Neural Network Approximation: Three Hidden Layers Are Enough

Zuowei Shen[a], Haizhao Yang[b], Shijun Zhang[a]

[a]*Department of Mathematics, National University of Singapore*
[b]*Department of Mathematics, Purdue University*

Abstract

A three-hidden-layer neural network with super approximation power is introduced. This network is built with the Floor function ($\lfloor x \rfloor$), the exponential function (2^x), the step function ($\mathbb{1}_{x\geq 0}$), or their compositions as the activation function in each neuron and hence we call such networks as Floor-Exponential-Step (FLES) networks. For any width hyper-parameter $N \in \mathbb{N}^+$, it is shown that FLES networks with width $\max\{d, N\}$ and three hidden layers can uniformly approximate a Hölder function f on $[0,1]^d$ with an exponential approximation rate $3\lambda(2\sqrt{d})^\alpha 2^{-\alpha N}$, where $\alpha \in (0,1]$ and $\lambda > 0$ are the Hölder order and constant, respectively. 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 $2\omega_f(2\sqrt{d})2^{-N} + \omega_f(2\sqrt{d}\,2^{-N})$. Moreover, we extend such a result to general continuous functions on a bounded set $E \subseteq \mathbb{R}^d$. As a consequence, this new class of networks overcomes the curse of dimensionality in approximation power when the variation of $\omega_f(r)$ as $r \to 0$ is moderate (e.g., $\omega_f(r) \lesssim r^\alpha$ for Hölder continuous functions), since the major term to be concerned in our approximation rate is essentially $\sqrt{d}$ times a function of N independent of d within the modulus of continuity. Finally, we extend our analysis to derive similar approximation results in the L^p-norm for $p \in [1, \infty)$ via replacing Floor-Exponential-Step activation functions by continuous activation functions.

Keywords:
Exponential Convergence, Curse of Dimensionality, Deep Neural Network,

Email addresses: matzuows@nus.edu.sg (Zuowei Shen), haizhao@purdue.edu (Haizhao Yang), zhangshijun@u.nus.edu (Shijun Zhang)

Floor-Exponential-Step Activation Function, Continuous Function.

1. Introduction

This paper studies the approximation power of neural networks and shows that three hidden layers are enough for neural networks to achieve super approximation capacity. In particular, leveraging the power of advanced yet simple activation functions, we will introduce new theories and network architectures with only three hidden layers achieving exponential convergence and avoiding the curse of dimensionality simultaneously for (Hölder) continuous functions with an explicit approximation bound. The theories established in this paper would provide new insights to explain why deeper neural networks are better than one-hidden-layer neural networks for large-scale and high-dimensional problems. The approximation theories here are constructive (i.e., with explicit formulas to specify network parameters) and quantitative (i.e., results valid for essentially arbitrary width and/or depth without lower bound constraints) with explicit error bounds working for three-hidden-layer networks with arbitrary width.

Constructive approximation with quantitative results and explicit error bounds would provide important guides for deciding the network sizes in deep learning. For example, the (nearly) optimal approximation rates of deep ReLU networks with width $\mathcal{O}(N)$ and depth $\mathcal{O}(L)$ for a Lipschitz continuous function and a C^s function f on $[0,1]^d$ are $\mathcal{O}(\sqrt{d}N^{-2/d}L^{-2/d})$ and $\mathcal{O}(\|f\|_{C^s}(\frac{N}{\ln N})^{-2s/d}(\frac{L}{\ln L})^{-2s/d})$ Shen et al. (2020); Lu et al. (2020), respectively. For results in terms of the number of nonzero parameters, the reader is referred to Yarotsky (2017); Schmidt-Hieber (2020b); Petersen and Voigtlaender (2018); Yarotsky (2018); Gühring et al. (2019); Yarotsky and Zhevnerchuk (2019) and the reference therein. Obviously, the curse of dimensionality exists in ReLU networks for these generic functions and, therefore, ReLU networks would need to be exponentially large in d to maintain a reasonably good approximation accuracy. The curse could be lessened when target function spaces are smaller. To name a few, Poggio et al. (2017); Barron and Klusowski (2018); E et al. (2019); Montanelli et al. (2020); Chen et al. (2019a); Hutzenthaler et al. (2020) and reference therein for ReLU networks. The limitation of ReLU networks motivated the work in Shen et al. (2020) to introduce Floor-ReLU networks built with either a Floor ($\lfloor x \rfloor$) or ReLU ($\max\{0, x\}$) activation function in each neuron. It was shown

by construction in Shen et al. (2020) that Floor-ReLU networks with width $\max\{d, 5N+13\}$ and depth $64dL+3$ can uniformly approximate a Hölder continuous function f on $[0,1]^d$ with a root-exponential approximation rate $3\lambda d^{\alpha/2}N^{-\alpha\sqrt{L}}$ without the curse of dimensionality.

The most important message of Shen et al. (2020) (and probably also of Yarotsky and Zhevnerchuk (2019)) is that the combination of simple activation functions can create super approximation power. In the Floor-ReLU networks mentioned above, the power of depth is fully reflected in the approximation rate $3\lambda d^{\alpha/2}N^{-\alpha\sqrt{L}}$ that is root-exponential in depth. However, the power of width is much weaker and the approximation rate is polynomial in width if depth is fixed. This seems to be inconsistent with recent development of network optimization theory Jacot et al. (2018); Du et al. (2019); Mei et al. (2018); Wu et al. (2018); Chen et al. (2019b); Lu et al. (2020); Luo and Yang (2020), where larger width instead of depth can ease the challenge of highly noncovex optimization. The mystery of the power of width and depth remains and it motivates us to demonstrate that width can also enable super approximation power when armed with appropriate activation functions.

In particular, we explore the floor function, the exponential function (2^x), the step function ($\mathbb{1}_{x\geq 0}$), or their compositions as activation functions to build fully-connected feed-forward neural networks. These networks are called Floor-Exponential-Step (FLES) networks. As we shall prove by construction, Theorem 1.1 below shows that FLES networks with width $\max\{d, N\}$ and three hidden layers can uniformly approximate a continuous function f on $[0,1]^d$ with an exponential approximation rate $2\omega_f(2\sqrt{d})2^{-N} + \omega_f(2\sqrt{d}\,2^{-N})$, where $\omega_f(\cdot)$ is the modulus of continuity defined as

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

In particular, there are three kinds of activation functions denoted as σ_1, σ_2, and σ_3 in FLES networks (see Figure 1 for an illustration):

$$\sigma_1(x) \coloneqq \lfloor x\rfloor, \quad \sigma_2(x)\coloneqq 2^x, \quad \text{and} \quad \sigma_3 \coloneqq \mathcal{T}(x-\lfloor x\rfloor - \tfrac{1}{2}), \quad \text{for any } x\in\mathbb{R},$$

where

$$\mathcal{T}(x) \coloneqq \mathbb{1}_{x\geq 0} = \begin{cases} 1, & x\geq 0,\\ 0, & x<0, \end{cases} \qquad \text{for any } x\in\mathbb{R}.$$

Theorem 1.1. *Given an arbitrary continuous function f defined on $[0,1]^d$, for any $N \in \mathbb{N}^+$, there exist $a_1, a_2, \cdots, a_N \in [0, \frac{1}{2})$ such that*

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le 2\omega_f(2\sqrt{d})2^{-N} + \omega_f(2\sqrt{d}\,2^{-N}),$$

for any $\boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in [0,1]^d$, where ϕ is defined by a formula in $a_1, a_2, \cdots, a_N$ as follows.

$$\phi(\boldsymbol{x}) = 2\omega_f(2\sqrt{d}) \sum_{j=1}^{N} 2^{-j} \sigma_3\Big(a_j \cdot \sigma_2\Big(1 + \sum_{i=1}^{d} 2^{(i-1)N} \sigma_1(2^{N-1} x_i) \Big) \Big) + f(\boldsymbol{0}) - \omega_f(2\sqrt{d}).$$

We remark that ϕ in Theorem 1.1 is determined by N parameters $a_1, a_2, \cdots, a_N$, which can be trained by a $(\sigma_1, \sigma_2, \sigma_3)$-activated network with width $\max\{d, N\}$, three hidden layers, and $2(d+N+1)$ nonzero parameters. See Figure 1 for an illustration.

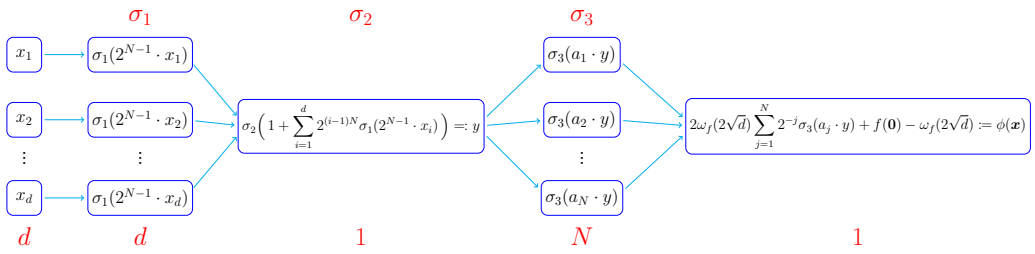


Figure 1: An illustration of the desired three-hidden-layer network in Theorem 1.1 for any $\boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}$. Each of the red functions "σ_1", "σ_2", and "σ_3" above the network is the activation function of the corresponding hidden layer. The number of neurons in each hidden layer is indicated by the red number below it.

The rate in $\omega_f(2\sqrt{d}\,2^{-N})$ implicitly depends on N through the modulus of continuity of f, while the rate in $2\omega_f(2\sqrt{d})2^{-N}$ is explicit in N. Simplifying the implicit approximation rate to make it explicitly depend on N is challenging in general. However, if f is a Hölder continuous function on $[0,1]^d$ of order $\alpha \in (0,1]$ with a constant $\lambda > 0$, i.e., $f(\boldsymbol{x})$ satisfying

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \le \lambda \|\boldsymbol{x} - \boldsymbol{y}\|_2^{\alpha}, \quad \text{for any } \boldsymbol{x}, \boldsymbol{y} \in [0,1]^d, \tag{1}$$

then $\omega_f(r) \le \lambda r^{\alpha}$ for any $r \ge 0$. Therefore, in the case of Hölder continuous functions, the approximation rate is simplified to $3\lambda(2\sqrt{d})^{\alpha}2^{-\alpha N}$ as shown in the following corollary. In the special case of Lipschitz continuous functions with a Lipschitz constant $\lambda > 0$, the approximation rate is simplified to $6\lambda\sqrt{d}\,2^{-N}$.

Corollary 1.2. *Given any a Hölder continuous function f on $[0,1]^d$ of order $\alpha \in (0,1]$ with a constant $\lambda > 0$, for any $N \in \mathbb{N}^+$, there exists $a_1, a_2, \cdots, a_N$ such that*

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le 3\lambda(2\sqrt{d})^{\alpha} 2^{-\alpha N}, \quad \text{for any } \boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in [0,1]^d,$$

where ϕ is defined by a formula in $a_1, a_2, \cdots, a_N$ as follows.

$$\phi(\boldsymbol{x}) = 2\omega_f(2\sqrt{d}) \sum_{j=1}^{N} 2^{-j} \sigma_3\left(a_j \cdot \sigma_2\left(1 + \sum_{i=1}^{d} 2^{(i-1)N} \sigma_1(2^{N-1} x_i) \right) \right) + f(\mathbf{0}) - \omega_f(2\sqrt{d}).$$

First, Theorem 1.1 and Corollary 1.2 show that the approximation capacity of three-hidden-layer neural networks with simple activation functions for continuous functions can be exponentially improved by increasing the network width, and the approximation error can be explicitly characterized in terms of the width $\mathcal{O}(N)$. Second, this new class of networks overcomes the curse of dimensionality in the approximation power when the modulus of continuity is moderate, since the approximation order is essentially $\sqrt{d}$ times a function of N independent of d within the modulus of continuity. Therefore, three hidden layers are enough for neural networks to achieve exponential convergence and avoid the curse of dimensionality for generic functions. The width is also powerful in network approximation.

The rest of this paper is organized as follows. In Section 2, we discuss the application scope of our theory, study the connection between the approximation error and the VapnikChervonenkis (VC) dimension, establish Corollary 2.3 to extend our analysis to general continuous functions on a bounded set, and compare related works in the literature. We will prove Theorem 1.1 and Corollary 2.3 in Section 3. In Section 4, we explore alternative continuous activation functions other than σ_1, σ_2, and σ_3 for super approximation power. Finally, we conclude this paper in Section 5.

2. Discussion

In this section, we will further interpret our results and discuss related research in the field of neural network approximation.

2.1. Application scope of our theory in machine learning

Let $\phi(\boldsymbol{x}; \boldsymbol{\theta})$ denote a function computed by a (fully-connected) network with $\boldsymbol{\theta}$ as the set of parameters. Given a target function f, consider the

expected error/risk of $\phi(\boldsymbol{x};\boldsymbol{\theta})$

$$R_{\mathcal{D}}(\boldsymbol{\theta}) \coloneqq \mathbb{E}_{\boldsymbol{x}\sim U(\mathcal{X})}\left[\ell(\phi(\boldsymbol{x};\boldsymbol{\theta}), f(\boldsymbol{x}))\right]$$

with a loss function typically taken as $\ell(y,y') = \frac{1}{2}|y-y'|^2$, where $U(\mathcal{X})$ is an unknown data distribution over $\mathcal{X}$. For example, when $\ell(y,y') = \frac{1}{2}|y-y'|^2$ and U is a uniform distribution over $\mathcal{X} = [0,1]^d$,

$$R_{\mathcal{D}}(\boldsymbol{\theta}) = \int_{[0,1]^d} \tfrac{1}{2}|\phi(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x})|^2 d\boldsymbol{x}.$$

The expected risk minimizer $\boldsymbol{\theta}_{\mathcal{D}}$ is defined as

$$\boldsymbol{\theta}_{\mathcal{D}} \coloneqq \arg\min_{\boldsymbol{\theta}} R_{\mathcal{D}}(\boldsymbol{\theta}).$$

It is unachievable in practice since f and $U(\mathcal{X})$ are not available. Instead, we only have samples of f.

Given samples $\{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i=1}^n$, the empirical risk is defined as

$$R_{\mathcal{S}}(\boldsymbol{\theta}) \coloneqq \frac{1}{n}\sum_{i=1}^{n} \ell\big(\phi(\boldsymbol{x}_i;\boldsymbol{\theta}), f(\boldsymbol{x}_i)\big).$$

And we usually use it to approximate/model the expected risk $R_{\mathcal{D}}(\boldsymbol{\theta})$. The goal of supervised learning is to identify the empirical risk minimizer

$$\boldsymbol{\theta}_{\mathcal{S}} = \arg\min_{\boldsymbol{\theta}} R_{\mathcal{S}}(\boldsymbol{\theta}), \tag{2}$$

to obtain $\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{S}}) \approx f(\boldsymbol{x})$. When a numerical optimization method is applied to solve (2), it may result in a numerical solution (denoted as $\boldsymbol{\theta}_{\mathcal{N}}$) that is not a global minimizer. Hence, the actually learned function generated by a neural network is $\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{N}})$. And the discrepancy between the target function f and the actually learned function $\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{N}})$ is measured by an inference error

$$R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}}) = \mathbb{E}_{\boldsymbol{x}\sim U(\mathcal{X})}\left[\ell(\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{N}}), f(\boldsymbol{x}))\right] \overset{e.g.}{=} \int_{[0,1]^d} \tfrac{1}{2}|\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{N}}) - f(\boldsymbol{x})|^2 d\boldsymbol{x},$$

where the second equality holds when $\ell(y,y') = \frac{1}{2}|y-y'|^2$ and U is a uniform distribution over $\mathcal{X} = [0,1]^d$.

Since $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}})$ is the expected inference error over all possible data samples, it can quantify how good the learned function $\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{N}})$ is. Note that

$$\begin{aligned} R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}}) &= \underbrace{[R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{N}})]}_{\text{GE}} + \underbrace{[R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{N}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{S}})]}_{\text{OE}} + \underbrace{[R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{S}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{D}})]}_{\leq 0 \text{ by Eq. (2)}} + \underbrace{[R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{D}}) - R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})]}_{\text{GE}} + \underbrace{R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})}_{\text{AE}} \\ &\leq \underbrace{R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})}_{\text{Approximation error (AE)}} + \underbrace{[R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{N}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{S}})]}_{\text{Optimization error (OE)}} + \underbrace{[R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{N}})] + [R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{D}}) - R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})]}_{\text{Generalization error (GE)}}. \end{aligned} \tag{3}$$

where the inequality comes from the fact that $[R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{S}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{D}})] \leq 0$ since $\boldsymbol{\theta}_{\mathcal{S}}$ is a global minimizer of $R_{\mathcal{S}}(\boldsymbol{\theta})$. Constructive approximation provides an upper bound of $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$ in terms of the network size, e.g., in terms of the network width and depth, or in terms of the number of parameters. The second term of Equation (3) is bounded by the optimization error of the numerical algorithm applied to solve the empirical loss minimization problem in Equation (2). Note that one only needs to make $R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{N}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{S}})$ small, but not $\boldsymbol{\theta}_{\mathcal{N}} - \boldsymbol{\theta}_{\mathcal{S}}$. The study of the bounds for the third and fourth terms is referred to as the generalization error analysis of neural networks. See Figure 2 for the intuitions of these three errors.

One of the key targets in the area of deep learning is to develop algorithms to reduce $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}})$. The constructive approximation established in this paper and in the literature provides upper bounds of the approximation error $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$ for several function spaces, which is crucial to estimate an upper bound of $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}})$. Instead of deriving an approximator to attain the approximation error bound, deep learning algorithms aim at identifying a solution $\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{N}})$ reducing the generalization and optimization errors in Equation (3). Solutions minimizing both generalization and optimization errors will lead to a good solution only if we also have a good upper bound estimate of $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$ as shown in Equation (3). Independent of whether our analysis here leads to a good approximator, which is an interesting topic to pursue, the theory here does provide a key ingredient in the error analysis of deep learning algorithms.

Theorem 1.1 and Corollary 1.2 provide an upper bound of $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$. This bound only depends on the given budget of neurons and layers of FLES networks. Hence, this bound is independent of the empirical loss minimization in (2) and the optimization algorithm used to compute the numerical solution of (2). In other words, Theorem 1.1 and Corollary 1.2 quantify the approximation power of FLES networks with a given size. Designing efficient optimization algorithms and analyzing the generalization bounds for FLES

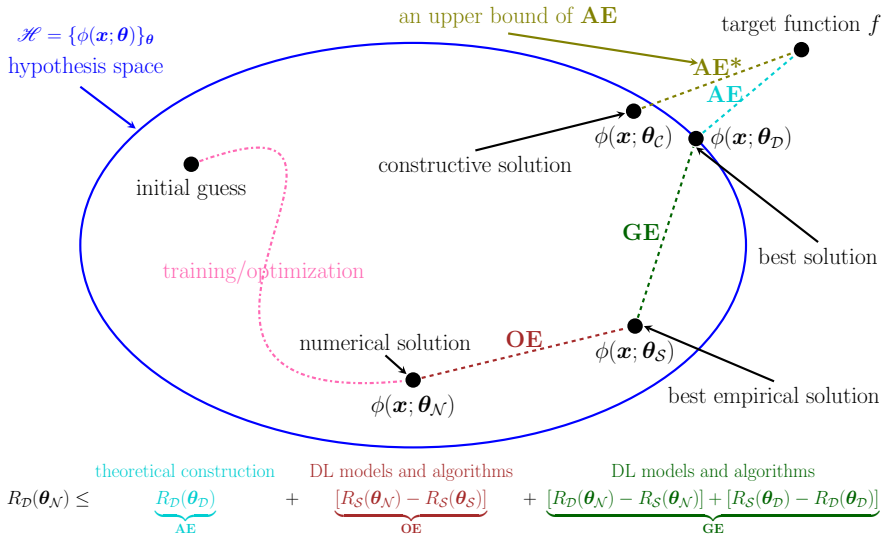


Figure 2: The intuitions of the approximation error (AE), the optimization error (OE), and the generalization error (GE). DL is short of deep learning. One needs to control AE, OE, and GE in order to bound the discrepancy between the target function f and the numerical solution $\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{N}})$ (what we can get in practice), measured by $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}}) = \mathbb{E}_{\boldsymbol{x}\sim U(\mathcal{X})}\left[\ell(\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{N}}), f(\boldsymbol{x}))\right]$.

networks are two other separate future directions.

2.2. Connection between approximation error and VC-dimension

The approximation error and the VapnikChervonenkis (VC) dimension are two important measures of the capacity (complexity) of a set of functions. In this section, we discuss the connection between them.

Let us first present the definitions of VC-dimension and related concepts. Let H be a class of functions mapping from a general domain $\mathcal{X}$ to $\{0,1\}$. We say H shatters the set $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m\} \subseteq \mathcal{X}$ if

$$\left|\left\{\left[h(\boldsymbol{x}_1), h(\boldsymbol{x}_2), \cdots, h(\boldsymbol{x}_m)\right]^T \in \{0,1\}^m : h \in H\right\}\right| = 2^m,$$

where $|\cdot|$ means the size of a set. This equation means, given any $\theta_i \in \{0,1\}$ for $i = 1, 2, \cdots, m$, there exists $h \in H$ such that $h(\boldsymbol{x}_i) = \theta_i$ for all i.

For any $m \in \mathbb{N}^+$, we define the growth function of H as

$$\Pi_H(m) \coloneqq \max_{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m \in \mathcal{X}} \left|\left\{\left[h(\boldsymbol{x}_1), h(\boldsymbol{x}_2), \cdots, h(\boldsymbol{x}_m)\right]^T \in \{0,1\}^m : h \in H\right\}\right|.$$

Definition 2.1 (VC-dimension)**.** Let H be a class of functions from $\mathcal{X}$ to $\{0,1\}$. The VC-dimension of H, denoted by $\mathrm{VCDim}(H)$, is the size of the largest shattered set, namely,

$$\mathrm{VCDim}(H) \coloneqq \sup\left\{m \in \mathbb{N}^+ : \Pi_H(m) = 2^m\right\}.$$

Let $\mathscr{F}$ be a class of functions from $\mathcal{X}$ to $\mathbb{R}$. The VC-dimension of $\mathscr{F}$, denoted by $\mathrm{VCDim}(\mathscr{F})$, is defined by $\mathrm{VCDim}(\mathscr{F}) \coloneqq \mathrm{VCDim}(\mathcal{T} \circ \mathscr{F})$, where

$$\mathcal{T}(t) \coloneqq \begin{cases} 1, & t \geq 0, \\ 0, & t < 0 \end{cases} \quad \text{and} \quad \mathcal{T} \circ \mathscr{F} \coloneqq \{\mathcal{T} \circ f : f \in \mathscr{F}\}.$$

In particular, the expression "VC-dimension of a network (architecture)" means the VC-dimension of the function set that consists of all functions implemented by this network (architecture).

As shown in Yarotsky (2018, 2017); Shen et al. (2019, 2020); Lu et al. (2020); Shen et al. (2020); Zhang (2020), VC-dimension essentially determines the lower bound of the approximation errors of networks. For simplicity, we use $\mathrm{Hölder}([0,1]^d, \alpha, \lambda)$ as an example, where $\mathrm{Hölder}([0,1]^d, \alpha, \lambda)$ denotes the space of Hölder continuous functions of order $\alpha \in (0,1]$ and a Hölder constant $\lambda > 0$. Without loss of generality, we assume $\lambda = 1$. Theorem 2.2 below shows that the best possible approximation error of functions in $\mathrm{Hölder}([0,1]^d, \alpha, 1)$ approximated by functions in $\mathscr{F}$ is bounded by a formula characterized by $\mathrm{VCDim}(\mathscr{F})$.

Theorem 2.2 (Theorem 4.17 of Zhang (2020))**.** *Assume $\mathscr{F}$ is a function set with all elements defined on $[0,1]^d$. Given any $\varepsilon \in (0, 2/9)$, suppose*

$$\inf_{\phi \in \mathscr{F}} \|\phi - f\|_{L^\infty([0,1]^d)} \leq \varepsilon, \quad \text{for any } f \in \mathrm{Hölder}([0,1]^d, \alpha, 1). \tag{4}$$

Then $\mathrm{VCDim}(\mathscr{F}) \geq (9\varepsilon)^{-d/\alpha}$.

This theorem investigates the connection between VC-dimension of $\mathscr{F}$ and the approximation errors of functions in $\mathrm{Hölder}([0,1]^d, \alpha, 1)$ approximated by elements of $\mathscr{F}$. Denote the best approximation error of functions in $\mathrm{Hölder}([0,1]^d, \alpha, 1)$ approximated by the elements of $\mathscr{F}$ as

$$\mathcal{E}_{\alpha,d}(\mathscr{F}) \coloneqq \sup_{f \in \mathrm{Hölder}([0,1]^d, \alpha, 1)} \left(\inf_{\phi \in \mathscr{F}} \|\phi - f\|_{L^\infty([0,1]^d)} \right).$$

Then, Theorem 2.2 implies that

$$\mathrm{VCDim}(\mathscr{F})^{-\alpha/d}/9 \;\le\; \mathcal{E}_{\alpha,d}(\mathscr{F}), \tag{5}$$

which means that the best possible approximation error is controlled by $\mathrm{VCDim}(\mathscr{F})^{-\alpha/d}/9$. A typical application of this theorem is to prove the optimality of approximation errors when using ReLU networks to approximate functions in $\text{Hölder}([0,1]^d,\alpha,1)$. It is shown in Harvey et al. (2017) that the VC-dimension of $\mathscr{F}_{N,L}$ is bounded by

$$\mathrm{VCDim}(\mathscr{F}_{N,L}) \le \mathcal{O}(N^2L\cdot L\cdot \ln(N^2L)) \le \mathcal{O}(N^2L^2\ln(NL)),$$

where $\mathscr{F}_{N,L}$ is the space consisting of all functions implemented by ReLU networks with width N and depth L. It is shown in Section 4.4.1 of Zhang (2020) that

$$C_1(\alpha,d)\cdot\Big(N^2L^2\ln(NL)\Big)^{-\alpha/d} \;\le\; \mathcal{E}_{\alpha,d}(\mathscr{F}_{N,L}) \;\le\; C_2(\alpha,d)\cdot\Big(N^2L^2\Big)^{-\alpha/d},$$

for any $N,L\in\mathbb{N}^+$, where $C_1(\alpha,d)$ and $C_2(\alpha,d)$ are two positive constants determined by α and d, and $C_2(\alpha,d)$ can be **explicitly** represented.

Finally, we would like to point out that a large VC-dimension of the hypothesis space $\mathscr{F}$ is a **necessary** condition of a good approximation error, but cannot guarantee a good approximation error, which also relies on other properties of the hypothesis space $\mathscr{F}$. For example, it is easy to check by Proposition 4.2 that

$$\mathrm{VCDim}\Big(\big\{\phi:\phi(x)=\cos(ax),\;\; a\in\mathbb{R}\big\}\Big)=\infty.$$

However, $\big\{\phi:\phi(x)=\cos(ax),\;\; a\in\mathbb{R}\big\}$ cannot achieve a good approximation error when approximating Hölder continuous functions. Designing a hypothesis space with a large VC-dimension is the first step for a good approximation toll, but to realize the desired approximation power requires refined design of the hypothesise space, which is also the philosophy we followed in this paper. Our initial goal is to design a network architecture with a fixed depth (e.g., three hidden layers) to generate a hypothesis space with a sufficiently large VC-dimension (∞). As we shall see later, Proposition 3.2 implies that the VC-dimension of FLES networks is infinity, which is a necessary condition for our FLES networks to attain super approximation power.

2.3. Further interpretation of our theory

In the interpretation of our theory, three more aspects are important to discuss. The first one is whether it is possible to extend our theory to functions on a more general domain, e.g, $E \subseteq [-R, R]^d$ for any $R > 0$, because $R > 1$ may cause an implicit curse of dimensionality in some existing theory. The second one is how bad the modulus of continuity would be since it is related to a high-dimensional function f that may lead to an implicit curse of dimensionality in our approximation rate. The last one is the discussion of overcoming the zero derivative in training FLES networks.

First, we can generalize Theorem 1.1 to the function space $C(E)$ with $E \subseteq [-R, R]^d$ for any $R > 0$ in the following corollary with the modulus of continuity $\omega_f^E(\cdot)$ defined as follows. For an arbitrary set $E \subseteq \mathbb{R}^d$, $\omega_f^E(r)$ is defined via

$$\omega_f^E(r) \coloneqq \sup\big\{|f(\boldsymbol{x}) - f(\boldsymbol{y})| : \|\boldsymbol{x} - \boldsymbol{y}\|_2 \le r,\ \boldsymbol{x}, \boldsymbol{y} \in E\big\}, \qquad \text{for any } r \ge 0.$$

As defined earlier, in the case $E = [0, 1]^d$, $\omega_f^E(r)$ is abbreviated to $\omega_f(r)$. The proof of this corollary will be presented in Section 3.2.

Corollary 2.3. *Given an arbitrary continuous function f on $E \subseteq [-R, R]^d$ for any $R > 0$, for any $N \in \mathbb{N}^+$, there exist $a_1, a_2, \cdots, a_N \in [0, \frac{1}{2})$ such that*

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le 2\omega_f^E(3R\sqrt{d})2^{-N} + \omega_f^E(3R\sqrt{d}\,2^{-N}),$$

for any $\boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in E$, where ϕ is defined by a formula in $a_1, a_2, \cdots, a_N$ as follows.

$$\phi(\boldsymbol{x}) = 2\omega_f^E(3R\sqrt{d}) \sum_{j=1}^{N} 2^{-j} \sigma_3\bigg(a_j \cdot \sigma_2\Big(1 + \sum_{i=1}^{d} 2^{(i-1)N} \sigma_1(2^N \tfrac{x_i + R}{3R}) \Big) \bigg) + C_f,$$

where C_f is a constant determined by f.

Hence, the volume of the function domain $E \subseteq [-R, R]^d$ only has a mild influence on the approximation rate of our FLES networks. FLES networks can still avoid the curse of dimensionality and achieve exponential convergence for continuous functions on $E \subseteq [-R, R]^d$ when $R > 1$. For example, in the case of Hölder continuous functions of order $\alpha \in (0, 1]$ with a constant $\lambda > 0$ on $E \subseteq [-R, R]^d$, our approximation rate becomes $3\lambda(3R\sqrt{d}\,2^{-N})^\alpha$.

Second, most interesting continuous functions in practice have a good modulus of continuity such that there is no implicit curse of dimensionality

hidden in $\omega_f(\cdot)$. For example, we have discussed the case of Hölder continuous functions previously. We would like to remark that the class of Hölder continuous functions implicitly depends on d through its definition in Equation (1), but this dependence is moderate since the ℓ^2-norm in Equation (1) is the square root of a sum with d terms. Let us now discuss several cases of $\omega_f(\cdot)$ when we cannot achieve exponential convergence or cannot avoid the curse of dimensionality. The first example is $\omega_f(r) = \frac{1}{\ln(1/r)}$ for small $r > 0$, which leads to an approximation rate

$$3(N \ln 2 - \tfrac{1}{2} \ln d - \ln 2)^{-1}, \quad \text{for large } N \in \mathbb{N}^+.$$

Apparently, the above approximation rate still avoids the curse of dimensionality but there is no exponential convergence, which has been canceled out by "ln" in $\omega_f(\cdot)$. The second example is $\omega_f(r) = \frac{1}{\ln^{1/d}(1/r)}$ for small $r > 0$, which leads to an approximation rate

$$3(N \ln 2 - \tfrac{1}{2} \ln d - \ln 2)^{-1/d}, \quad \text{for large } N \in \mathbb{N}^+.$$

The power $1/d$ further weakens the approximation rate and hence the curse of dimensionality exists. The last example we would like to discuss is $\omega_f(r) = r^{\alpha/d}$ for small $r > 0$, which results in the approximation rate

$$3\lambda(2\sqrt{d})^{\alpha/d} 2^{-\alpha N/d}, \quad \text{for large } N \in \mathbb{N}^+,$$

which achieves the exponential convergence and avoids the curse of dimensionality when we use very wide networks. Though we have provided several examples of immoderate $\omega_f(\cdot)$, to the best of our knowledge, we are not aware of useful continuous functions with $\omega_f(\cdot)$ that is immoderate.

Finally, we would like to point out that the training of FLES networks in practice may encounter two issues. First, network weights in our main theorems require high-precision computation that might not be available in existing computer systems when the dimension d and the network size parameter N are large. But there is no theoretical evidence to exclude the possibility that similar approximation results can be achieved with reasonable weights in practical computation. Second, the vanishing gradient of piecewise constant activation functions makes standard SGD infeasible. There are two possible directions to solve the optimization problem for FLES networks: 1) gradient-free optimization methods, e.g., Nelder-Mead method Nelder and Mead (1965), genetic algorithm Holland (1992), simulated annealing Kirkpatrick et al. (1983), particle swarm optimization Kennedy and Eberhart

(1995), and consensus-based optimization Pinnau et al. (2017); Carrillo et al. (2019); 2) applying optimization algorithms for quantized networks that also have piecewise constant activation functions Lin et al. (2019); Boo et al. (2020); Bengio et al. (2013); Wang et al. (2018); Hubara et al. (2017); Yin et al. (2019). For example, an empirical way is to use a straight-through estimator (STE) by setting the incoming gradients to the activation function (e.g., Floor) equal to its outgoing gradients, disregarding the derivative of the activation function itself. It would be interesting future work to explore efficient learning algorithms based on the FLES network.

2.4. Kolmogorov-Arnold Superposition Theorem

A closely related research topic is the Kolmogorov-Arnold representation theorem (KST) Kolmogorov (1956); Arnold (1957); Kolmogorov (1957) and its approximation in a form of modern neural networks. Our FLES networks admit super approximation power with a fixed number of layers for continuous functions and the KST exactly represent continuous functions using two hidden layers and $\mathcal{O}(d)$ neurons. More specifically, given any $f \in C([0,1]^d)$, the KST shows that there exist continuous functions $\phi_q : \mathbb{R} \to \mathbb{R}$ and $\psi_{q,p} : \mathbb{R} \to \mathbb{R}$ such that

$$f(\boldsymbol{x}) = \sum_{q=0}^{2d} \phi_q \left(\sum_{p=1}^{d} \psi_{q,p}(x_p) \right), \quad \text{for any } \boldsymbol{x} = (x_1, \cdots, x_d) \in [0,1]^d. \tag{6}$$

Note that the activation functions $\{\phi_q\}$ (also called outer functions) of the neural network in Equation (6) have to depend on the target function f, though $\{\psi_{q,p}\}$ (also called inner functions) can be independent of f. The modulus of continuity of $\{\psi_{q,p}\}$ can be constructed such that they moderately depend on d, but the modulus of continuity of $\{\phi_q\}$ would be exponentially bad in d. In sum, the outer functions are too pathological such that there is no existing numerical algorithms to evaluate these activation functions, even though they are shown to exist by iterative construction Braun and Griebel (2009).

There has been an active research line to develop more practical network approximation based on KST Kůrková (1991, 1992); Maiorov and Pinkus (1999); Guliyev and Ismailov (2018); Montanelli and Yang (2020); Igelnik and Parikh (2003); Schmidt-Hieber (2020a) by relaxing the exact representation to network approximation with an ε-error. The key issue these KST-related networks attempting to address is the f-dependency of the activation functions and the main goal is to construct neural networks conquering the curse

of dimensionality in a more practical way computationally. The main idea of these variants is to apply computable activation functions independent of f to construct neural networks to approximate the outer and inner functions of the KST, resulting in a larger network that can approximate a continuous function with the desired accuracy. Using this idea, the seminal work in Kůrková (1992) applied sigmoid activation functions and constructed two-hidden-layer networks to approximate $f \in C([0,1]^d)$. Though the activation functions are independent of f, the number of neurons scales exponentially in d and the curse of dimensionality exists. Cubic-splines and piecewise linear functions have also been used to approximate the outer and inner functions of KST in Igelnik and Parikh (2003); Montanelli and Yang (2020); Schmidt-Hieber (2020a), resulting in cubic-spline networks or deep ReLU networks to approximate $f \in C([0,1]^d)$. But the approximation bounds in these works still suffer from the curse of dimensionality unless f has simple outer functions in the KST. It is still an open problem to characterize the class of functions with a moderate outer function in KST.

To the best of our knowledge, the most successful construction of neural networks with f-independent activation functions conquering the curse of dimensionality is in Maiorov and Pinkus (1999); Guliyev and Ismailov (2018), where a two-hidden-layer network with $\mathcal{O}(d)$ neurons can approximate $f \in C([0,1]^d)$ within an arbitrary error ε. Let us briefly summerize their main ideas to obtain such an exciting result here. 1) Identify a dense and countable subset $\{u_k\}_{k=1}^{\infty}$ of $C([-1,1])$, e.g., polynomials with rational coefficients. 2) Construct an activation function ϱ to "store" all $u_k(x)$ for $x \in [-1,1]$. For example, divide the domain of $\varrho(x)$ into countable pieces and each piece is a connected interval of length 2 associated with a u_k. In particular, let $\varrho(x+4k+1) = a_k + b_k x + c_k u_k(x)$ for any $x \in [-1,1]$ with carefully chosen constants a_k, b_k, and c_k such that $\varrho(x)$ can be a sigmoid function. 3) By construction, there exists a one-hidden layer network with width 3 and $\varrho(x)$ as the activation function to approximate any outer or inner function in KST with an arbitrary accuracy parameter δ. Only the parameters of the one-hidden-layer network depend on the target function and accuracy. 4) Replace the inner and outer function in KST with these one-hidden-layer networks to achieve a two-hidden-layer network with $\varrho(x)$ as the activation function and width $\mathcal{O}(d)$ to approximate an arbitrary $f \in C([0,1]^d)$ within an arbitrary error ε. Unfortunately, the construction of the parameters of this magic network relies on the evaluation of the outer and inner functions of KST, which is not computationally feasible even if computation with arbitrary

Table 1: A comparison of several KST-related results for approximating $f \in C([0,1]^d)$.

paper	number of hidden layers	width	activation function(s)	error	remark
Kolmogorov (1956); Arnold (1957); Kolmogorov (1957)	2	$2d+1$	f-dependent	0	original KST
Maiorov and Pinkus (1999); Guliyev and Ismailov (2018)	2	$\mathcal{O}(d)$	f-independent	arbitrary error ε	based on KST
Shen et al. (2019)	3	$\mathcal{O}(dN)$	ReLU	$\mathcal{O}(\omega_f(N^{-2/d}))$	not based on KST
this paper	3	$\max\{d, N\}$	$(\sigma_1, \sigma_2, \sigma_3)$	$2\omega_f(\sqrt{d})2^{-N} + \omega_f(\sqrt{d}2^{-N})$	not based on KST

precision is allowed.

We would like to remark that, though the approximation rate of FLES networks in this paper is relatively worse than the approximation rate in Maiorov and Pinkus (1999); Guliyev and Ismailov (2018), our activation functions are much simpler and there are explicit formulas to specify the parameters of FLES networks. If computation with an arbitrary precision is allowed and the target function f can be arbitrarily sampled, we can specify all the weights in FLES networks. Besides, our approximation rate is sufficiently attractive since it is exponential and avoids the curse of dimensionality. For a large dimension d, the width parameter of our FLES network can be chosen as $N = d$, which leads to a FLES network of size $\mathcal{O}(d)$ with an approximation accuracy $\mathcal{O}(2^{-d})$ for Lipschitz continuous functions. $\mathcal{O}(2^{-d})$ is sufficiently attractive. In practice, when d is very large, N could be much smaller than d and our approximation rate is still attractive.

Finally, we list several KST-related results in Table 1 for a quick comparison.[1] As shown in Table 1, there exists a trade-off between the complexity of activation functions and the network size when the approximation error is fixed. A key advantage of our FLES networks is to use simple and explicit activation functions to attain an exponential convergence rate.

2.5. Discussion on the literature

In this section, we will discuss other recent development of neural network approximation. Our discussion will be divided into mainly three parts according to the analysis methodology in the references: 1) functions admitting integral representations; 2) linear approximation; 3) bit extraction.

In the seminal work of Barron (1993), its variants or generalization Barron and Klusowski (2018); E et al. (2019); Chen and Wu (2019); Montanelli et al. (2020), and related references therein, d-dimensional functions of the

[1] The result in Shen et al. (2019) is for Hölder functions, but can be easily generalized to general continuous functions.

following form were considered:

$$f(\boldsymbol{x}) = \int_{\widetilde{\Omega}} a(\boldsymbol{w}) K(\boldsymbol{w} \cdot \boldsymbol{x}) d\mu(\boldsymbol{w}), \tag{7}$$

where $\widetilde{\Omega} \subseteq \mathbb{R}^d$, $\mu(\boldsymbol{w})$ is a Lebesgue measure in $\boldsymbol{w}$, and $\boldsymbol{x} \in \Omega \subseteq \mathbb{R}^d$. The above integral representation is equivalent to the expectation of a high-dimensional random function when $\boldsymbol{w}$ is treated as a random variable. By the law of large number theory, the average of N samples of the integrand leads to an approximation of $f(\boldsymbol{x})$ with an approximation error bounded by $\frac{C_f\sqrt{\mu(\Omega)}}{\sqrt{N}}$ measured in $L^2(\Omega, \mu)$ (Equation (6) of Barron (1993)), where $\mathcal{O}(N)$ is the total number of parameters in the network, C_f is a d-dimensional integral with an integrand related to f, and $\mu(\Omega)$ is the Lebesgue measure of Ω. As discussed in Barron (1993), $\mu(\Omega)$ and C_f would be exponential in d and standard smoothness properties of f alone are not enough to remove the exponential dependence of C_f on d. Therefore, the curse of dimensionality exists in the whole approximation error while the curse does not exist in the approximation rate in N.

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. Typical examples include approximation via orthogonal polynomials, Fourier series expansion, etc. Inspired by the seminal work in Yarotsky (2017), where deep ReLU networks were constructed to approximate polynomials with exponential convergence, subsequent works in E and Wang (2018); Opschoor et al. (2019); Montanelli and Du (2019); Chen and Wu (2019); Montanelli et al. (2020); Yarotsky and Zhevnerchuk (2019); Lu et al. (2020); Montanelli and Yang (2020); Yang and Wang (2020) have constructed deep ReLU networks to approximate various smooth function spaces. The main idea of these works is to approximate smooth functions via (piecewise) polynomial approximation first and then construct deep ReLU networks to approximate the ensemble of polynomials. But the curse of dimensionality exists since polynomial approximation cannot avoid the curse. Finally, a different approach is used in Li et al. (2019). The authors of Li et al. (2019) use a dynamic system based approach to obtain a universal approximation property of residual networks.

The bit extraction proposed in Bartlett et al. (1998) has been a very important technique to develop nearly optimal approximation rates of deep ReLU neural networks Yarotsky (2018); Shen et al. (2020); Lu et al. (2020); Yang and Wang (2020); Zhang (2020) and the optimality is based on the

nearly optimal VC-dimension bound of ReLU networks in Harvey et al. (2017). The bit extraction was also applied in Shen et al. (2020); Schmidt-Hieber (2020a) and this paper to develop network approximation theories. In the first step, an efficient projection map in a form of a ReLU, or a Floor-ReLU, or a FLES network is constructed to project high-dimensional points to one-dimensional points such that the high-dimensional approximation problem is reduced to a one-dimensional approximation problem. In the first step, the one-dimensional approximation problem is solved by constructing a ReLU, or a Floor-ReLU, or a FLES network, which can be efficiently compressed via the bit extraction. Although shallower neural networks can also carry out the above two steps, bit extraction can take full advantage of the power of depth and construct deep neural networks with a nearly optimal number of parameters or neurons to fulfill the above two steps.

3. Theoretical Analysis

In this section, we first introduce basic notations in this paper in Section 3.1. Then we prove Theorem 1.1 and Corollary 2.3 in Section 3.2 based on Theorem 3.1, which is proved in Section 3.3.

3.1. Notations

The main notations of this paper are listed as follows.

- Vectors and matrices are denoted in a bold font. Standard vectorization is adopted in the matrix and vector computation. For example, adding a scalar and a vector means adding the scalar to each entry of the vector.
- Let $\mathbb{Z}$, $\mathbb{N}$, and $\mathbb{N}^+$ denote the set of integers, natural numbers, all positive integers, respectively, i.e., $\mathbb{Z} = \{0, 1, 2, \cdots\} \cup \{-1, -2, -3, \cdots\}$, $\mathbb{N} = \{0, 1, 2, \cdots\}$, and $\mathbb{N}^+ = \{1, 2, 3, \cdots\}$.
- For any $p \in [1, \infty)$, the p-norm of a vector $\boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d$ is defined by
$$\|\boldsymbol{x}\|_p \coloneqq \left(|x_1|^p + |x_2|^p + \cdots + |x_d|^p\right)^{1/p}.$$
- Let $\sigma : \mathbb{R} \to \mathbb{R}$ denote the rectified linear unit (ReLU), i.e. $\sigma(x) = \max\{0, x\}$. With a slight abuse of notation, 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 $\boldsymbol{x} = (x_1, \cdots, x_d) \in \mathbb{R}^d$.

- The floor function (Floor) is defined as $\lfloor x \rfloor \coloneqq \max\{n : n \le x,\ n \in \mathbb{Z}\}$ for any $x \in \mathbb{R}$.
- 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 $\operatorname{bin} 0.\theta_1\theta_2\cdots\theta_L$ to denote the L-term binary representation of θ, i.e., $\operatorname{bin} 0.\theta_1\theta_2\cdots\theta_L \coloneqq \sum_{\ell=1}^{L} \theta_\ell 2^{-\ell}$.
- 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.

3.2. Proof of Theorem 1.1 and Corollary 2.3

In this section, we will prove Theorem 1.1 and Corollary 2.3. To this end, we first introduce Theorem 3.1 that works only for $[0,1)^d$, regraded as a weaker variant of Theorem 1.1.

Theorem 3.1. *Given an arbitrary continuous function f on $[0,1]^d$, for any $N \in \mathbb{N}^+$, there exist $a_1, a_2, \cdots, a_N \in [0, \frac{1}{2})$ such that*

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le 2\omega_f(\sqrt{d})2^{-N} + \omega_f(\sqrt{d}\,2^{-N}),$$

for any $\boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in [0,1)^d$, where ϕ is defined by a formula in $a_1, a_2, \cdots, a_N$ as follows.

$$\phi(\boldsymbol{x}) = 2\omega_f(\sqrt{d}) \sum_{j=1}^{N} 2^{-j} \sigma_3\Bigg(a_j \cdot \sigma_2\Big(1 + \sum_{i=1}^{d} 2^{(i-1)N} \sigma_1(2^N x_i) \Big) \Bigg) + f(\boldsymbol{0}) - \omega_f(\sqrt{d}).$$

We will prove Theorem 1.1 and Corollary 2.3 based on Theorem 3.1, the proof of which can be found in Section 3.3.

First, let us prove Theorem 1.1 by assuming Theorem 3.1 is true.

Proof of Theorem 1.1. Given any $f \in C([0,1]^d)$, by Lemma 4.2 of Shen et al. (2020) via setting $E = [0,1]^d$ and $S = \mathbb{R}^d$, there exists $g \in C(\mathbb{R}^d)$ such that

- $g(\boldsymbol{x}) = f(\boldsymbol{x})$ for any $\boldsymbol{x} \in E = [0,1]^d$;
- $\omega_g^S(r) = \omega_f^E(r) = \omega_f(r)$ for any $r \ge 0$.

Define $\widetilde{g}(\boldsymbol{x}) \coloneqq g(2\boldsymbol{x})$ for any $\boldsymbol{x} \in \mathbb{R}^d$. By applying Theorem 3.1 to $\widetilde{g} \in C(\mathbb{R}^d)$, there exist $a_1, a_2, \cdots, a_N \in [0, \frac{1}{2})$ such that

$$|\widetilde{\phi}(\boldsymbol{x}) - \widetilde{g}(\boldsymbol{x})| \le 2\omega_{\widetilde{g}}^S(\sqrt{d})2^{-N} + \omega_{\widetilde{g}}^S(\sqrt{d}\,2^{-N}), \quad \text{for any } \boldsymbol{x} \in [0,1)^d, \tag{8}$$

where

$$\widetilde{\phi}(\boldsymbol{x}) = 2\omega_{\widetilde{g}}^S(\sqrt{d}) \sum_{j=1}^{N} 2^{-j} \sigma_3\bigg(a_j \cdot \sigma_2\Big(1 + \sum_{i=1}^{d} 2^{(i-1)N} \sigma_1(2^N x_i) \Big) \bigg) + \widetilde{g}(\boldsymbol{0}) - \omega_{\widetilde{g}}^S(\sqrt{d}).$$

Recall that $f(\boldsymbol{x}) = g(\boldsymbol{x}) = \widetilde{g}(\frac{\boldsymbol{x}}{2})$ for any $\boldsymbol{x} \in E = [0,1]^d$ and

$$\omega_{\widetilde{g}}^S(r) = \omega_g^S(2r) = \omega_f^E(2r) = \omega_f(2r), \quad \text{for any } r \ge 0.$$

Define $\phi(\boldsymbol{x}) \coloneqq \widetilde{\phi}(2\boldsymbol{x})$ for any $\boldsymbol{x} \in \mathbb{R}^d$. Then by Equation (8), for any $\boldsymbol{x} \in [0,1]^d = E$, we have $\frac{\boldsymbol{x}}{2} \in [0, \frac{1}{2}]^d \subseteq [0,1)^d$, implying

$$\begin{aligned} |\phi(\boldsymbol{x}) - f(\boldsymbol{x})| = |\phi(\boldsymbol{x}) - g(\boldsymbol{x})| &= |\widetilde{\phi}(\tfrac{\boldsymbol{x}}{2}) - \widetilde{g}(\tfrac{\boldsymbol{x}}{2})| \\ &\le 2\omega_{\widetilde{g}}^S(\sqrt{d})2^{-N} + \omega_{\widetilde{g}}^S(\sqrt{d}\,2^{-N}) \\ &= 2\omega_f(2\sqrt{d})2^{-N} + \omega_f(2\sqrt{d}\,2^{-N}), \end{aligned}$$

where $\phi(\boldsymbol{x}) \coloneqq \widetilde{\phi}(\frac{\boldsymbol{x}}{2})$ can be represented by

$$2\omega_f(2\sqrt{d}) \sum_{j=1}^{N} 2^{-j} \sigma_3\bigg(a_j \cdot \sigma_2\Big(1 + \sum_{i=1}^{d} 2^{(i-1)N} \sigma_1(2^{N-1} x_i) \Big) \bigg) + f(\boldsymbol{0}) - \omega_f(2\sqrt{d}).$$

With the discussion above, we have proved Theorem 1.1. □

Next, we present the proof of Corollary 2.3 below.

Proof of Corollary 2.3. Given any $f \in C(E)$, by Lemma 4.2 of Shen et al. (2020) via setting $S = \mathbb{R}^d$, there exists $g \in C(\mathbb{R}^d)$ such that

- $g(\boldsymbol{x}) = f(\boldsymbol{x})$ for any $\boldsymbol{x} \in E \subseteq [-R, R]^d$;
- $\omega_g^S(r) = \omega_f^E(r)$ for any $r \ge 0$.

Define

$$\widetilde{g}(\boldsymbol{x}) \coloneqq g(3R\boldsymbol{x} - R), \quad \text{for any } \boldsymbol{x} \in \mathbb{R}^d.$$

By applying Theorem 3.1 to $\widetilde{g} \in C(\mathbb{R}^d)$, there exist $a_1, a_2, \cdots, a_N \in [0, \frac{1}{2})$ such that

$$|\widetilde{\phi}(\boldsymbol{x}) - \widetilde{g}(\boldsymbol{x})| \le 2\omega_{\widetilde{g}}^S(\sqrt{d})2^{-N} + \omega_{\widetilde{g}}^S(\sqrt{d}\,2^{-N}), \quad \text{for any } \boldsymbol{x} \in [0,1)^d, \tag{9}$$

where

$$\widetilde{\phi}(\boldsymbol{x}) = 2\omega_{\widetilde{g}}^S(\sqrt{d}) \sum_{j=1}^{N} 2^{-j} \sigma_3\Big(a_j \cdot \sigma_2\Big(1 + \sum_{i=1}^{d} 2^{(i-1)N} \sigma_1(2^N x_i) \Big) \Big) + \widetilde{g}(\mathbf{0}) - \omega_{\widetilde{g}}^S(\sqrt{d}).$$

Recall that $f(\boldsymbol{x}) = g(\boldsymbol{x}) = \widetilde{g}(\frac{\boldsymbol{x}+R}{3R})$ for any $\boldsymbol{x} \in E \subseteq [-R, R]^d$ and

$$\omega_{\widetilde{g}}^S(r) = \omega_g^S(3Rr) = \omega_f^E(3Rr), \qquad \text{for any } r \ge 0.$$

Define $\phi(\boldsymbol{x}) \coloneqq \widetilde{\phi}(\frac{\boldsymbol{x}+R}{3R})$ for any $\boldsymbol{x} \in \mathbb{R}^d$. Then by Equation (9), for any $\boldsymbol{x} \in E \subseteq [-R, R]^d$, we have $\frac{\boldsymbol{x}+R}{3R} \in [0, \frac{2}{3}]^d \subseteq [0,1)^d$, implying

$$\begin{aligned} |\phi(\boldsymbol{x}) - f(\boldsymbol{x})| = |\phi(\boldsymbol{x}) - g(\boldsymbol{x})| &= |\widetilde{\phi}(\tfrac{\boldsymbol{x}+R}{3R}) - \widetilde{g}(\tfrac{\boldsymbol{x}+R}{3R})| \\ &\le 2\omega_{\widetilde{g}}^S(\sqrt{d})2^{-N} + \omega_{\widetilde{g}}^S(\sqrt{d}\,2^{-N}) \\ &= 2\omega_f(3R\sqrt{d})2^{-N} + \omega_f(3R\sqrt{d}\,2^{-N}), \end{aligned}$$

where $\phi(\boldsymbol{x}) = \widetilde{\phi}(\frac{\boldsymbol{x}+R}{3R})$ can be represented by

$$2\omega_f(3R\sqrt{d}) \sum_{j=1}^{N} 2^{-j} \sigma_3\Big(a_j \cdot \sigma_2\Big(1 + \sum_{i=1}^{d} 2^{(i-1)N} \sigma_1(2^N \tfrac{x_i+R}{3R}) \Big) \Big) + C_f,$$

where $C_f = \widetilde{g}(\mathbf{0}) - \omega_{\widetilde{g}}^S(\sqrt{d})$ is a constant essentially determined by f. With the discussion above, we have proved Corollary 2.3. □

3.3. Proof of Theorem 3.1

To prove Theorem 3.1, we first present the proof sketch. Shortly speaking, we construct piecewise constant functions to approximate continuous functions. There are five key steps in our construction.

1. Normalize f as $\widetilde{f}$ satisfying $\widetilde{f}(\boldsymbol{x}) \in [0,1]$ for any $\boldsymbol{x} \in [0,1]^d$, divide $[0,1)^d$ into a set of non-overlapping cubes $\{Q_{\boldsymbol{\beta}}\}_{\boldsymbol{\beta} \in \{0,1,\cdots,J-1\}^d}$, and denote $\boldsymbol{x}_{\boldsymbol{\beta}}$ as the vertex of $Q_{\boldsymbol{\beta}}$ with minimum $\|\cdot\|_1$ norm, where J is an integer determined later. See Figure 3 for the illustrations of $Q_{\boldsymbol{\beta}}$ and $\boldsymbol{x}_{\boldsymbol{\beta}}$.

2. Construct a vector-valued function $\mathbf{\Phi}_1 : \mathbb{R}^d \to \mathbb{R}^d$ projecting the whole cube $Q_{\boldsymbol{\beta}}$ to the index $\boldsymbol{\beta}$, i.e., $\mathbf{\Phi}_1(\boldsymbol{x}) = \boldsymbol{\beta}$ for all $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and each $\boldsymbol{\beta} \in \{0, 1, \cdots, J-1\}^d$.

3. Construct a linear function $\phi_2 : \mathbb{R}^d \to \mathbb{R}$ bijectively mapping $\boldsymbol{\beta} \in \{0, 1, \cdots, J-1\}^d$ to $\phi_2(\boldsymbol{\beta}) \in \{1, 2, \cdots, J^d\}$.

4. Construct a function $\phi_3 : \mathbb{R} \to \mathbb{R}$ mapping $\phi_2(\boldsymbol{\beta}) \in \{1, 2, \cdots, J^d\}$ approximately to $\widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}})$, i.e., $\phi_3(\phi_2(\boldsymbol{\beta})) \approx \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}})$ for each $\boldsymbol{\beta} \in \{0, 1, \cdots, J-1\}^d$.

5. Define $\widetilde{\phi} \coloneqq \phi_3 \circ \phi_2 \circ \mathbf{\Phi}_1$. Then $\widetilde{\phi}$ is a piecewise constant function mapping $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ to $\phi_3(\phi_2(\boldsymbol{\beta})) \approx \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}})$ for each $\boldsymbol{\beta} \in \{0, 1, \cdots, J-1\}^d$, implying $\widetilde{\phi} \approx \widetilde{f}$. Finally, re-scale and shift $\widetilde{\phi}$ to obtain the final function ϕ approximating f well.

Recall that

$$\sigma_1(x) \coloneqq \lfloor x \rfloor, \quad \sigma_2(x) \coloneqq 2^x, \quad \text{and} \quad \sigma_3 \coloneqq \mathcal{T}(x - \lfloor x \rfloor - \tfrac{1}{2}), \quad \text{for any } x \in \mathbb{R},$$

where

$$\mathcal{T}(x) \coloneqq \mathbb{1}_{x \geq 0} = \begin{cases} 1, & x \geq 0, \\ 0, & x < 0, \end{cases} \quad \text{for any } x \in \mathbb{R}.$$

Step 1 and 5 are straightforward. To implement Step 2, we introduce σ_1 since it can help to significantly simplify the construction of the vector-valued projecting function $\mathbf{\Phi}_1$. The implementation of Step 3 is based on the J-ary representation, namely, let $\phi_2(\boldsymbol{x}) \coloneqq 1 + \sum_{i=1}^{d} J^{i-1} x_i$. The most technical step above is Step 4, which is essentially a point fitting problem. Solving such a problem eventually relies on the bit extraction technique in Shen et al. (2020); Lu et al. (2020); Shen et al. (2020); Harvey et al. (2017); Bartlett et al. (1998); Zhang (2020); Yarotsky (2018). To extract sufficient many bits with a limited neuron budget, we introduce two powerful activation functions σ_2 and σ_3, as shown in the proposition below.

Proposition 3.2. *Given any $K \in \mathbb{N}^+$ and arbitrary $\theta_1, \theta_2, \cdots, \theta_K \in \{0, 1\}$, it holds that*

$$\sigma_3(2^k \cdot a) = \theta_k, \quad \text{for any } k \in \{1, 2, \cdots, K\},$$

where

$$a = \sum_{j=1}^{K} 2^{-j-1} \cdot \theta_j \ \in [0, \tfrac{1}{2}).$$

Proof. Since $\theta_j \in \{0,1\}$ for $j \in \{1,2,\cdots,K\}$, we have

$$0 \le \sum_{j=1}^{K} 2^{-j-1} \cdot \theta_j \le \sum_{j=1}^{K} 2^{-j-1} < \tfrac{1}{2},$$

implying $a \in [0, \frac{1}{2})$.

Next, fix $k \in \{1,2,\cdots,K\}$ for the proof below. It holds that

$$2^k \cdot a = 2^k \cdot \sum_{j=1}^{K} 2^{-j-1} \cdot \theta_j = \underbrace{\sum_{j=1}^{k-1} 2^{k-j-1} \cdot \theta_j}_{\text{an integer}} + \overbrace{\tfrac{1}{2}\theta_k}^{0 \text{ or } \frac{1}{2}} + \underbrace{\sum_{j=k+1}^{K} 2^{k-j-1} \cdot \theta_j}_{\text{in } [0, \frac{1}{2})}. \quad (10)$$

[2]

Clearly, the first term in Equation (10) $\sum_{j=1}^{k-1} 2^{k-j-1} \cdot \theta_j$ is a non-negative integer since $\theta_j \in \{0,1\}$ for any $j \in \{1,2,\cdots,K\}$. As for the third term in Equation (10), we have

$$0 \le \sum_{j=k+1}^{K} 2^{k-j-1} \cdot \theta_j \le \sum_{j=k+1}^{K} 2^{k-j-1} < \tfrac{1}{2}$$

Therefore, by Equation (10), we have

$$2^k \cdot a \in \bigcup_{n \in \mathbb{N}} [n, n+\tfrac{1}{2}), \text{ if } \theta_k = 0 \quad \text{and} \quad 2^k \cdot a \in \bigcup_{n \in \mathbb{N}} [n+\tfrac{1}{2}, n+1), \text{ if } \theta_k = 1. \quad (11)$$

Recall that $\sigma_3(x) = \mathcal{T}(x - \lfloor x \rfloor - \frac{1}{2})$, where $\mathcal{T}(x) = \begin{cases} 1, & x \ge 0, \\ 0, & x < 0 \end{cases}$. It is easy to verify that

$$\sigma_3(x) = 0 \text{ if } x \in \bigcup_{n \in \mathbb{N}} [n, n+\tfrac{1}{2}) \quad \text{and} \quad \sigma_3(x) = 1 \text{ if } x \in \bigcup_{n \in \mathbb{N}} [n+\tfrac{1}{2}, n+1).$$

If $\theta_k = 0$, by Equation (11), we have

$$2^k \cdot a \in \bigcup_{n \in \mathbb{N}} [n, n+\tfrac{1}{2}) \implies \sigma_3(2^k \cdot a) = 0 = \theta_k.$$

Similarly, if $\theta_k = 1$, by Equation (11), we have

$$2^k \cdot a \in \bigcup_{n \in \mathbb{N}} [n+\tfrac{1}{2}, n+1) \implies \sigma_3(2^k \cdot a) = 1 = \theta_k.$$

Since $k \in \{1,2,\cdots,K\}$ is arbitrary, we have $\sigma_3(2^k \cdot a) = \theta_k$ for any $k \in \{1,2,\cdots,K\}$. So we finish the proof. □

[2] By convention, $\sum_{j=n}^{m} a_j = 0$ if $n > m$ no matter what a_j is for each j.

We would like to point out that Proposition 3.2 indicates that the VC-dimension of the function space

$$\{f : f(x) = \sigma_3(a \cdot x), \text{ for } a \in \mathbb{R}\}$$

is infinity, which implies that the VC-dimension of FLES networks is also infinity. As discussed previously in Section 2.2, having an infinite VC-dimension is a necessary condition for our FLES networks to attain super approximation power.

With Proposition 3.2 in hand, we are ready to prove Theorem 3.1.

Proof of Theorem 3.1. The proof consists of five steps.

Step 1: Set up.

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

$$\widetilde{f} \coloneqq \frac{f - f(\mathbf{0}) + \omega_f(\sqrt{d})}{2\omega_f(\sqrt{d})}. \tag{12}$$

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

Set $J = 2^N$ and divide $[0,1)^d$ into J^d cubes $\{Q_{\boldsymbol{\beta}}\}_{\boldsymbol{\beta}}$. To be exact, defined $\boldsymbol{x}_{\boldsymbol{\beta}} \coloneqq \boldsymbol{\beta}/J$ and

$$Q_{\boldsymbol{\beta}} \coloneqq \Big\{\boldsymbol{x} = (x_1, x_2, \cdots, x_d) : x_i \in [\tfrac{\beta_i}{J}, \tfrac{\beta_i+1}{J}) \text{ for } i = 1, 2, \cdots, d\Big\},$$

for each $\boldsymbol{\beta} = (\beta_1, \beta_2, \cdots, \beta_d) \in \{0, 1, \cdots, J-1\}^d$. See Figure 3 for illustrations.

Step 2: Construct $\boldsymbol{\Phi}_1$ mapping $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ to $\boldsymbol{\beta}$ for each $\boldsymbol{\beta} \in \{0, 1, \cdots, J-1\}^d$.

Define

$$\boldsymbol{\Phi}_1(\boldsymbol{x}) \coloneqq \Big(\sigma_1(Jx_1), \sigma_1(Jx_2), \cdots, \sigma_1(Jx_d)\Big) = \Big(\lfloor Jx_1 \rfloor, \lfloor Jx_2 \rfloor, \cdots, \lfloor Jx_d \rfloor\Big),$$

for any $\boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d$. Then, for any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and each $\boldsymbol{\beta} \in \{0, 1, \cdots, J-1\}^d$, we have

$$\boldsymbol{\Phi}_1(\boldsymbol{x}) = \Big(\lfloor Jx_1 \rfloor, \lfloor Jx_2 \rfloor, \cdots, \lfloor Jx_d \rfloor\Big) = (\beta_1, \beta_2, \cdots, \beta_d) = \boldsymbol{\beta}. \tag{13}$$

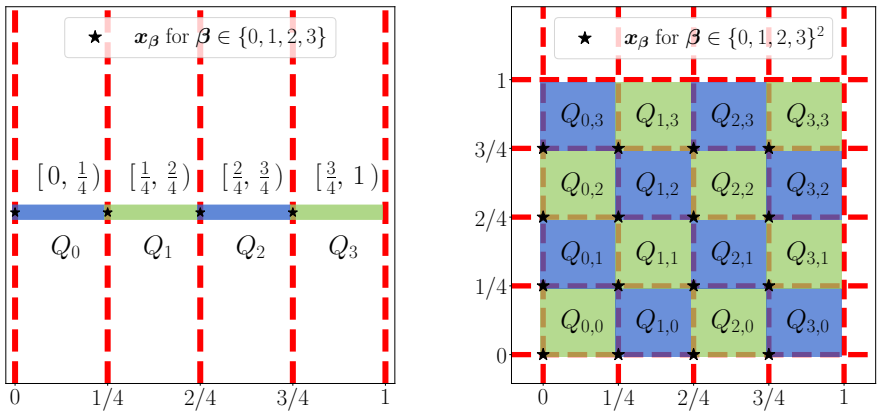


Figure 3: Illustrations of $Q_{\boldsymbol{\beta}}$ and $\boldsymbol{x}_{\boldsymbol{\beta}}$ for any $\boldsymbol{\beta} \in \{0,1,\cdots,J-1\}^d$. (a) $J=4,\ d=1$. (b) $J=4,\ d=2$.

Step 3: Construct ϕ_2 bijectively mapping $\boldsymbol{\beta} \in \{0,1,\cdots,J-1\}^d$ to $\phi_2(\boldsymbol{\beta}) \in \{1,2,\cdots,J^d\}$.

Inspired by the J-ary representation, we define a linear function

$$\phi_2(\boldsymbol{x}) := 1 + \sum_{i=1}^{d} J^{i-1} x_i, \quad \text{for each } \boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d.$$

Then ϕ_2 is a bijection from $\{0,1,\cdots,J-1\}^d$ to $\{1,2,\cdots,J^d\}$.

Step 4: Construct ϕ_3 mapping $\phi_2(\boldsymbol{\beta}) \in \{1,2,\cdots,J^d\}$ approximately to $\widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}})$.

For each $k \in \{1,2,\cdots,J^d\}$, there exists a unique $\boldsymbol{\beta} \in \{0,1,\cdots,J-1\}^d$ such that $\phi_2(\boldsymbol{\beta}) = k$. Thus, define

$$\xi_k := \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}}) \in [0,1], \quad \text{for any } k \in \{1,2,\cdots,J^d\} \text{ with } k = \phi_2(\boldsymbol{\beta}). \tag{14}$$

For each $k \in \{1,2,\cdots,J^d\}$, there exist $\theta_{k,1}, \theta_{k,2}, \cdots, \theta_{k,N} \in \{0,1\}$ such that

$$|\xi_k - \mathrm{bin}0.\theta_{k,1}\theta_{k,2}\cdots\theta_{k,N}| \le 2^{-N}. \tag{15}$$

For each $j \in \{1,2,\cdots,N\}$, by Proposition 3.2 (set $K = J^d$ therein), there exists $a_j \in [0, \frac{1}{2})$ such that

$$\sigma_3(2^k \cdot a_j) = \theta_{k,j}, \quad \text{for any } k \in \{1,2,\cdots,J^d\}.$$

Define

$$\phi_3(x) \coloneqq \sum_{j=1}^{N} 2^{-j}\sigma_3\big(a_j \cdot \sigma_2(x)\big) = \sum_{j=1}^{N} 2^{-j}\sigma_3(2^x \cdot a_j), \quad \text{for any } x \in \mathbb{R}.$$

Then, for any $k \in \{1, 2, \cdots, J^d\}$, we have

$$\phi_3(k) = \sum_{j=1}^{N} 2^{-j}\sigma_3(2^k \cdot a_j) = \sum_{j=1}^{N} 2^{-j} \cdot \theta_{k,j} = \mathrm{bin}0.\theta_{k,1}\theta_{k,2}\cdots\theta_{k,N}. \tag{16}$$

Step 5: Define $\widetilde{\phi} \coloneqq \phi_3 \circ \phi_2 \circ \mathbf{\Phi}_1$ approximating $\widetilde{f}$ well, and re-scale and shift $\widetilde{\phi}$ to obtain ϕ approximating f well.

Define $\widetilde{\phi} \coloneqq \phi_3 \circ \phi_2 \circ \mathbf{\Phi}_1$, by Equation (13), (14), (15), and (16), we have, for any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and each $\boldsymbol{\beta} \in \{0, 1, \cdots, J-1\}^d$ with $k = \phi_2(\boldsymbol{\beta})$,

$$\begin{aligned}
|\widetilde{\phi}(\boldsymbol{x}) - \widetilde{f}(\boldsymbol{x})| &= |\phi_3 \circ \phi_2 \circ \mathbf{\Phi}_1(\boldsymbol{x}) - \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}})| + |\widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}}) - \widetilde{f}(\boldsymbol{x})| \\
&\le |\phi_3 \circ \phi_2(\boldsymbol{\beta}) - \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}})| + \omega_{\widetilde{f}}(\tfrac{\sqrt{d}}{J}) \\
&\le |\phi_3(k) - \xi_k| + \omega_{\widetilde{f}}(\tfrac{\sqrt{d}}{J}) \\
&\le |\mathrm{bin}0.\theta_{k,1}\theta_{k,2}\cdots\theta_{k,N} - \xi_k| + \omega_{\widetilde{f}}(\tfrac{\sqrt{d}}{J}) \le 2^{-N} + \omega_{\widetilde{f}}(\tfrac{\sqrt{d}}{J}).
\end{aligned}$$

Finally, define $\phi \coloneqq 2\omega_f(\sqrt{d})\widetilde{\phi} + f(\mathbf{0}) - \omega_f(\sqrt{d})$. Equation (12) implies $\omega_f(r) = 2\omega_f(\sqrt{d})\omega_{\widetilde{f}}(r)$ for any $r \ge 0$, deducing

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

for any $\boldsymbol{x} \in \bigcup_{\boldsymbol{\beta}\in\{0,1,\cdots,J-1\}^d} Q_{\boldsymbol{\beta}} = [0, 1)^d$. It follows from $J = 2^N$ and the definitions of $\mathbf{\Phi}_1$, ϕ_2, and ϕ_3 that

$$\begin{aligned}
\phi(\boldsymbol{x}) &= 2\omega_f(\sqrt{d})\phi_3 \circ \phi_2 \circ \mathbf{\Phi}_1(\boldsymbol{x}) + f(\mathbf{0}) - \omega_f(\sqrt{d}) \\
&= 2\omega_f(\sqrt{d})\phi_3\Big(1 + \sum_{i=1}^{d} J^{i-1}\sigma_1(Jx_i)\Big) + f(\mathbf{0}) - \omega_f(\sqrt{d}) \\
&= 2\omega_f(\sqrt{d})\sum_{j=1}^{N} 2^{-j}\sigma_3\bigg(a_j \cdot \sigma_2\Big(1 + \sum_{i=1}^{d} 2^{(i-1)N}\sigma_1(2^N x_i)\Big)\bigg) + f(\mathbf{0}) - \omega_f(\sqrt{d}).
\end{aligned}$$

So we finish the proof. □

4. Approximation with continuous activation functions

As discussed previously, our FLES networks can attain super approximation power. However, two activation functions in FLES networks are piecewise constant functions that would lead to challenges in numerical algorithm design. It is interesting to explore continuous activation functions achieving similar results. To this end, we introduce three new activation functions as follows. First, for any $\delta \in (0,1)$, we define

$$\varrho_{1,\delta}(x) \coloneqq \begin{cases} n-1, & x \in [n-1, n-\delta] \text{ for any } n \in \mathbb{Z}, \\ (x-n+\delta)/\delta, & x \in (n-\delta, n] \text{ for any } n \in \mathbb{Z}. \end{cases}$$

In fact, $\varrho_{1,\delta}$ can be regarded as a "continuous version" of the floor function. Next, we define

$$\varrho_2(x) \coloneqq 3^x, \quad \text{and} \quad \varrho_3(x) \coloneqq \widetilde{\mathcal{T}}(\cos(2\pi x)), \quad \text{for any } x \in \mathbb{R},$$

where

$$\widetilde{\mathcal{T}}(x) \coloneqq \begin{cases} 0, & x \in (\cos(\frac{4\pi}{9}), \infty), \\ 1 - x/\cos(\frac{4\pi}{9}), & x \in [0, \cos(\frac{4\pi}{9})], \\ 1, & x \in (-\infty, 0), \end{cases} \qquad \text{for any } x \in \mathbb{R},$$

is a continuous piecewise linear function. ϱ_2 plays the same role of $\sigma_2(x) = 2^x$ and ϱ_3 is essentially a "continuous version" of σ_3 in FLES.

With these three activation functions in hand, we have the following theorem.

Theorem 4.1. *Let f be an arbitrary continuous function defined on $[0,1]^d$. For any $\delta \in (0,1)$, $N \in \mathbb{N}^+$, and $p \in [1,\infty)$, there exist $a_1, a_2, \cdots, a_N \in [0, \frac{2}{9})$ such that*

$$\|\phi - f\|^p_{L^p([0,1]^d)} \le \Big(2\omega_f(\sqrt{d})2^{-N} + \omega_f(\sqrt{d}\,2^{-N})\Big)^p + 2d\delta\big(|f(\mathbf{0})| + \omega_f(\sqrt{d})\big)^p,$$

where ϕ is defined by a formula in $a_1, a_2, \cdots, a_N$ as follows

$$\phi(\boldsymbol{x}) = 2\omega_f(\sqrt{d}) \sum_{j=1}^{N} 2^{-j} \varrho_3\bigg(a_j \cdot \varrho_2\Big(1 + \sum_{i=1}^{d} 2^{(i-1)N} \varrho_{1,\delta}(2^N x_i) \Big) \bigg) + f(\mathbf{0}) - \omega_f(\sqrt{d}).$$

The approximation error in Theorem 4.1 is characterized by L^p-norm for $p \in [1,\infty)$ instead of a pointwise error estimate in Theorem 1.1. By

using ideas in Lu et al. (2020); Zhang (2020), we can extend this result to L^∞-norm. However, this extension requires $2d+3$ hidden layers instead of 3 hidden layers. Since our focus here is the approximation using three hidden layers, we will leave this extension as future work.

To prove Theorem 4.1, we need the following proposition.

Proposition 4.2. *Given any $K \in \mathbb{N}^+$ and arbitrary $\theta_1, \theta_2, \cdots, \theta_K \in \{0,1\}$, it holds that*

$$\varrho_3(3^k \cdot a) = \theta_k, \quad \text{for any } k \in \{1,2,\cdots,K\},$$

where

$$a = \sum_{j=1}^{K} 3^{-j-1} \cdot \theta_j \ \in [0, \tfrac{2}{9}).$$

Proof. Since $\theta_j \in \{0,1\}$ for $j \in \{1,2,\cdots,K\}$, we have

$$0 \le \sum_{j=1}^{K} 3^{-j-1} \cdot \theta_j \le \sum_{j=1}^{K} 3^{-j-1} < \tfrac{2}{9},$$

implying $a \in [0, \frac{2}{9})$.

Next, fix $k \in \{1,2,\cdots,K\}$ for the proof below. It holds that

$$3^k \cdot a = 3^k \cdot \sum_{j=1}^{K} 3^{-j-1} \cdot \theta_j = \underbrace{\sum_{j=1}^{k-1} 3^{k-j-1} \cdot \theta_j}_{\text{an integer}} + \overbrace{\tfrac{1}{3}\theta_k}^{0 \text{ or } \frac{1}{3}} + \underbrace{\sum_{j=k+1}^{K} 3^{k-j-1} \cdot \theta_j}_{\text{in } [0, \frac{2}{9})}. \tag{17}$$

Clearly, the first term $\sum_{j=1}^{k-1} 3^{k-j-1} \cdot \theta_j$ in Equation (17) is a non-negative integer since $\theta_j \in \{0,1\}$ for any $j \in \{1,2,\cdots,K\}$. As for the third term in Equation (17), we have

$$0 \le \sum_{j=k+1}^{K} 3^{k-j-1} \cdot \theta_j \le \sum_{j=k+1}^{K} 3^{k-j-1} < \tfrac{2}{9}.$$

Recall that

$$\cos(2\pi x) \in (\cos(\tfrac{4\pi}{9}), 1], \quad \text{for any } x \in \bigcup_{n \in \mathbb{N}} [n, n + \tfrac{2}{9}),$$

and

$$\cos(2\pi x) \in [-1, \cos(\tfrac{2\pi}{3})] \subseteq [-1, 0], \quad \text{for any } x \in \bigcup_{n\in\mathbb{N}} [n + \tfrac{1}{3}, n + \tfrac{5}{9}).$$

Recall that

$$\widetilde{\mathcal{T}}(x) \coloneqq \begin{cases} 0, & x \in (\cos(\frac{4\pi}{9}), \infty), \\ 1 - x/\cos(\frac{4\pi}{9}), & x \in [0, \cos(\frac{4\pi}{9})], \\ 1, & x \in (-\infty, 0), \end{cases} \qquad \text{for any } x \in \mathbb{R}.$$

Therefore, if $\theta_k = 0$, by Equation (17), we have

$$3^k \cdot a \in \bigcup_{n\in\mathbb{N}} [n, n + \tfrac{2}{9}) \quad \Longrightarrow \quad \varrho_3(3^k \cdot a) = \widetilde{\mathcal{T}}(\cos(2\pi \cdot 3^k \cdot a)) = 0 = \theta_k.$$

Similarly, if $\theta_k = 1$, by Equation (17), we have

$$3^k \cdot a \in \bigcup_{n\in\mathbb{N}} [n + \tfrac{1}{3}, n + \tfrac{5}{9}) \quad \Longrightarrow \quad \varrho_3(3^k \cdot a) = \widetilde{\mathcal{T}}(\cos(2\pi \cdot 3^k \cdot a)) = 1 = \theta_k.$$

Since $k \in \{1, 2, \cdots, K\}$ is arbitrary, we have $\varrho_3(3^k \cdot a) = \theta_k$ for any $k \in \{1, 2, \cdots, K\}$. So we finish the proof. □

Before proving Theorem 4.1, let us define a small region as follows to simplify the notation. Given any $J \in \mathbb{N}^+$ and $\delta \in (0, 1)$, define a small region $\Lambda([0, 1]^d, J, \delta)$ as

$$\Lambda([0, 1]^d, J, \delta) \coloneqq \bigcup_{i=1}^{d} \left\{ \boldsymbol{x} = (x_1, \cdots, x_d) \in [0, 1]^d : x_i \in \bigcup_{j=1}^{J-1} [\tfrac{j-\delta}{J}, \tfrac{j}{J}] \right\}. \tag{18}$$

In particular, $\Lambda([0, 1]^d, J, \delta) = \varnothing$ if $J = 1$. See Figure 4 for two examples of trifling regions.

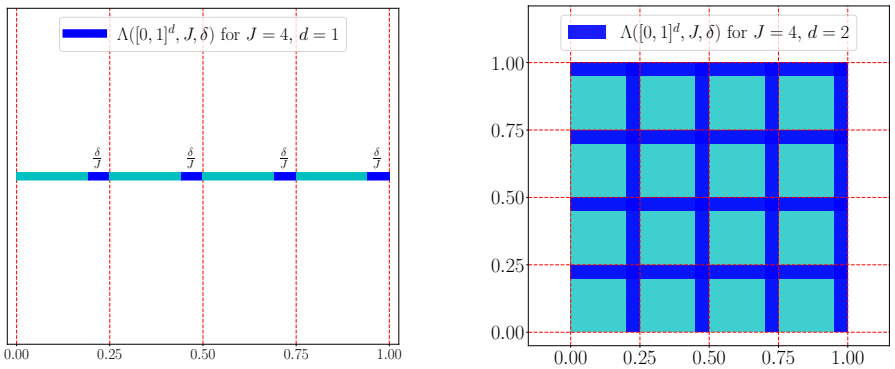
With Proposition 4.2 in hand, we are ready to prove Theorem 4.1.

Proof of Theorem 4.1. The proof consists of five steps.

Step 1: Set up.

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

$$\widetilde{f} \coloneqq \frac{f - f(\boldsymbol{0}) + \omega_f(\sqrt{d})}{2\omega_f(\sqrt{d})}. \tag{19}$$



(a) (b)

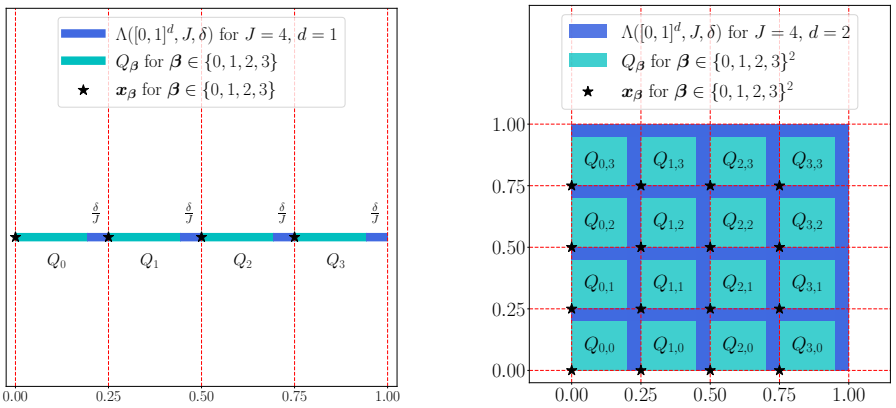
Figure 4: Illustrations of $\Lambda([0,1]^d, J, \delta)$. (a) $J = 4$, $d = 1$. (b) $J = 4$, $d = 2$.

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

Set $J = 2^N$ and divide $[0,1]^d$ into J^d cubes $\{Q_{\boldsymbol{\beta}}\}_{\boldsymbol{\beta}}$ and a small region $\Lambda([0,1]^d, J, \delta)$. To be exact, define $\boldsymbol{x}_{\boldsymbol{\beta}} \coloneqq \boldsymbol{\beta}/J$ and

$$Q_{\boldsymbol{\beta}} \coloneqq \Big\{\boldsymbol{x} = (x_1, x_2, \cdots, x_d) : x_i \in [\tfrac{\beta_i}{J}, \tfrac{\beta_i + 1 - \delta}{J}] \text{ for } i = 1, 2, \cdots, d\Big\},$$

for each $\boldsymbol{\beta} = (\beta_1, \beta_2, \cdots, \beta_d) \in \{0, 1, \cdots, J-1\}^d$. See Figure 5 for illustrations.



(a) (b)

Figure 5: Illustrations of $\Lambda([0,1]^d, J, \delta)$, $Q_{\boldsymbol{\beta}}$, and $\boldsymbol{x}_{\boldsymbol{\beta}}$ for any $\boldsymbol{\beta} \in \{0, 1, \cdots, J-1\}^d$. (a) $J = 4$, $d = 1$. (b) $J = 4$, $d = 2$.

Step 2: Construct $\boldsymbol{\Phi}_1$ mapping $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ to $\boldsymbol{\beta}$ for each $\boldsymbol{\beta} \in \{0, 1, \cdots, J-1\}^d$.

Define

$$\mathbf{\Phi}_1(\boldsymbol{x}) \coloneqq \Big(\varrho_{1,\delta}(Jx_1), \varrho_{1,\delta}(Jx_2), \cdots, \varrho_{1,\delta}(Jx_d)\Big),$$

for any $\boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d$. Then, for any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and each $\boldsymbol{\beta} \in \{0, 1, \cdots, J-1\}^d$, we have

$$\mathbf{\Phi}_1(\boldsymbol{x}) = \Big(\varrho_{1,\delta}(Jx_1), \varrho_{1,\delta}(Jx_2), \cdots, \varrho_{1,\delta}(Jx_d)\Big) = (\beta_1, \beta_2, \cdots, \beta_d) = \boldsymbol{\beta}. \tag{20}$$

Step 3: Construct ϕ_2 bijectively mapping $\boldsymbol{\beta} \in \{0, 1, \cdots, J-1\}^d$ to $\phi_2(\boldsymbol{\beta}) \in \{1, 2, \cdots, J^d\}$.

Inspired by the J-ary representation, we define an affine linear map

$$\phi_2(\boldsymbol{x}) \coloneqq 1 + \sum_{i=1}^{d} J^{i-1} x_i, \quad \text{for each } \boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d.$$

Then ϕ_2 is a bijection from $\{0, 1, \cdots, J-1\}^d$ to $\{1, 2, \cdots, J^d\}$.

Step 4: Construct ϕ_3 mapping $\phi_2(\boldsymbol{\beta}) \in \{1, 2, \cdots, J^d\}$ approximately to $\widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}})$.

For each $k \in \{1, 2, \cdots, J^d\}$, there exists a unique $\boldsymbol{\beta} \in \{0, 1, \cdots, J-1\}^d$ such that $\phi_2(\boldsymbol{\beta}) = k$. Thus, define

$$\xi_k \coloneqq \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}}) \in [0, 1], \quad \text{for any } k \in \{1, 2, \cdots, J^d\} \text{ with } k = \phi_2(\boldsymbol{\beta}). \tag{21}$$

For each $k \in \{1, 2, \cdots, J^d\}$, there exist $\theta_{k,1}, \theta_{k,2}, \cdots, \theta_{k,N} \in \{0, 1\}$ such that

$$|\xi_k - \mathrm{bin}0.\theta_{k,1}\theta_{k,2}\cdots\theta_{k,N}| \le 2^{-N}. \tag{22}$$

For each $j \in \{1, 2, \cdots, N\}$, by Proposition 4.2 (set $K = J^d$ therein), there exists $a_j \in [0, \frac{2}{9})$ such that

$$\varrho_3(3^k \cdot a_j) = \theta_{k,j}, \quad \text{for any } k \in \{1, 2, \cdots, J^d\}.$$

Define

$$\phi_3(x) \coloneqq \sum_{j=1}^{N} 2^{-j} \varrho_3\big(a_j \cdot \varrho_2(x)\big) = \sum_{j=1}^{N} 2^{-j} \varrho_3(3^x \cdot a_j), \quad \text{for any } x \in \mathbb{R}.$$

We get

$$\varrho_3(x) \in [0,1], \quad \text{for any } x \in \mathbb{R} \quad \Longrightarrow \quad \phi_3(x) \in [0,1], \quad \text{for any } x \in \mathbb{R}, \tag{23}$$

and

$$\phi_3(k) = \sum_{j=1}^{N} 2^{-j} \varrho_3(3^k \cdot a_j) = \sum_{j=1}^{N} 2^{-j} \cdot \theta_{k,j} = \text{bin}0.\theta_{k,1}\theta_{k,2}\cdots\theta_{k,N}, \tag{24}$$

for any $k \in \{1, 2, \cdots, J^d\}$.

Step 5: Define $\widetilde{\phi} \coloneqq \phi_3 \circ \phi_2 \circ \mathbf{\Phi}_1$ approximating $\widetilde{f}$ well, and re-scale and shift $\widetilde{\phi}$ to obtain ϕ approximating f well.

Define $\widetilde{\phi} \coloneqq \phi_3 \circ \phi_2 \circ \mathbf{\Phi}_1$, by Equation (20), (21), (22), and (24), we have, for any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and each $\boldsymbol{\beta} \in \{0, 1, \cdots, J-1\}^d$ with $k = \phi_2(\boldsymbol{\beta})$,

$$\begin{aligned}
|\widetilde{\phi}(\boldsymbol{x}) - \widetilde{f}(\boldsymbol{x})| &= |\phi_3 \circ \phi_2 \circ \mathbf{\Phi}_1(\boldsymbol{x}) - \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}})| + |\widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}}) - \widetilde{f}(\boldsymbol{x})| \\
&\leq |\phi_3 \circ \phi_2(\boldsymbol{\beta}) - \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}})| + \omega_{\widetilde{f}}(\tfrac{\sqrt{d}}{J}) \\
&\leq |\phi_3(k) - \xi_k| + \omega_{\widetilde{f}}(\tfrac{\sqrt{d}}{J}) \\
&\leq |\text{bin}0.\theta_{k,1}\theta_{k,2}\cdots\theta_{k,N} - \xi_k| + \omega_{\widetilde{f}}(\tfrac{\sqrt{d}}{J}) \leq 2^{-N} + \omega_{\widetilde{f}}(\tfrac{\sqrt{d}}{J}).
\end{aligned}$$

Finally, define $\phi \coloneqq 2\omega_f(\sqrt{d})\widetilde{\phi} + f(\mathbf{0}) - \omega_f(\sqrt{d})$. Equation (19) implies $\omega_f(r) = 2\omega_f(\sqrt{d})\omega_{\widetilde{f}}(r)$ for any $r \geq 0$, deducing

$$\begin{aligned}
|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| = 2\omega_f(\sqrt{d})|\widetilde{\phi}(\boldsymbol{x}) - \widetilde{f}(\boldsymbol{x})| &\leq 2\omega_f(\sqrt{d})\big(2^{-N} + \omega_{\widetilde{f}}(\tfrac{\sqrt{d}}{J})\big) \\
&= 2\omega_f(\sqrt{d})2^{-N} + \omega_f(\tfrac{\sqrt{d}}{J}),
\end{aligned}$$

for any $\boldsymbol{x} \in \bigcup_{\boldsymbol{\beta}\in\{0,1,\cdots,J-1\}^d} Q_{\boldsymbol{\beta}}$. By Equation (23) and the definition of

$$\phi \coloneqq 2\omega_f(\sqrt{d})\phi_3 \circ \phi_2 \circ \mathbf{\Phi}_1 + f(\mathbf{0}) - \omega_f(\sqrt{d}),$$

we have $\|\phi\|_{L^\infty(\mathbb{R}^d)} \leq |f(\mathbf{0})| + \omega_f(\sqrt{d})$. Let $\mu(\cdot)$ denote the Lebesgue measure. Recall that $\|f\|_{[0,1]^d} \leq |f(\mathbf{0})| + \omega_f(\sqrt{d})$. If follows from $\mu(\Lambda([0,1]^d, J, \delta)) \leq$

$Jd\frac{\delta}{J} = d\delta$ that

$$
\begin{aligned}
&\|\phi - f\|^p_{L^p([0,1]^d)} = \int_{[0,1]^d} |\phi(\boldsymbol{x}) - f(\boldsymbol{x})|^p d\boldsymbol{x} \\
&= \sum_{\boldsymbol{\beta}\in\{0,1,\cdots,J-1\}^d} \int_{Q_{\boldsymbol{\beta}}} |\phi(\boldsymbol{x}) - f(\boldsymbol{x})|^p d\boldsymbol{x} + \int_{\Lambda([0,1]^d,J,\delta)} |\phi(\boldsymbol{x}) - f(\boldsymbol{x})|^p d\boldsymbol{x} \\
&\le \sum_{\boldsymbol{\beta}\in\{0,1,\cdots,J-1\}^d} \mu(Q_{\boldsymbol{\beta}}) \Big(2\omega_f(\sqrt{d})2^{-N} + \omega_f(\tfrac{\sqrt{d}}{J})\Big)^p + \big(2|f(\mathbf{0})| + 2\omega_f(\sqrt{d})\big)^p d\delta \\
&\le \Big(2\omega_f(\sqrt{d})2^{-N} + \omega_f(\sqrt{d}\,2^{-N})\Big)^p + 2^p d\delta\big(|f(\mathbf{0})| + \omega_f(\sqrt{d})\big)^p.
\end{aligned}
$$

By the definitions of $\boldsymbol{\Phi}_1$, ϕ_2, and ϕ_3, we have

$$
\begin{aligned}
\phi(\boldsymbol{x}) &= 2\omega_f(\sqrt{d})\phi_3 \circ \phi_2 \circ \boldsymbol{\Phi}_1(\boldsymbol{x}) + f(\mathbf{0}) - \omega_f(\sqrt{d}) \\
&= 2\omega_f(\sqrt{d})\phi_3\Big(1 + \sum_{i=1}^{d} J^{i-1}\varrho_{1,\delta}(Jx_i)\Big) + f(\mathbf{0}) - \omega_f(\sqrt{d}) \\
&= 2\omega_f(\sqrt{d}) \sum_{j=1}^{N} 2^{-j}\varrho_3\Bigg(a_j \cdot \varrho_2\Big(1 + \sum_{i=1}^{d} 2^{(i-1)N}\varrho_{1,\delta}(2^N x_i)\Big)\Bigg) + f(\mathbf{0}) - \omega_f(\sqrt{d}).
\end{aligned}
$$

So we finish the proof. □

5. Conclusion

This paper has introduced a theoretical framework to show that three hidden layers are enough for neural network approximation to achieve exponential convergence and avoid the curse of dimensionality for approximating functions as general as (Hölder) continuous functions. The key idea is to leverage the power of multiple simple activation functions: the Floor function ($\lfloor x \rfloor$), the exponential function (2^x), the step function ($\mathbb{1}_{x\ge 0}$), or their compositions. This new class of networks is called the FLES network. Given a Lipschitz continuous function f on $[0,1]^d$, it was shown by construction that FLES networks with width $\max\{d, N\}$ and three hidden layers admit a uniform approximation rate $6\lambda\sqrt{d}\,2^{-N}$, where λ 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 $2\omega_f(2\sqrt{d})2^{-N} + \omega_f(2\sqrt{d}\,2^{-N})$. We also extend such a result to general continuous functions on a bounded set $E \subseteq \mathbb{R}^d$. The results in this paper provide a theoretical lower bound of the power of FLES networks. Whether or not

this bound is achievable in actual computation relies on advanced algorithm design as a separate line of research. Finally, we have also derived similar approximation results in the L^p-norm for $p \in [1, \infty)$ using continuous activation functions.

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

Arnold, V.I., 1957. On functions of three variables. Dokl. Akad. Nauk SSSR , 679–681.

Barron, A.R., 1993. Universal approximation bounds for superpositions of a sigmoidal function. IEEE Transactions on Information Theory 39, 930–945. doi:10.1109/18.256500.

Barron, A.R., Klusowski, J.M., 2018. Approximation and estimation for high-dimensional deep learning networks. arXiv:1809.03090.

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

Bengio, Y., Léonard, N., Courville, A., 2013. Estimating or propagating gradients through stochastic neurons for conditional computation arXiv:1308.3432.

Boo, Y., Shin, S., Sung, W., 2020. Quantized neural networks: Characterization and holistic optimization arXiv:2006.00530.

Braun, J., Griebel, M., 2009. On a constructive proof of kolmogorov's superposition theorem. Constructive Approximation 30, 653–675.

Carrillo, J.A.T., Jin, S., Li, L., Zhu, Y., 2019. A consensus-based global optimization method for high dimensional machine learning problems. arXiv:1909.09249 .

Chen, L., 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, 3400–3404. doi:10.1002/mma.5575.

Chen, M., Jiang, H., Liao, W., Zhao, T., 2019a. 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., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 32. Curran Associates, Inc., pp. 8174–8184.

Chen, Z., Cao, Y., Zou, D., Gu, Q., 2019b. How much over-parameterization is sufficient to learn deep ReLU networks? CoRR arXiv:1911.12360. URL: https://arxiv.org/abs/1911.12360.

Du, S.S., Zhai, X., Poczos, B., Singh, A., 2019. Gradient descent provably optimizes over-parameterized neural networks, in: International Conference on Learning Representations. URL: https://openreview.net/forum?id=S1eK3i09YQ.

E, W., Ma, C., Wu, L., 2019. A priori estimates of the population risk for two-layer neural networks. Communications in Mathematical Sciences 17, 1407 – 1425.

E, W., Wang, Q., 2018. Exponential convergence of the deep neural network approximation for analytic functions. CoRR abs/1807.00297. URL: http://arxiv.org/abs/1807.00297, arXiv:1807.00297.

Gühring, I., Kutyniok, G., Petersen, P., 2019. Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms. arXiv:1902.07896.

Guliyev, N.J., Ismailov, V.E., 2018. Approximation capability of two hidden layer feedforward neural networks with fixed weights. Neurocomputing 316, 262 – 269. doi:10.1016/j.neucom.2018.07.075.

Harvey, N., Liaw, C., Mehrabian, A., 2017. Nearly-tight VC-dimension bounds for piecewise linear neural networks, in: Kale, S., Shamir, O. (Eds.), Proceedings of the 2017 Conference on Learning Theory, PMLR, Amsterdam, Netherlands. pp. 1064–1068. URL: http://proceedings.mlr.press/v65/harvey17a.html.

Holland, J.H., 1992. Genetic algorithms. Scientific American 267, 66–73. URL: http://www.jstor.org/stable/24939139.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y., 2017. Quantized neural networks: Training neural networks with low precision weights and activations. J. Mach. Learn. Res. 18, 68696898.

Hutzenthaler, M., Jentzen, A., Wurstemberger, v.W., 2020. Overcoming the curse of dimensionality in the approximative pricing of financial derivatives with default risks. Electron. J. Probab. 25, 73 pp. doi:10.1214/20-EJP423.

Igelnik, B., Parikh, N., 2003. Kolmogorov's spline network. IEEE Transactions on Neural Networks 14, 725–733.

Jacot, A., Gabriel, F., Hongler, C., 2018. Neural tangent kernel: Convergence and generalization in neural networks, in: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.), Advances in Neural Information Processing Systems. Curran Associates, Inc.. volume 31, pp. 8571–8580. URL: https://proceedings.neurips.cc/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, pp. 1942–1948 vol.4.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680. doi:10.1126/science.220.4598.671.

Kolmogorov, A.N., 1956. On the representation of continuous functions of several variables by superposition of continuous functions of a smaller number of variables. Dokl. Akad. Nauk SSSR , 179–182.

Kolmogorov, A.N., 1957. On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. Dokl. Akad. Nauk SSSR , 953–956.

Kůrková, V., 1991. Kolmogorov's theorem is relevant. Neural Computation 3, 617–622. doi:10.1162/neco.1991.3.4.617.

Kůrková, V., 1992. Kolmogorov's theorem and multilayer neural networks. Neural Networks 5, 501 – 506. doi:10.1016/0893-6080(92)90012-8.

Li, Q., Lin, T., Shen, Z., 2019. Deep learning via dynamical systems: An approximation perspective. arXiv e-prints arXiv:1912.10382.

Lin, Y., Lei, M., Niu, L., 2019. Optimization strategies in quantized neural networks: A review, in: 2019 International Conference on Data Mining Workshops (ICDMW), pp. 385–390.

Lu, J., Shen, Z., Yang, H., Zhang, S., 2020. Deep network approximation for smooth functions. arXiv e-prints arXiv:2001.03040.

Lu, Y., Ma, C., Lu, Y., Lu, J., Ying, L., 2020. A mean-field analysis of deep resnet and beyond: Towards provable optimization via overparameterization from depth. CoRR abs/2003.05508. URL: https://arxiv.org/abs/2003.05508, arXiv:2003.05508.

Luo, T., Yang, H., 2020. Two-layer neural networks for partial differential equations: Optimization and generalization theory arXiv:abs/2006.15733.

Maiorov, V., Pinkus, A., 1999. Lower bounds for approximation by MLP neural networks. Neurocomputing 25, 81 – 91. doi:10.1016/S0925-2312(98)00111-8.

Mei, S., Montanari, A., Nguyen, P.M., 2018. A mean field view of the landscape of two-layer neural networks. Proceedings of the National Academy of Sciences 115, E7665–E7671. doi:10.1073/pnas.1806579115.

Montanelli, H., Du, Q., 2019. New error bounds for deep ReLU networks using sparse grids. SIAM Journal on Mathematics of Data Science 1, 78–92. doi:10.1137/18M1189336.

Montanelli, H., Yang, H., 2020. Error bounds for deep ReLU networks using the Kolmogorov-Arnold superposition theorem. Neural Networks 129, 1 – 6. doi:10.1016/j.neunet.2019.12.013.

Montanelli, H., Yang, H., Du, Q., 2020. Deep ReLU networks overcome the curse of dimensionality for bandlimited functions. Journal of Computational Mathematics .

Nelder, J., Mead, R., 1965. A simplex method for function minimization. Comput. J. 7, 308–313. doi:10.1093/comjnl/7.4.308.

Opschoor, J.A., Schwab, C., Zech, J., 2019. Exponential ReLU DNN expression of holomorphic maps in high dimension 2019-35. URL: https://math.ethz.ch/sam/research/reports.html?id=839.

Petersen, P., Voigtlaender, F., 2018. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. Neural Networks 108, 296 – 330. doi:10.1016/j.neunet.2018.08.019.

Pinnau, R., Totzeck, C., Tse, O., Martin, S., 2017. A consensus-based model for global optimization and its mean-field limit. Mathematical Models and Methods in Applied Sciences 27, 183–204. doi:10.1142/S0218202517400061.

Poggio, T., Mhaskar, H.N., Rosasco, L., Miranda, B., 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.

Schmidt-Hieber, J., 2020a. The kolmogorov-arnold representation theorem revisited. arXiv:2007.15884.

Schmidt-Hieber, J., 2020b. Nonparametric regression using deep neural networks with ReLU activation function. Annals of Statistics 48, 1875–1897. URL: https://projecteuclid.org/euclid.aos/1597370649.

Shen, Z., Yang, H., Zhang, S., 2019. Nonlinear approximation via compositions. Neural Networks 119, 74 – 84. doi:10.1016/j.neunet.2019.07.011.

Shen, Z., Yang, H., Zhang, S., 2020. Deep network approximation characterized by number of neurons. Communications in Computational Physics 28, 1768–1811. doi:10.4208/cicp.OA-2020-0149.

Shen, Z., Yang, H., Zhang, S., 2020. Deep network with approximation error being reciprocal of width to power of square root of depth. arXiv e-prints arXiv:2006.12231.

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

Wu, L., Ma, C., E, W., 2018. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective, in: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 31. Curran Associates, Inc., pp. 8279–8288.

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

Yarotsky, D., 2017. Error bounds for approximations with deep ReLU networks. Neural Networks 94, 103 – 114. doi:10.1016/j.neunet.2017.07.002.

Yarotsky, D., 2018. Optimal approximation of continuous functions by very deep ReLU networks, in: Bubeck, S., Perchet, V., Rigollet, P. (Eds.), Proceedings of the 31st Conference On Learning Theory, PMLR. pp. 639–649. URL: http://proceedings.mlr.press/v75/yarotsky18a.html.

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

Yin, P., Lyu, J., Zhang, S., Osher, S., Qi, Y., Xin, J., 2019. Understanding straight-through estimator in training activation quantized neural nets arXiv:1903.05662.

Zhang, S., 2020. Deep neural network approximation via function compositions. PhD Thesis, National University of Singapore URL: https://scholarbank.nus.edu.sg/handle/10635/186064.