Deep Network Approximation with Discrepancy Being Reciprocal of Width to Power of Depth

Zuowei Shen
matzuows@nus.edu.sg
Department of Mathematics, National University of Singapore

Haizhao Yang
haizhao@purdue.edu
Department of Mathematics, Purdue University

Shijun Zhang
zhangshijun@u.nus.edu
Department of Mathematics, National University of Singapore

Keywords: Exponential Convergence, Curse of Dimensionality, Deep Neural Network, Floor-ReLU Activation Function, Approximation via Compositions, Continuous Function.

Abstract

A new network with super approximation power is introduced. This network is built with Floor ($\lfloor x \rfloor$) and ReLU ($\max\{0, x\}$) activation functions and hence we call such networks as Floor-ReLU networks. It is shown by construction that Floor-ReLU networks with width $\max\{d,\, 5N + 13\}$ and depth $64dL + 3$ can pointwise approximate a Lipschitz continuous function f on $[0, 1]^d$ with an exponential approximation rate $3\mu\sqrt{d}\, N^{-\sqrt{L}}$, where μ is the Lipschitz constant of f. More generally for an arbitrary continuous function f on $[0, 1]^d$ with a modulus of continuity $\omega_f(\cdot)$, the constructive approximation rate is $\omega_f(\sqrt{d}\, N^{-\sqrt{L}}) + 2\omega_f(\sqrt{d})N^{-\sqrt{L}}$. As a consequence, this new network overcomes the curse of dimensionality in approximation power since this approximation order is essentially $\sqrt{d}$ times a function of N and L independent of d.

1 Introduction

Recently, there has been a large number of successful real-world applications of deep neural networks in many fields of computer science and engineering, especially for

large-scale and high-dimensional learning problems. Understanding the approximation capacity of deep neural networks has become a fundamental research direction for revealing the advantages of deep learning versus traditional methods. This paper introduces new theories and network architectures achieving exponential convergence and avoiding the curse of dimensionality for continuous functions for the first time in deep network approximation, which might be two foundational laws supporting the application of deep network approximation in large-scale and high-dimensional problems. The approximation theories here are quantitative and work for networks with essentially arbitrary width and depth. They would shed new light on the design of the efficient numerical implementation of deep learning.

Deep ReLU networks can achieve the approximation rate $O(N^{-L})$ for polynomials on $[0,1]^d$ (Lu et al., 2020) but it is not true for general functions, e.g., the optimal approximation rates of deep ReLU networks for continuous functions and C^s functions f on $[0,1]^d$ are $O(\sqrt{d}N^{-2/d}L^{-2/d})$ and $O(\|f\|_{C^s}N^{-2s/d}L^{-2s/d})$ (Shen et al., 2019; Lu et al., 2020), respectively. The limitation of ReLU networks motivates us to explore other types of network architectures to seek the answers to two fundamental questions: Do deep neural networks with an arbitrary width N and an arbitrary depth L admit an approximation rate $O(c(d)N^{-L})$ for general functions in a d-dimensional space? How small the function $c(d)$ in d could be?

In particular, we introduce the Floor-ReLU network, which is a fully connected neural network (FNN) built with either Floor ($\lfloor x \rfloor$) or ReLU ($\max\{0,x\}$) activation function[1] in each neuron. Mathematically, if we let $N_0 = d$, $N_{L+1} = 1$, and N_ℓ be the number of neurons in ℓ-th hidden layer of a Floor-ReLU network for $\ell = 1, 2, \cdots, L$, then the architecture of this network with input $\boldsymbol{x}$ and output $\phi(\boldsymbol{x})$ can be described as

$$\boldsymbol{x} = \widetilde{\boldsymbol{h}}_0 \xrightarrow{\boldsymbol{W}_0, \boldsymbol{b}_0} \boldsymbol{h}_1 \xrightarrow{\sigma \text{ or } \lfloor\cdot\rfloor} \widetilde{\boldsymbol{h}}_1 \quad \cdots \quad \xrightarrow{\boldsymbol{W}_{L-1}, \boldsymbol{b}_{L-1}} \boldsymbol{h}_L \xrightarrow{\sigma \text{ or } \lfloor\cdot\rfloor} \widetilde{\boldsymbol{h}}_L \xrightarrow{\boldsymbol{W}_L, \boldsymbol{b}_L} \boldsymbol{h}_{L+1} = \phi(\boldsymbol{x}),$$

where $\boldsymbol{W}_\ell \in \mathbb{R}^{N_{\ell+1} \times N_\ell}$, $\boldsymbol{b}_\ell \in \mathbb{R}^{N_{\ell+1}}$, $\boldsymbol{h}_{\ell+1} := \boldsymbol{W}_\ell \cdot \widetilde{\boldsymbol{h}}_\ell + \boldsymbol{b}_\ell$ for $\ell = 0, 1, \cdots, L$, and $\widetilde{\boldsymbol{h}}_{\ell,n}$ equals to $\sigma(\boldsymbol{h}_{\ell,n})$ or $\lfloor \boldsymbol{h}_{\ell,n} \rfloor$ for $\ell = 1, 2, \cdots, L$ and $n = 1, 2, \cdots, N_\ell$, where $\boldsymbol{h}_\ell = (\boldsymbol{h}_{\ell,1}, \cdots, \boldsymbol{h}_{\ell,N_\ell})$ and $\widetilde{\boldsymbol{h}}_\ell = (\widetilde{\boldsymbol{h}}_{\ell,1}, \cdots, \widetilde{\boldsymbol{h}}_{\ell,N_\ell})$ for $\ell = 1, 2, \cdots, L$.

In Theorem 1.1 below, we show by construction that Floor-ReLU networks with width $\max\{d, 5N+13\}$ and depth $64dL+3$ can pointwise approximate an arbitrary continuous function f on $[0,1]^d$ with an exponential approximation rate $\omega_f(\sqrt{d}\, N^{-\sqrt{L}}) + 2\omega_f(\sqrt{d})N^{-\sqrt{L}}$, where $\omega_f(\cdot)$ is the modulus of continuity defined as

$$\omega_f(r) := \sup\big\{|f(\boldsymbol{x}) - f(\boldsymbol{y})| : \|\boldsymbol{x} - \boldsymbol{y}\|_2 \le r,\ \boldsymbol{x}, \boldsymbol{y} \in [0,1]^d\big\}, \quad \text{for any } r \ge 0,$$

where $\|\boldsymbol{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_d^2}$ for any $\boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d$.

Theorem 1.1. *Given any $N, L \in \mathbb{N}^+$ and a continuous function f on $[0,1]^d$, there exists a function ϕ implemented by a Floor-ReLU network with width $\max\{d, 5N+13\}$ and depth $64dL+3$ such that*

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le \omega_f(\sqrt{d}\, N^{-\sqrt{L}}) + 2\omega_f(\sqrt{d})N^{-\sqrt{L}}, \quad \text{for any } \boldsymbol{x} \in [0,1]^d.$$

[1] Our results can be easily generalized to Ceiling-ReLU networks, namely, feed-forward neural networks with Ceiling ($\lceil x \rceil$) and ReLU ($\max\{0,x\}$) activation functions.

The rate in $\omega_f(\sqrt{d}\,N^{-\sqrt{L}})$ implicitly depends on N and L through the modulus of continuity of f while the rate in $2\omega_f(\sqrt{d})N^{-\sqrt{L}}$ is explicit and independent of f. Simplifying the implicit approximation rate to make it explicitly depending on N and L is challenging in general. However, if f is a Lipschitz continuous function on $[0,1]^d$ with a Lipschitz constant μ, then $\omega_f(r) \le \mu r$ for any $r \ge 0$. Therefore, in the case of Lipschitz continuous functions, the approximation rate is simplified to $3\mu\sqrt{d}N^{-\sqrt{L}}$ as shown in the following corollary.

Corollary 1.2. *Given any $N, L \in \mathbb{N}^+$ and a Lipschitz continuous function f on $[0,1]^d$ with a Lipschitz constant μ, there exists a function ϕ implemented by a Floor-ReLU network with width $\max\{d, 5N+13\}$ and depth $64dL+3$ such that*

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le 3\mu\sqrt{d}N^{-\sqrt{L}}, \quad \text{for any } \boldsymbol{x} \in [0,1]^d.$$

First, Theorem 1.1 and Corollary 1.2 show that the approximation capacity of deep networks for continuous functions can be exponentially improved by increasing the network depth, and the approximation error can be explicitly characterized in terms of the width $O(N)$ and depth $O(L)$. Second, this new network overcomes the curse of dimensionality in approximation power since this approximation order is essentially $\sqrt{d}$ times a function of N and L independent of d. Finally, applying piecewise constant and integer-valued functions as activation functions and integer numbers as parameters have been explored in quantized neural networks (Hubara et al., 2017; Yin et al., 2019) with efficient training algorithms for the purpose of low computational complexity (Wang et al., 2018). As we shall see in our constructive proof of Theorem 1.1, most parameters in the Floor-ReLU network are integers. Therefore, the proposed network is also attractive for efficient computation. Though there might not be an existing optimization algorithm to identify an approximant with the approximation rate in this paper, Theorem 1.1 can provide an expected accuracy before a learning task and how much the current optimization algorithms could be improved.

Characterizing deep network approximation in terms of N and L simultaneously is fundamental and indispensable in realistic applications, while quantifying the deep network approximation based on the number of nonzero parameters W is probably only of interest in theory as far as we know. Theorem 1.1 can provide practical guidance for choosing network sizes in realistic applications while theories in terms of W cannot tell how large a network should be to guarantee a target accuracy. The width and depth are two most direct and amenable hyper-parameters in choosing a specific network for a learning task, while the number of nonzero parameters W is hardly controlled efficiently. Theories in terms of W essentially have a single variable to control the network size in three types of structures: 1) fixing the width N and varying the depth L; 2) fixing the depth L and changing the width N; 3) both the width and depth are controlled by the same parameter like the target accuracy ε in a specific way (e.g., N is a polynomial of $\frac{1}{\varepsilon^d}$ and L is a polynomial of $\log(\frac{1}{\varepsilon})$). Considering the non-uniqueness of structures for realizing the same W, it is impractical to develop approximation rates in terms of W covering all these structures. If one network structure has been chosen in a certain application, there might not be a known theory in terms of W to quantify the performance of this structure.

Almost all existing approximation theories for deep neural networks so far focus on the approximation rate in W (Yarotsky, 2017; Petersen and Voigtlaender, 2018; Yarotsky, 2018; Montanelli et al., 2019; Liang and Srikant, 2016; E and Wang, 2018; Opschoor et al., 2019; Barron, 1993; Montanelli and Du, 2017; Chen and Wu, 2019; Poggio et al., 2017; Yarotsky and Zhevnerchuk, 2019; Montanelli and Yang, 2020). From the point of view of theoretical difficulty, controlling two variables N and L in our theory is more challenging than controlling one variable W in the literature. In terms of mathematical logic, the characterization of deep network approximation in terms of N and L addresses the question in terms of W, while it is not true the other way around. As we have discussed in the last paragraph, existing theories essentially have a single variable to control the network size in three types of structures. Let us use the first type of structures, which includes the best-known result for a nearly optimal approximation rate $O(W^{-2/d})$ for continuous functions in terms of W (Yarotsky, 2018), as an example to show how Theorem 1.1 in terms of N and L can be applied to show a significantly much better result in terms of W. It is similar to apply Theorem 1.1 to obtain other corollaries with other types of structures in terms of W. The main idea is to specify the value of N and L in Theorem 1.1 to show the desired corollary. For example, we let the width parameter $N = 2$ and the depth parameter $L = W$ in Theorem 1.1, then the width is $\max\{d, 23\}$, the depth is $64dW + 3$, and the total number of parameters is bounded by $\max\{d^2, 23^2\}(64dW + 3) = O(W)$. Therefore, we can prove Corollary 1.3 below stating that our Floor-ReLU network can provide an approximation accuracy of $O(\sqrt{d}2^{-\sqrt{W}})$.

Corollary 1.3. *Given any $W \in \mathbb{N}^+$ and a continuous function f on $[0,1]^d$, there exists a function ϕ implemented by a Floor-ReLU network with $\mathcal{O}(W)$ nonzero parameters, width $\max\{d, 23\}$ and depth $64dW + 3$, such that*

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le \omega_f(\sqrt{d}\,2^{-\sqrt{W}}) + 2\omega_f(\sqrt{d})2^{-\sqrt{W}}, \quad \text{for any } \boldsymbol{x} \in [0,1]^d.$$

To the best of our knowledge, the neural network constructed here is the first to achieve exponential convergence and no curse of dimensionality simultaneously for a function class as general as continuous functions, while existing theories only work for functions with an intrinsic low complexity (e.g., the exponential convergence for polynomials (Yarotsky, 2017; Montanelli et al., 2019; Lu et al., 2020), smooth functions (Montanelli et al., 2019; Liang and Srikant, 2016), analytic functions (E and Wang, 2018), functions admitting a holomorphic extension to a Bernstein polyellipse (Opschoor et al., 2019); no curse of dimensionality (or the curse is lessened) for Barron spaces (Barron, 1993), Korobov spaces (Montanelli and Du, 2017), band-limited functions (Chen and Wu, 2019; Montanelli et al., 2019), compositional functions (Poggio et al., 2017), and smooth functions (Yarotsky and Zhevnerchuk, 2019; Lu et al., 2020; Montanelli and Yang, 2020; Yang and Wang, 2020)). The prefactor in our approximation rate is a known constant of size $O(\sqrt{d})$ in the case of Lipschitz continuous functions, while the prefactor of all existing theories for much smaller function classes is unknown or grows exponentially in d. Our proof fully explores the advantage of the compositional structure and the nonlinearity of deep networks, while existing theories were built on traditional approximation tools (e.g., polynomial approximation, multiresolution analysis, and Monte Carlo sampling) making it impossible for existing theories

to obtain a theoretical breakthrough. Let us review these existing works in more detail below.

In terms of no curse of dimensionality, (Barron, 1993) and its variants in (Chen and Wu, 2019; Montanelli et al., 2019) considered d-dimensional functions with Fourier integral representations, which can be approximated by the sum of N samples of the integrant at N frequencies in the same spirit of Monte Carlo sampling by the law of large numbers with an approximation error of $O(\frac{1}{\sqrt{N}})$. Target functions in (Barron, 1993; Chen and Wu, 2019; Montanelli et al., 2019) are hence required to be sufficiently smooth and the approximation error contains a prefactor that is exponentially large in d. Similarly in (Montanelli and Du, 2017), d-dimensional functions in the Korobov space are approximated by the linear combination of basis functions of a sparse grid, each of which is approximated by a ReLU network. Though the curse of dimensionality has been lessened, target functions have to be sufficiently smooth and the approximation error contains a large prefactor that is exponential in d. Similarly, the works in (Yarotsky and Zhevnerchuk, 2019; Lu et al., 2020; Yang and Wang, 2020) take advantage of the polynomial approximation to smooth functions and ReLU networks are constructed to approximate polynomials. Generally speaking, in almost all these works, the approximation power for no curse of dimensionality essentially comes from traditional tools instead of networks.

Similarly, the approximation power for exponential approximation rate in existing works comes from traditional tools for approximating a small class of functions instead of networks. In (E and Wang, 2018; Opschoor et al., 2019; Chen and Wu, 2019; Montanelli et al., 2019), highly smooth functions are first approximated by the linear combination of special polynomials with high degrees (e.g., Chebyshev polynomials, Legendre polynomials) with an exponential approximation rate, i.e., to achieve an ε-accuracy, a linear combination of only $O(p(\log(\frac{1}{\varepsilon})))$ polynomials is required, where p is a polynomial with a degree that may depend on the dimension d. Then each polynomial is approximated by a ReLU network with $O(\log(\frac{1}{\varepsilon}))$ parameters. Finally, all ReLU networks are assembled to form a large network approximating the target function with an exponential approximation rate.

Finally, deep network approximation is in fact a special case of function approximation via compositions, where an approximant space is generated as the composition of several simple latent spaces. Function compositions can significantly enhance the approximation power. The central question is to characterize the relation of the approximant space and the latent spaces so as to design simple latent spaces to generate complex approximant spaces. This is because balancing the computational complexity and the approximation capacity is crucial in designing an efficient approximation tool in realistic applications. In terms of computational efficiency, latent spaces should remain simple and structured such that efficient numerical algorithms can be designed to identify an approximant in $\mathcal{S}$. In terms of approximation efficiency, latent spaces should be complex enough such that they can generate a wide class of functions.

The rest of this paper is organized as follows. In Section 2, we prove Theorem 1.1 based on Proposition 2.2. Next, this basic proposition is proved in Section 3. In Section 4, we introduce function approximation via compositions as a more general framework that includes deep network approximation. Finally, we conclude this paper in Section 5.

2 Approximation of continuous functions

In this section, we first introduce basic notations in this paper in Section 2.1. Then we prove the first main theorem, Theorem 1.1, based on Proposition 2.2 in Section 3.

2.1 Notations

The main notations of this paper are listed as follows.

- Let $\mathbb{N}^+$ denote the set containing all positive integers, i.e., $\mathbb{N}^+ = \{1, 2, 3, \cdots\}$.
- Let $\sigma : \mathbb{R} \to \mathbb{R}$ denote the rectified linear unit (ReLU), i.e. $\sigma(x) = \max\{0, x\}$. With the abuse of notations, we define $\sigma : \mathbb{R}^d \to \mathbb{R}^d$ as $\sigma(\boldsymbol{x}) = \begin{bmatrix} \max\{0, x_1\} \\ \vdots \\ \max\{0, x_d\} \end{bmatrix}$ for any $\boldsymbol{x} = (x_1, \cdots, x_d) \in \mathbb{R}^d$.
- For a one-dimensional function set $\Theta = \{\rho_1(x), \rho_2(x), \ldots, \rho_s(x)\}$, $\boldsymbol{\varrho} \in \Theta^d$ means $\boldsymbol{\varrho}$ is a vector function of length d with each entry as a function in Θ. With the abuse of notation, $\boldsymbol{\varrho} \odot (\boldsymbol{x})$ is used to denote $\begin{bmatrix} \rho_1(x_1) \\ \vdots \\ \rho_d(x_d) \end{bmatrix}$ with $\rho_i \in \Theta$ for $i = 1, \ldots, d$.
- The floor function (Floor) is defined as $\lfloor x \rfloor := \max\{n : n \le x,\ n \in \mathbb{Z}\}$ for any $x \in \mathbb{R}$. $\lfloor \boldsymbol{x} \rfloor$ means applying $\lfloor \cdot \rfloor$ entrywise to $\boldsymbol{x}$.
- For $\theta \in [0, 1)$, suppose its binary representation is $\theta = \sum_{\ell=1}^{\infty} \theta_\ell 2^{-\ell}$ with $\theta_\ell \in \{0, 1\}$, we introduce a special notation $\mathrm{bin} 0.\theta_1\theta_2\cdots\theta_L$ to denote the L-term binary representation of θ, i.e., $\sum_{\ell=1}^{L} \theta_\ell 2^{-\ell}$.
- The expression "a network ϕ" is short of "a function ϕ that implemented by a network".
- The expression "a network with width N and depth L" means
  - The maximum width of this network for all hidden layers is no more than N.
  - The number of hidden layers of this network is no more than L.

2.2 Proof of Theorem 1.1

The proof of Theorem 1.1 is an immediate result of Theorem 2.1 below.

Theorem 2.1. *Given any $N, L \in \mathbb{N}^+$ and a continuous function f on $[0, 1]^d$, there exists a function ϕ implemented by a Floor-ReLU network with width $\max\{d, 2N^2 + 5N\}$ and depth $7dL^2 + 3$ such that*

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le \omega_f(\sqrt{d}\, N^{-L}) + 2\omega_f(\sqrt{d}) 2^{-NL}, \quad \text{for any } \boldsymbol{x} \in [0, 1]^d.$$

This theorem will be proved later in this section. Now let us prove Theorem 1.1 based on Theorem 2.1.

Proof of Theorem 1.1. Given any $N, L \in \mathbb{N}^+$, there exist $\widetilde{N}, \widetilde{L} \in \mathbb{N}^+$ with $\widetilde{N} \geq 2$ and $\widetilde{L} \geq 3$ such that

$$(\widetilde{N}-1)^2 \leq N < \widetilde{N}^2 \quad \text{and} \quad (\widetilde{L}-1)^2 \leq 4L < \widetilde{L}^2.$$

By Theorem 2.1, there exists a function ϕ implemented by a Floor-ReLU network with width $\max\{d, 2\widetilde{N}^2+5\widetilde{N}\}$ and depth $7d\widetilde{L}^2+3$ such that

$$|\phi(\boldsymbol{x})-f(\boldsymbol{x})| \leq \omega_f(\sqrt{d}\,\widetilde{N}^{-\widetilde{L}}) + 2\omega_f(\sqrt{d})2^{-\widetilde{N}\widetilde{L}}, \quad \text{for any } \boldsymbol{x} \in [0,1]^d.$$

Note that

$$2^{-\widetilde{N}\widetilde{L}} \leq \widetilde{N}^{-\widetilde{L}} = (\widetilde{N}^2)^{-\frac{1}{2}\sqrt{\widetilde{L}^2}} \leq N^{-\frac{1}{2}\sqrt{4L}} \leq N^{-\sqrt{L}}.$$

Then we have

$$|\phi(\boldsymbol{x})-f(\boldsymbol{x})| \leq \omega_f(\sqrt{d}\,N^{-\sqrt{L}}) + 2\omega_f(\sqrt{d})N^{-\sqrt{L}}, \quad \text{for any } \boldsymbol{x} \in [0,1]^d.$$

For $\widetilde{N}, \widetilde{L} \in \mathbb{N}^+$ with $\widetilde{N} \geq 2$ and $\widetilde{L} \geq 3$, we have

$$2\widetilde{N}^2+\widetilde{N} \leq 5(\widetilde{N}-1)^2+13 \leq 5N+13 \quad \text{and} \quad 7\widetilde{L}^2 \leq 16(\widetilde{L}-1)^2 \leq 64L.$$

Therefore, ϕ can be computed by a Floor-ReLU network with width $\max\{d, 2\widetilde{N}^2+5\widetilde{N}\} \leq \max\{d, 5N+13\}$ and depth $7d\widetilde{L}^2+3 \leq 64dL+3$, as desired. So we finish the proof. □

To prove Theorem 2.1, we first present the proof sketch. Shortly speaking, we construct piecewise constant functions implemented by Floor-ReLU networks to approximate continuous functions. There are six key steps in our construction.

1. Normalize f as $\widetilde{f}$ satisfying $\widetilde{f}(\boldsymbol{x}) \in [0,1]$ for any $\boldsymbol{x} \in [0,1]^d$ and divide $[0,1]^d$ into a set of non-overlapping cubes $\{Q_{\boldsymbol{\alpha}}\}_{\boldsymbol{\alpha} \in \{0,1,\cdots,M-1\}^d}$, where M is an integer determined later.

2. Construct a vector-valued function $\boldsymbol{\Phi}_1 : \mathbb{R}^d \to \mathbb{R}^d$ mapping $\boldsymbol{x} \in Q_{\boldsymbol{\alpha}}$ to the index $\boldsymbol{\alpha}$ for each $\boldsymbol{\alpha} \in \{0,1,\cdots,M-1\}^d$.

3. Construct a function $\phi_2 : \mathbb{R}^d \to \mathbb{R}$ projecting $\boldsymbol{\alpha} \in \{0,1,\cdots,M-1\}^d$ to $\phi_2(\boldsymbol{\alpha}) \in \{1,2,\cdots,M^d\}$.

4. Construct a function $\phi_3 : \mathbb{R} \to \mathbb{R}$ mapping $\phi_2(\boldsymbol{\alpha}) \in \{1,2,\cdots,M^d\}$ to $\phi_3(\phi_2(\boldsymbol{\alpha})) \approx \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\alpha}})$, where $\boldsymbol{x}_{\boldsymbol{\alpha}}$ is a pre-specified point of $Q_{\boldsymbol{\alpha}}$.

5. Define $\widetilde{\phi} \coloneqq \phi_3 \circ \phi_2 \circ \boldsymbol{\Phi}_1$. Then $\widetilde{\phi}$ is a piecewise constant function mapping $\boldsymbol{x} \in Q_{\boldsymbol{\alpha}}$ to $\phi_3(\phi_2(\boldsymbol{\alpha})) \approx \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\alpha}})$.

6. Re-scale and shift $\widetilde{\phi}$ to obtain the final function ϕ approximating f well.

It is not difficult to construct Floor-ReLU networks with the desired width and depth to implement $\mathbf{\Phi}_1$ and ϕ_2. The most technical part is the construction of a Floor-ReLU network with the desired width and depth computing ϕ_3, which needs the following proposition based on the "bit extraction" technique introduced in (Bartlett et al., 1998; Harvey et al., 2017).

Proposition 2.2. *Given any $N, L \in \mathbb{N}^+$ and arbitrary $\theta_m \in \{0,1\}$ for $m = 1, 2, \cdots, N^L$, there exists a function ϕ computed by a Floor-ReLU network with width $2N+2$ and depth $7L-2$ such that*

$$\phi(m) = \theta_m, \quad \text{for } m = 1, 2, \cdots, N^L.$$

The proof of this proposition is presented in Section 3. By this proposition and the definition of VC-dimension (e.g., see (Harvey et al., 2017)), it is easy to prove that the VC-dimension of Floor-ReLU networks with constant width and depth $\mathcal{O}(L)$ has a lower bound 2^L. Such a lower bound is much larger than $\mathcal{O}(L^2)$, which is a VC-dimension upper bound of ReLU networks with the same width and depth due to Theorem 8 of (Harvey et al., 2017). This means Floor-ReLU networks are much more powerful than ReLU networks from the perspective of VC-dimension.

Based on the proof sketch stated just above, we are ready to give the detailed proof of Theorem 2.1 as follows.

Proof of Theorem 2.1. Assume f is not a constant function since it is a trivial case. Then $\omega_f(r) > 0$ for any $r > 0$. Clearly, $|f(\boldsymbol{x}) - f(\mathbf{0})| \le \omega_f(\sqrt{d})$ for any $\boldsymbol{x} \in [0,1]^d$. Define

$$\widetilde{f} := \big(f - f(\mathbf{0}) + \omega_f(\sqrt{d})\big)\big/\big(2\omega_f(\sqrt{d})\big). \tag{1}$$

It follows that $\widetilde{f}(\boldsymbol{x}) \in [0,1]$ for any $\boldsymbol{x} \in [0,1]^d$.

Set $M = N^L$, $E_{M-1} = [\frac{M-1}{M}, 1]$, and $E_m = [\frac{m}{M}, \frac{m+1}{M})$ for $m = 0, 1, \cdots, M-2$. Define a step function ϕ_1 as

$$\phi_1(t) := \big\lfloor -\sigma(-Mt + M - 1) + M - 1 \big\rfloor, \quad \text{for any } t \in \mathbb{R}.^{2}$$

See Figure 1 for an example of ϕ_1. It follows from the definition of ϕ_1 that

$$\phi_1(t) = m, \quad \text{if } t \in E_m, \text{ for } m = 0, 1, \cdots, M-1.$$

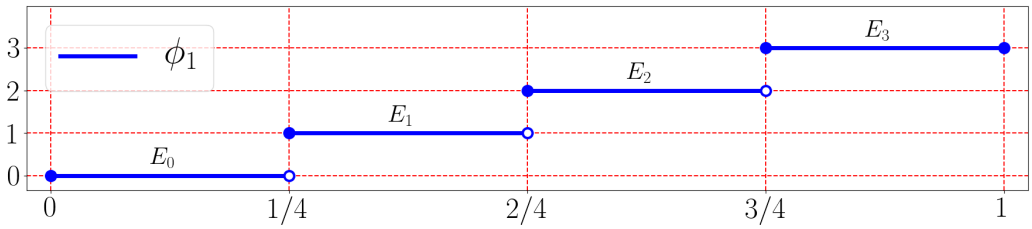


Figure 1: An illustration of ϕ_1 on $[0,1]$ for $M = 4$.

[2] If we just define $\phi_1(t) = \lfloor Mt \rfloor$, then $\phi_1(1) = M \neq M - 1$ even though $1 \in E_{M-1}$.

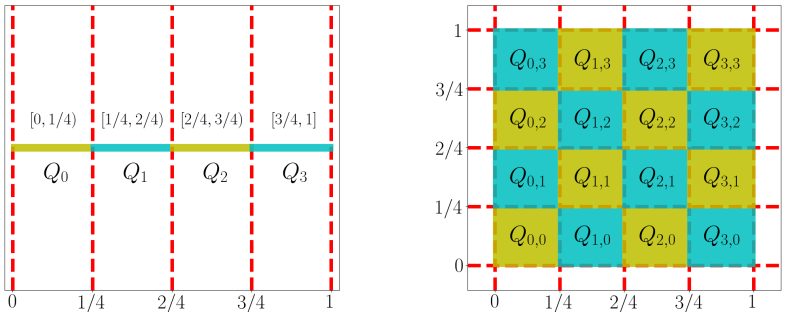
Define

$$\boldsymbol{\Phi}_1(\boldsymbol{x}) \coloneqq \big(\phi_1(x_1), \phi_1(x_2), \cdots, \phi_1(x_d)\big), \quad \text{for any } \boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d,$$

and

$$Q_{\boldsymbol{\alpha}} \coloneqq \Big\{\boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d : x_j \in E_{\alpha_j} \text{ for } j = 1, 2, \cdots, d\Big\},$$

for any $\boldsymbol{\alpha} = (\alpha_1, \alpha_2 \cdots, \alpha_d) \in \{0, 1, \cdots, M-1\}^d$. See Figure 2 for the examples of $Q_{\boldsymbol{\alpha}}$, $\boldsymbol{\alpha} \in \{0, 1, \cdots, M-1\}^d$, for $M = 4$ and $d = 1, 2$.



(a) (b)

Figure 2: Illustrations of $Q_{\boldsymbol{\alpha}}$ for $\boldsymbol{\alpha} \in \{0, 1, \cdots, M-1\}^d$. (a) $M = 4$, $d = 1$. (b) $M = 4$, $d = 2$.

Clearly, we have, for $\boldsymbol{x} \in Q_{\boldsymbol{\alpha}}$ and $\boldsymbol{\alpha} \in \{0, 1, \cdots, M-1\}^d$,

$$\boldsymbol{\Phi}_1(\boldsymbol{x}) = \big(\phi_1(x_1), \phi_1(x_2), \cdots, \phi_1(x_d)\big) = (\alpha_1, \alpha_2, \cdots, \alpha_d) = \boldsymbol{\alpha}.$$

Using the idea of M-ary representation, we define a projection function ϕ_2 via

$$\phi_2(\boldsymbol{y}) \coloneqq 1 + \sum_{j=1}^{d} y_j M^{j-1}, \quad \text{for any } \boldsymbol{y} = (y_1, y_2, \cdots, y_d) \in \mathbb{R}^d.$$

Then ϕ_2 is a bijection (one-to-one correspondence) from $\{0, 1, \cdots, M-1\}^d$ to $\{1, 2, \cdots, M^d\}$.

Given any $i \in \{1, 2, \cdots, M^d\}$, there exists a unique $\boldsymbol{\alpha} \in \{0, 1, \cdots, M-1\}^d$ such that $i = \phi_2(\boldsymbol{\alpha})$. Then define

$$\xi_i \coloneqq \widetilde{f}\big(\tfrac{\boldsymbol{\alpha}}{M}\big) \in [0, 1], \quad \text{for } i = \phi_2(\boldsymbol{\alpha}) \text{ and } \boldsymbol{\alpha} \in \{0, 1, \cdots, M-1\}^d,$$

where $\widetilde{f}$ is the normalization of f defined in Equation (1). It follows that there exists $\xi_{i,\ell} \in \{0, 1\}$ for $\ell = 1, 2, \cdots, NL$ such that

$$|\xi_i - \mathrm{bin}\, 0.\xi_{i,1}\xi_{i,2}\cdots, \xi_{i,NL}| \le 2^{-NL}, \quad \text{for } i = 1, 2, \cdots, M^d.$$

By $M^d = (N^L)^d = N^{dL}$ and Proposition 2.2, there exists a function $\phi_{3,\ell}$ computed by a Floor-ReLU network with width $2N+2$ and depth $7dL-2$, for each $\ell = 1, 2, \cdots, NL$, such that

$$\phi_{3,\ell}(i) = \xi_{i,\ell}, \quad \text{for } i = 1, 2, \cdots, M^d.$$

By defining $\phi_3 \coloneqq \sum_{\ell=1}^{NL} 2^{-\ell}\phi_{3,\ell}$, we have, for $i = \phi_2(\boldsymbol{\alpha})$ and $\boldsymbol{\alpha} \in \{0, 1, \cdots, M-1\}^d$,

$$\begin{aligned} |\widetilde{f}(\tfrac{\boldsymbol{\alpha}}{M}) - \phi_3(\phi_2(\boldsymbol{\alpha}))| = |\xi_i - \phi_3(i)| &= |\xi_i - \sum_{\ell=1}^{NL} 2^{-\ell}\phi_{3,\ell}(i)| \\ &= |\xi_i - \text{bin}\, 0.\xi_{i,1}\xi_{i,2}\cdots\xi_{i,NL}| \le 2^{-NL}. \end{aligned} \tag{2}$$

Define $\widetilde{\phi} \coloneqq \phi_3 \circ \phi_2 \circ \boldsymbol{\Phi}_1$, i.e., for any $\boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d$,

$$\widetilde{\phi}(\boldsymbol{x}) = \phi_3 \circ \phi_2 \circ \boldsymbol{\Phi}_1(\boldsymbol{x}) = \phi_3\Big(\phi_2\big(\phi_1(x_1), \phi_1(x_2), \cdots, \phi_1(x_d)\big)\Big).$$

Note that $|\boldsymbol{x} - \frac{\boldsymbol{\alpha}}{M}| \le \frac{\sqrt{d}}{M}$ for any $\boldsymbol{x} \in Q_{\boldsymbol{\alpha}}$ and $\boldsymbol{\alpha} \in \{0, 1, \cdots, M-1\}^d$. Then we have, for any $\boldsymbol{x} \in Q_{\boldsymbol{\alpha}}$ and $\boldsymbol{\alpha} \in \{0, 1, \cdots, M-1\}^d$,

$$\begin{aligned} |\widetilde{f}(\boldsymbol{x}) - \widetilde{\phi}(\boldsymbol{x})| &\le |\widetilde{f}(\boldsymbol{x}) - \widetilde{f}(\tfrac{\boldsymbol{\alpha}}{M})| + |\widetilde{f}(\tfrac{\boldsymbol{\alpha}}{M}) - \widetilde{\phi}(\boldsymbol{x})| \\ &\le \omega_{\widetilde{f}}(\tfrac{\sqrt{d}}{M}) + |\widetilde{f}(\tfrac{\boldsymbol{\alpha}}{M}) - \phi_3(\phi_2(\boldsymbol{\Phi}_1(\boldsymbol{x})))| \\ &\le \omega_{\widetilde{f}}(\tfrac{\sqrt{d}}{M}) + |\widetilde{f}(\tfrac{\boldsymbol{\alpha}}{M}) - \phi_3(\phi_2(\boldsymbol{\alpha}))| \le \omega_{\widetilde{f}}(\tfrac{\sqrt{d}}{M}) + 2^{-NL}, \end{aligned}$$

where the last inequality comes form Equation (2).

Note $\boldsymbol{x} \in Q_{\boldsymbol{\alpha}}$ and $\boldsymbol{\alpha} \in \{0, 1, \cdots, M-1\}^d$ are arbitrary. Since $[0,1]^d = \cup_{\boldsymbol{\alpha}\in\{0,1,\cdots,M-1\}^d} Q_{\boldsymbol{\alpha}}$, we have

$$|\widetilde{f}(\boldsymbol{x}) - \widetilde{\phi}(\boldsymbol{x})| \le \omega_{\widetilde{f}}(\tfrac{\sqrt{d}}{M}) + 2^{-NL}, \quad \text{for any } \boldsymbol{x} \in [0,1]^d.$$

Define $\phi \coloneqq 2\omega_f(\sqrt{d})\widetilde{\phi} + f(\boldsymbol{0}) - \omega_f(\sqrt{d})$. By $M = N^L$ and $\omega_f(r) = 2\omega_f(\sqrt{d}) \cdot \omega_{\widetilde{f}}(r)$ for any $r \ge 0$, we have, for any $\boldsymbol{x} \in [0,1]^d$,

$$\begin{aligned} |f(\boldsymbol{x}) - \phi(\boldsymbol{x})| = 2\omega_f(\sqrt{d})\big|\widetilde{f}(\boldsymbol{x}) - \widetilde{\phi}(\boldsymbol{x})\big| &\le 2\omega_f(\sqrt{d})\Big(\omega_{\widetilde{f}}(\tfrac{\sqrt{d}}{M}) + 2^{-NL}\Big) \\ &\le \omega_f(\tfrac{\sqrt{d}}{M}) + 2\omega_f(\sqrt{d})2^{-NL} \\ &\le \omega_f(\sqrt{d}\,N^{-L}) + 2\omega_f(\sqrt{d})2^{-NL}. \end{aligned}$$

Since ϕ is defined via re-scaling and shifting $\widetilde{\phi}$, what remains is to determine the width and depth of the Floor-ReLU network computing $\widetilde{\phi}$. Clearly, ϕ_3 can be implemented by the architecture in Figure 3.

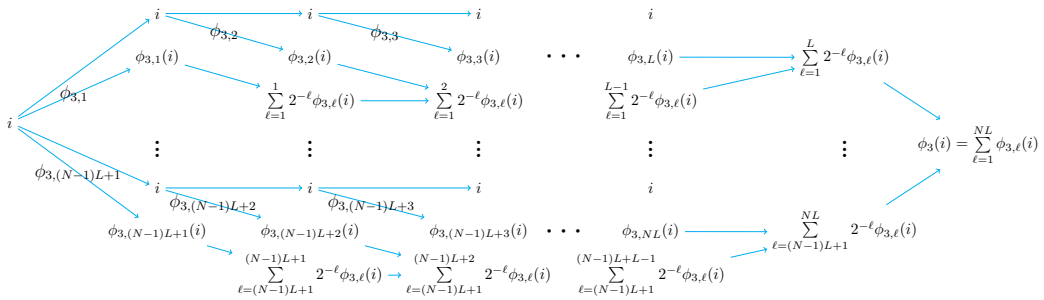


Figure 3: An illustration of the desired network architecture computing ϕ_3. We omit some ReLU (σ) activation functions if inputs are obviously non-negative.

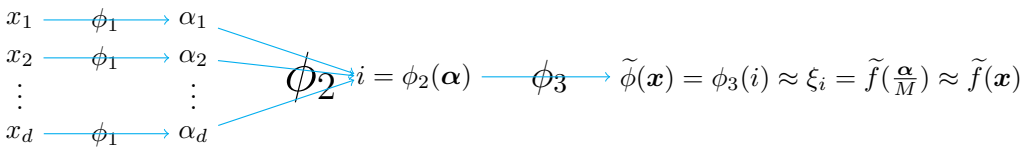


Figure 4: An illustration of the desired network architecture computing $\widetilde{\phi}$, where $\widetilde{\phi}(\boldsymbol{x}) = \phi_3 \circ \phi_2 \circ \boldsymbol{\Phi}_1(\boldsymbol{x})$ for any $\boldsymbol{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$. We omit some ReLU ($\sigma$) activation functions if inputs are obviously non-negative.

As we can see from Figure 3, ϕ_3 can be computed by a Floor-ReLU network with width $N(2N+2+3) = 2N^2+5N$ and depth $L(7dL-2+1)+1 = L(7dL-1)+1$. With the network architecture computing ϕ_3 in hand, $\widetilde{\phi}$ can be implemented by the network architecture shown in Figure 4.

By Figure 4, ϕ and $\widetilde{\phi}$ can be implemented by a Floor-ReLU network with width $\max\{d,\, 2N^2+5N\}$ and depth $L(7dL-1)+1+3 \le 7dL^2+3$. So we finish the proof. □

3 Proof of Proposition 2.2

The proof of Proposition 2.2 mainly relies on the "bit extraction" technique. As we shall see later, our key idea is to apply the Floor activation function to make "bit extraction" more powerful to reduce network sizes. In particular, Floor-ReLU networks can extract much more bits than ReLU networks with the same network size.

Let us first establish a basic lemma to extract $1/N$ of total bits stored in a new binary number from an input binary number.

Lemma 3.1. *Given any $J, N \in \mathbb{N}^+$, there exists a function $\phi : \mathbb{R}^2 \to \mathbb{R}$ that can be implemented by a Floor-ReLU network with width $2N$ and depth 4 such that, for any $\theta_j \in \{0,1\}$, $j = 1, \cdots, NJ$, we have*

$$\phi(\text{bin}\,0.\theta_1 \cdots \theta_{NJ},\, n) = \text{bin}\,0.\theta_{(n-1)J+1} \cdots \theta_{nJ}, \quad \text{for } n = 1, 2, \cdots, N.$$

Proof. Given any $\theta_j \in \{0,1\}$ for $j = 1, \cdots, NJ$, denote

$$s = \text{bin}\,0.\theta_1 \cdots \theta_{NJ} \quad \text{and} \quad s_n = \text{bin}\,0.\theta_{(n-1)J+1} \cdots \theta_{nJ}, \quad \text{for } n = 1, 2, \cdots, N.$$

Then our goal is to construct a function $\phi : \mathbb{R}^2 \to \mathbb{R}$ computed by a Floor-ReLU network with the desired width and depth that satisfies

$$\phi(s, n) = s_n, \quad \text{for } n = 1, 2, \cdots, N.$$

Based on the properties of the binary representation, it is easy to check that

$$s_n = \left\lfloor 2^{nJ} s \right\rfloor \Big/ 2^{(n-1)J} - \left\lfloor 2^{(n-1)J} s \right\rfloor, \quad \text{for } n = 1, 2, \cdots, N. \tag{3}$$

With formulas to return $s_1, s_2, \cdots, s_N$, it is still technical to construct a network outputting s_n for a given index $n \in \{1, 2, \cdots, N\}$.

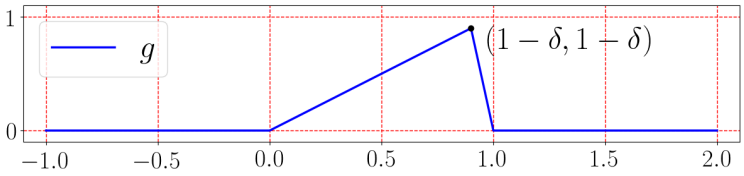


Figure 5: An illustration of $g(x) = \sigma\Big(\sigma(x) - \sigma\big(\frac{x+\delta-1}{\delta}\big)\Big)$, where $\sigma(x) = \max\{0, x\}$.

Set $\delta = 2^{-NJ}$ and define g (see Figure 5) as

$$g(x) \coloneqq \sigma\Big(\sigma(x) - \sigma\big(\tfrac{x+\delta-1}{\delta}\big)\Big), \quad \text{where } \sigma(x) = \max\{0, x\}.$$

Since $s_n \in [0, 1-\delta]$ for $n = 1, 2, \cdots, N$, we have

$$s_n = \sum_{k=1}^{N} g(s_k + k - n), \quad \text{for } n = 1, 2, \cdots, N. \tag{4}$$

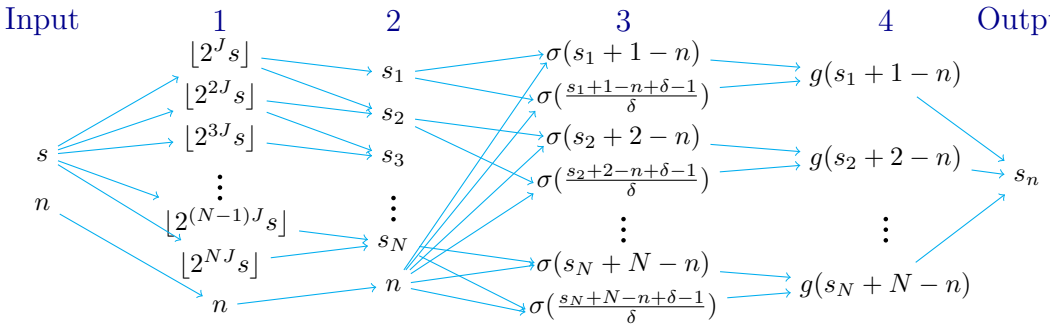


Figure 6: This architecture is based on Equation (3) and (4). We omit some ReLU (σ) activation functions provided inputs are obviously non-negative.

As shown in Figure 6, the desired function ϕ can be computed by a Floor-ReLU network with width $2N$ and depth 4. Moreover, it holds that

$$\phi(s, n) = s_n, \quad \text{for } n = 1, 2, \cdots, N.$$

So we finish the proof. □

The next lemma constructs a Floor-ReLU network that can extract any bit from a binary number according to a specific index.

Lemma 3.2. *Given any $N, L \in \mathbb{N}^+$, there exists a function $\phi : \mathbb{R}^2 \to \mathbb{R}$ implemented by a Floor-ReLU network with width $2N + 2$ and depth $7L - 3$ such that, for any $\theta_m \in \{0, 1\}$, $m = 1, 2, \cdots, N^L$, we have*

$$\phi(\mathrm{bin}\, 0.\theta_1\theta_2\cdots\theta_{N^L}, m) = \theta_m, \quad \text{for } m = 1, 2, \cdots, N^L.$$

Proof. The proof is based on repeated applications of Lemma 3.1. To be exact, we construct a sequence of functions $\phi_1, \phi_2, \cdots, \phi_L$ implemented by Floor-ReLU networks by induction to satisfy the following two conditions for each $\ell \in \{1, 2, \cdots, L\}$.

(i) $\phi_\ell : \mathbb{R}^2 \to \mathbb{R}$ can be implemented by a Floor-ReLU network with width $2N + 2$ and depth $7\ell - 3$.

(ii) For any $\theta_m \in \{0, 1\}$, $m = 1, 2, \cdots, N^\ell$, we have

$$\phi_\ell(\mathrm{bin}0.\theta_1\theta_2\cdots\theta_{N^\ell},\, m) = \mathrm{bin}0.\theta_m, \quad \text{for } m = 1, 2, \cdots, N^\ell.$$

Firstly, consider the case $\ell = 1$. By Lemma 3.1 (set $J = 1$ therein), there exists a function ϕ_1 implemented by a Floor-ReLU network with width $4 \le 6$ and depth $4 = 7 - 3$ such that, for any $\theta_m \in \{0, 1\}$, $m = 1, 2, \cdots, N$, we have

$$\phi_1(\mathrm{bin}0.\theta_1\theta_2\cdots\theta_N,\, m) = \mathrm{bin}0.\theta_m, \quad \text{for } m = 1, 2, \cdots, N.$$

It follows that Condition (i) and (ii) hold for $\ell = 1$.

Next, assume Condition (i) and (ii) hold for $\ell = k$. We would like to construct ϕ_{k+1} to make Condition (i) and (ii) true for $\ell = k + 1$. By Lemma 3.1 (set $J = N^k$ therein), there exists a function ψ implemented by a Floor-ReLU network with width $2N + 2$ and depth 4 such that, for any $\theta_m \in \{0, 1\}$, $m = 1, 2, \cdots, N^k$, we have

$$\psi(\mathrm{bin}0.\theta_1\cdots\theta_{N^{k+1}},\, n) = \mathrm{bin}0.\theta_{(n-1)N^k+1}\cdots\theta_{(n-1)N^k+N^k}, \quad \text{for } n = 1, 2, \cdots, N. \tag{5}$$

By the hypothesis of induction, we have

- $\phi_k : \mathbb{R}^2 \to \mathbb{R}$ can be implemented by a Floor-ReLU network with width $2N + 2$ and depth $7k - 3$.
- For any $\theta_j \in \{0, 1\}$, $j = 1, 2, \cdots, N^k$, we have

$$\phi_k(\mathrm{bin}0.\theta_1\theta_2\cdots\theta_{N^k},\, j) = \mathrm{bin}0.\theta_j, \quad \text{for } j = 1, 2, \cdots, N^k. \tag{6}$$

Given any $m \in \{1, 2, \cdots, N^{k+1}\}$, there exist $n \in \{1, 2, \cdots, N\}$ and $j \in \{1, 2, \cdots, N^k\}$ such that $m = (n - 1)N^k + j$, and such k, j can be obtained by

$$n = \lfloor (m - 1)/N^k \rfloor + 1 \quad \text{and} \quad j = m - (n - 1)N^k. \tag{7}$$

Then the desired architecture of the Floor-ReLU network implementing ϕ_{k+1} is shown in Figure 7.

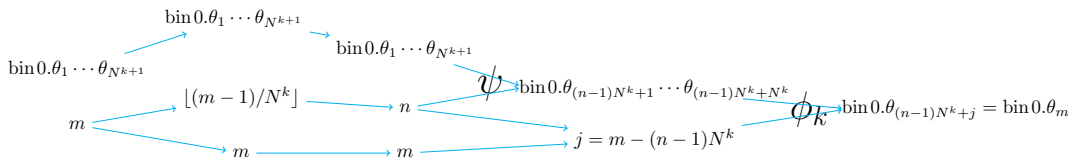


Figure 7: This architecture is based on Equation (5), (6), and (7). We omit some ReLU (σ) activation functions provided inputs are obviously non-negative.

Note that ψ can be computed by a Floor-ReLU network with width $2N$ and depth 4. By Figure 7, we have

- $\phi_{k+1} : \mathbb{R}^2 \to \mathbb{R}$ can be implemented by a Floor-ReLU network with width $2N + 2$ and depth $7 + (7k - 3) = 7(k + 1) - 3$, which implies Condition (i) for $\ell = k + 1$.
- For any $\theta_m \in \{0, 1\}$, $m = 1, 2, \cdots, N^{k+1}$, we have

$$\phi_{k+1}(\text{bin}0.\theta_1\theta_2\cdots\theta_{N^{k+1}}, m) = \text{bin}0.\theta_m, \quad \text{for } m = 1, 2, \cdots, N^{k+1}.$$

  That is, Condition (ii) holds for $\ell = k + 1$.

So we finish the process of induction.

By the principle of induction, there exists a function $\phi_L : \mathbb{R}^2 \to \mathbb{R}$ such that

- ϕ_L can be implemented by a Floor-ReLU network with width $2N + 2$ and depth $7L - 3$.
- For any $\theta_m \in \{0, 1\}$, $m = 1, 2, \cdots, N^L$, we have

$$\phi_L(\text{bin}0.\theta_1\theta_2\cdots\theta_{2^L}, m) = \text{bin}0.\theta_m, \quad \text{for } m = 1, 2, \cdots, N^L.$$

Finally, define $\phi \coloneqq 2\phi_L$. Then ϕ can also be implemented by a Floor-ReLU network with width $2N + 2$ and depth $7L - 3$. Moreover, for any $\theta_m \in \{0, 1\}$, $m = 1, 2, \cdots, N^L$, we have

$$\phi(\text{bin}0.\theta_1\theta_2\cdots\theta_{N^L}, m) = 2 \cdot \phi_L(\text{bin}0.\theta_1\theta_2\cdots\theta_{N^L}, m) = 2 \cdot \text{bin}0.\theta_m = \theta_m,$$

for $m = 1, 2, \cdots, N^L$. So we finish the proof. □

With Lemma 3.2 in hand, we are ready to prove Proposition 2.2.

Proof of Proposition 2.2. By Lemma 3.2, there exists a function $\widetilde{\phi} : \mathbb{R}^2 \to \mathbb{R}$ computed by a Floor-ReLU network with a fixed architecture with width $2N + 2$ and depth $7L - 3$ such that, for any $z_m \in \{0, 1\}$, $m = 1, 2, \cdots, N^L$, we have

$$\widetilde{\phi}(\text{bin}0.z_1z_2\cdots z_{N^L}, m) = z_m, \quad \text{for } m = 1, 2, \cdots, N^L.$$

Based on $\theta_m \in \{0, 1\}$ for $m = 1, 2, \cdots, N^L$ given in Proposition 2.2, we define the final function ϕ as

$$\phi(x) \coloneqq \widetilde{\phi}\big(\sigma(x \cdot 0 + \text{bin}0.\theta_1\theta_2\cdots\theta_{N^L}), \sigma(x)\big), \quad \text{where } \sigma(x) = \max\{0, x\}.$$

Clearly, ϕ can be implemented by a Floor-ReLU network with width $2N + 2$ and depth $(7L - 3) + 1 = 7L - 2$. Moreover, we have, for any $m \in \{1, 2, \cdots, N^L\}$,

$$\phi(m) \coloneqq \widetilde{\phi}\big(\sigma(m \cdot 0 + \text{bin}0.\theta_1\theta_2\cdots\theta_{N^L}), \sigma(m)\big) = \widetilde{\phi}(\text{bin}0.\theta_1\theta_2\cdots\theta_{N^L}, m) = \theta_m.$$

So we finish the proof. □

We shall point out that only the properties of Floor on $[0, \infty)$ are used in our proof. Thus, the Floor can be replaced by the truncation function that can be easily computed by truncating the decimal part.

4 Approximation via compositions

In this section, we will discuss function compositions for approximation and its a few examples including deep network approximation. Let us first formulate the problem of function approximation via compositions, where an approximant space $\mathcal{S}$ is generated by the composition of latent spaces $\mathcal{S}_\ell$ as follows.

Definition 4.1. Suppose $N_0 = d$, $N_{L+1} = k$, $N_\ell \in \mathbb{N}_+$, and $\mathcal{S}_\ell$ is a space of functions from $\mathbb{R}^{N_{\ell-1}}$ to $\mathbb{R}^{N_\ell}$ for $\ell = 1, \ldots, L+1$. Let

$$\mathcal{S} := \left\{ \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{h}_{L+1} \circ \boldsymbol{h}_L \circ \cdots \circ \boldsymbol{h}_1(\boldsymbol{x}) : \boldsymbol{h}_\ell \in \mathcal{S}_\ell \text{ for } \ell = 1, \ldots, L+1 \right\},$$

then $\mathcal{S}$ is called the approximant space generated by latent spaces $\mathcal{S}_\ell$ for $\ell = 1, \ldots, L+1$ with L compositions.

A best approximant of $\boldsymbol{f}(\boldsymbol{x})$ in $\mathcal{S}$ is identified by solving

$$\boldsymbol{\phi}(\boldsymbol{x}) = \underset{\boldsymbol{\phi}(\boldsymbol{x}) \in \mathcal{S}}{\arg\min} \| \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{\phi}(\boldsymbol{x}) \|_*, \tag{8}$$

where $\| \cdot \|_*$ is an appropriate norm depending on applications. Function compositions can significantly enhance the approximation power. Composing a fixed latent space several times could generate a much richer approximant space. However, this idea was not considered in the literature previously due to the expensive computation in solving the optimization problem (8). Deep learning and its related optimization algorithms (e.g., backpropagation techniques (Werbos, 1975; Fukushima, 1980; Rumelhart et al., 1986), parallel computing techniques (Scherer et al., 2010; Cireşan et al., 2011), and stochastic algorithms (Duchi et al., 2011; Johnson and Zhang, 2013)) indicate that solving Equation (8) has become feasible and hence compositions can be a practical choice for function approximation.

4.1 Classical approximation

Though function approximation via compositions is relatively new, most existing approximation techniques can be considered as its special cases.

Linear approximation

Let us first discuss the linear approximation through the lens of approximation via compositions. Linear approximation is an efficient approximation tool for smooth functions that computes the approximant of a target function via a linear projection to a Hilbert space or a Banach space as the approximant space. The linear projection can be computed efficiently through the orthogonality of basis functions in the Hilbert space or via (quasi) interpolation in the Banach space. Typically, target functions are required to be sufficiently smooth to obtain a good numerical approximation via the projection to a fixed set of finitely many (e.g., N) basis functions. Approximation theories and numerical tools have been well-developed in linear approximation, e.g. polynomial approximations, Fourier analysis, finite element approximation, spline approximation, etc. They have become powerful tools for approximating smooth functions.

Linear approximation can be considered as a special case of function approximation via compositions when $L = 1$, $k = 1$, and $N_1 = N$. For simplicity, let us take the one-dimensional orthogonal polynomial approximation as an example. The approximant space $\mathcal{S}$, in this case, is generated by

$$\mathcal{S}_1 = \left\{ \boldsymbol{\varrho}(x) = [\varrho_0(x), \varrho_1(x), \ldots, \varrho_{N-1}(x)]^T \right\},$$

and

$$\mathcal{S}_2 = \left\{ f(\boldsymbol{x}) = \boldsymbol{W} \cdot \boldsymbol{x} : \boldsymbol{W} \in \mathbb{R}^N \right\}, \tag{9}$$

where $\{\varrho_i(x)\}_{i=0}^{N-1}$ is a set of orthogonal polynomials with degrees from zero to $N-1$. The computation between a target function and an approximant is highly efficient due to the orthogonality.

Similarly, when the target function space is $L^2(\mathbb{R}^d)$ and $\mathcal{S}_1$ consists of a vector function of length N with entries as basis functions in the Fourier series of $\mathbb{R}^d$. N here restricts approximants to the first N-term Fourier series expansion. Fourier basis functions admit a good structure, orthogonality, which makes it simple to compute an approximant in $\mathcal{S}$ to approximate a target function f and to reconstruct f from its representation.

In general, the computational complexity of tools in the case of $L = 1$ could be as low as nearly optimal, e.g., $O(N)$ operations ignoring a logarithm factor. Nevertheless, their approximation capacity suffers from the curse of dimensionality and there is no exponential approximation rate for general continuous functions. The approximation accuracy of $\mathcal{S}$ when $L = 1$ and $N_1 = N$ is usually $O(N^{-1/d})$ for a continuous function, which is far from $O(N^{-\sqrt{L}})$ when d is large.

Nonlinear approximation

Nonlinear approximation (DeVore, 1998) has become a popular technique in recent decades for piecewise-smooth function approximation. A typical algorithm in nonlinear approximation is to design a highly redundant nonlinear dictionary, $\mathcal{D}$, and to identify the optimal approximant as a linear combination of N elements of $\mathcal{D}$. Given a dictionary $\mathcal{D}$ and a target function $f(\boldsymbol{x})$, nonlinear approximation seeks $\{g_n\}$ and $\{T_n\}$ such that

$$\{\{T_n\}, \{g_n\}\} = \underset{\{g_n\}\subseteq\mathbb{R}, \{T_n\}\subseteq\mathcal{D}}{\arg\min} \|f(\boldsymbol{x}) - \sum_{n=1}^{N} g_n T_n(\boldsymbol{x})\|_*, \tag{10}$$

which is also called the best N-term approximation with an appropriate norm $\|\cdot\|_*$.

Traditional dictionaries in nonlinear approximation can be considered as a special case of function approximation via compositions when $L = 1$, $k = 1$, and $N_1 = N$. Wavelet frames of $L^2([0,1]^d)$ built with the dilation and translation of a mother wavelet can serve as a typical example. The approximant space $\mathcal{S}$ of wavelet frames can be generated by latent spaces

$$\mathcal{S}_1 = \left\{ \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{\varrho} \odot (\boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b}) : \boldsymbol{W} \in \mathbb{R}^{N\times d}, \boldsymbol{b} \in \mathbb{R}^N, \boldsymbol{\varrho} \in \Theta^N, 1 \le i \le N \right\},$$

and

$$\mathcal{S}_2 = \left\{ h(\boldsymbol{x}) = \boldsymbol{W} \cdot \boldsymbol{x} : \boldsymbol{W} \in \mathbb{R}^N \right\},$$

where $\Theta = \{\varrho(x)\}$ with $\varrho(x)$ as a mother wavelet, $\boldsymbol{W}$ corresponds to dilation, and $\boldsymbol{b}$ determines translation. Though wavelet frames are redundant and hence there is no orthogonality, under the assumption that target functions have a sparse approximant, it is

still computationally efficient to determine a best approximant in $\mathcal{S}$ and reconstruct a target function via the ℓ_1-regularization to Equation (10) in realistic applications. Similar to the case of linear approximation, the approximation capacity of traditional nonlinear approximation suffers from the curse of dimensionality and there is no exponential approximation rate for general continuous functions, e.g. typically $O(N^{-1/d})$ for a continuous function.

Nonlinear approximation via compositions proposed in (Shen et al., 2019) can provide a more attractive approximation rate. The key idea is to use function compositions to generate a dictionary in nonlinear approximation. For example, in the case when $N_L = N$, $k = 1$, and $\mathcal{S}_{L+1}$ is defined by Equation (9), seeking a best function approximation via compositions by solving Equation (8) is equivalent to Equation (10) when we let

$$\mathcal{D} = \mathcal{D}_L := \{[\boldsymbol{h}_L \circ \cdots \circ \boldsymbol{h}_1(\boldsymbol{x})]_j : \boldsymbol{h}_\ell \in \mathcal{S}_\ell, \ell = 1, \ldots, L, j = 1, \ldots, N\},$$

where $[\boldsymbol{h}(\boldsymbol{x})]_j$ means the j-th output of the vector function $\boldsymbol{h}(\boldsymbol{x})$. Nonlinear approximation concerns the quantification of the best N-term approximation rate in N defined as

$$\varepsilon_f(N) = \min_{\{g_n\}\subseteq\mathbb{R},\{T_n\}\subseteq\mathcal{D}} \|f(\boldsymbol{x}) - \sum_{n=1}^{N} g_n T_n(\boldsymbol{x})\|_*.$$

Hence, when we use $\mathcal{D}_L$ as the dictionary, it is interesting to quantify

$$\varepsilon_{L,f}(N) = \min_{\{g_n\}\subseteq\mathbb{R},\{T_n\}\subseteq\mathcal{D}_L} \|f(\boldsymbol{x}) - \sum_{n=1}^{N} g_n T_n(\boldsymbol{x})\|_*.$$

Function compositions can significantly enrich the dictionary of nonlinear approximation. For example, if the dictionary is built with the Floor-ReLU networks proposed in this paper, $\varepsilon_{L,f}(N) \leq O(N^{-\sqrt{L}})$ for Lipchitz continuous functions f. This rate is much better than existing rates for much smaller function classes, e.g. $O(N^{-s/d})$ for functions in Besov spaces with smoothness s (DeVore and Ron, 2010; Hangelbroek and Ron, 2010), and $O(N^{-\frac{1}{2d}})$ for Hölder continuous functions of order 1 on $[0,1]^d$ (Xie and Cao, 2013).

4.2 Approximation by compositions

Since approximation via compositions has not been fully explored yet, there are many new research directions remaining. The central question is to characterize the relation of the approximant space $\mathcal{S}$ and the latent spaces $\mathcal{S}_\ell$ so as to guide the design of latent spaces according to the requirement of the approximant space. In terms of computational efficiency, latent spaces $\mathcal{S}_\ell$ should remain simple and structured such that it is easy to parametrize $\mathcal{S}_\ell$ with an efficient numerical algorithm to identify an approximant in $\mathcal{S}$. In terms of approximation efficiency, latent spaces should be complex enough such that they can generate a wide class of functions.

Generating an approximant space with a large number of latent spaces is a natural preference balancing the computational complexity and the approximation capacity. There is a dilemma making it difficult to design a powerful approximant space $\mathcal{S}$ when L is small. For example, when $L = 1$, it is required that $\mathcal{S}_1$ is sufficiently simple and complex simultaneously. However, when L is large, there is much room to use simple

latent spaces $\mathcal{S}_\ell$ to generate a complicated approximant space $\mathcal{S}$, which is the most prevailing advantage of function approximation via compositions.

For example, the approximant space of deep ReLU networks with depth L can be generated by latent spaces defined as

$$\mathcal{S}_\ell = \left\{ \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{\varrho} \odot (\boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b}) : \boldsymbol{W} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}, \boldsymbol{b} \in \mathbb{R}^{N_\ell}, \boldsymbol{\varrho} \in \Theta^{N_\ell} \right\}, \tag{11}$$

with $\Theta = \{\max\{0, x\}\}$, $1 \le i \le N_\ell$, $\ell = 1, \ldots, L$, and

$$\mathcal{S}_{L+1} = \left\{ h(\boldsymbol{x}) = \boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b} : \boldsymbol{W} \in \mathbb{R}^{k \times N_L}, \boldsymbol{b} \in \mathbb{R}^k \right\}.$$

Recall that, for a set of one-dimensional functions Θ, $\boldsymbol{\varrho} \odot (\boldsymbol{x}) \in \Theta^d$ is used to denote $\boldsymbol{\varrho} \odot (\boldsymbol{x}) = \begin{bmatrix} \rho_1(x_1) \\ \vdots \\ \rho_d(x_d) \end{bmatrix}$ with $\rho_i \in \Theta$ for $i = 1, \ldots, d$.

Similarly, the space of Floor-ReLU neural networks studied in this paper can be generated by latent spaces $\mathcal{S}_\ell$ defined as

$$\mathcal{S}_\ell = \left\{ \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{\varrho} \odot (\boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b}) : \boldsymbol{W} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}, \boldsymbol{b} \in \mathbb{R}^{N_\ell}, \boldsymbol{\varrho} \in \Theta^{N_\ell} \right\}, \tag{12}$$

with $\Theta = \{\max\{0, x\}, \lfloor x \rfloor\}$, $1 \le i \le N_\ell$, for $\ell = 1, \ldots, L$, and

$$\mathcal{S}_{L+1} = \left\{ h(\boldsymbol{x}) = \boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b} : \boldsymbol{W} \in \mathbb{R}^{k \times N_L}, \boldsymbol{b} \in \mathbb{R}^k \right\}.$$

Motivated by Floor-ReLU networks, it would be interesting to investigate the approximant space $\mathcal{S}_{\Theta,\boldsymbol{N}}$ generated by latent spaces $\mathcal{S}_\ell$ defined via

$$\mathcal{S}_\ell = \left\{ \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{\varrho} \odot (\boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b}) : \boldsymbol{W} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}, \boldsymbol{b} \in \mathbb{R}^{N_\ell}, \boldsymbol{\varrho} \in \Theta^{N_\ell} \right\} \tag{13}$$

for $\ell = 1, \ldots, L$, and

$$\mathcal{S}_{L+1} = \left\{ \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b} : \boldsymbol{W} \in \mathbb{R}^{k \times N_L}, \boldsymbol{b} \in \mathbb{R}^k \right\},$$

where $\boldsymbol{N} = (N_0, N_1, \ldots, N_L, N_{L+1}) \in \mathbb{N}_+^{L+2}$, $N_0 = d$, $N_{L+1} = k$, and Θ is a finite set of one-dimensional functions. The formulation in Equation (13) keeps it neat and easy for computation. Though functions in $\mathcal{S}_\ell$ defined in Equation (13) are high-dimensional, they only consist of a simple high-dimensional linear transform and a few one-dimensional nonlinear functions. Therefore, the essential complexity of each function in $\mathcal{S}_\ell$ is $O(N_\ell)$. The formulation in Equation (13) is sufficiently powerful to make it unnecessary to consider more complex latent spaces, because any continuous latent space can be approximated by the compositions of latent spaces in Equation (13) as shown by Theorem 1.1.

There are mainly two research directions to characterize the approximation power of compositions: 1) fixing one kind of latent spaces and varying target function spaces; 2) fixing a target function space as general as possible and explore different kinds of latent spaces. For simplicity, let us assume $N_\ell = N$ for $\ell = 1, \ldots, L$ and $k = 1$ in Definition 4.1.

In the first research direction, there has been extensive research in the literature starting from shallow neural networks with the Sigmoid activation function to deep neural

networks with the ReLU activation function (Cybenko, 1989; Hornik et al., 1989; Barron, 1993; Yarotsky, 2018, 2017; Bölcskei et al., 2019; Zhou, 2019; Chui et al., 2018; Gribonval et al., 2019; Gühring et al., 2019; Suzuki, 2019; Nakada and Imaizumi, 2019; Chen et al., 2019; Bao et al., 2019; Li et al., 2019; Montanelli and Yang, 2020). For example, it was shown in (Lu et al., 2020) that deep ReLU neural networks can achieve an approximation rate $O(N^{-L})$ for multi-dimensional polynomials, but $O(N^{-L})$ is not true for general smooth functions, e.g. a nearly optimal approximation rate for $C^s([0,1]^d)$ functions is $O(N^{-2s/d})$. For a larger function class, e.g., $C([0,1]^d)$ functions, a nearly optimal approximation rate of ReLU networks is $O(N^{-2/d})$ as shown in (Shen et al., 2019). Hence, the first research direction with an existing network structure may not provide an appealing approximation rate unless the target function space is sufficiently small and structured.

The second research direction is more promising and open to new theories. It was stated in (Yarotsky and Zhevnerchuk, 2019) with a sketchy discussion that, when Θ in Equation (13) contains both the ReLU and sin functions, the approximant space $\mathcal{S}_{\Theta,\boldsymbol{N}}$ with $\max\{N_1,\dots,N_L\} = O(1)$ can approximate smooth functions on $[0,1]^d$ with an approximation rate $O(e^{-\sqrt{L}})$. Though the power of function compositions can be reflected by the exponent $\sqrt{L}$, the contribution of width is missing and the exponent is not proportional to L. Furthermore, the target function space is much smaller than the continuous function space. When Θ in Equation (13) contains both the ReLU and Floor activation functions, by applying Theorem 1.1 for k times and assembling the resulting k networks into a large network, we can show that an approximation rate of $O(N^{-\sqrt{L}})$ is achievable as in Corollary 4.2 below. It would be very interesting to explore other Θ's to see whether $O(N^{-L})$ is achievable and what is the key characterization of latent spaces to achieve this rate.

Corollary 4.2. *There exist* $\Theta = \{\max\{0,x\}, \lfloor x \rfloor\}$ *such that given any* $N, L \in \mathbb{N}^+$, *for an arbitrary continuous vector function* $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^k$ *on* $[0,1]^d$, *there exists a function* $\boldsymbol{\phi} \in \mathcal{S}_{\Theta,\boldsymbol{N}}$ *with* $\boldsymbol{N} = \{d, N_1, \dots, N_{\tilde{L}}, k\}$, $\tilde{L} = 64dL + 3$, *and* $\max\{dk, k(5N+13)\} \geq \max\{N_1,\dots,N_{\tilde{L}}\}$ *such that*

$$\|\boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x})\|_{\ell^\infty} \leq \omega_{\boldsymbol{f}}\left(\sqrt{d}N^{-\sqrt{L}}\right) + 2\omega_{\boldsymbol{f}}(\sqrt{d})N^{-\sqrt{L}}$$

for any $\boldsymbol{x} \in [0,1]^d$, *where* $\omega_{\boldsymbol{f}}(x) := \max\{\omega_{f_1}(x), \dots, \omega_{f_k}(x)\}$.

5 Conclusion

This paper has introduced the first theoretical framework to show that deep network approximation can achieve exponential convergence and avoid the curse of dimensionality for approximating functions as general as continuous functions. Given a Lipschitz continuous function f on $[0,1]^d$, it was shown by construction that Floor-ReLU networks with width $\max\{d, 5N+13\}$ and depth $64dL+3$ admit a uniform approximation rate $3\mu\sqrt{d}\,N^{-\sqrt{L}}$, where μ is the Lipschitz constant of f. More generally for an arbitrary continuous function f on $[0,1]^d$ with a modulus of continuity $\omega_f(\cdot)$, the constructive approximation rate is $\omega_f(\sqrt{d}\,N^{-\sqrt{L}}) + 2\omega_f(\sqrt{d})N^{-\sqrt{L}}$. Function approximation via

compositions was also introduced as a more general framework including deep network approximation.

Acknowledgments. Z. Shen is supported by Tan Chin Tuan Centennial Professorship. H. Yang was partially supported by the US National Science Foundation under award DMS-1945029.

References

Bao, C., Li, Q., Shen, Z., Tai, C., Wu, L., and Xiang, X. (2019). Approximation analysis of convolutional neural networks.

Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945.

Bartlett, P., Maiorov, V., and Meir, R. (1998). Almost linear VC-dimension bounds for piecewise polynomial networks. *Neural Computation*, 10:217–3.

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

Chen, L. and Wu, C. (2019). A note on the expressive power of deep rectified linear unit networks in high-dimensional spaces. *Mathematical Methods in the Applied Sciences*, 42(9):3400–3404.

Chen, M., Jiang, H., Liao, W., and Zhao, T. (2019). Efficient approximation of deep relu networks for functions on low dimensional manifolds. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8174–8184. Curran Associates, Inc.

Chui, C. K., Lin, S.-B., and Zhou, D.-X. (2018). Construction of neural networks for realization of localized deep learning. *Frontiers in Applied Mathematics and Statistics*, 4:14.

Cireşan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pages 1237–1242. AAAI Press.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *MCSS*, 2:303–314.

DeVore, R. and Ron, A. (2010). Approximation using scattered shifts of a multivariate function. *Transactions of the American Mathematical Society*, 362(12):6205–6229.

DeVore, R. A. (1998). Nonlinear approximation. *Acta Numerica*, 7:51–150.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159.

E, W. and Wang, Q. (2018). Exponential convergence of the deep neural network approximation for analytic functions. *CoRR*, abs/1807.00297.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202.

Gribonval, R., Kutyniok, G., Nielsen, M., and Voigtlaender, F. (2019). Approximation spaces of deep neural networks. *arXiv e-prints*, page arXiv:1905.01208.

Gühring, I., Kutyniok, G., and Petersen, P. (2019). Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms. *arXiv e-prints*, page arXiv:1902.07896.

Hangelbroek, T. and Ron, A. (2010). Nonlinear approximation using gaussian kernels. *Journal of Functional Analysis*, 259(1):203 – 219.

Harvey, N., Liaw, C., and Mehrabian, A. (2017). Nearly-tight VC-dimension bounds for piecewise linear neural networks. In Kale, S. and Shamir, O., editors, *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 1064–1068, Amsterdam, Netherlands. PMLR.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. (2017). Quantized neural networks: Training neural networks with low precision weights and activations. *J. Mach. Learn. Res.*, 18(1):6869–6898.

Johnson, R. and Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, pages 315–323, USA. Curran Associates Inc.

Li, Q., Lin, T., and Shen, Z. (2019). Deep Learning via Dynamical Systems: An Approximation Perspective. *arXiv e-prints*, page arXiv:1912.10382.

Liang, S. and Srikant, R. (2016). Why deep neural networks? *CoRR*, abs/1610.04161.

Lu, J., Shen, Z., Yang, H., and Zhang, S. (2020). Deep Network Approximation for Smooth Functions. *arXiv e-prints*, page arXiv:2001.03040.

Montanelli, H. and Du, Q. (2017). New error bounds for deep networks using sparse grids.

Montanelli, H. and Yang, H. (2020). Error bounds for deep relu networks using the kolmogorov–arnold superposition theorem. *Neural Networks*, 129:1 – 6.

Montanelli, H., Yang, H., and Du, Q. (2019). Deep ReLU networks overcome the curse of dimensionality for bandlimited functions.

Nakada, R. and Imaizumi, M. (2019). Adaptive approximation and estimation of deep neural network with intrinsic dimensionality.

Opschoor, J. A., Schwab, C., and Zech, J. (2019). Exponential ReLU DNN expression of holomorphic maps in high dimension. Technical report, Zurich.

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

Poggio, T., Mhaskar, H. N., Rosasco, L., Miranda, B., and Liao, Q. (2017). Why and when can deep—but not shallow—networks avoid the curse of dimensionality: A review. *International Journal of Automation and Computing*, 14:503–519.

Rumelhart, D., McClelland, J., Group, P. R., and University of California, S. D. P. R. G. (1986). *Psychological and Biological Models*. Number v. 2 in A Bradford Book. MIT Press.

Scherer, D., Müller, A., and Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In Diamantaras, K., Duch, W., and Iliadis, L. S., editors, *Artificial Neural Networks – ICANN 2010*, pages 92–101, Berlin, Heidelberg. Springer Berlin Heidelberg.

Shen, Z., Yang, H., and Zhang, S. (2019). Deep network approximation characterized by number of neurons. *arXiv e-prints*, page arXiv:1906.05497.

Shen, Z., Yang, H., and Zhang, S. (2019). Nonlinear approximation via compositions. *Neural Networks*, 119:74 – 84.

Suzuki, T. (2019). 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*.

Wang, P., Hu, Q., Zhang, Y., Zhang, C., Liu, Y., and Cheng, J. (2018). Two-step quantization for low-bit neural networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4376–4384.

Werbos, P. (1975). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University.

Xie, T. F. and Cao, F. L. (2013). The rate of approximation of gaussian radial basis neural networks in continuous function space. *Acta Mathematica Sinica, English Series*, 29(2):295–302.

Yang, Y. and Wang, Y. (2020). Approximation in shift-invariant spaces with deep ReLU neural networks. *arXiv e-prints*, page arXiv:2005.11949.

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

Yarotsky, D. (2018). Optimal approximation of continuous functions by very deep ReLU networks. In Bubeck, S., Perchet, V., and Rigollet, P., editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 639–649. PMLR.

Yarotsky, D. and Zhevnerchuk, A. (2019). The phase diagram of approximation rates for deep neural networks. *arXiv e-prints*, page arXiv:1906.09477.

Yin, P., Lyu, J., Zhang, S., Osher, S., Qi, Y., and Xin, J. (2019). Understanding straight-through estimator in training activation quantized neural nets. *ArXiv*, abs/1903.05662.

Zhou, D.-X. (2019). Universality of deep convolutional neural networks. *Applied and Computational Harmonic Analysis*.