals, proving the claim by induction.

Define f_k as the piecewise linear interpolation of f on points $\frac{i}{2^k}, i = 0, 1, \ldots, 2^k$, i.e. f_k is linear on $[\frac{i}{2^k}, \frac{i+1}{2^k}]$ and $f_k(\frac{i}{2^k}) := (\frac{i}{2^k})^2$. Note that $f_k - f_{k+1}$ is also piecewise linear with breakpoints $\frac{i}{2^{k+1}}, i = 0, 1, \ldots, 2^{k+1}$, with the value

$$f_k(\frac{2i}{2^{k+1}}) - f_{k+1}(\frac{2i}{2^{k+1}}) = 0,$$

and

$$f_k(\frac{2i+1}{2^{k+1}}) - f_{k+1}(\frac{2i+1}{2^{k+1}})$$
$$= \left(\frac{i}{2^k}\right)^2 + \frac{\left(\frac{i+1}{2^k}\right)^2 - \left(\frac{i}{2^k}\right)^2}{\frac{1}{2^k}} \frac{1}{2^{k+1}} - \left(\frac{2i+1}{2^{k+1}}\right)^2$$
$$= \frac{1}{2^{2(k+1)}}.$$

Therefore, (9) implies that $2^{2(k+1)}(f_k - f_{k+1})$ is equal to $h^{(k+1)}$. Summing over k, we derive

$$\begin{aligned}\sum_{k=1}^{n} 2^{-2k} h^{(k)}(x) &= \sum_{k=1}^{n} (f_{k-1}(x) - f_k(x)) \\ &= f_0(x) - f_n(x) \\ &= x - f_n(x),\end{aligned}$$

yielding:

$$f_n(x) = x - \sum_{k=1}^{n} 2^{-2k} h^{(k)}(x).$$

It is then straightforward to show that

$$|x^2 - f_n(x)| \leq 2^{-2n} \text{ and } |f_n(x)| \leq 1, \quad \forall x \in [0,1].$$

Since $h^{(n)}$ has a compositional structure with intermediate layers producing $h^{(k)}$, $k < n$, we now only need to add $\sigma(x) - \sigma(-x)$ to keep their summation to the last layer. This implies that f_n can be realized by a neural network with bounded width and $O(n)$ layers. By choosing $\varepsilon = 2^{-2n}$, we derive $n = O(|\ln \varepsilon|)$ and f_n has at most $c|\ln \varepsilon|$ layers and $c|\ln \varepsilon|$ neurons for some constant $c > 0$.

Combining the above lemma with Lemma 1, extension to general univariate polynomials is rather straightforward. Employing $xy = \frac{1}{2}(x+y)^2 - \frac{1}{2}x^2 - \frac{1}{2}y^2$, we are able to further approximate multivariate polynomials efficiently.

Theorem 5 ([110]). *Given $M > 0$ and let $f(x,y) = xy$, $(x,y) \in [-M,M]$. Then for any $\varepsilon > 0$, there exists a ReLU neural network ϕ with bounded width and at most $O(|\ln \varepsilon|))$ layers and neurons such that*

$$\|f - \phi\|_{L_\infty([-M,M]^2)} \leq \varepsilon.$$

In addition, ϕ satisfies $\phi(x,y) = 0$ if $xy = 0$.

Proof. Firstly, we rewrite $f(x,y)$ as

$$f(x,y) = 2M^2 \left(\left(\frac{x+y}{2M} \right)^2 - \left(\frac{x}{2M} \right)^2 - \left(\frac{y}{2M} \right)^2 \right).$$

Let ϕ_ε be a ReLU neural network from Lemma 4 such that $\|\phi_\varepsilon - x^2\|_{L_\infty([0,1])} \leq \varepsilon/6M^2$. Define

$$\phi(x,y) = 2M^2 \left(\phi_\varepsilon \left(\frac{x+y}{2M} \right) - \phi_\varepsilon \left(\frac{x}{2M} \right) - \phi_\varepsilon \left(\frac{y}{2M} \right) \right).$$

Using the triangle inequality, we obtain $\|f - \phi\|_{L_\infty([-M,M]^2)} \leq \varepsilon$. The proof is completed.

One can immediately extend Theorem 5 to higher-order products by

$$\begin{aligned}
&\psi_1 = x_1, \\
&\psi_2 = \phi_\varepsilon\left(\psi_1, x_2\right), \\
&\psi_3 = \phi_\varepsilon\left(\psi_2, x_3\right), \\
&\quad\vdots \\
&\psi_d = \phi_\varepsilon\left(\psi_{d-1}, x_d\right),
\end{aligned} \tag{11}$$

where we take ϕ_ε to be able to approximate $x \cdot y$ within error tolerance ε over a large enough domain $[-M, M]^2$. Notably, we can show $\psi_k(x_1, \ldots, x_k) = 0$ when $x_1 \times x_2 \times \cdots \times x_k = 0$ by induction. Then

$$\begin{aligned}
\left|\prod_{i=1}^{d} x_i - \psi_d\right| &\leq \left|\prod_{i=1}^{d} x_i - x_d \cdot \psi_{d-1}\right| + \left|x_d \cdot \psi_{d-1} - \phi_\varepsilon\left(\psi_{d-1}, x_d\right)\right| \\
&\leq |x_d| \left|\prod_{i=1}^{d-1} x_i - \psi_{d-1}\right| + \varepsilon.
\end{aligned}$$

Iteratively, we can prove that the ReLU neural network ψ_d has $O(|\ln \varepsilon|)$ layers and $O(|\ln \varepsilon|)$ neurons and is capable of realizing $\prod_{i=1}^{d} x_i$ within error ε. Similar techniques were employed by [105] for approximating rational functions with ReLU neural networks. More specifically, to approximate a rational function of degree no more than r, we can find a ReLU neural network with no more than $O(|\ln \varepsilon|^3)$ neurons that achieves error tolerance ε, which is also very efficient. Conversely, for some ReLU neural networks with bounded width and $2k$ layers, any rational function with $O(2^k)$ terms in both the numerator and denominator in total have a lower bound $\frac{1}{64}$ to approximate the functions given by ReLU neural networks in terms of L_1 norm, indicating the power of the compositional structure in neural networks.

4.2 Approximating locally with ReLU neural networks

The partition of unity is an important tool for establishing approximation results of deep neural networks [110, 61]. Let us set parameters in (7) as $a := \frac{m}{N} - \frac{2}{3N}$, $b := \frac{m}{N} + \frac{2}{3N}$, $\delta = \frac{1}{3N}$. Then

$$\begin{aligned}
h\left(x; \frac{m}{N}\right) &:= h(x; [a, b], \delta) \\
&= \begin{cases} 1, & x \in \left[\frac{m}{N} - \frac{1}{3N}, \frac{m}{N} + \frac{1}{3N}\right], \\ \text{linear}, & x \in \left(\frac{m}{N} - \frac{2}{3N}, \frac{m}{N} - \frac{1}{3N}\right) \cup \left(\frac{m}{N} + \frac{1}{3N}, \frac{m}{N} + \frac{2}{3N}\right), \\ 0, & x \notin \left[\frac{m}{N} - \frac{2}{3N}, \frac{m}{N} + \frac{2}{3N}\right]. \end{cases}
\end{aligned} \tag{12}$$

Summing over all $m \in [N]_0$ where we recall $[N]_0 = \{0, 1, \dots, N\}$, we obtain the following equality holds for any $x \in [0, 1]$,

$$\sum_{m=0}^{N} h\left(x; \frac{m}{N}\right) = 1.$$

To see this, let us consider the interval between $\frac{m}{N}$ and $\frac{m+1}{N}$ with break points $\frac{m}{N} + \frac{i}{3N}$, $i = 1, 2$. Notice that the localization property indicates that only $h(x; \frac{m}{N})$ and $h(x; \frac{m+1}{N})$ are nonzero over $[\frac{m}{N}, \frac{m+1}{N}]$. When x is within the distance $\frac{1}{3N}$ of $\frac{m}{N}$ or $\frac{m+1}{N}$, then the summation is equal to one. Between $\frac{m}{N} + \frac{1}{3N}$ and $\frac{m}{N} + \frac{2}{3N}$, the linearity implies the summation equals one. Hence, we conclude the above equality.

Similarly, we have the following extended partition of unity on higher-dimensional cubes

$$\sum_{\boldsymbol{m} \in [N]_0^d} h_{\boldsymbol{m}}(\boldsymbol{x}) = 1,$$

where

$$h_{\boldsymbol{m}}(\boldsymbol{x}) = \prod_{i=1}^{d} h\left(x_i; \frac{m_i}{N}\right).$$

The above partition of unity has several properties, namely:

(a) $\|h_{\boldsymbol{m}}\|_{L_\infty([0,1]^d)} = 1$,

(b) supp $h_{\boldsymbol{m}} \subset \prod_{i=1}^{d} \left[\frac{m_i}{N} - \frac{1}{N}, \frac{m_i}{N} + \frac{1}{N}\right]$.

Using (11), we can construct a neural network that approximates $h_{\boldsymbol{m}}$ well in the sense of

$$\left\| h_{\boldsymbol{m}}(x_1, \dots, x_d) - \psi_d\left(h\left(x_1; \frac{m_1}{N}\right), \dots, h\left(x_d; \frac{m_i}{N}\right)\right)\right\|_{L_\infty([0,1]^d)} \leq \varepsilon, \qquad (13)$$

where the neural network has depth $O(|\ln \varepsilon|)$ and number of parameters $O(|\ln \varepsilon|)$. Now we can decompose f and consider the approximation on each small patch

$$\begin{aligned} f(\boldsymbol{x}) &= \sum_{\boldsymbol{m} \in [N]_0^d} h_{\boldsymbol{m}}(\boldsymbol{x}) f(\boldsymbol{x}) \\ &= \sum_{\boldsymbol{m}: |x_i - \frac{m_i}{N}| \leq \frac{1}{N}} h_{\boldsymbol{m}}(\boldsymbol{x}) f(\boldsymbol{x}), \end{aligned}$$

and for any fixed $\boldsymbol{x} \in [0, 1]^d$, there are at most 2^d terms in the summation of $(N + 1)^d$ terms that make $h_{\boldsymbol{m}}$ nonzero, which is efficient.

4.3 Neural networks and sawtooth functions

The following lemma indicates that depth could be more important than width.

Lemma 5 ([104]). *Let f and g be t_f-sawtooth and t_g-sawtooth, respectively. Then $f+g$ and $g \circ f$ are $(t_f + t_g)$-sawtooth and $t_f t_g$-sawtooth respectively.*

Proof. Let $\{(x_{f,i}, x_{f,i+1}) : i \in [t_f]\}$ be non-overlap intervals such that on each interval f is an affine function. Denote x_j, $j \in [t_g - 1]$ be break points of g. Without loss of generality, we assume that $t_g < t_f$. Then there are at most $t_g - 1$ intervals, let us say, (x_{f,i_j}, x_{f,i_j+1}), $j \in [t_g - 1]$ such that $x_j \in (x_{f,i_j}, x_{f,i_j+1})$. It indicates that $f + g$ is affine linear on both (x_{f,i_j}, x_j) and (x_j, x_{f,i_j+1}). Hence, there are at most $t_f + t_g$ intervals, with the form of $(x_{f,i}, x_{f,i+1})$, (x_{f,i_j}, x_j), or (x_j, x_{f,i_j+1}) such that $f + g$ is affine on them. It completes the claim for $f + g$.

Then let us consider $g \circ f\left((x_{f,i}, x_{f,i+1})\right)$. We assume that $f(x) = ax + b$, $x \in (x_{f,i}, x_{f,i+1})$ for some $a > 0$, $b \in \mathbb{R}$. Then $f\left((x_{f,i}, x_{f,i+1})\right) = (ax_{f,i} + b, ax_{f,i+1} + b)$. The fact that g is t_g sawtooth implies that $g \circ f$ is at most t_g-sawtooth on $(ax_{f,i} + b, ax_{f,i+1} + b)$. Applying the above analysis for any i, we conclude that $g \circ f$ is $t_f t_g$-sawtooth.

Lemma 6 ([104]). *Every ReLU neural network with depth L and width W is $(2W)^L$-sawtooth.*

Proof. This is a direct consequence of Lemma 5.

Equation (9) provides an example that there exists a ReLU neural network with bounded width and L layers such that it is $O(2^L)$-sawtooth. For a deeper analysis of the number of affine regions that neural networks can generate when partitioning the input space, please refer to [78, 50].

4.4 Lower bounds for approximating C^2 functions

In the following, we consider the rate of approximating C^2 functions that provide a lower bound. In practice, it could be an advantage to increase depth instead of width.

Theorem 6 ([110]). *Let $f \in C^2([0,1]^d)$ be a non-affine function. Consider a neural network ϕ with depth L and width W. If $|f(\boldsymbol{x}) - \phi(\boldsymbol{x})| \leq \varepsilon$ for any $\boldsymbol{x} \in [0,1]^d$, the network ϕ satisfies $\varepsilon \geq c(2W)^{-2L}$ where the constant $c > 0$ is independent of L and W.*

Proof. Since f is non-affine, there exists a point $\boldsymbol{x}_0$ such that $f''(\boldsymbol{x}_0) \geq c_0 > 0$ for some constant c_0. Due to the regularity of f, there exists a direction $\boldsymbol{d} \in \mathbb{R}^d$ such that for any $t \in [0,1]$, we have $\boldsymbol{x}_0 + t\boldsymbol{d} \in [0,1]^d$ and function $f_1(t) := f(\boldsymbol{x}_0 + t\boldsymbol{d})$ is strictly convex, satisfying

$$\min_{t \in [0,1]} |f_1''| \geq 2c > 0,$$

for some constant c. Now let us consider a neural network $\phi_1(t) := \phi(\boldsymbol{x}_0 + t\boldsymbol{d})$.

Lemma 6 implies that there exists an interval $[a, b] \subset [0, 1]$ such that ϕ_1 is affine linear on $[a, b]$ and $b - a \geq (2W)^{-L}$. Let $g = f_1 - \phi_1$. Then we have $\|g\|_{L_\infty([0,1])} \leq \|\phi - f\|_{L_\infty([0,1]^d)} \leq \varepsilon$ and $\min_{t\in[a,b]} |g''| \geq 2c$. Hence, g is strongly convex, and it follows that

$$g(b) \geq g\left(\frac{a+b}{2}\right) + g'\left(\frac{a+b}{2}\right)\left(b - \frac{a+b}{2}\right) + c\left(b - \frac{a+b}{2}\right)^2,$$
$$g(a) \geq g\left(\frac{a+b}{2}\right) + g'\left(\frac{a+b}{2}\right)\left(a - \frac{a+b}{2}\right) + c\left(a - \frac{a+b}{2}\right)^2$$

Combining the above inequalities, we obtain

$$\varepsilon \geq \max\{g(a), g(b)\} - g\left(\frac{a+b}{2}\right) \geq c\left(\frac{b-a}{2}\right)^2 \geq c'(2W)^{-2L},$$

where $c' = c/4$.

Notice that the total number of neurons is $N = WL$. Hence in the case of $L = 1$, Theorem 6 implies that $N = W \geq \sqrt{\frac{c}{4\varepsilon}}$. When a network has bounded width $W = C$ for some constant $C \in \mathbb{N}_+$, then we can estimate the number of layers and neurons by

$$L \geq \frac{\log(c/\varepsilon)}{2\log 2C} = O(|\log\varepsilon|), \quad N = CL \geq \frac{C\log(c/\varepsilon)}{2\log 2C} = O(|\log\varepsilon|).$$

The lower bound on N is $O(|\log\varepsilon|)$ for deep neural networks and $O(\varepsilon^{-1/2})$ for shallow neural networks. It implies that increasing depth could be much better for approximation. The lower bound also implies that the construction for approximating x^2 in Lemma 4 is optimal.

Other works have also attempted to explain the power of depth in neural networks. Specifically, in the work [37], it was demonstrated that there exists a function on $\mathbb{R}^d$ that is expressible by a depth-3 neural network with width $O(d^{19/4})$ but cannot be approximated arbitrarily well by any 2-layer network with width $O(e^{cd})$ for some $c > 0$. Additionally, the contributions [84, 110] considered general target function spaces on investigating the relationships between the accuracy and layers.

5 Convergence rates of deep neural networks

In this section, we shall explore the approximation properties of deep ReLU neural networks with various architectures for smooth functions developed in a series of recent works [110, 116, 76, 28, 15, 42, 99]. Smoothness is crucial for machine learning tasks, as highlighted in [84]. To illustrate this, consider an image classification task over a dataset of cats and dogs labeled by 0 or 1. Deep learning models are designed to estimate the probability distribution across possible

labels. When presented with an image of a dog, neural networks should indicate a probability of over 0.5 for the label 1. Furthermore, when the images are perturbed by noise, our model should remain stable and provide similar predictions, demonstrating that the model's outputs should not fluctuate significantly within any small region. Consequently, target functions with smoothness are of paramount importance in the expressivity of neural networks.

First, we introduce the ability of ReLU neural networks to approximate Sobolev functions nearly optimally. Subsequently, we extend to the Korobov space to see how to alleviate the curse of dimensionality. Besides, we consider the universality of convolutional neural networks and low bounds for sparsely connected neural networks. Finally, we discuss the universality of self-attention, which exhibits at least the same expressive power as deep fully connected neural networks.

5.1 Near optimal error bounds for approximating Sobolev functions

In the following, we denote by $L_p(\Omega)$ the standard Lebesgue space equipped with $\|\cdot\|_{L_p} := \|\cdot\|_{L_p(\Omega)}$ norm [4]. We shall first introduce the Sobolev spaces, which are important smooth function spaces and are widely studied in approximation theory.

Definition 5 (Sobolev spaces). *Let $r \in \mathbb{N}$ and $p \in [1, \infty]$. The Sobolev space $W_p^r((0,1)^d)$ is defined as a subspace of $L_p((0,1)^d)$*

$$W_p^r((0,1)^d) := \{f \in L_p((0,1)^d) : D^{\boldsymbol{n}} f \in L_p((0,1)^d), \|\boldsymbol{n}\|_1 \le r\},$$

with the following norm for $p \in [1, \infty)$

$$\|f\|_{W_p^r} := \left(\sum_{\|\boldsymbol{n}\|_1 \le r} \|D^{\boldsymbol{n}} f\|_{L_p((0,1)^d)}^p \right)^{1/p},$$

and

$$\|f\|_{W_\infty^r} := \max_{\|\boldsymbol{n}\|_1 \le r} \operatorname{ess\,sup}_{\boldsymbol{x} \in (0,1)^d} |D^{\boldsymbol{n}} f|,$$

where $D^{\boldsymbol{n}} f$ is the corresponding weak derivative.

Under the hypothesis of continuous weight selection assumption, any mapping that approximates W_∞^r functions should have at least $O(\varepsilon^{-d/r})$ free parameters [32]. The following result shows that ReLU neural networks can achieve the optimal rate up to a logarithmic term.

Theorem 7 ([110]). *Let $d, r \in \mathbb{N}_+$ and $f \in W_\infty^r([0,1]^d)$ with $\|f\|_{W_\infty^r} \le 1$. For any $\varepsilon > 0$, there exits a ReLU neural network ϕ with depth $O(|\ln \varepsilon|)$ and $O(\varepsilon^{-d/r} |\ln \varepsilon|)$ neurons such that $\|f - \phi\|_{L_\infty([0,1]^d)} \le \varepsilon$.*

Proof. Let $P_{\boldsymbol{m}}$ be the $(r-1)$-degree Taylor polynomial of f at $\boldsymbol{x}_{\boldsymbol{m}} = \boldsymbol{m}/N = (m_1/N, \ldots, m_d/N)$,

$$P_{\boldsymbol{m}}(\boldsymbol{x}) = \sum_{\|\boldsymbol{n}\|_1 < r} \frac{D^{\boldsymbol{n}} f(\boldsymbol{x}_{\boldsymbol{m}})}{\boldsymbol{n}!} (\boldsymbol{x} - \boldsymbol{x}_{\boldsymbol{m}})^{\boldsymbol{n}},$$

where $\boldsymbol{n}! = \prod_{i=1}^d n_i!$ and $(\boldsymbol{x}-\boldsymbol{x}_{\boldsymbol{m}})^{\boldsymbol{n}} = \prod_{i=1}^d (x_i - (\boldsymbol{x}_{\boldsymbol{m}})_i)^{n_i}$. Then there exists a localized approximation f_N of f, defined as

$$\begin{aligned} f_N(\boldsymbol{x}) &= \sum_{\boldsymbol{m} \in [N]_0^d} h_{\boldsymbol{m}}(\boldsymbol{x}) P_{\boldsymbol{m}}(\boldsymbol{x}) \\ &= \sum_{\boldsymbol{m} \in [N]_0^d} \sum_{\|\boldsymbol{n}\|_1 < r} \frac{D^{\boldsymbol{n}} f(\boldsymbol{x}_{\boldsymbol{m}})}{\boldsymbol{n}!} h_{\boldsymbol{m}}(\boldsymbol{x}) (\boldsymbol{x} - \boldsymbol{x}_{\boldsymbol{m}})^{\boldsymbol{n}}. \end{aligned} \tag{14}$$

where $h_{\boldsymbol{m}}$ forms the partition of unity of $[0,1]^d$, as discussed in Sect. 4.2. Applying the localization property of $h_{\boldsymbol{m}}$ and Taylor expansion of f, we obtain that the approximation error is bounded by

$$\begin{aligned} |f(\boldsymbol{x}) - f_N(\boldsymbol{x})| &\le \sum_{\boldsymbol{m} \in [N]_0^d} h_{\boldsymbol{m}}(\boldsymbol{x}) \, |f(\boldsymbol{x}) - P_{\boldsymbol{m}}(\boldsymbol{x})| \\ &\le \sum_{\boldsymbol{m} \in \Lambda} |f(\boldsymbol{x}) - P_{\boldsymbol{m}}(\boldsymbol{x})| \\ &\le 2^d \max_{\boldsymbol{m}: |x_i - \frac{m_i}{N}| < \frac{1}{N}} |f(\boldsymbol{x}) - P_{\boldsymbol{m}}(\boldsymbol{x})| \\ &\le \frac{2^d d^r}{r!} \frac{1}{N^r}, \end{aligned}$$

where $\Lambda = \{\boldsymbol{m} \in [N]_0^d : h_{\boldsymbol{m}}(\boldsymbol{x}) \neq 0\}$. Next, we only need to construct a neural network ϕ that is built with similar terms in f_N by approximating products. For this, denote

$$\phi(\boldsymbol{x}) = \sum_{\boldsymbol{m} \in [N]_0^d} \sum_{\|\boldsymbol{n}\|_1 < r} a_{\boldsymbol{m},\boldsymbol{n}} \phi_{\boldsymbol{m}}(\boldsymbol{x}),$$

where $a_{\boldsymbol{m},\boldsymbol{n}}$ is the coefficient in f_N and $\phi_{\boldsymbol{m}}(\boldsymbol{x})$ approximates the product between and within $(\boldsymbol{x}-\boldsymbol{x}_{\boldsymbol{m}})^{\boldsymbol{n}}$ and $h_{\boldsymbol{m}}(\boldsymbol{x})$. Within error tolerance δ, we can approximate the $(d+r-1)$-dimensional product $h_{\boldsymbol{m}}(\boldsymbol{x})(\boldsymbol{x}-\boldsymbol{x}_{\boldsymbol{m}})^{\boldsymbol{n}}$ with a neural network $\phi_{\boldsymbol{m}}$ with depth at most $O(|\ln \delta|)$ and at most $O(|\ln \delta|)$ neurons, as shown in (11) and (13).

Combing (13) and Lemma 4, we obtain the error of using ϕ to approximate f_N as

$$\begin{aligned} |f_N(\boldsymbol{x}) - \phi(\boldsymbol{x})| &\le \sum_{\boldsymbol{m} \in [N]_0^d} \sum_{\|\boldsymbol{n}\|_1 < r} |a_{\boldsymbol{m},\boldsymbol{n}}| \, |h_{\boldsymbol{m}}(\boldsymbol{x})(\boldsymbol{x} - \boldsymbol{x}_{\boldsymbol{m}})^{\boldsymbol{n}} - \phi_{\boldsymbol{m}}(\boldsymbol{x})| \\ &\le 2^d d^r \delta. \end{aligned}$$

Let us now choose

$$N := \left\lceil \left(\frac{r! \cdot \varepsilon}{2^{d+1} d^r} \right)^{-1/r} \right\rceil, \quad \delta := \frac{\varepsilon}{2^{d+1} d^r}. \tag{15}$$

Then we can conclude that

$$|f(\boldsymbol{x}) - \phi(\boldsymbol{x})| \leq |f(\boldsymbol{x}) - f_N(\boldsymbol{x})| + |f_N(\boldsymbol{x}) - \phi(\boldsymbol{x})| \leq \varepsilon,$$

where ϕ has depth $O(|\ln \varepsilon|)$ and $O((N+1)^d d^r |\ln \varepsilon|) = O(\varepsilon^{-d/r} |\ln \varepsilon|)$ neurons. The proof is completed.

The core concept of proving Theorem 7 involves utilizing Taylor polynomial approximation. Analogously, developing averaged Taylor polynomials, one can generalize approximation results for functions belonging to the W_p^r space with the error measured by W_p^s norm, where $1 \leq p \leq \infty$ and $s \in [0, 1]$. Let us first introduce the fractional-order spaces $W_p^s((0,1)^d)$.

Definition 6 (Sobolev-Slobodeckij spaces). *Let $s \in (0,1)$ and $p \in [1, \infty]$. The Sobolev-Slobodeckij space $W_p^s((0,1)^d)$ is defined as a subspace of $L_p((0,1)^d)$*

$$W_p^s((0,1)^d) := \{f \in L_p((0,1)^d) : \|f\|_{W_p^s} < \infty\},$$

equipped with the norm

$$\|f\|_{W_p^s} := \left(\|f\|_{L_p((0,1)^d)}^p + \int_{(0,1)^d} \int_{(0,1)^d} \left(\frac{|f(\boldsymbol{x}) - f(\boldsymbol{y})|}{\|\boldsymbol{x} - \boldsymbol{y}\|_2^{s+d/p}} \right)^p d\boldsymbol{x} d\boldsymbol{y} \right)^{1/p},$$

for $p \in [1, \infty)$, and

$$\|f\|_{W_\infty^s} := \max \left\{ \|f\|_{L_\infty((0,1)^d)}, \operatorname*{ess\,sup}_{\boldsymbol{x}, \boldsymbol{y} \in (0,1)^d} \frac{|f(\boldsymbol{x}) - f(\boldsymbol{y})|}{\|\boldsymbol{x} - \boldsymbol{y}\|_2^s} \right\}.$$

The Sobolev-Slobodeckij spaces defined in Definition 6 are Banach spaces. The generalized result with W_p^s norm developed in [42] aligns with Theorem 7 when $s = 0$ and $p = \infty$, as described below.

Theorem 8 ([42]). *Let $d, r \in \mathbb{N}_+$ with $r \geq 2$, $p \in [1, \infty]$, $s \in [0, 1]$, let $f \in W_p^r((0,1)^d)$ with $\|f\|_{W_p^r} \leq 1$. For any $0 < \varepsilon < 1/2$, there exists a ReLU neural network ϕ with depth $O(|\ln \varepsilon^{r/(r-s)}|)$ and $O(\varepsilon^{-d/(r-s)} |\ln \varepsilon^{r/(r-s)}|)$ neurons such that $\|f - \phi\|_{W_p^s((0,1)^d)} \leq \varepsilon$.*

As highlighted in [42], if we set $s = 1$ and $p = \infty$, both the target function f and its weak derivative can be uniformly approximated by ϕ and the corresponding weak gradient, thereby demonstrating the expressive power of deep neural networks.

The main idea for proving Theorem 8 is similar to that of Theorem 7. We need to construct a local polynomial approximator f_N which is similar to (14) to decompose the error into two parts

$$\|f-\phi\|_{W_p^s} \leq \underbrace{\|f-f_N\|_{W_p^s}}_{\text{(I)}} + \underbrace{\|f_N-\phi\|_{W_p^s}}_{\text{(II)}}. \tag{16}$$

The challenge here is how to derive the analysis for W_p^s-error. The key technique to solve this problem relies on the properties of interpolation spaces.

Given two Banach spaces B_0, B_1. We call (B_0, B_1) an interpolation couple if B_1 is continuously embedded in B_0.

Definition 7 (Interpolation spaces). *Let (B_0, B_1) be an interpolation couple. Let $s \in (0,1)$ and $p \in [1,\infty]$. The interpolation space $B_{s,p}$ is defined as a subspace of B_0*

$$B_{s,p} := (B_0, B_1)_{s,p} := \{f \in B_0 : \|f\|_{(B_0,B_1)_{s,p}} < \infty\},$$

equipped with the norm

$$\|f\|_{(B_0,B_1)_{s,p}} := \begin{cases} \left(\int_0^\infty t^{-sp} K(t,f,B_0,B_1)^p \frac{dt}{t}\right)^{1/p}, & p \in [1,\infty), \\ \sup\limits_{t\in(0,\infty)} t^{-s} K(t,f,B_0,B_1), & p = \infty, \end{cases}$$

where $K(t,f,B_0,B_1)$ is defined as

$$K(t,f,B_0,B_1) := \inf_{g\in B_1} \left(\|f-g\|_{B_0} + t\|g\|_{B_1}\right), \quad t > 0.$$

If we define the interpolation space

$$\widetilde{W}_p^s((0,1)^d) := \left(L_p((0,1)^d), W_p^1((0,1)^d)\right)_{s,p},$$

then

$$W_p^s((0,1)^d) = \widetilde{W}_p^s((0,1)^d),$$

with equivalence of the norms [42, Theorem B.14]. This indicates that the analysis of W_p^s can benefit from the properties of interpolation spaces. Particularly, the analysis of both terms (I) and (II) in (16) can be reduced to the case $s \in \{0,1\}$.

To see this, we denote $\mathcal{L}(X,Y)$ the space of all linear bounded operators from X to Y equipped with the norm $\|T\|_{\mathcal{L}(X,Y)} := \{\|Tx\|_Y : \|x\|_X = 1\}$. The results [66, Theorem 1.6] and [66, Proposition 1.4] imply that we only need to consider the case $s \in \{0,1\}$ for the term (I) in (16) if we can construct a linear operator $Tf := f - f_N$ that satisfies conditions of the following result.

Theorem 9 ([66]). *Let $(L_p((0,1)^d), W_p^1((0,1)^d))$ be an interpolation couple. If $T \in \mathcal{L}(W_p^r, L_p) \cap \mathcal{L}(W_p^r, W_p^1)$, then $T \in \mathcal{L}(W_p^r, W_p^s)$ for any $s \in (0,1)$ and $p \in [1,\infty]$ and*

$$\|T\|_{\mathcal{L}(W_p^r,W_p^s)} \leq \|T\|^{1-s}_{\mathcal{L}(W_p^r,L_p)} \|T\|^s_{\mathcal{L}(W_p^r,W_p^1)}.$$

For the term (II), instead, we can utilize the following result [66, Corollary 1.7] to relax the condition on s to be $s \in \{0, 1\}$.

Corollary 1 ([66]). *Let $(L_p((0,1)^d), W_p^1((0,1)^d))$ be an interpolation couple and $s \in (0,1)$ and $p \in [1, \infty]$. Then there is a constant $c := c(s,p)$ such that for any $f \in W_p^1((0,1)^d)$, we have*

$$\|f\|_{W_p^s} \leq c\|f\|_{L_p}^{1-s}\|f\|_{W_p^1}^s.$$

The proof details of Theorem 8 can be found in [42], while additional properties of interpolation spaces are discussed in [66]. For the main properties of localized polynomials, please refer to [18, Chapter 4].

5.2 Alleviating the curse of dimensionality with Korobov functions

One significant challenge in considering Sobolev spaces is the curse of dimensionality. As the number of neurons grows rapidly, proportional to $\varepsilon^{-d/r}$, these spaces often become impractical for image processing tasks, where images typically consist of around 100×100 pixels, leading to approximately $30,000$ input dimensions when considering color channels. This limitation emphasizes the importance of research aimed at overcoming the curse of dimensionality. To address this issue, two main remedies have been proposed.

One approach involves considering function spaces that are smaller than Sobolev spaces but still reasonably large for practical applications. Examples include Korobov functions [76, 62, 59], bandlimited functions [77], and other function classes [53, 84]. Another strategy leverages the manifold assumption, which helps reduce the intrinsic dimension of inputs. By doing so, the rate of growth depends primarily on the manifold dimension rather than the high input dimension [61, 24, 93, 92, 80, 20, 115].

In the following, we shall investigate why the Korobov space can help to alleviate the curse of dimensionality.

Definition 8. *Let $2 \leq p \leq \infty$. The Korobov space $X_p^2([0,1]^d)$ is defined as the space of L_p functions which vanish on the boundry of $[0,1]^d$*

$$X_p^2([0,1]^d) := \{f \in L_p([0,1]^d) : f|_{\partial[0,1]^d} = 0, D^{\boldsymbol{n}} f \in L_p([0,1]^d), \|\boldsymbol{n}\|_\infty \leq r\},$$

with the seminorm

$$|f|_{\boldsymbol{n},\infty} := \|D^{\boldsymbol{n}} f\|_{L_\infty}, \quad \|\boldsymbol{n}\|_\infty \leq r.$$

From the definition, we can see Korobov spaces require more conditions on derivatives $\|\boldsymbol{n}\|_\infty \leq r$ compared to $\|\boldsymbol{n}\|_1 \leq r$ and hence are subspaces of Sobolev spaces W_p^r. The benefit of considering the Korobov space is that any Korobov function can be approximated well by a very few number of piecewise linear functions.

Let us consider tensor product functions on each grid point $\boldsymbol{x}_{\boldsymbol{\ell},\boldsymbol{i}} = \boldsymbol{i} \cdot \boldsymbol{\Delta}_{\boldsymbol{\ell}} := (i_1 \Delta_{\ell_1}, \dots, i_d \Delta_{\ell_d})$, $0 \le i_j \le 2^{\ell_j}$, $\Delta_{\ell_j} = 2^{-\ell_j}$, $\ell \in \mathbb{N}^d$,

$$h_{\boldsymbol{\ell},\boldsymbol{i}}(\boldsymbol{x}) := \prod_{j=1}^{d} h\left(\frac{x_j - i_j \Delta_{\ell_j}}{\Delta_{\ell_j}}\right) := \prod_{j=1}^{d} h_{\ell_j,i_j}(x_j),$$

which can be generated by a mother hat function $h(x) := h(x;[-1,1],1)$, defined in (7) as

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

We further consider the function space spanned by these $h_{\boldsymbol{\ell},\boldsymbol{i}}$ with index $\boldsymbol{i}$ belongs to $\Lambda_{\boldsymbol{\ell}} := \{\boldsymbol{i} \in \mathbb{N}^d, 1 \le i_j \le 2^{\ell_j} - 1, i_j \text{ odd}\}$

$$W_{\boldsymbol{\ell}} = \text{span } \{h_{\boldsymbol{\ell},\boldsymbol{i}} : \boldsymbol{i} \in \Lambda_{\boldsymbol{\ell}}\}$$

and

$$V_n^{(1)} := \bigoplus_{\|\boldsymbol{\ell}\|_1 \le n+d-1} W_{\boldsymbol{\ell}}.$$

The support of $h_{\boldsymbol{\ell},\boldsymbol{i}}$ in $W_{\boldsymbol{\ell}}$ is mutually disjoint since the support of $h_{\boldsymbol{\ell},\boldsymbol{i}}$ is centered at $x_{\boldsymbol{\ell},\boldsymbol{i}}$ with radius controlled by $\boldsymbol{\Delta}_{\boldsymbol{\ell}}$ and we only take odd $\boldsymbol{i}$. The grid corresponds to the space $V_n^{(1)}$ is the so-called sparse grids [19]. The number of grid points of $V_n^{(1)}$ was shown to be $O(2^n n^{d-1})$ in Lemma 3.6 [19]. Define $f_n^{(1)} \in V_n^{(1)}$ as

$$f_n^{(1)} := \sum_{\|\boldsymbol{\ell}\|_1 \le n+d-1} \sum_{\boldsymbol{i} \in \Delta_{\boldsymbol{\ell}}} v_{\boldsymbol{\ell},\boldsymbol{i}} h_{\boldsymbol{\ell},\boldsymbol{i}},$$

with coefficients

$$v_{\boldsymbol{\ell},\boldsymbol{i}} := \int_{[0,1]^d} \prod_{j=1}^{d} \left(-2^{-\ell_j - 1} h_{\ell_j,i_j}(x_j)\right) \frac{\partial^{2d} f}{\partial x_1^2 \cdots \partial x_d^2} dx.$$

[19, Lemma 3.13] shows that given $f \in X_p^2([0,1]^d)$, for any $\varepsilon > 0$, we can achieve $\|f - f_n^{(1)}\|_\infty \le \varepsilon$ with $O(\varepsilon^{-1/2} |\ln \varepsilon|^{3(d-1)/2})$ sparse grids. However, by setting (15), Theorem 7 utilizes full grids $(N+1)^d = O(\varepsilon^{-d/r})$ for localized polynomials to approximate Sobolev functions, which grows much faster than using sparse grids. The simple expression $h_{\boldsymbol{\ell},\boldsymbol{i}}$ also helps improve the approximation rates. These benefits lead to the following improvement for the curse of dimensionality.

Theorem 10 ([76]). *Let $f \in X_p^2([0,1]^d)$ with $|f|_{\mathbf{2},\infty} \le 1$. Then there exists a ReLU neural network ϕ with depth $O(|\ln \varepsilon|)$ and $O(\varepsilon^{-1/2} |\ln \varepsilon|^{3(d-1)/2+1})$ neurons such that $\|f - \phi\|_{L_\infty([0,1]^d)} \le \varepsilon$.*

Proof. Naturally and similarly as the proof of Theorem 7, let us define a ReLU neural network ϕ as

$$\phi(\boldsymbol{x}) := \sum_{\|\boldsymbol{\ell}\|_1 \le n+d-1} \sum_{\boldsymbol{i} \in \boldsymbol{\Delta}_{\boldsymbol{\ell}}} v_{\boldsymbol{\ell},\boldsymbol{i}} \tilde{h}_{\boldsymbol{\ell},\boldsymbol{i}}(\boldsymbol{x}),$$

where we construct $\tilde{h}_{\boldsymbol{\ell},\boldsymbol{i}}(\boldsymbol{x})$ such that it can ε-approximate $h_{\boldsymbol{\ell},\boldsymbol{i}}$, defined as

$$\tilde{h}_{\boldsymbol{\ell},\boldsymbol{i}}(\boldsymbol{x}) := \psi_d \left(h_{\ell_1,i_1}(x_1), \dots, h_{\ell_d,i_d}(x_d) \right).$$

Here we use the network ψ_d defined in (11). The compact support of $\tilde{h}_{\boldsymbol{\ell},\boldsymbol{i}}$ is originated from the property of ψ_d and h_{ℓ_j,i_j}, which leads to

$$\begin{aligned} |f_n^{(1)}(\boldsymbol{x}) - \phi(\boldsymbol{x})| &\le \sum_{\|\boldsymbol{\ell}\|_1 \le n+d-1} \sum_{\boldsymbol{i} \in \boldsymbol{\Delta}_{\boldsymbol{\ell}}} |v_{\boldsymbol{\ell},\boldsymbol{i}}| |h_{\boldsymbol{\ell},\boldsymbol{i}}(\boldsymbol{x}) - \tilde{h}_{\boldsymbol{\ell},\boldsymbol{i}}(\boldsymbol{x})| \\ &\le \sum_{\|\boldsymbol{\ell}\|_1 \le n+d-1} |v_{\boldsymbol{\ell},\boldsymbol{i}_{\boldsymbol{\ell}}}| |h_{\boldsymbol{\ell},\boldsymbol{i}_{\boldsymbol{\ell}}}(\boldsymbol{x}) - \tilde{h}_{\boldsymbol{\ell},\boldsymbol{i}_{\boldsymbol{\ell}}}(\boldsymbol{x})| \\ &\le \varepsilon \sum_{\|\boldsymbol{\ell}\|_1 \le n+d-1} |v_{\boldsymbol{\ell},\boldsymbol{i}_{\boldsymbol{\ell}}}|, \end{aligned}$$

where in the second step, we use the fact that supports of elements in $W_{\boldsymbol{\ell}}$ are mutually disjoint. [19, Lemma 3.3] gives an estimation $|v_{\boldsymbol{\ell},\boldsymbol{i}}| \le 2^{-d-2\|\boldsymbol{\ell}\|_1} |f|_{\mathbf{2},\infty}$. Then we can further relax the bound to

$$\sum_{\|\boldsymbol{\ell}\|_1 \le n+d-1} |v_{\boldsymbol{\ell},\boldsymbol{i}_{\boldsymbol{\ell}}}| \le \sum_{\|\boldsymbol{\ell}\|_1 \le n+d-1} 2^{-d-2\|\boldsymbol{\ell}\|_1} \le 1.$$

Finally, we have

$$|f_n^{(1)}(\boldsymbol{x}) - \phi(\boldsymbol{x})| \le \varepsilon,$$

Together with the fact that $|f(\boldsymbol{x}) - f_n^{(1)}(\boldsymbol{x})| \le \varepsilon$, we obtain

$$\|f - \phi\|_\infty \le \|f - f_n^{(1)}\|_\infty + \|f_n^{(1)} - \phi\|_\infty.$$

The depth of ϕ is $O(|\ln \varepsilon|)$, given by the depth of ψ_d and it contains $O(|\ln \varepsilon| \times \varepsilon^{-1/2} |\ln \varepsilon|^{3(d-1)/2}) = O(\varepsilon^{-1/2} |\ln \varepsilon|^{3(d-1)/2+1})$ neurons.

5.3 Approximation error of convolutional neural networks

As illustrated in Fig. 3, convolutional layers can be regarded as specialized fully connected neural networks characterized by Toeplitz-type weight matrices (1). The primary advantage of convolutional neural networks for image processing lies in their sparsity and efficiency, as they rely solely on the parameters of convolutional kernels. Despite utilizing fewer parameters, CNNs are not mathematically a proper subset of fully connected neural networks. In fact, as we will demonstrate, with sufficient layers, CNNs can become equally powerful, capable of producing dense affine transformations in fully connected networks.

Theorem 11 ([116]). *Let $2 \leq s \leq d$, $m \in \mathbb{N}_+$, and $\Omega = [-1,1]^d$. Given a set of vectors $\boldsymbol{a}_i \in \mathbb{R}^d$ with $\|\boldsymbol{a}_i\|_1 = 1$ and $t_i \in \mathbb{R}$, $i \in [m]_0$. There exists a ReLU CNN ϕ with depth $L < \frac{(m+1)d-1}{s-1} + 1$ and kernel size $s+1$ such that for any $\boldsymbol{x} \in \Omega$, some elements of its outputs are equal to $\sigma(\langle \boldsymbol{a}_i, \boldsymbol{x}\rangle + t_i)$ and others are zero.*

Proof. Define a sequence $\boldsymbol{w} := [\boldsymbol{a}_m^\top, \ldots, \boldsymbol{a}_0^\top]$ with components of $\boldsymbol{a}_i$ reversed. Theorem 3 [116] implies there exists a sequence of filter masks $\{\boldsymbol{w}^{(\ell)}\}_{\ell=1}^L$ with support $[s+1]$ and $L < \frac{(m+1)d-1}{s-1}+1$ such that $\boldsymbol{w} = \boldsymbol{w}^{(L)} * \cdots * \boldsymbol{w}^{(0)}$. The definition of convolution and $\boldsymbol{w}$ implies that $\boldsymbol{T}^{\boldsymbol{w}} \in \mathbb{R}^{(d+(m+1)d-1)\times d}$ and its $(i+1)d$-th row of $\boldsymbol{T}^{\boldsymbol{w}}$ is exactly the transpose of $\boldsymbol{a}_i$. Denote $M = \max_\ell \sum_{k=-\infty}^{\infty} |\boldsymbol{w}^{(\ell)}|$ and $\boldsymbol{T}^{(\ell)} = T^{\boldsymbol{w}^{(\ell)}}$. Then we have

$$\|\boldsymbol{T}^{(\ell)} * \cdots * \boldsymbol{T}^{(1)} x\|_\infty \leq M^\ell, \quad \forall \boldsymbol{x} \in \Omega, \ell \geq 1.$$

Let us choose $\boldsymbol{b}^{(1)} = M\mathbf{1}_{d_1}$ and $\boldsymbol{b}^{(\ell)} = -M^{\ell-1}T^{(\ell)}\mathbf{1}_{d_{\ell-1}} + M^\ell \mathbf{1}_{d_\ell}$, $\ell > 1$. Then the ℓ-th layer of CNN ϕ parameterized by $\boldsymbol{T}^{(\ell)}$, $\boldsymbol{b}^{(\ell)}$, $\ell = 1, \ldots, \ell$ is equal to

$$\phi^{(\ell)}(\boldsymbol{x}) = \boldsymbol{T}^{(\ell)} * \cdots * \boldsymbol{T}^{(1)} x + M^\ell \mathbf{1}_{d_\ell},$$

Define the last bias vector $\boldsymbol{b}^{(L)}$ according to the following rule

$$\begin{cases} -M^{L-1}(\boldsymbol{T}^{(L)}\mathbf{1}_{d_{L-1}})_k + t_i, & k = (i+1)d, \\ -M^{L-1}(\boldsymbol{T}^{(L)}\mathbf{1}_{d_{L-1}})_k - M^L, & k \neq (i+1)d. \end{cases}$$

We finally get

$$\phi_k^{(L)}(\boldsymbol{x}) = \begin{cases} \sigma(\langle \boldsymbol{a}_i, \boldsymbol{x}\rangle + t_i), & k = (i+1)d, \\ 0, & k \neq (i+1)d, \end{cases}$$

which completes the proof.

Theorem 11 indicates that with enough convolutional layers, any inner product can be realized. In the following, we apply the convergence rate of shallow neural networks presented in [53] to demonstrate the universality of CNNs. To make CNNs functions, we apply a linear combination to elements in the outputs of CNNs in Definition 2.

Theorem 12 ([116]). *Let $d \in \mathbb{N}_+$, $2 \leq s \leq d$ and $\Omega = [-1,1]^d$. Given $f = F|_\Omega$ with $F \in H^r(\mathbb{R}^d)$ and $r > 2 + d/2$. For any small enough $\varepsilon > 0$, there exists a ReLU CNN ϕ with depth*

$$O(\varepsilon^{-2d/(d+2)} |\ln \varepsilon|^{2d/(d+2)})$$

and

$$O(\varepsilon^{-2d/(d+2)} |\ln \varepsilon|^{2d/(d+2)})$$

kernels with kernel size $s+1$ such that $\|f - \phi\|_{C(\Omega)} \leq \varepsilon$.

Proof. We apply the results of [53] that there exists a ridge approximation F_m of the form

$$F_m(\boldsymbol{x}) = t_0 + \langle \boldsymbol{a}_0, \boldsymbol{x} \rangle + \frac{v}{m} \sum_{i=1}^{m} \beta_i \sigma \left(\langle \boldsymbol{a}_0, \boldsymbol{x} \rangle - t_k \right),$$

with $\beta_k \in [-1, 1]$, $\|\boldsymbol{a}_i\|_1 = 1$, $0 \leq t_k \leq 1$, $t_0 = F(0)$, $\boldsymbol{a}_0 = \nabla F(0)$ and $|v| \leq 2v_{F,2} = 2 \int_{\mathbb{R}^d} \|w\|_1^2 |\hat{F}(w)| dw$ such that

$$\|F - F_m\|_{C(\Omega)} \leq c v_{F,2} \sqrt{d + \ln m}\ m^{-1/2-1/d},$$

Together with Theorem 11, we can find a convolutional neural network that realizes F_m, i.e.

$$\|f - \phi\|_{C(\Omega)} \leq c v_{F,2} \sqrt{d + \ln m}\ m^{-1/2-1/d},$$

Let $\tilde{L} := \lceil \frac{(m+1)d-1}{s-1} \rceil + 1$ and $\boldsymbol{w}^{(L)} = \boldsymbol{w}^{(L+1)} = \cdots = \boldsymbol{w}^{(\tilde{L})}$ be the delta sequence which takes the value 1 at only one position and zero otherwise. Then we can extend ϕ with kernels $\boldsymbol{w}^{(\ell)}$, $\ell > L$ from L layers to $\tilde{L}$ layers. Since $L < \frac{(m+1)d-1}{s-1} + 1 \leq \tilde{L}$, we have $m < \tilde{L}s/d$ and the upper bound can be controlled by

$$\|f - \phi\|_{C(\Omega)} \leq c_{s,d} \sqrt{\ln \tilde{L}}\ \tilde{L}^{-1/2-1/d}$$

Choose $\tilde{L} = \varepsilon^{-2d/(d+2)} |\ln \varepsilon|^{2d/(d+2)}$. Finally we obtain $\|f - \phi\|_{C(\Omega)} \leq \varepsilon$.

In Theorem 11, we focus on single-channel CNNs and establish a depth bound $O(d)$. We shall see that by using multi-channel CNNs, we can improve the bound on the convolutional layers to $O(\ln d)$, explaining the power of multi-channel convolutions.

Theorem 13 ([57]). *Let $L \in \mathbb{N}_+$, $d = 2^L$, and $\Omega \in [-1, 1]^d$. Given a set of vectors $\boldsymbol{a}_i \in \mathbb{R}^d$ with $\|\boldsymbol{a}_i\|_1 = 1$ and $t_i \in \mathbb{R}$, $i \in [m]_0$. Then there exists a ReLU multi-channel CNN ϕ with depth L and kernel size 2 such that for any $\boldsymbol{x} \in \Omega$, its i-th channel satisfies $\phi(\boldsymbol{x})_i = \sigma(\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle + t_i)$.*

Proof. First, we define a sequence of vectors

$$\{\boldsymbol{r}_{\ell,k} \in \mathbb{R}^{2^\ell}, k = 1, \ldots, n_\ell\}$$

such that their linear combination can generate any length-2^ℓ piece of $\boldsymbol{a}_i$, i.e.

$$\left\{ \left((\boldsymbol{a}_i)_{j2^\ell+1}, (\boldsymbol{a}_i)_{j2^\ell+2}, \ldots, (\boldsymbol{a}_i)_{(j+1)2^\ell} \right)^\top \right\}_{j \in [2^{L-\ell}-1]_0, i \in [m]_0}.$$

Then, we split $\boldsymbol{r}_{\ell+1,k}$ into two length-2^ℓ vectors $\boldsymbol{r}^{(1)}_{\ell+1,k}, \boldsymbol{r}^{(2)}_{\ell+1,k}$. There exists $\boldsymbol{w}^{(\ell)}_{j,k} = (w^{(\ell,1)}_{j,k}, w^{(\ell,2)}_{j,k})$ such that

$$\boldsymbol{r}^{(1)}_{\ell+1,k} = \sum_{j=1}^{n_\ell} w^{(\ell,1)}_{j,k} \boldsymbol{r}_{\ell,j},$$

$$\boldsymbol{r}^{(2)}_{\ell+1,k} = \sum_{j=1}^{n_\ell} w^{(\ell,2)}_{j,k} \boldsymbol{r}_{\ell,j}.$$

Hence, we can reorganize the column vector $\boldsymbol{r}_{\ell+1,k}$

$$\begin{aligned}\boldsymbol{r}_{\ell+1,k}^{\top} &= \left(\left(\boldsymbol{r}_{\ell+1,k}^{(1)}\right)^{\top}, \left(\boldsymbol{r}_{\ell+1,k}^{(2)}\right)^{\top}\right) \\ &= \sum_{j=1}^{n_\ell} \boldsymbol{w}_{j,k}^{(\ell)} \left(\begin{bmatrix} \boldsymbol{r}_{\ell,j}^{\top} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{r}_{\ell,j}^{\top} \end{bmatrix}\right).\end{aligned} \tag{17}$$

Notice that convolution (1) becomes a diagonal matrix when both kernel size and stride are equal to 2. Let

$$\mathrm{Diag}_k(\boldsymbol{x}) := \begin{bmatrix} \boldsymbol{z}^{\top} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{z}^{\top} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \ddots & \ddots & \boldsymbol{0} \\ \vdots & \cdots & \boldsymbol{0} & \boldsymbol{z}^{\top} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \cdots & \boldsymbol{z}^{\top} \end{bmatrix} \in \mathbb{R}^{k\times 2k},$$

for any $\boldsymbol{z} \in \mathbb{R}^2$. Then we can write (17) in matrix product

$$\mathrm{Diag}_{2^{L-\ell-1}}(\boldsymbol{r}_{\ell+1,k}) = \sum_{j=1}^{n_\ell} \mathrm{Diag}_{2^{L-\ell-1}}(\boldsymbol{w}_{j,k}^{(\ell)}) \mathrm{Diag}_{2^{L-\ell}}(\boldsymbol{r}_{\ell,j}),$$

In particular, when $\ell+1 = L$, we have $\mathrm{Diag}_{2^{L-\ell-1}}(\boldsymbol{r}_{\ell+1,k})\boldsymbol{x} = \langle \boldsymbol{r}_{\ell+1,k}, \boldsymbol{x}\rangle$. We now can choose $\boldsymbol{r}_{L,k}$ as $\boldsymbol{a}_k$ and $\{\boldsymbol{w}_{j,k}^{(\ell)}, \boldsymbol{r}_{1,k}\}$ as convolutional kernels. Iteratively, we show that there exists a sequence of multi-channel convolutional operators such that in the i-th channel, the final output equals $\langle \boldsymbol{a}_i, \boldsymbol{x}\rangle$. Similar to Theorem 11, we conclude the result by choosing proper bias $\boldsymbol{b}^{(\ell)}$.

The number of neurons in each layer in Theorem 12 is linearly increasing, which can be easily seen from (1). It was found that by utilizing the downsampling operation, neurons in each layer can be reduced [115]. Besides, for approximating ridge functions, the upper bound of free parameters is $O(\varepsilon^{-1/\alpha})$ where α only depends on the Lipschitz property of the target functions. This bound is independent of the input dimension, which shows the nice approximation properties of CNNs. It has also been shown that for various target functions that have compositional structures, the curse of dimensionality can also be circumvented [69, 68, 39, 57, 8]. Notably, when approximating radial basis functions, CNNs were proved theoretically better than shallow neural networks [68]. Beyond Theorem 11 for single-channel CNNs, many results also considered the multichannel CNNs [8, 57, 44, 61, 82]. In [61, 82], the analysis considered the relationship between fully connected neural networks and multichannel CNNs with residual connections and extended the results for approximating functions from Barron space, Hölder space, and Besov space. In addition, [44] considered multichannel 2D convolutional neural networks, which are more popular in image processing and cannot be directly deduced from, for example, Theorem 11. The idea is also first to realize a kernel decomposition result as Theorem 3 in [116]

for 2D cases and then explore the corresponding L_2 approximation property. In [57], the realization of the inner products can be extended to higher-dimensional convolutions, like 3D convolutions, which are also widely used in 3D MRI volume image segmentation [74].

To understand the superiority of CNNs in applications, convolutional layers are connected with layered thresholding algorithms for approximating sparse outputs of deep sparse coding problems [83]. The special structure of convolutional operators requires mild sparsity conditions. Other variants were considered in [1, 89, 101, 102]. Recently, it was shown that the error bound for approximating deep sparse features is $O(e^{cL})$ for some $c > 0$, where L is the depth of the CNN [58]. This demonstrates the efficiency and expressivity of CNNs.

5.4 Approximation error of sparsely connected neural networks

In the previous subsection, we considered CNNs and noted that convolution operators have special sparse structures, as given by Equation (1), which make them efficient in various applications. In this subsection, we will discuss the results for general sparsely connected neural networks. The connectivity of a neural network ϕ is defined as the total number of nonzero elements in weight matrices $\{\boldsymbol{W}^{(\ell)}\}_{\ell=1}^{L}$. We denote this connectivity of ϕ as $\mathcal{M}(\phi)$. Analyzing the bounds on $\mathcal{M}(\phi)$ can provide insights into the relationship between the memory requirements of deep neural networks and their expressivity. The corresponding approximation results for deep neural networks were developed based on the min-max rate-distortion framework, as presented in [15]. The central concept in this framework is the minimax code length, which quantifies the optimal minimal coding length required to achieve a given error during data compression.

Definition 9. *Let $d \in \mathbb{N}$, $\Omega \in \mathbb{R}$, and $\mathcal{C} \subset C(\Omega)$. For each $\ell \in \mathbb{N}$, we denote by*

$$\mathfrak{E}^{\ell} := \{E : \mathcal{C} \to \{0,1\}^{\ell}\}, \quad \mathfrak{D}^{\ell} := \{D : \{0,1\}^{\ell} \to L_2(\Omega)\},$$

the set of binary encoders and the set of binary decoders of length ℓ, respectively. Then, for $\varepsilon > 0$, the minimax code length $L(\varepsilon, \mathcal{C})$ is defined as

$$L(\varepsilon, \mathcal{C}) := \min\left\{\ell \in \mathbb{N} : \exists (E, D) \in \mathfrak{E}^{\ell} \times \mathfrak{D}^{\ell} : \sup_{f \in \mathcal{C}} \|D(E(f)) - f\| \leq \varepsilon\right\}.$$

Moreover, the optimal exponent $\gamma^*(\mathcal{C})$ is defined as

$$\gamma^*(\mathcal{C}) := \sup\left\{\gamma \in \mathbb{R} : L(\varepsilon, \mathcal{C}) \in O(\varepsilon^{-1/\gamma}), \varepsilon \to 0\right\}.$$

The optimal exponent $\gamma^*(\mathcal{C})$ describes the asymptotic behavior of the minimax code length $L(\varepsilon, \mathcal{C})$ as the required error tends to zero. When we have a larger $\gamma^*(\mathcal{C})$, the growth rate of $L(\varepsilon, \mathcal{C})$ is smaller and the minimal necessary code length could be much shorter and result in better memory requirements.

The following theorem describes the lower bound of the connectivity of a neural network by the optimal exponent.

Theorem 14 ([15]). *Let $d \in \mathbb{N}$, $\Omega \subset \mathbb{R}^d$, $c > 0$, and $\mathcal{C} \subset L_2(\Omega)$. If for any pair $(\varepsilon, f) \in (0, \frac{1}{2}) \times \mathcal{C}$ there exists a neural network $\phi_{\varepsilon,f}$ with weights being representable by no more than $\lceil c(\ln \varepsilon)^{-1} \rceil$ bits and $\|f - \phi_{\varepsilon,f}\|_{L_2} \leq \varepsilon$, then we have*

$$\sup_{f \in \mathcal{C}} \mathcal{M}(\phi_{\varepsilon,f}) \notin O(\varepsilon^{-1/\gamma}), \varepsilon \to 0, \quad \text{for all } \gamma > \gamma^*(\mathcal{C}).$$

As pointed out in [15], the optimal exponent $\gamma^*(\mathcal{C})$ has been studied in [27, 33, 41] for Besov spaces and cartoon-like functions. Combining these results with Theorem 14, lower bounds can be given for approximating these functions.

For the optimal approximation results, the idea is to consider utilizing the classical approximation results of representation systems.

Definition 10 ([15]). *Let $d \in \mathbb{N}$, $\Omega \subset \mathbb{R}^d$, and $\mathcal{D} := \{g_i\}_{i \in I} \subset L_2(\mathbb{R})$ be a representation system. Then $\mathcal{D}$ is said to be representable by neural networks if there exist $L, M \in \mathbb{N}$ such that for all $\varepsilon > 0$, there is a sequence of neural networks $\{\phi_{i,\varepsilon}\}_{i \in I}$ with exactly L layers and connectivity no more than M, satisfying*

$$\sup_{i \in I} \|g_i - \phi_{i,\varepsilon}\|_{L_2} \leq \varepsilon.$$

Based on the above definition, we can transfer the approximation results of representation systems to the approximation results of deep neural networks.

Theorem 15 ([15]). *Let $d \in \mathbb{N}$, $\Omega \subset \mathbb{R}^d$, and $\mathcal{D} := \{g_i\}_{i \in I} \subset L_2(\mathbb{R})$. We assume that $\mathcal{D}$ is representable by neural networks. Let $f \in L_2(\Omega)$ and for $M \in \mathbb{N}$, let $f_M = \sum_{i \in I_M} c_i g_i$, $I_M \subset I$, $\#I_M = M$ satisfy*

$$\|f - f_M\|_{L_2} \leq \varepsilon,$$

where $\varepsilon \in (0, 1/2)$. Then there exists a neural network ϕ with L (depending only on $\mathcal{D}$) layers and connectivity $O(M)$ such that

$$\|f - \phi\|_{L_2} \leq 2\varepsilon.$$

This theorem shows that for any $f \in L_2(\Omega)$, the connectivity of neural networks has at least the same order as the M-term approximation by $\mathcal{D}$. This result can also be generalized to neural networks with quantized weights [15]. Combining these results, one of the key contributions in [15] can be concluded: if a function class is optimally represented by an affine system, then it is also optimally represented by neural networks. Readers who are interested in this topic are encouraged to refer to [15].

5.5 Expressivity of self-attention and Transformer

In the paper [28], the relationship between self-attention and convolutional layers was discussed with the following revised attention score

$$\tilde{A}_{(i,j),(q,k)} = \left\langle \boldsymbol{X}_{i,j,:}^\top \boldsymbol{W}_{\text{qry}}\,,\ \boldsymbol{X}_{q,k,:}^\top \boldsymbol{W}_{\text{key}} \right\rangle + \left\langle \boldsymbol{X}_{i,j,:}^\top \boldsymbol{W}_{\text{qry}}\,,\ \boldsymbol{R}_{q-i,k-j}^\top \widehat{\boldsymbol{W}}_{\text{key}} \right\rangle \quad (18)$$

$$+ \left\langle \boldsymbol{u}^\top\,,\ \boldsymbol{X}_{q,k,:}^\top \boldsymbol{W}_{\text{key}} \right\rangle + \left\langle \boldsymbol{v}^\top\,,\ \boldsymbol{R}_{q-i,k-j}^\top \widehat{\boldsymbol{W}}_{\text{key}} \right\rangle, \quad (19)$$

where $\boldsymbol{u} \in \mathbb{R}^t$ and $\boldsymbol{v} \in \mathbb{R}^{t'}$ are unique for heads and $\boldsymbol{R}_{q-i,k-j} \in \mathbb{R}^{t'}$ is the relatively positional encoding shared by all layers and heads. Matrices $\boldsymbol{W}_{\text{key}}$ and $\widehat{\boldsymbol{W}}_{\text{key}}$ are separately applied to the input vectors and relative positional encoding. The above attention score differs from (4), aiming at learning the position difference between key vectors and query vectors [30].

Theorem 16 ([28]). *Let $d, K, c_{in}, c_{out} \in \mathbb{N}$ and $\boldsymbol{W} \in [0,1]^{K\times K\times c_{in}\times c_{out}}$. For any $\varepsilon > 0$, there exists a multi-head self-attention layer $\mathcal{SA}_{mul}$ with $N_h = K^2$ heads and a relative positional encoding such that for any $\boldsymbol{X} \in [0,1]^{d\times d\times c_{in}}$, we have*

$$\left\| (\mathcal{SA}_{mul}(\boldsymbol{X})_{i,j,:})^\top - \sum_{m,n=1}^{K} \boldsymbol{X}^\top_{i+m-1,j+n-1,:} \boldsymbol{W}_{m,n,:,:} \right\|_\infty \leq \varepsilon, \quad \forall (i,j) \in [d]^2$$

Proof. Let us set

$$\boldsymbol{W}_{\text{qry}} = \boldsymbol{W}_{\text{key}} = \boldsymbol{0}, \widehat{\boldsymbol{W}}_{\text{key}} = \boldsymbol{I}.$$

Then $\tilde{\boldsymbol{A}}_{(i,j),(q,k)} = \boldsymbol{v}^\top \boldsymbol{R}_{(q-i,k-j)}$. Define

$$\boldsymbol{\mu} = (q-i, k-j)^\top, \boldsymbol{\eta} \in [K-1]_0^2,$$

$$\boldsymbol{v} = -\alpha(1, -2\eta_1, -2\eta_2), \quad \boldsymbol{R}_{q-i,k-j} = (\|\boldsymbol{\mu}\|_2^2, \mu_1, \mu_2).$$

Substituting the above vectors in the attention score with relative positional encoding, we get the following form

$$\tilde{A}_{(i,j),(q,k)} = -\alpha(\|\boldsymbol{\mu}\|_2^2 - 2\boldsymbol{\mu}^\top \boldsymbol{\eta}) = -\alpha(\|\boldsymbol{\mu} - \boldsymbol{\eta}\|_2^2 - \|\boldsymbol{\eta}\|_2^2).$$

Denote $\boldsymbol{\mu}' = (q'-i, k'-j)^\top$. Then when $\boldsymbol{\mu} = \boldsymbol{\eta}$, the corresponding attention probability tends to 1 since

$$\begin{aligned} \tilde{P}_{(i,j),(q,k)} &= \frac{\exp\left(-\alpha(\|\boldsymbol{\mu} - \boldsymbol{\eta}\|_2^2 - \|\boldsymbol{\eta}\|_2^2)\right)}{\sum_{q',k'} \exp\left(-\alpha(\|\boldsymbol{\mu}' - \boldsymbol{\eta}\|_2^2 - \|\boldsymbol{\eta}\|_2^2)\right)} \\ &= \frac{1}{1 + \sum_{(q',k')\neq(q,k)} \exp\left(-\alpha(\|\boldsymbol{\mu}' - \boldsymbol{\eta}\|_2^2)\right)} \\ &\to 1, \quad \alpha \to +\infty. \end{aligned}$$

When $\boldsymbol{\mu} \neq \boldsymbol{\eta}$, we have $\tilde{P}_{(i,j),(q,k)} \to 0$, $\alpha \to +\infty$. Notice that $\boldsymbol{\mu} = \boldsymbol{\eta}$ implies $q = i + \eta_1$ and $k = j + \eta_2$. Therefore, the summation over q, k in (5) only has one non-zero term and multi-head self attention becomes

$$(\mathcal{SA}_{\text{mul}}(\boldsymbol{X})_{i,j,:})^\top \to \sum_{h=1}^{N_h} \boldsymbol{X}^\top_{i+\eta_1,j+\eta_2,:} \boldsymbol{W}^{(h)}_{\text{val}} \boldsymbol{W}_h, \quad \alpha \to +\infty$$

Let g be a bijective mapping between head indices $[N_h]$ and convolutional kernel indices $[K-1]_0^2$ (here we assume that $\#[N_h] = \#[K-1]_0^2$). Now if we set $\boldsymbol{W}$ such that $\boldsymbol{W}_{\eta_1+1,\eta_2+1,:,:} = \boldsymbol{W}_{\text{val}}^{(h)}\boldsymbol{W}_h = \boldsymbol{W}_{\text{val}}^{(g(\eta_1,\eta_2))}\boldsymbol{W}_{g(\eta_1,\eta_2)}$, then

$$\begin{aligned}(\mathcal{SA}_{\text{mul}}(\boldsymbol{X})_{i,j,:})^\top &\to \sum_{\eta_1,\eta_2=0}^{K-1} \boldsymbol{X}_{i+\eta_1,j+\eta_2,:}^\top \boldsymbol{W}_{\eta_1+1,\eta_2+1,:,:} \\ &= \sum_{m,n=1}^{K} \boldsymbol{X}_{i+m-1,j+n-1,:}^\top \boldsymbol{W}_{m,n,:,:}, \quad \alpha \to +\infty,\end{aligned}$$

which is a multi-channel convolution.

Combining the above results with existing analysis for CNNs as discussed in the previous section, we can obtain the universality of self-attention. Since the Transformer is built by iteratively performing multi-head self-attention and fully connected layers, similarly, we have that the Transformer is also a universal approximator. The contribution [6] also considered the universal approximation property of different variants of Transformers.

5.6 A look into the future of expressivity

In recent years, the field of deep learning approximation has been widely studied for functions with different properties, such as smooth/piecewise smooth functions [110, 84, 43, 112, 97, 103, 63, 108] and continuous functions [111, 94, 96, 95]. Yarotsky provided in [110] an in-depth discussion on upper and lower bounds for the approximation of deep neural networks. Specifically, as introduced in the previous subsection, a theoretical analysis was developed for Sobolev spaces W_∞^r and achieved nearly optimal approximation rate $O(\varepsilon^{-d/r}|\ln\varepsilon|)$ in terms of the number of neurons within approximation error ε. Subsequently, this was refined to $O(\varepsilon^{-d/r})$ with respect to the L_p norm, $p \in [1,\infty)$ in [84] and [43] with respect to L_∞. The contribution [42] provided a general upperbound $O(\varepsilon^{-d/(r-s)}|\ln\varepsilon|)$ with respect to the W_p^s norm, $s \in [0,1]$. Beyond considering an error analysis in terms of the number of neurons, other contributions have focused on the characterization in terms of depth L and width W. For instance, in [112], the rate $O(L^{-2r/d}|\ln L|^{2r/d})$ was achieved for L-layer neural networks with bounded width (only depends on the input dimension) and the work [103] instead considered those networks with finite layers. Furthermore, these results were improved in [96, 63, 97] and recent work [108] proved the rate $O(W^{-2r/d}L^{-2r/d})$ for Sobolev and Besov spaces.

For approximating continuous functions, the modulus of continuity is often used for error analysis, which is defined as

$$\omega_f(r) = \max\{|f|(\boldsymbol{x}) - f(\boldsymbol{y})| : |\boldsymbol{x}-\boldsymbol{y}| \le r\}.$$

Bounded-width neural networks were considered in [111] and they provided an approximation rate $O(\omega_f(O(L^{-2/d})))$ for continuous functions. For any $L \in$

$\mathbb{N}_+$ and $W \in \mathbb{N}_+$, the techniques in [94] helped achieving a better result: a nearly optimal rate $O(\omega_f(W^{-2/d}L^{-2/d}))$. Later this was further improved to $O(\omega_f(W^{-2/d}L^{-2/d}|\ln W|^{-1/d}))$ [96].

As we discussed in Sect. 4, the approximation theory of ReLU neural networks relies on efficient interpolation, which inspired the work [17] to explore rational functions as activation functions. They discovered that for an error $\varepsilon > 0$, the number of neurons required in rational neural networks to approximate ReLU is $O(\ln\ln\varepsilon)$. Rational neural networks incorporate x^2 term. However, as mentioned in Sect. 4, the lower bound $O(\ln\varepsilon)$ is achieved for approximating x^2 by ReLU neural networks. These suggest that rational neural networks could offer better performance. SignReLU, a ReLU-type activation function, has been proven to be even stronger than rational activations, with an upper bound $O(1)$ for approximating both rational functions and ReLU, implying that it could be a good choice [59].

Quadratic neurons represent another approach to improving the approximation results. For example, quadratic neuron defined in [38] is given by

$$\boldsymbol{x} \to \sigma\left((\boldsymbol{w}_1^\top \boldsymbol{x} + b_1)(\boldsymbol{w}_2^\top \boldsymbol{x} + b_2) + \boldsymbol{w}_3^\top(\boldsymbol{x} \odot \boldsymbol{x}) + b_3\right),$$

where $\odot$ denotes Hadamard product. Given that x^2 is straightforward to implement, parametric efficiency is observed for approximating various functions [38].

In [114], it was further found that with a class of elementary universal activation functions, neural networks are dense in $C([0,1]^d)$ with depth $O(1)$ and width $O(d^2)$. This implies that even with finite many neurons, neural networks could serve as universal approximators. In particular, one interesting finding is that this result can be applied to certain sigmoidal activation functions.

Nowadays, deep learning is actively employed in solving Partial Differential Equations (PDEs). One such example is DeepOnet, which is based on the universal approximation theorem for operators [64, 21]. Let us consider a compact set V of $C(\Omega)$ and a nonlinear continuous operator $G : V \to C(\Omega)$. For any $\varepsilon > 0$, there exist parameters $c_i^k, \theta_i^k, b_k \in \mathbb{R}$, $\boldsymbol{w}_k, \boldsymbol{x}_j \in \mathbb{R}^d$ such that

$$\left| G(u)(\boldsymbol{y}) - \sum_{k=1}^{p}\sum_{i=1}^{n} c_i^k \left(\sum_{j=1}^{m} \xi_{i,j}^k u(\boldsymbol{x}_j) + \theta_i^k\right) \sigma(\boldsymbol{w}_k^\top \boldsymbol{y} + b_k) \right| \le \varepsilon,$$

for all $u \in V, \boldsymbol{y} \in \Omega$. This approximation result underscores the potential of powerful neural networks like DeepOnet as PDE solvers and demonstrates the remarkable capabilities of neural networks.

Spiking neural networks (SNNs) represent a pioneering computational model inspired by the behavior of biological neurons in the brain. SNNs are designed to mimic the neural coding and information processing mechanisms observed in biological systems. Unlike traditional artificial neural networks (ANNs) that rely on continuous-valued activations, SNNs utilize discrete spikes or pulses to transmit information. These spikes can be seen as the neuronal action potentials emitted at specific time points.

Definition 11. *A spiking neural network ϕ is a simple finite directed graph (V, E) and consists of a finite set V of spiking neurons and a set $E \subset V \times V$ of synapses. Each synapse $(u, v) \in E$ is associated with a tuple $(w_{uv}, d_{uv}, \varepsilon_{uv})$ where $w_{uv} > 0$ is a synaptic weight, d_{uv} is a synaptic delay, and $\varepsilon_{uv} : \mathbb{R}_+ \to \mathbb{R}$ is a response function. Each non-input neuron v is associated with a firing threshold $\theta_v > 0$ and a membrane potential $P_v : \mathbb{R} \to \mathbb{R}$*

$$P_v(t) := \sum_{(u,v)\in E} \sum_{t_u^f \in F_u} w_{uv}\varepsilon_{uv}(t - t_u^f),$$

where $F_u := \left\{t_u^f : f \in [1, n], n \in \mathbb{N}\right\}$ denotes the set of firing times of a neuron u, i.e., times t whenever $P_u(t)$ reaches θ_u from below.

As shown in [99], SNNs can realize any continuous piecewise linear functions. Furthermore, it was also shown that a two-layer SNN can realize ReLU on a compact set and this is not true for a one-layer SNN. For a general deep ReLU neural network, the following characterization describes the connection between SNNs and ReLU neural networks.

Theorem 17 ([99]). *Let $d, L \in \mathbb{N}$, $[a, b]^d \in \mathbb{R}^d$ and ϕ_{ReLU} be an arbitrary ReLU neural network with L layers, width d and N neurons in total. Then there exists response functions ε_{uv}, $(u, v) \in E$ such that there is an SNN ϕ_{SNN} with $3L - 2$ layers and $N + (2d + 3)L - (2d - 2)$ neurons satisfying $\phi_{SNN}(\boldsymbol{x}) = \phi_{ReLU}(\boldsymbol{x})$, $\forall \boldsymbol{x} \in [a, b]^d$.*

This result indicates that SNNs could be more powerful than deep ReLU neural networks in terms of the flexibility of the choice of the response function. It was also proved that SNNs can effectively realize continuous piecewise linear functions with fewer neurons compared with that of ReLU neural networks [99]. This further demonstrates the expressivity of SNNs.

Acknowledgement

J. Li gratefully acknowledges support from the project CONFORM, funded by the German Federal Ministry of Education and Research (BMBF) as well as the German Research Foundation under the Grant DFG-SPP-2298. This work of G. Kutyniok was supported in part by the Konrad Zuse School of Excellence in Reliable AI (DAAD), the Munich Center for Machine Learning (MCML) as well as the German Research Foundation under Grants DFG-SPP-2298, KU 1446/31-1 and KU 1446/32-1. Furthermore, G. Kutyniok acknowledges additional support by the project "Next Generation AI Computing (gAIn)", funded by the Bavarian Ministry of Science and the Arts and the Saxon Ministry for Science, Culture, and Tourism as well as by the Hightech Agenda Bavaria.

References

1. Aberdam, A., Sulam, J., Elad, M.: Multi-layer sparse coding: The holistic way. SIAM Journal on Mathematics of Data Science **1**(1), 46–77 (2019)
2. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
3. Adamopoulou, E., Moussiades, L.: Chatbots: History, technology, and applications. Machine Learning with Applications **2**, 100,006 (2020)
4. Adams, R.A., Fournier, J.J.: Sobolev spaces. Elsevier (2003)
5. Aggarwal, C.C.: Neural networks and deep learning. Springer (2018)
6. Alberti, S., Dern, N., Thesing, L., Kutyniok, G.: Sumformer: Universal approximation for efficient transformers. In: Topological, Algebraic and Geometric Learning Workshops 2023, pp. 72–86. PMLR (2023)
7. Bach, F.: Breaking the curse of dimensionality with convex neural networks. Journal of Machine Learning Research **18**(19), 1–53 (2017)
8. Bao, C., Li, Q., Shen, Z., Cheng, T., Lei, W., Xiang, X.: Approximation analysis of convolutional neural networks. East Asian Journal on Applied Mathematics **13**(3), 524–549 (2023)
9. Barron, A.R.: Universal approximation bounds for superpositions of a sigmoidal function. IEEE Transactions on Information Theory **39**(3), 930–945 (1993)
10. Basodi, S., Ji, C., Zhang, H., Pan, Y.: Gradient amplification: An efficient way to train deep neural networks. Big Data Mining and Analytics **3**(3), 196–207 (2020)
11. Bengio, Y., Ducharme, R., Vincent, P.: A neural probabilistic language model. Advances in Neural Information Processing Systems **13** (2000)
12. Bengio, Y., Frasconi, P., Simard, P.: The problem of learning long-term dependencies in recurrent networks. In: IEEE International Conference on Neural Networks, pp. 1183–1188. IEEE (1993)
13. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks **5**(2), 157–166 (1994)
14. Berner, J., Grohs, P., Kutyniok, G., Petersen, P.: The modern mathematics of deep learning. In: P. Grohs, G. Kutyniok (eds.) Mathematical Aspects of Deep Learning, pp. 1–111. Cambridge University Press, Cambridge (2023)
15. Bolcskei, H., Grohs, P., Kutyniok, G., Petersen, P.: Optimal approximation with sparsely connected deep neural networks. SIAM Journal on Mathematics of Data Science **1**(1), 8–45 (2019)
16. Borkakoti, N., Thornton, J.M.: AlphaFold2 protein structure prediction: Implications for drug discovery. Current Opinion in Structural Biology **78**, 102,526 (2023)
17. Boulle, N., Nakatsukasa, Y., Townsend, A.: Rational neural networks. In: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 14,243–14,253. Curran Associates, Inc. (2020)
18. Brenner, S.C.: The mathematical theory of finite element methods. Springer (2008)
19. Bungartz, H.J., Griebel, M.: Sparse grids. Acta Numerica **13**, 147–269 (2004)
20. Chen, M., Jiang, H., Liao, W., Zhao, T.: Efficient approximation of deep ReLU networks for functions on low dimensional manifolds. Advances in Neural Information Processing Systems **32** (2019)

21. Chen, T., Chen, H.: Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. IEEE transactions on neural networks **6**(4), 911–917 (1995)
22. Chui, C.K., Li, X.: Approximation by ridge functions and neural networks with one hidden layer. Journal of Approximation Theory **70**(2), 131–141 (1992)
23. Chui, C.K., Li, X., Mhaskar, H.N.: Limitations of the approximation capabilities of neural networks with one hidden layer. Advances in Computational Mathematics **5**(1), 233–243 (1996)
24. Chui, C.K., Mhaskar, H.N.: Deep nets for local manifold learning. Frontiers in Applied Mathematics and Statistics **4**, 12 (2018)
25. Ciarlet, P.G.: Linear and nonlinear functional analysis with applications. SIAM (2013)
26. Clevert, D.A.: Fast and accurate deep network learning by exponential linear units (ELUs). arXiv preprint arXiv:1511.07289 (2015)
27. Cohen, A., Dahmen, W., Daubechies, I., DeVore, R.: Tree approximation and optimal encoding. Applied and Computational Harmonic Analysis **11**(2), 192–226 (2001)
28. Cordonnier, J.B., Loukas, A., Jaggi, M.: On the relationship between self-attention and convolutional layers. In: Eighth International Conference on Learning Representations-ICLR 2020 (2020)
29. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems **2**(4), 303–314 (1989)
30. Dai, Z., Yang, Z., Yang, Y., Carbonell, J.G., Le, Q.V., Salakhutdinov, R.: Transformer-XL: Attentive language models beyond a fixed-length context. In: Annual Meeting of the Association for Computational Linguistics (2019)
31. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: J. Burstein, C. Doran, T. Solorio (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186 (2019)
32. DeVore, R.A., Howard, R., Micchelli, C.: Optimal nonlinear approximation. Manuscripta Mathematica **63**, 469–478 (1989)
33. Donoho, D.L.: Sparse components of images and optimal atomic decompositions. Constructive Approximation **17**, 353–382 (2001)
34. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2020)
35. Dubey, S.R., Singh, S.K., Chaudhuri, B.B.: Activation functions in deep learning: A comprehensive survey and benchmark. Neurocomputing **503**, 92–108 (2022)
36. Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., Garcia, R.: Incorporating second-order functional knowledge for better option pricing. Advances in Neural Information Processing Systems **13** (2000)
37. Eldan, R., Shamir, O.: The power of depth for feedforward neural networks. In: Conference on Learning Theory, pp. 907–940. PMLR (2016)
38. Fan, F.L., Dong, H.C., Wu, Z., Ruan, L., Zeng, T., Cui, Y., Liao, J.X.: One neuron saved is one neuron earned: On parametric efficiency of quadratic networks. arXiv preprint arXiv:2303.06316 (2023)
39. Fang, Z., Feng, H., Huang, S., Zhou, D.X.: Theory of deep convolutional neural networks II: Spherical analysis. Neural Networks **131**, 154–162 (2020)

40. Georgiev, S.G., Georgiev: Theory of Distributions. Springer (2015)
41. Grohs, P., Keiper, S., Kutyniok, G., Schäfer, M.: Cartoon approximation with α α-curvelets. Journal of Fourier Analysis and Applications **22**, 1235–1293 (2016)
42. Gühring, I., Kutyniok, G., Petersen, P.: Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms. Analysis and Applications **18**(05), 803–859 (2020)
43. Han, Z., Yu, S., Lin, S.B., Zhou, D.X.: Depth selection for deep ReLU nets in feature extraction and generalization. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(4), 1853–1868 (2022)
44. He, J., Li, L., Xu, J.: Approximation properties of deep ReLU CNNs. Research in the Mathematical Sciences **9**(3), 38 (2022)
45. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
46. Hochreiter, S.: Untersuchungen zu dynamischen neuronalen netzen. Diploma, Technische Universität München **91**(1), 31 (1991)
47. Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al.: Gradient flow in recurrent nets: the difficulty of learning long-term dependencies (2001)
48. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural networks **2**(5), 359–366 (1989)
49. Hu, Y., Kuang, W., Qin, Z., Li, K., Zhang, J., Gao, Y., Li, W., Li, K.: Artificial intelligence security: Threats and countermeasures. ACM Computing Surveys (CSUR) **55**(1), 1–36 (2021)
50. Huchette, J., Muñoz, G., Serra, T., Tsay, C.: When deep learning meets polyhedral theory: A survey. arXiv preprint arXiv:2305.00241 (2023)
51. Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., et al.: ChatGPT for good? on opportunities and challenges of large language models for education. Learning and Individual Differences **103**, 102,274 (2023)
52. Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S.: Self-normalizing neural networks. Advances in Neural Information Processing Systems **30** (2017)
53. Klusowski, J.M., Barron, A.R.: Approximation by combinations of ReLU and squared ReLU ridge functions with ℓ^1 and ℓ^0 controls. IEEE Transactions on Information Theory **64**(12), 7649–7656 (2018)
54. Kuhail, M.A., Alturki, N., Alramlawi, S., Alhejori, K.: Interacting with educational chatbots: A systematic review. Education and Information Technologies **28**(1), 973–1018 (2023)
55. Leshno, M., Lin, V.Y., Pinkus, A., Schocken, S.: Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural networks **6**(6), 861–867 (1993)
56. Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J.Z., Langer, D., Pink, O., Pratt, V., et al.: Towards fully autonomous driving: Systems and algorithms. In: 2011 IEEE Intelligent Vehicles Symposium (IV), pp. 163–168. IEEE (2011)
57. Li, J., Feng, H., Zhou, D.X.: Approximation analysis of CNNs from a feature extraction view. Analysis and Applications **22**(03), 635–654 (2024)
58. Li, J., Feng, H., Zhou, D.X.: Convergence analysis for deep sparse coding via convolutional neural networks. arXiv preprint arXiv:2408.05540 (2024)
59. Li, J., Feng, H., Zhou, D.X.: SignReLU neural network and its approximation ability. Journal of Computational and Applied Mathematics **440**, 115,551 (2024)

60. Lin, V.Y., Pinkus, A.: Fundamentality of ridge functions. Journal of Approximation Theory **75**(3), 295–311 (1993)
61. Liu, H., Chen, M., Zhao, T., Liao, W.: Besov function approximation and binary classification on low-dimensional manifolds using convolutional residual networks. In: International Conference on Machine Learning, pp. 6770–6780. PMLR (2021)
62. Liu, Y., Mao, T., Zhou, D.X.: Approximation of functions from Korobov spaces by shallow neural networks. Information Sciences **670**, 120,573 (2024)
63. Lu, J., Shen, Z., Yang, H., Zhang, S.: Deep network approximation for smooth functions. SIAM Journal on Mathematical Analysis **53**(5), 5465–5506 (2021)
64. Lu, L., Jin, P., Pang, G., Zhang, Z., Karniadakis, G.E.: Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature Machine Intelligence **3**(3), 218–229 (2021)
65. Lu, Z., Pu, H., Wang, F., Hu, Z., Wang, L.: The expressive power of neural networks: A view from the width. Advances in Neural Information Processing Systems **30** (2017)
66. Lunardi, A.: Interpolation theory, vol. 16. Springer (2018)
67. Maas, A.L., Hannun, A.Y., Ng, A.Y., et al.: Rectifier nonlinearities improve neural network acoustic models. In: Proc. ICML, vol. 30, p. 3. Atlanta, GA (2013)
68. Mao, T., Shi, Z., Zhou, D.X.: Theory of deep convolutional neural networks III: Approximating radial functions. Neural Networks **144**, 778–790 (2021)
69. Mao, T., Shi, Z., Zhou, D.X.: Approximating functions with multi-features by deep convolutional neural networks. Analysis and Applications **21**(01), 93–125 (2023)
70. Mao, T., Zhou, D.X.: Rates of approximation by ReLU shallow neural networks. Journal of Complexity **79**, 101,784 (2023)
71. Medsker, L.R., Jain, L., et al.: Recurrent neural networks. Design and Applications **5**(64-67), 2 (2001)
72. Mhaskar, H.N.: Approximation properties of a multilayered feedforward artificial neural network. Advances in Computational Mathematics **1**(1), 61–80 (1993)
73. Mhaskar, H.N.: Function approximation with zonal function networks with activation functions analogous to the rectified linear unit functions. Journal of Complexity **51**, 1–19 (2019)
74. Milletari, F., Navab, N., Ahmadi, S.A.: V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: 2016 Fourth International Conference on 3D Vision (3DV), pp. 565–571. Ieee (2016)
75. Misra, D.: Mish: A self regularized non-monotonic activation function. In: The 31st British Machine Vision Conference (2020)
76. Montanelli, H., Du, Q.: New error bounds for deep ReLU networks using sparse grids. SIAM Journal on Mathematics of Data Science **1**(1), 78–92 (2019)
77. Montanelli, H., Yang, H., Du, Q.: Deep ReLU networks overcome the curse of dimensionality for generalized bandlimited functions. Journal of Computational Mathematics **39**(6) (2021)
78. Montufar, G.F., Pascanu, R., Cho, K., Bengio, Y.: On the number of linear regions of deep neural networks. Advances in Neural Information Processing Systems **27** (2014)
79. Murdoch, B.: Privacy and artificial intelligence: challenges for protecting health information in a new era. BMC Medical Ethics **22**, 1–5 (2021)
80. Nakada, R., Imaizumi, M.: Adaptive approximation and generalization of deep neural network with intrinsic dimensionality. Journal of Machine Learning Research **21**(174), 1–38 (2020)

81. Niu, Z., Zhong, G., Yu, H.: A review on the attention mechanism of deep learning. Neurocomputing **452**, 48–62 (2021)
82. Oono, K., Suzuki, T.: Approximation and non-parametric estimation of ResNet-type convolutional neural networks. In: International Conference on Machine Learning, pp. 4922–4931. PMLR (2019)
83. Papyan, V., Romano, Y., Elad, M.: Convolutional neural networks analyzed via convolutional sparse coding. Journal of Machine Learning Research **18**(83), 1–52 (2017)
84. Petersen, P., Voigtlaender, F.: Optimal approximation of piecewise smooth functions using deep ReLU neural networks. Neural Networks **108**, 296–330 (2018)
85. Petersen, P., Zech, J.: Mathematical theory of deep learning. arXiv preprint arXiv:2407.18384 (2024)
86. Pinkus, A.: Approximation theory of the MLP model in neural networks. Acta Numerica **8**, 143–195 (1999)
87. Ramachandran, P., Zoph, B., Le, Q.V.: Searching for activation functions. arXiv preprint arXiv:1710.05941 (2017)
88. Ren, F., Ding, X., Zheng, M., Korzinkin, M., Cai, X., Zhu, W., Mantsyzov, A., Aliper, A., Aladinskiy, V., Cao, Z., et al.: AlphaFold accelerates artificial intelligence powered drug discovery: efficient discovery of a novel CDK20 small molecule inhibitor. Chemical Science **14**(6), 1443–1452 (2023)
89. Rey-Otero, I., Sulam, J., Elad, M.: Variations on the convolutional sparse coding model. IEEE Transactions on Signal Processing **68**, 519–528 (2020)
90. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical image computing and computer-assisted intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18, pp. 234–241. Springer (2015)
91. Schauperl, M., Denny, R.A.: AI-based protein structure prediction in drug discovery: impacts and challenges. Journal of Chemical Information and Modeling **62**(13), 3142–3156 (2022)
92. Schmidt-Hieber, J.: Deep ReLU network approximation of functions on a manifold. arXiv preprint arXiv:1908.00695 (2019)
93. Shaham, U., Cloninger, A., Coifman, R.R.: Provable approximation properties for deep neural networks. Applied and Computational Harmonic Analysis **44**(3), 537–557 (2018)
94. Shen, Z., Yang, H., Zhang, S.: Deep network approximation characterized by number of neurons. arXiv preprint arXiv:1906.05497 (2019)
95. Shen, Z., Yang, H., Zhang, S.: Nonlinear approximation via compositions. Neural Networks **119**, 74–84 (2019)
96. Shen, Z., Yang, H., Zhang, S.: Optimal approximation rate of ReLU networks in terms of width and depth. Journal de Mathématiques Pures et Appliquées **157**, 101–135 (2022)
97. Siegel, J.W.: Optimal approximation rates for deep ReLU neural networks on Sobolev and Besov spaces. Journal of Machine Learning Research **24**(357), 1–52 (2023)
98. Siegel, J.W., Xu, J.: High-order approximation rates for shallow neural networks with cosine and ReLUk activation functions. Applied and Computational Harmonic Analysis **58**, 1–26 (2022)
99. Singh, M., Fono, A., Kutyniok, G.: Expressivity of spiking neural networks through the spike response model. In: UniReps: the First Workshop on Unifying Representations in Neural Models (2023)

100. Soori, M., Arezoo, B., Dastres, R.: Artificial intelligence, machine learning and deep learning in advanced robotics, a review. Cognitive Robotics **3**, 54–70 (2023)
101. Sulam, J., Aberdam, A., Beck, A., Elad, M.: On multi-layer basis pursuit, efficient algorithms and convolutional neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **42**(8), 1968–1980 (2019)
102. Sulam, J., Papyan, V., Romano, Y., Elad, M.: Multilayer convolutional sparse modeling: Pursuit and dictionary learning. IEEE Transactions on Signal Processing **66**(15), 4090–4104 (2018)
103. Suzuki, T.: Adaptivity of deep ReLU network for learning in Besov and mixed smooth Besov spaces: optimal rate and curse of dimensionality. In: International Conference on Learning Representations (2019)
104. Telgarsky, M.: Representation benefits of deep feedforward networks. arXiv preprint arXiv:1509.08101 (2015)
105. Telgarsky, M.: Neural networks and rational functions. In: International Conference on Machine Learning, pp. 3387–3393. PMLR (2017)
106. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc. (2017)
107. Vostrecov, B., Kreines, M.: Approximation of continuous functions by superpositions of plane waves. In: Dokl. Akad. Nauk SSSR, vol. 140, pp. 1237–1240 (1961)
108. Yang, Y.: On the optimal approximation of Sobolev and Besov functions using deep ReLU neural networks. arXiv preprint arXiv:2409.00901 (2024)
109. Yang, Y., Zhou, D.X.: Optimal rates of approximation by shallow ReLUk neural networks and applications to nonparametric regression. Constructive Approximation pp. 1–32 (2024)
110. Yarotsky, D.: Error bounds for approximations with deep ReLU networks. Neural networks **94**, 103–114 (2017)
111. Yarotsky, D.: Optimal approximation of continuous functions by very deep ReLU networks. In: S. Bubeck, V. Perchet, P. Rigollet (eds.) Proceedings of the 31st Conference On Learning Theory, *Proceedings of Machine Learning Research*, vol. 75, pp. 639–649. PMLR (2018)
112. Yarotsky, D., Zhevnerchuk, A.: The phase diagram of approximation rates for deep neural networks. Advances in Neural Information Processing Systems **33**, 13,005–13,015 (2020)
113. Yurtsever, E., Lambert, J., Carballo, A., Takeda, K.: A survey of autonomous driving: Common practices and emerging technologies. IEEE Access **8**, 58,443–58,469 (2020)
114. Zhang, S., Shen, Z., Yang, H.: Deep network approximation: Achieving arbitrary accuracy with fixed number of neurons. Journal of Machine Learning Research **23**(276), 1–60 (2022)
115. Zhou, D.X.: Theory of deep convolutional neural networks: Downsampling. Neural Networks **124**, 319–327 (2020)
116. Zhou, D.X.: Universality of deep convolutional neural networks. Applied and Computational Harmonic Analysis **48**(2), 787–794 (2020)