

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 continuous 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 bounded 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 $r \rightarrow 0$ 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)

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 \(2020\)](#); [Petersen and Voigtlaender \(2018\)](#); [Yarotsky \(2018\)](#); [Gühring et al. \(2019\)](#); [Yarotsky and Zhevnerchuk \(2020\)](#) 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 the reference therein for ReLU networks. The limitation of ReLU networks motivated the work in [Shen et al. \(2021\)](#) 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. (2021) 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, where $\alpha \in (0, 1]$ and $\lambda > 0$ are the Hölder order and constant, respectively.

The most important message of Shen et al. (2021) (and probably also of Yarotsky and Zhevnerchuk (2020)) 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 nonconvex 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

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

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):

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

where

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

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

$$70 \quad |\phi(\mathbf{x}) - f(\mathbf{x})| \leq 2\omega_f(2\sqrt{d})2^{-N} + \omega_f(2\sqrt{d})2^{-N},$$

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

$$73 \quad \phi(\mathbf{x}) = 2\omega_f(2\sqrt{d}) \sum_{j=1}^N 2^{-j} \sigma_3 \left(a_j \cdot \sigma_2 \left(1 + \sum_{i=1}^d 2^{(i-1)N} \sigma_1(2^{N-1} x_i) \right) \right) + f(\mathbf{0}) - \omega_f(2\sqrt{d}).$$

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

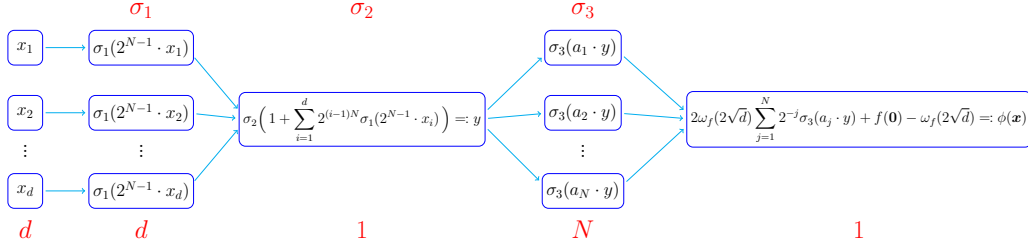


Figure 1: An illustration of the desired three-hidden-layer network in Theorem 1.1 for any $\mathbf{x} = (x_1, x_2, \dots, x_d) \in \mathbb{R}$. Each of the red functions “ σ_1 ”, “ σ_2 ”, and “ σ_3 ” above the network is the activation function of the corresponding hidden layer. The number of neurons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

78 The rate in $\omega_f(2\sqrt{d})2^{-N}$ implicitly depends on N through the modulus
 79 of continuity of f , while the rate in $2\omega_f(2\sqrt{d})2^{-N}$ is explicit in N . Sim-
 80 plifying the implicit approximation rate to make it explicitly depend on N
 81 is challenging in general. However, if f is a Hölder continuous function on
 82 $[0, 1]^d$ of order $\alpha \in (0, 1]$ with a Hölder constant $\lambda > 0$, i.e., $f(\mathbf{x})$ satisfying

$$83 \quad |f(\mathbf{x}) - f(\mathbf{y})| \leq \lambda \|\mathbf{x} - \mathbf{y}\|_2^\alpha, \quad \text{for any } \mathbf{x}, \mathbf{y} \in [0, 1]^d, \quad (1)$$

84 then $\omega_f(r) \leq \lambda r^\alpha$ for any $r \geq 0$. Therefore, in the case of Hölder continuous
 85 functions, the approximation rate is simplified to $3\lambda(2\sqrt{d})^\alpha 2^{-\alpha N}$ as shown in
 86 the following corollary. In the special case of Lipschitz continuous functions
 87 with a Lipschitz constant $\lambda > 0$, the approximation rate is simplified to
 88 $6\lambda\sqrt{d}2^{-N}$.

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

$$92 \quad |\phi(\mathbf{x}) - f(\mathbf{x})| \leq 3\lambda(2\sqrt{d})^\alpha 2^{-\alpha N}, \quad \text{for any } \mathbf{x} = (x_1, x_2, \dots, x_d) \in [0, 1]^d,$$

93 *where ϕ is defined by a formula in a_1, a_2, \dots, a_N as follows.*

$$94 \quad \phi(\mathbf{x}) = 2\omega_f(2\sqrt{d}) \sum_{j=1}^N 2^{-j} \sigma_3 \left(a_j \cdot \sigma_2 \left(1 + \sum_{i=1}^d 2^{(i-1)N} \sigma_1(2^{N-1} x_i) \right) \right) + f(\mathbf{0}) - \omega_f(2\sqrt{d}).$$

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

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

114 2. Discussion

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

117 2.1. Application scope of our theory in machine learning

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

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

$$121 \quad R_{\mathcal{D}}(\boldsymbol{\theta}) := \mathbb{E}_{\mathbf{x} \sim U(\mathcal{X})} [\ell(\phi(\mathbf{x}; \boldsymbol{\theta}), f(\mathbf{x}))]$$

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

$$125 \quad R_{\mathcal{D}}(\boldsymbol{\theta}) = \int_{[0,1]^d} \frac{1}{2} |\phi(\mathbf{x}; \boldsymbol{\theta}) - f(\mathbf{x})|^2 d\mathbf{x}.$$

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

$$127 \quad \boldsymbol{\theta}_{\mathcal{D}} := \arg \min_{\boldsymbol{\theta}} R_{\mathcal{D}}(\boldsymbol{\theta}).$$

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

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

$$131 \quad R_{\mathcal{S}}(\boldsymbol{\theta}) := \frac{1}{n} \sum_{i=1}^n \ell(\phi(\mathbf{x}_i; \boldsymbol{\theta}), f(\mathbf{x}_i)).$$

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

$$134 \quad \boldsymbol{\theta}_{\mathcal{S}} = \arg \min_{\boldsymbol{\theta}} R_{\mathcal{S}}(\boldsymbol{\theta}), \tag{2}$$

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

$$140 \quad R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}}) = \mathbb{E}_{\mathbf{x} \sim U(\mathcal{X})} [\ell(\phi(\mathbf{x}; \boldsymbol{\theta}_{\mathcal{N}}), f(\mathbf{x}))] \stackrel{e.g.}{=} \int_{[0,1]^d} \frac{1}{2} |\phi(\mathbf{x}; \boldsymbol{\theta}_{\mathcal{N}}) - f(\mathbf{x})|^2 d\mathbf{x},$$

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

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

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

148 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
 149 $\boldsymbol{\theta}_{\mathcal{S}}$ is a global minimizer of $R_{\mathcal{S}}(\boldsymbol{\theta})$. Constructive approximation provides
 150 an upper bound of $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$ in terms of the network size, e.g., in terms of
 151 the network width and depth, or in terms of the number of parameters.
 152 The second term of Equation (3) is bounded by the optimization error of the
 153 numerical algorithm applied to solve the empirical loss minimization problem
 154 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,
 155 but not $\boldsymbol{\theta}_{\mathcal{N}} - \boldsymbol{\theta}_{\mathcal{S}}$. The study of the bounds for the third and fourth terms
 156 is referred to as the generalization error analysis of neural networks. See
 157 Figure 2 for the intuitions of these three errors.

158 One of the key targets in the area of deep learning is to develop algo-
 159 rithms to reduce $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}})$. The constructive approximation established in
 160 this paper and in the literature provides upper bounds of the approxima-
 161 tion error $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$ for several function spaces, which is crucial to estimate
 162 an upper bound of $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}})$. Instead of deriving an approximator to attain
 163 the approximation error bound, deep learning algorithms aim to identify
 164 a solution $\phi(\mathbf{x}; \boldsymbol{\theta}_{\mathcal{N}})$ reducing the generalization and optimization errors in
 165 Equation (3). Solutions minimizing both generalization and optimization er-
 166 rors will lead to a good solution only if we also have a good upper bound
 167 estimate of $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$ as shown in Equation (3). Independent of whether our
 168 analysis here leads to a good approximator, which is an interesting topic to
 169 pursue, the theory here does provide a key ingredient in the error analysis of
 170 deep learning algorithms.

171 Theorem 1.1 and Corollary 1.2 provide an upper bound of $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$. This
 172 bound only depends on the given budget of neurons and layers of FLES net-
 173 works. Hence, this bound is independent of the empirical loss minimization
 174 in Equation (2) and the optimization algorithm used to compute the numeri-
 175 cal solution of Equation (2). In other words, Theorem 1.1 and Corollary 1.2
 176 quantify the approximation power of FLES networks with a given size. De-
 177 signing efficient optimization algorithms and analyzing the generalization

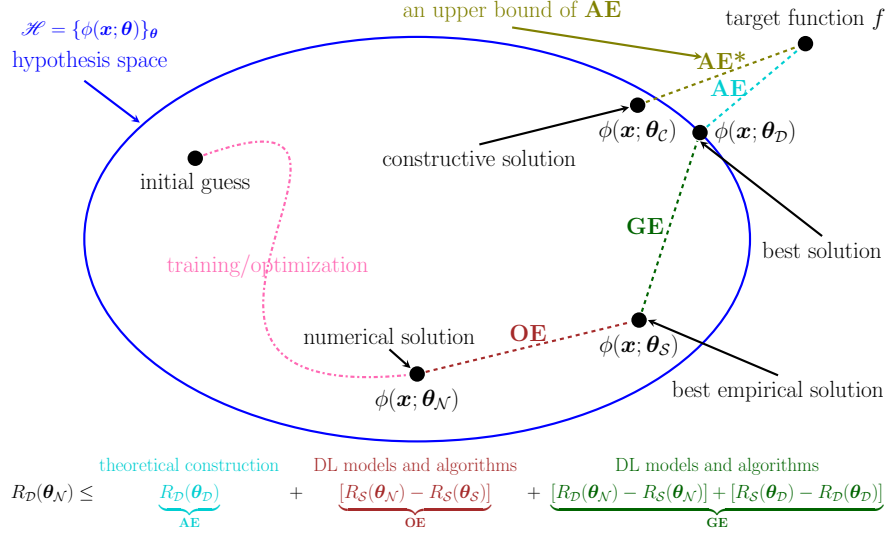


Figure 2: The intuitions of the approximation error (AE), the optimization error (OE), and the generalization error (GE). DL is short of deep learning. One needs to control AE, OE, and GE in order to bound the discrepancy between the target function f and the numerical solution $\phi(\mathbf{x}; \theta_N)$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

178 bounds for FLES networks are two other separate future directions.

179 2.2. Connection between approximation error and VC-dimension

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

183 Let us first present the definitions of VC-dimension and related concepts.
 184 Assume H is a class of functions mapping from a general domain \mathcal{X} to $\{0, 1\}$.
 185 We say H shatters a set of points $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\} \subseteq \mathcal{X}$ if

$$186 \quad \left| \left\{ [h(\mathbf{x}_1), h(\mathbf{x}_2), \dots, h(\mathbf{x}_m)]^T \in \{0, 1\}^m : h \in H \right\} \right| = 2^m,$$

187 where $|\cdot|$ means the size of a set. The above equation means, given any
 188 $\theta_i \in \{0, 1\}$ for $i = 1, 2, \dots, m$, there exists $h \in H$ such that $h(\mathbf{x}_i) = \theta_i$ for all i .
 189 For a general function set \mathcal{F} with its elements mapping from \mathcal{X} to \mathbb{R} , we say
 190 \mathcal{F} shatters $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\} \subseteq \mathcal{X}$ if $\mathcal{T} \circ \mathcal{F}$ does, where

$$191 \quad \mathcal{T}(t) := \begin{cases} 1, & t \geq 0, \\ 0, & t < 0 \end{cases} \quad \text{and} \quad \mathcal{T} \circ \mathcal{F} := \{\mathcal{T} \circ f : f \in \mathcal{F}\}.$$

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

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

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

$$\text{VCDim}(H) := \sup \{m \in \mathbb{N}^+ : \Pi_H(m) = 2^m\}$$

in the case that $\{m \in \mathbb{N}^+ : \Pi_H(m) = 2^m\}$ is not empty. If $\{m \in \mathbb{N}^+ : \Pi_H(m) = 2^m\} = \emptyset$, we may define $\text{VCDim}(H) = 0$.

Let \mathcal{F} be a class of functions from \mathcal{X} to \mathbb{R} . The VC-dimension of \mathcal{F} , denoted by $\text{VCDim}(\mathcal{F})$, is defined by $\text{VCDim}(\mathcal{F}) := \text{VCDim}(\mathcal{T} \circ \mathcal{F})$,¹ where

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

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

As shown in [Yarotsky \(2018, 2017\)](#); [Shen et al. \(2019, 2020\)](#); [Lu et al. \(2020\)](#); [Shen et al. \(2021\)](#); [Zhang \(2020\)](#); [Shen et al. \(to appear\)](#), VC-dimension essentially determines the lower bound of the approximation errors of networks. For simplicity, we use $\text{H\"older}([0, 1]^d, \alpha, \lambda)$ as an example, where $\text{H\"older}([0, 1]^d, \alpha, \lambda)$ denotes the space of H\"older continuous functions of order $\alpha \in (0, 1]$ and a H\"older 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 $\text{H\"older}([0, 1]^d, \alpha, 1)$ approximated by functions in \mathcal{F} is bounded by a formula characterized by $\text{VCDim}(\mathcal{F})$.

Theorem 2.2 (Theorem 2.4 of [Shen et al. \(to appear\)](#) or Theorem 4.17 of [Zhang \(2020\)](#)). Assume \mathcal{F} is a function set with all elements defined on $[0, 1]^d$. Given any $\varepsilon > 0$, suppose $\text{VCDim}(\mathcal{F}) \geq 1$ and

$$\inf_{\phi \in \mathcal{F}} \|\phi - f\|_{L^\infty([0, 1]^d)} \leq \varepsilon, \quad \text{for any } f \in \text{H\"older}([0, 1]^d, \alpha, 1).$$

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

¹One may also define $\text{VCDim}(\mathcal{F}) := \text{VCDim}(\widehat{\mathcal{T}} \circ \mathcal{F})$, where $\widehat{\mathcal{T}}(t) := \begin{cases} 1, & t > 0, \\ 0, & t \leq 0. \end{cases}$

220 This theorem investigates the connection between VC-dimension of \mathcal{F}
 221 and the approximation errors of functions in $\text{Hölder}([0, 1]^d, \alpha, 1)$ approxi-
 222 mated by elements of \mathcal{F} . Denote the best approximation error of functions
 223 in $\text{Hölder}([0, 1]^d, \alpha, 1)$ approximated by the elements of \mathcal{F} as

$$224 \quad \mathcal{E}_{\alpha,d}(\mathcal{F}) := \sup_{f \in \text{Hölder}([0,1]^d, \alpha, 1)} \left(\inf_{\phi \in \mathcal{F}} \|\phi - f\|_{L^\infty([0,1]^d)} \right).$$

225 Then, Theorem 2.2 implies that

$$226 \quad \text{VCDim}(\mathcal{F})^{-\alpha/d}/9 \leq \mathcal{E}_{\alpha,d}(\mathcal{F}),$$

227 which means that the best possible approximation error is controlled by
 228 $\text{VCDim}(\mathcal{F})^{-\alpha/d}/9$. A typical application of this theorem is to prove the op-
 229 timality of approximation errors when using ReLU networks to approximate
 230 functions in $\text{Hölder}([0, 1]^d, \alpha, 1)$. It is shown in Harvey et al. (2017) that the
 231 VC-dimension of $\mathcal{F}_{N,L}$ is bounded by

$$232 \quad \text{VCDim}(\mathcal{F}_{N,L}) \leq \mathcal{O}(N^2 L \cdot L \cdot \ln(N^2 L)) \leq \mathcal{O}(N^2 L^2 \ln(NL)),$$

233 where $\mathcal{F}_{N,L}$ is the space consisting of all functions implemented by ReLU
 234 networks with width N and depth L . It is shown in Section 4.4.1 of Zhang
 235 (2020) that

$$236 \quad C_1(\alpha, d) \cdot (N^2 L^2 \ln(NL))^{-\alpha/d} \leq \mathcal{E}_{\alpha,d}(\mathcal{F}_{N,L}) \leq C_2(\alpha, d) \cdot (N^2 L^2)^{-\alpha/d},$$

237 where $C_1(\alpha, d)$ and $C_2(\alpha, d)$ are two positive constants determined by α and
 238 d .

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

$$244 \quad \text{VCDim}(\{\phi : \phi(x) = \cos(ax), a \in \mathbb{R}\}) = \infty.$$

245 However, $\{\phi : \phi(x) = \cos(ax), a \in \mathbb{R}\}$ cannot achieve a good approximation
 246 error when approximating Hölder continuous functions. Designing a hypothe-
 247 sis space with a large VC-dimension is the first step for a good approximation

248 toll, but to realize the desired approximation power requires refined design of
 249 the hypothesis space, which is also the philosophy we followed in this paper.
 250 Our initial goal is to design a network architecture with a fixed depth (e.g.,
 251 three hidden layers) to generate a hypothesis space with a sufficiently large
 252 VC-dimension (∞). As we shall see later, Proposition 3.2 implies that the
 253 VC-dimension of FLES networks is infinity, which is a necessary condition
 254 for our FLES networks to attain super approximation power.

255 2.3. Further interpretation of our theory

256 In the interpretation of our theory, three more aspects are important
 257 to discuss. The first one is whether it is possible to extend our theory to
 258 functions on a more general domain, e.g, $E \subseteq [-R, R]^d$ for any $R > 0$, because
 259 $R > 1$ may cause an implicit curse of dimensionality in some existing theory.
 260 The second one is how bad the modulus of continuity would be since it is
 261 related to a high-dimensional function f that may lead to an implicit curse
 262 of dimensionality in our approximation rate. The last one is the discussion
 263 of overcoming the zero derivative in training FLES networks.

264 First, we can generalize Theorem 1.1 to the function space $C(E)$ with
 265 $E \subseteq [-R, R]^d$ for any $R > 0$ in the following corollary with the modulus of
 266 continuity $\omega_f^E(\cdot)$ defined as follows. For an arbitrary set $E \subseteq \mathbb{R}^d$, $\omega_f^E(r)$ is
 267 defined via

$$268 \quad \omega_f^E(r) := \sup \{ |f(\mathbf{x}) - f(\mathbf{y})| : \|\mathbf{x} - \mathbf{y}\|_2 \leq r, \mathbf{x}, \mathbf{y} \in E \}, \quad \text{for any } r \geq 0.$$

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

271 **Corollary 2.3.** *Given an arbitrary bounded continuous function f on $E \subseteq$
 272 $[-R, R]^d$ where R is an arbitrary positive real number, for any $N \in \mathbb{N}^+$, there
 273 exist $a_1, a_2, \dots, a_N \in [0, \frac{1}{2})$ such that*

$$274 \quad |\phi(\mathbf{x}) - f(\mathbf{x})| \leq 2\omega_f^E(3R\sqrt{d})2^{-N} + \omega_f^E(3R\sqrt{d}2^{-N}),$$

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

$$277 \quad \phi(\mathbf{x}) = 2\omega_f^E(3R\sqrt{d}) \sum_{j=1}^N 2^{-j} \sigma_3 \left(a_j \cdot \sigma_2 \left(1 + \sum_{i=1}^d 2^{(i-1)N} \sigma_1 \left(2^N \frac{x_i + R}{3R} \right) \right) \right) + C_f,$$

278 where C_f is a constant determined by f .

279 Hence, the volume of the function domain $E \subseteq [-R, R]^d$ only has a mild
 280 influence on the approximation rate of our FLES networks. FLES networks
 281 can still avoid the curse of dimensionality and achieve exponential conver-
 282 gence for continuous functions on $E \subseteq [-R, R]^d$ when $R > 1$. For example,
 283 in the case of Hölder continuous functions of order $\alpha \in (0, 1]$ with a constant
 284 $\lambda > 0$ on $E \subseteq [-R, R]^d$, our approximation rate becomes $3\lambda(3R\sqrt{d}2^{-N})^\alpha$.

285 Second, most interesting continuous functions in practice have a good
 286 modulus of continuity such that there is no implicit curse of dimensionality
 287 hidden in $\omega_f(\cdot)$. For example, we have discussed the case of Hölder contin-
 288 uous functions previously. We would like to remark that the class of Hölder
 289 continuous functions implicitly depends on d through its definition in Equa-
 290 tion (1), but this dependence is moderate since the ℓ^2 -norm in Equation (1)
 291 is the square root of a sum with d terms. Let us now discuss several cases of
 292 $\omega_f(\cdot)$ when we cannot achieve exponential convergence or cannot avoid the
 293 curse of dimensionality. The first example is $\omega_f(r) = \frac{1}{\ln(1/r)}$ for small $r > 0$,
 294 which leads to an approximation rate

$$295 \quad 3(N \ln 2 - \tfrac{1}{2} \ln d - \ln 2)^{-1}, \quad \text{for large } N \in \mathbb{N}^+.$$

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

$$300 \quad 3(N \ln 2 - \tfrac{1}{2} \ln d - \ln 2)^{-1/d}, \quad \text{for large } N \in \mathbb{N}^+.$$

301 The power $1/d$ further weakens the approximation rate and hence the curse
 302 of dimensionality exists. The last example we would like to discuss is $\omega_f(r) =$
 303 $r^{\alpha/d}$ for small $r > 0$, which results in the approximation rate

$$304 \quad 3\lambda(2\sqrt{d})^{\alpha/d}2^{-\alpha N/d}, \quad \text{for large } N \in \mathbb{N}^+,$$

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

309 Finally, we would like to point out that the training of FLES networks
 310 in practice may encounter two issues. First, network weights in our main
 311 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 $\phi_q : \mathbb{R} \rightarrow \mathbb{R}$ and $\psi_{q,p} : [0, 1] \rightarrow \mathbb{R}$ such that

$$f(\mathbf{x}) = \sum_{q=0}^{2d} \phi_q \left(\sum_{p=1}^d \psi_{q,p}(x_p) \right), \quad \text{for any } \mathbf{x} = (x_1, \dots, x_d) \in [0, 1]^d. \quad (4)$$

Note that the activation functions $\{\phi_q\}$ (also called outer functions) of the neural network in Equation (4) 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 \(1991, 1992\)](#); [Maierov and Pinkus \(1999\)](#); [Guliyev and Ismailov \(2018\)](#); [Montanelli and Yang \(2020\)](#); [Igel'nik and Parikh \(2003\)](#); [Schmidt-Hieber \(2021\)](#) 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 [Igel'nik and Parikh \(2003\)](#); [Montanelli and Yang \(2020\)](#); [Schmidt-Hieber \(2021\)](#), 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 [Maierov 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 summarize 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 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 [Maierov 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.² 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.

²The result in [Shen et al. \(2019\)](#) is for Hölder functions, but can be easily generalized to general continuous functions.

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
Maïorov 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

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 following form were considered:

$$f(\mathbf{x}) = \int_{\tilde{\Omega}} a(\mathbf{w}) K(\mathbf{w} \cdot \mathbf{x}) d\mu(\mathbf{w}),$$

where $\tilde{\Omega} \subseteq \mathbb{R}^d$, $\mu(\mathbf{w})$ is a Lebesgue measure in \mathbf{w} , and $\mathbf{x} \in \Omega \subseteq \mathbb{R}^d$. The above integral representation is equivalent to the expectation of a high-dimensional random function when \mathbf{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 $f(\mathbf{x})$ 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 (2020); 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. \(to appear\)](#). The authors of [Li et al. \(to appear\)](#) use a dynamic system based approach to obtain a universal approximation property of residual neural 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 \(2020\)](#); [Shen et al. \(to appear\)](#) 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. \(2021\)](#); [Schmidt-Hieber \(2021\)](#) 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 second 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.

- 482 • Let \mathbb{R} denote the set of real numbers.
- 483 • Let \mathbb{Z} , \mathbb{N} , and \mathbb{N}^+ denote the set of integers, natural numbers, all
484 positive integers, respectively, i.e., $\mathbb{Z} = \{0, 1, 2, \dots\} \cup \{-1, -2, -3, \dots\}$,
485 $\mathbb{N} = \{0, 1, 2, \dots\}$, and $\mathbb{N}^+ = \{1, 2, 3, \dots\}$.
- 486 • For any $p \in [1, \infty)$, the p -norm of a vector $\mathbf{x} = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d$ is
487 defined by
$$488 \quad \|\mathbf{x}\|_p := (|x_1|^p + |x_2|^p + \dots + |x_d|^p)^{1/p}.$$
- 489 • The floor function (Floor) is defined as $\lfloor x \rfloor := \max\{n : n \leq x, n \in \mathbb{Z}\}$ for
490 any $x \in \mathbb{R}$.
- 491 • For $\theta \in [0, 1)$, suppose its binary representation is $\theta = \sum_{\ell=1}^{\infty} \theta_{\ell} 2^{-\ell}$ with
492 $\theta_{\ell} \in \{0, 1\}$, we introduce a special notation $\text{bin}0.\theta_1\theta_2\cdots\theta_L$ to denote the
493 L -term binary representation of θ , i.e., $\text{bin}0.\theta_1\theta_2\cdots\theta_L := \sum_{\ell=1}^L \theta_{\ell} 2^{-\ell}$.
- 494 • The expression “a network with width N and depth L ” means
 - 495 – The maximum width of this network for all **hidden** layers is no
496 more than N .
 - 497 – The number of **hidden** layers of this network is no more than L .

498 3.2. Proof of Theorem 1.1 and Corollary 2.3

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

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

$$504 \quad |\phi(\mathbf{x}) - f(\mathbf{x})| \leq 2\omega_f(\sqrt{d})2^{-N} + \omega_f(\sqrt{d}2^{-N}),$$

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

$$507 \quad \phi(\mathbf{x}) = 2\omega_f(\sqrt{d}) \sum_{j=1}^N 2^{-j} \sigma_3 \left(a_j \cdot \sigma_2 \left(1 + \sum_{i=1}^d 2^{(i-1)N} \sigma_1(2^N x_i) \right) \right) + f(\mathbf{0}) - \omega_f(\sqrt{d}).$$

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

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

511 *Proof of Theorem 1.1.* Given any $f \in C([0, 1]^d)$, by Lemma 4.2 of Shen et al.
 512 (2020) via setting $E = [0, 1]^d$ and $S = \mathbb{R}^d$, there exists $g \in C(\mathbb{R}^d)$ such that

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

515 Define $\tilde{g}(\mathbf{x}) := g(2\mathbf{x})$ for any $\mathbf{x} \in \mathbb{R}^d$. By applying Theorem 3.1 to
 516 $\tilde{g} \in C(\mathbb{R}^d)$, there exist $a_1, a_2, \dots, a_N \in [0, \frac{1}{2})$ such that

$$517 \quad |\tilde{\phi}(\mathbf{x}) - \tilde{g}(\mathbf{x})| \leq 2\omega_{\tilde{g}}^S(\sqrt{d})2^{-N} + \omega_{\tilde{g}}^S(\sqrt{d}2^{-N}), \quad \text{for any } \mathbf{x} \in [0, 1]^d, \quad (5)$$

518 where

$$519 \quad \tilde{\phi}(\mathbf{x}) = 2\omega_{\tilde{g}}^S(\sqrt{d}) \sum_{j=1}^N 2^{-j} \sigma_3 \left(a_j \cdot \sigma_2 \left(1 + \sum_{i=1}^d 2^{(i-1)N} \sigma_1(2^N x_i) \right) \right) + \tilde{g}(\mathbf{0}) - \omega_{\tilde{g}}^S(\sqrt{d}).$$

520 Note that $f(\mathbf{x}) = g(\mathbf{x}) = \tilde{g}(\frac{\mathbf{x}}{2})$ for any $\mathbf{x} \in E = [0, 1]^d$ and

$$521 \quad \omega_{\tilde{g}}^S(r) = \omega_g^S(2r) = \omega_f^E(2r) = \omega_f(2r), \quad \text{for any } r \geq 0.$$

522 Define $\phi(\mathbf{x}) := \tilde{\phi}(2\mathbf{x})$ for any $\mathbf{x} \in \mathbb{R}^d$. Then by Equation (5), for any $\mathbf{x} \in$
 523 $[0, 1]^d = E$, we have $\frac{\mathbf{x}}{2} \in [0, \frac{1}{2}]^d \subseteq [0, 1)^d$, implying

$$\begin{aligned} |\phi(\mathbf{x}) - f(\mathbf{x})| &= |\phi(\mathbf{x}) - g(\mathbf{x})| = |\tilde{\phi}(\frac{\mathbf{x}}{2}) - \tilde{g}(\frac{\mathbf{x}}{2})| \\ &\leq 2\omega_{\tilde{g}}^S(\sqrt{d})2^{-N} + \omega_{\tilde{g}}^S(\sqrt{d}2^{-N}) \\ &= 2\omega_f(2\sqrt{d})2^{-N} + \omega_f(2\sqrt{d}2^{-N}), \end{aligned}$$

525 where $\phi(\mathbf{x}) := \tilde{\phi}(\frac{\mathbf{x}}{2})$ can be represented by

$$526 \quad 2\omega_f(2\sqrt{d}) \sum_{j=1}^N 2^{-j} \sigma_3 \left(a_j \cdot \sigma_2 \left(1 + \sum_{i=1}^d 2^{(i-1)N} \sigma_1(2^{N-1} x_i) \right) \right) + f(\mathbf{0}) - \omega_f(2\sqrt{d}).$$

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

528 Next, we present the proof of Corollary 2.3 below.

529 *Proof of Corollary 2.3.* Given any bounded continuous function $f \in C(E)$,
 530 by Lemma 4.2 of [Shen et al. \(2020\)](#) via setting $S = \mathbb{R}^d$, there exists $g \in C(\mathbb{R}^d)$
 531 such that

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

534 Define

$$535 \quad \tilde{g}(\mathbf{x}) := g(3R\mathbf{x} - R), \quad \text{for any } \mathbf{x} \in \mathbb{R}^d.$$

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

$$538 \quad |\tilde{\phi}(\mathbf{x}) - \tilde{g}(\mathbf{x})| \leq 2\omega_{\tilde{g}}^S(\sqrt{d})2^{-N} + \omega_{\tilde{g}}^S(\sqrt{d}2^{-N}), \quad \text{for any } \mathbf{x} \in [0, 1]^d, \quad (6)$$

539 where

$$540 \quad \tilde{\phi}(\mathbf{x}) = 2\omega_{\tilde{g}}^S(\sqrt{d}) \sum_{j=1}^N 2^{-j} \sigma_3 \left(a_j \cdot \sigma_2 \left(1 + \sum_{i=1}^d 2^{(i-1)N} \sigma_1(2^N x_i) \right) \right) + \tilde{g}(\mathbf{0}) - \omega_{\tilde{g}}^S(\sqrt{d}).$$

541 Note that $f(\mathbf{x}) = g(\mathbf{x}) = \tilde{g}(\frac{\mathbf{x}+R}{3R})$ for any $\mathbf{x} \in E \subseteq [-R, R]^d$ and

$$542 \quad \omega_g^S(r) = \omega_{\tilde{g}}^S(3Rr) = \omega_f^E(3Rr), \quad \text{for any } r \geq 0.$$

543 Define $\phi(\mathbf{x}) := \tilde{\phi}(\frac{\mathbf{x}+R}{3R})$ for any $\mathbf{x} \in \mathbb{R}^d$. Then by Equation (6), for any $\mathbf{x} \in E \subseteq$
 544 $[-R, R]^d$, we have $\frac{\mathbf{x}+R}{3R} \in [0, \frac{2}{3}]^d \subseteq [0, 1]^d$, implying

$$\begin{aligned} |\phi(\mathbf{x}) - f(\mathbf{x})| &= |\phi(\mathbf{x}) - g(\mathbf{x})| = |\tilde{\phi}(\frac{\mathbf{x}+R}{3R}) - \tilde{g}(\frac{\mathbf{x}+R}{3R})| \\ &\leq 2\omega_{\tilde{g}}^S(\sqrt{d})2^{-N} + \omega_{\tilde{g}}^S(\sqrt{d}2^{-N}) \\ &= 2\omega_f(3R\sqrt{d})2^{-N} + \omega_f(3R\sqrt{d}2^{-N}), \end{aligned}$$

546 where $\phi(\mathbf{x}) = \tilde{\phi}(\frac{\mathbf{x}+R}{3R})$ can be represented by

$$547 \quad 2\omega_f(3R\sqrt{d}) \sum_{j=1}^N 2^{-j} \sigma_3 \left(a_j \cdot \sigma_2 \left(1 + \sum_{i=1}^d 2^{(i-1)N} \sigma_1(2^N \frac{x_i+R}{3R}) \right) \right) + C_f,$$

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

550 3.3. Proof of Theorem 3.1

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

- 554 1. Normalize f as \tilde{f} satisfying $\tilde{f}(\mathbf{x}) \in [0, 1]$ for any $\mathbf{x} \in [0, 1]^d$, divide
 555 $[0, 1]^d$ into a set of non-overlapping cubes $\{Q_\beta\}_{\beta \in \{0, 1, \dots, J-1\}^d}$, and denote
 556 \mathbf{x}_β as the vertex of Q_β with minimum $\|\cdot\|_1$ norm, where J is an integer
 557 determined later. See Figure 3 for the illustrations of Q_β and \mathbf{x}_β .
- 558 2. Construct a vector-valued function $\Phi_1 : \mathbb{R}^d \rightarrow \mathbb{R}^d$ projecting the whole
 559 cube Q_β to the index β , i.e., $\Phi_1(\mathbf{x}) = \beta$ for all $\mathbf{x} \in Q_\beta$ and each
 560 $\beta \in \{0, 1, \dots, J-1\}^d$.
- 561 3. Construct a linear function $\phi_2 : \mathbb{R}^d \rightarrow \mathbb{R}$ bijectively mapping $\beta \in$
 562 $\{0, 1, \dots, J-1\}^d$ to $\phi_2(\beta) \in \{1, 2, \dots, J^d\}$.
- 563 4. Construct a function $\phi_3 : \mathbb{R} \rightarrow \mathbb{R}$ mapping $\phi_2(\beta) \in \{1, 2, \dots, J^d\}$ approx-
 564 imately to $\tilde{f}(\mathbf{x}_\beta)$, i.e., $\phi_3(\phi_2(\beta)) \approx \tilde{f}(\mathbf{x}_\beta)$ for each $\beta \in \{0, 1, \dots, J-1\}^d$.
- 565 5. Define $\tilde{\phi} := \phi_3 \circ \phi_2 \circ \Phi_1$. Then $\tilde{\phi}$ is a piecewise constant function mapping
 566 $\mathbf{x} \in Q_\beta$ to $\phi_3(\phi_2(\beta)) \approx \tilde{f}(\mathbf{x}_\beta)$ for each $\beta \in \{0, 1, \dots, J-1\}^d$, implying
 567 $\tilde{\phi} \approx \tilde{f}$. Finally, re-scale and shift $\tilde{\phi}$ to obtain the final function ϕ
 568 approximating f well.

569 Recall that

570 $\sigma_1(x) := \lfloor x \rfloor, \quad \sigma_2(x) := 2^x, \quad \text{and} \quad \sigma_3(x) := \mathcal{T}(x - \lfloor x \rfloor - \frac{1}{2}), \quad \text{for any } x \in \mathbb{R},$

571 where

572
$$\mathcal{T}(x) := \mathbb{1}_{x \geq 0} = \begin{cases} 1, & x \geq 0, \\ 0, & x < 0, \end{cases} \quad \text{for any } x \in \mathbb{R}.$$

573 Step 1 and 5 are straightforward. To implement Step 2, we introduce σ_1
 574 since it can help to significantly simplify the construction of the vector-valued
 575 projecting function Φ_1 . The implementation of Step 3 is based on the J -ary
 576 representation, namely, define $\phi_2(\mathbf{x}) := 1 + \sum_{i=1}^d J^{i-1} x_i$. The most technical
 577 step above is Step 4, which is essentially a point fitting problem. Solving
 578 such a problem eventually relies on the bit extraction technique in Shen et al.
 579 (2020); Lu et al. (2020); Shen et al. (2021); Harvey et al. (2017); Bartlett
 580 et al. (1998); Zhang (2020); Yarotsky (2018). To extract sufficient many bits

581 with a limited neuron budget, we introduce two powerful activation functions
 582 σ_2 and σ_3 , as shown in the proposition below.

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

$$585 \quad \sigma_3(a \cdot \sigma_2(k)) = \sigma_3(2^k \cdot a) = \theta_k, \quad \text{for any } k \in \{1, 2, \dots, K\},$$

586 where

$$587 \quad a = \sum_{j=1}^K 2^{-j-1} \cdot \theta_j \in [0, \frac{1}{2}).$$

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

$$589 \quad 0 \leq \sum_{j=1}^K 2^{-j-1} \cdot \theta_j \leq \sum_{j=1}^K 2^{-j-1} < \frac{1}{2},$$

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

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

$$592 \quad 2^k \cdot a = 2^k \cdot \sum_{j=1}^K 2^{-j-1} \cdot \theta_j = \underbrace{\sum_{j=1}^{k-1} 2^{k-j-1} \cdot \theta_j}_{\text{an integer}} + \overbrace{\frac{1}{2} \theta_k}^{0 \text{ or } \frac{1}{2}} + \underbrace{\sum_{j=k+1}^K 2^{k-j-1} \cdot \theta_j}_{\text{in } [0, \frac{1}{2})}.^3 \quad (7)$$

593 Clearly, the first term in Equation (7) $\sum_{j=1}^{k-1} 2^{k-j-1} \cdot \theta_j$ is a non-negative integer
 594 since $\theta_j \in \{0, 1\}$ for any $j \in \{1, 2, \dots, K\}$. As for the third term in Equation (7),
 595 we have

$$596 \quad 0 \leq \sum_{j=k+1}^K 2^{k-j-1} \cdot \theta_j \leq \sum_{j=k+1}^K 2^{k-j-1} < \frac{1}{2}$$

597 Therefore, by Equation (7), we have

$$598 \quad 2^k \cdot a \in \bigcup_{n \in \mathbb{N}} [n, n + \frac{1}{2}), \text{ if } \theta_k = 0 \quad \text{and} \quad 2^k \cdot a \in \bigcup_{n \in \mathbb{N}} [n + \frac{1}{2}, n + 1), \text{ if } \theta_k = 1. \quad (8)$$

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

$$601 \quad \sigma_3(x) = 0 \text{ if } x \in \bigcup_{n \in \mathbb{N}} [n, n + \frac{1}{2}) \quad \text{and} \quad \sigma_3(x) = 1 \text{ if } x \in \bigcup_{n \in \mathbb{N}} [n + \frac{1}{2}, n + 1).$$

³By convention, $\sum_{j=n}^m a_j = 0$ if $n > m$ no matter what a_j is for each j .

602 If $\theta_k = 0$, by Equation (8), we have

$$603 \quad 2^k \cdot a \in \bigcup_{n \in \mathbb{N}} [n, n + \frac{1}{2}) \implies \sigma_3(2^k \cdot a) = 0 = \theta_k.$$

604 Similarly, if $\theta_k = 1$, by Equation (8), we have

$$605 \quad 2^k \cdot a \in \bigcup_{n \in \mathbb{N}} [n + \frac{1}{2}, n + 1) \implies \sigma_3(2^k \cdot a) = 1 = \theta_k.$$

606 Since $k \in \{1, 2, \dots, K\}$ is arbitrary, we have $\sigma_3(a \cdot \sigma_2(k)) = \sigma_3(2^k \cdot a) = \theta_k$
 607 for any $k \in \{1, 2, \dots, K\}$. So we finish the proof. \square

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

$$610 \quad \{f : f(x) = \sigma_3(a \cdot x), \text{ for } a \in \mathbb{R}\}$$

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

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

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

617 **Step 1:** Set up.

618 Assume f is not a constant function since it is a trivial case. Then
 619 $\omega_f(r) > 0$ for any $r > 0$. Clearly, $|f(\mathbf{x}) - f(\mathbf{0})| \leq \omega_f(\sqrt{d})$ for any $\mathbf{x} \in [0, 1]^d$.
 620 Define

$$621 \quad \tilde{f} := \frac{f - f(\mathbf{0}) + \omega_f(\sqrt{d})}{2\omega_f(\sqrt{d})}. \quad (9)$$

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

623 Set $J = 2^N$ and divide $[0, 1]^d$ into J^d cubes $\{Q_\beta\}_\beta$. To be exact, defined
 624 $\mathbf{x}_\beta := \beta/J$ and

$$625 \quad Q_\beta := \left\{ \mathbf{x} = (x_1, x_2, \dots, x_d) : x_i \in [\frac{\beta_i}{J}, \frac{\beta_i+1}{J}) \text{ for } i = 1, 2, \dots, d \right\},$$

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

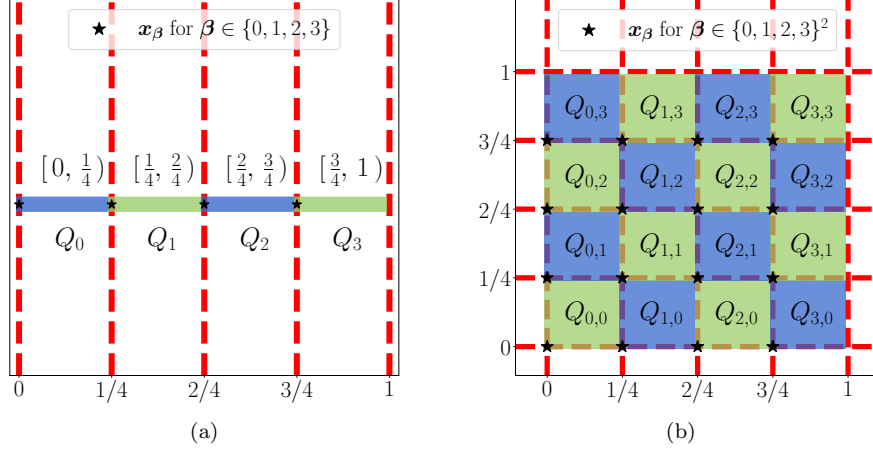


Figure 3: Illustrations of Q_β and \mathbf{x}_β for any $\beta \in \{0, 1, \dots, J-1\}^d$. (a) $J = 4, d = 1$. (b) $J = 4, d = 2$.

627 **Step 2:** Construct Φ_1 mapping $\mathbf{x} \in Q_\beta$ to β for each $\beta \in \{0, 1, \dots, J-1\}^d$.

628 Define

$$629 \quad \Phi_1(\mathbf{x}) := (\sigma_1(Jx_1), \sigma_1(Jx_2), \dots, \sigma_1(Jx_d)) = (\lfloor Jx_1 \rfloor, \lfloor Jx_2 \rfloor, \dots, \lfloor Jx_d \rfloor),$$

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

$$632 \quad \Phi_1(\mathbf{x}) = (\lfloor Jx_1 \rfloor, \lfloor Jx_2 \rfloor, \dots, \lfloor Jx_d \rfloor) = (\beta_1, \beta_2, \dots, \beta_d) = \beta. \quad (10)$$

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

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

$$636 \quad \phi_2(\mathbf{x}) := 1 + \sum_{i=1}^d J^{i-1} x_i, \quad \text{for each } \mathbf{x} = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d.$$

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

638 **Step 4:** Construct ϕ_3 mapping $\phi_2(\beta) \in \{1, 2, \dots, J^d\}$ approximately to $\tilde{f}(\mathbf{x}_\beta)$.

639

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

$$642 \quad \xi_k := \tilde{f}(\mathbf{x}_\beta) \in [0, 1], \quad \text{for any } k \in \{1, 2, \dots, J^d\} \text{ with } k = \phi_2(\beta). \quad (11)$$

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

$$644 \quad |\xi_k - \text{bin} 0.\theta_{k,1}\theta_{k,2}\dots\theta_{k,N}| \leq 2^{-N}. \quad (12)$$

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

$$647 \quad \sigma_3(2^k \cdot a_j) = \theta_{k,j}, \quad \text{for any } k \in \{1, 2, \dots, J^d\}.$$

648 Define

$$649 \quad \phi_3(x) := \sum_{j=1}^N 2^{-j} \sigma_3(a_j \cdot \sigma_2(x)) = \sum_{j=1}^N 2^{-j} \sigma_3(2^x \cdot a_j), \quad \text{for any } x \in \mathbb{R}.$$

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

$$651 \quad \phi_3(k) = \sum_{j=1}^N 2^{-j} \sigma_3(2^k \cdot a_j) = \sum_{j=1}^N 2^{-j} \cdot \theta_{k,j} = \text{bin} 0.\theta_{k,1}\theta_{k,2}\dots\theta_{k,N}. \quad (13)$$

652 **Step 5:** Define $\tilde{\phi} := \phi_3 \circ \phi_2 \circ \Phi_1$ approximating \tilde{f} well, and re-scale and
 653 shift $\tilde{\phi}$ to obtain ϕ approximating f well.

654 Define $\tilde{\phi} := \phi_3 \circ \phi_2 \circ \Phi_1$, by Equation (10), (11), (12), and (13), we have,
 655 for any $\mathbf{x} \in Q_\beta$ and each $\beta \in \{0, 1, \dots, J-1\}^d$ with $k = \phi_2(\beta)$,

$$\begin{aligned} |\tilde{\phi}(\mathbf{x}) - \tilde{f}(\mathbf{x})| &\leq |\phi_3 \circ \phi_2 \circ \Phi_1(\mathbf{x}) - \tilde{f}(\mathbf{x}_\beta)| + |\tilde{f}(\mathbf{x}_\beta) - \tilde{f}(\mathbf{x})| \\ 656 \quad &\leq |\phi_3 \circ \phi_2(\beta) - \tilde{f}(\mathbf{x}_\beta)| + \omega_{\tilde{f}}(\frac{\sqrt{d}}{J}) \leq |\phi_3(k) - \xi_k| + \omega_{\tilde{f}}(\frac{\sqrt{d}}{J}) \\ &\leq |\text{bin} 0.\theta_{k,1}\theta_{k,2}\dots\theta_{k,N} - \xi_k| + \omega_{\tilde{f}}(\frac{\sqrt{d}}{J}) \leq 2^{-N} + \omega_{\tilde{f}}(\frac{\sqrt{d}}{J}). \end{aligned}$$

657 Finally, define $\phi := 2\omega_f(\sqrt{d})\tilde{\phi} + f(\mathbf{0}) - \omega_f(\sqrt{d})$. Equation (9) implies
 658 $\omega_f(r) = 2\omega_f(\sqrt{d})\omega_{\tilde{f}}(r)$ for any $r \geq 0$, deducing

$$\begin{aligned} |\phi(\mathbf{x}) - f(\mathbf{x})| &= 2\omega_f(\sqrt{d})|\tilde{\phi}(\mathbf{x}) - \tilde{f}(\mathbf{x})| \leq 2\omega_f(\sqrt{d})(2^{-N} + \omega_{\tilde{f}}(\frac{\sqrt{d}}{J})) \\ 659 \quad &= 2\omega_f(\sqrt{d})2^{-N} + \omega_f(\frac{\sqrt{d}}{J}) \\ &= 2\omega_f(\sqrt{d})2^{-N} + \omega_f(\sqrt{d}2^{-N}), \end{aligned}$$

660 for any $\mathbf{x} \in \bigcup_{\beta \in \{0,1,\dots,J-1\}^d} Q_\beta = [0,1]^d$. It follows from $J = 2^N$ and the defini-
 661 tions of Φ_1 , ϕ_2 , and ϕ_3 that

$$\begin{aligned}
 \phi(\mathbf{x}) &= 2\omega_f(\sqrt{d})\phi_3 \circ \phi_2 \circ \Phi_1(\mathbf{x}) + f(\mathbf{0}) - \omega_f(\sqrt{d}) \\
 &= 2\omega_f(\sqrt{d})\phi_3\left(1 + \sum_{i=1}^d J^{i-1}\sigma_1(Jx_i)\right) + f(\mathbf{0}) - \omega_f(\sqrt{d}) \\
 &= 2\omega_f(\sqrt{d})\sum_{j=1}^N 2^{-j}\sigma_3\left(a_j \cdot \sigma_2\left(1 + \sum_{i=1}^d 2^{(i-1)N}\sigma_1(2^N x_i)\right)\right) + f(\mathbf{0}) - \omega_f(\sqrt{d}).
 \end{aligned}$$

663 So we finish the proof. \square

664 4. Approximation with continuous activation functions

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

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

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

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

675 where

$$\tilde{\mathcal{T}}(x) := \begin{cases} 0, & x \in (\cos(\frac{4\pi}{9}), \infty), \\ 1 - x/\cos(\frac{4\pi}{9}), & x \in [0, \cos(\frac{4\pi}{9})], \\ 1, & x \in (-\infty, 0) \end{cases}$$

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

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

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

$$684 \quad \|\phi - f\|_{L^p([0,1]^d)}^p \leq \left(2\omega_f(\sqrt{d})2^{-N} + \omega_f(\sqrt{d}2^{-N})\right)^p + 2d\delta(|f(\mathbf{0})| + \omega_f(\sqrt{d}))^p,$$

685 where ϕ is defined by a formula in a_1, a_2, \dots, a_N as follows

$$686 \quad \phi(\mathbf{x}) = 2\omega_f(\sqrt{d}) \sum_{j=1}^N 2^{-j} \varrho_3 \left(a_j \cdot \varrho_2 \left(1 + \sum_{i=1}^d 2^{(i-1)N} \varrho_{1,\delta}(2^N x_i) \right) \right) + f(\mathbf{0}) - \omega_f(\sqrt{d}).$$

687 The approximation error in Theorem 4.1 is characterized by L^p -norm
688 for $p \in [1, \infty)$ instead of a pointwise error estimate in Theorem 1.1. By
689 using ideas in Lu et al. (2020); Zhang (2020), we can extend this result to
690 L^∞ -norm. However, this extension requires $2d + 3$ hidden layers instead of 3
691 hidden layers. Since our focus here is the approximation using three hidden
692 layers, we will leave this extension as future work.

693 To prove Theorem 4.1, we need the following proposition.

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

$$696 \quad \varrho_3(a \cdot \varrho_2(k)) = \varrho_3(3^k \cdot a) = \theta_k, \quad \text{for any } k \in \{1, 2, \dots, K\},$$

697 where

$$698 \quad a = \sum_{j=1}^K 3^{-j-1} \cdot \theta_j \in [0, \frac{2}{9}).$$

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

$$700 \quad 0 \leq \sum_{j=1}^K 3^{-j-1} \cdot \theta_j \leq \sum_{j=1}^K 3^{-j-1} < \frac{2}{9},$$

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

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

$$703 \quad 3^k \cdot a = 3^k \cdot \sum_{j=1}^K 3^{-j-1} \cdot \theta_j = \underbrace{\sum_{j=1}^{k-1} 3^{k-j-1} \cdot \theta_j}_{\text{an integer}} + \overbrace{\frac{1}{3}\theta_k}^{0 \text{ or } \frac{1}{3}} + \underbrace{\sum_{j=k+1}^K 3^{k-j-1} \cdot \theta_j}_{\text{in } [0, \frac{2}{9})}. \quad (14)$$

Clearly, the first term $\sum_{j=1}^{k-1} 3^{k-j-1} \cdot \theta_j$ in Equation (14) is a non-negative integer since $\theta_j \in \{0, 1\}$ for any $j \in \{1, 2, \dots, K\}$. As for the third term in Equation (14), we have

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

Recall that

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

and

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

Note that

$$\tilde{\mathcal{T}}(x) := \begin{cases} 0, & x \in (\cos(\frac{4\pi}{9}), \infty), \\ 1 - x / \cos(\frac{4\pi}{9}), & x \in [0, \cos(\frac{4\pi}{9})], \\ 1, & x \in (-\infty, 0). \end{cases}$$

Therefore, if $\theta_k = 0$, by Equation (14), we have

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

Similarly, if $\theta_k = 1$, by Equation (14), we have

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

Since $k \in \{1, 2, \dots, K\}$ is arbitrary, we have $\varrho_3(a \cdot \varrho_2(k)) = \varrho_3(3^k \cdot a) = \theta_k$ for any $k \in \{1, 2, \dots, K\}$. So we finish the proof. \square

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

$$\Lambda([0, 1]^d, J, \delta) := \bigcup_{i=1}^d \left\{ \mathbf{x} = (x_1, \dots, x_d) \in [0, 1]^d : x_i \in \bigcup_{j=1}^{J-1} [\frac{j-\delta}{J}, \frac{j}{J}] \right\}.$$

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

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

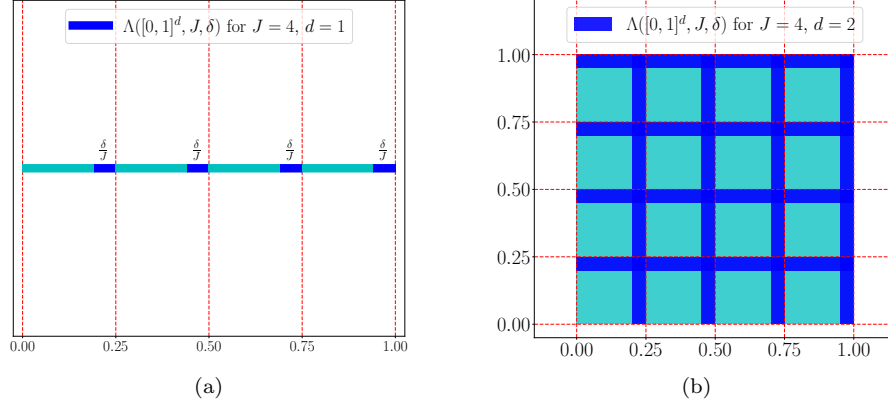


Figure 4: Illustrations of $\Lambda([0, 1]^d, J, \delta)$. (a) $J = 4$, $d = 1$. (b) $J = 4$, $d = 2$.

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

727 **Step 1:** Set up.

728 Assume f is not a constant function since it is a trivial case. Then
 729 $\omega_f(r) > 0$ for any $r > 0$. Clearly, $|f(\mathbf{x}) - f(\mathbf{0})| \leq \omega_f(\sqrt{d})$ for any $\mathbf{x} \in [0, 1]^d$.
 730 Define

$$731 \quad \tilde{f} := \frac{f - f(\mathbf{0}) + \omega_f(\sqrt{d})}{2\omega_f(\sqrt{d})}. \quad (15)$$

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

733 Set $J = 2^N$ and divide $[0, 1]^d$ into J^d cubes $\{Q_\beta\}_\beta$ and a small region
 734 $\Lambda([0, 1]^d, J, \delta)$. To be exact, define $\mathbf{x}_\beta := \beta/J$ and

$$735 \quad Q_\beta := \left\{ \mathbf{x} = (x_1, x_2, \dots, x_d) : x_i \in \left[\frac{\beta_i}{J}, \frac{\beta_i + 1 - \delta}{J} \right] \text{ for } i = 1, 2, \dots, d \right\},$$

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

737 **Step 2:** Construct Φ_1 mapping $\mathbf{x} \in Q_\beta$ to β for each $\beta \in \{0, 1, \dots, J-1\}^d$.

738 Define

$$739 \quad \Phi_1(\mathbf{x}) := \left(\varrho_{1,\delta}(Jx_1), \varrho_{1,\delta}(Jx_2), \dots, \varrho_{1,\delta}(Jx_d) \right),$$

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

$$742 \quad \Phi_1(\mathbf{x}) = \left(\varrho_{1,\delta}(Jx_1), \varrho_{1,\delta}(Jx_2), \dots, \varrho_{1,\delta}(Jx_d) \right) = (\beta_1, \beta_2, \dots, \beta_d) = \beta. \quad (16)$$

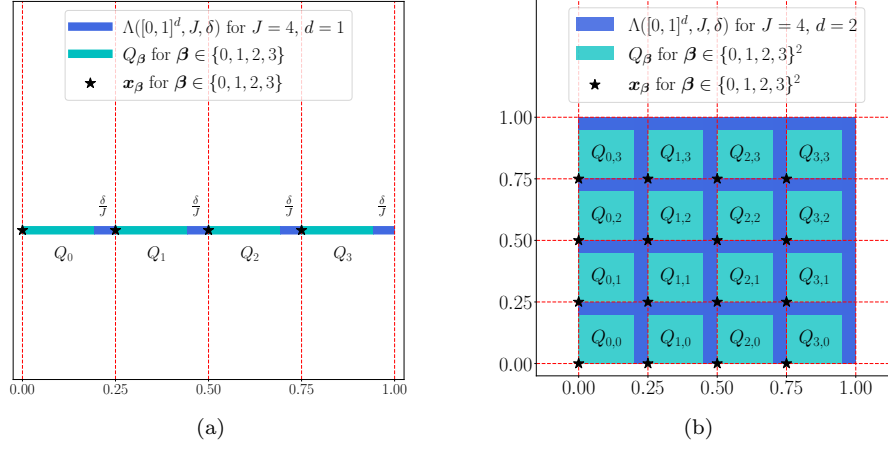


Figure 5: Illustrations of $\Lambda([0,1]^d, J, \delta)$, Q_β , and \mathbf{x}_β for any $\beta \in \{0,1,\dots,J-1\}^d$. (a) $J=4, d=1$. (b) $J=4, d=2$.

743 **Step 3:** Construct ϕ_2 bijectively mapping $\beta \in \{0,1,\dots,J-1\}^d$ to $\phi_2(\beta) \in$
744 $\{1,2,\dots,J^d\}$.

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

$$746 \quad \phi_2(\mathbf{x}) := 1 + \sum_{i=1}^d J^{i-1} x_i, \quad \text{for each } \mathbf{x} = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d.$$

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

748 **Step 4:** Construct ϕ_3 mapping $\phi_2(\beta) \in \{1,2,\dots,J^d\}$ approximately to $\tilde{f}(\mathbf{x}_\beta)$.

749

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

$$752 \quad \xi_k := \tilde{f}(\mathbf{x}_\beta) \in [0,1], \quad \text{for any } k \in \{1,2,\dots,J^d\} \text{ with } k = \phi_2(\beta). \quad (17)$$

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

$$754 \quad |\xi_k - \text{bin } 0.\theta_{k,1}\theta_{k,2}\dots\theta_{k,N}| \leq 2^{-N}. \quad (18)$$

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

$$757 \quad \varrho_3(3^k \cdot a_j) = \theta_{k,j}, \quad \text{for any } k \in \{1,2,\dots,J^d\}.$$

758 Define

$$759 \quad \phi_3(x) := \sum_{j=1}^N 2^{-j} \varrho_3(a_j \cdot \varrho_2(x)) = \sum_{j=1}^N 2^{-j} \varrho_3(3^x \cdot a_j), \quad \text{for any } x \in \mathbb{R}.$$

760 Then we have

$$761 \quad \varrho_3(x) \in [0, 1], \quad \text{for any } x \in \mathbb{R} \implies \phi_3(x) \in [0, 1], \quad \text{for any } x \in \mathbb{R}, \quad (19)$$

762 and

$$763 \quad \phi_3(k) = \sum_{j=1}^N 2^{-j} \varrho_3(3^k \cdot a_j) = \sum_{j=1}^N 2^{-j} \cdot \theta_{k,j} = \text{bin}0.\theta_{k,1}\theta_{k,2}\cdots\theta_{k,N}, \quad (20)$$

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

765 **Step 5:** Define $\tilde{\phi} := \phi_3 \circ \phi_2 \circ \Phi_1$ approximating \tilde{f} well, and re-scale and
766 shift $\tilde{\phi}$ to obtain ϕ approximating f well.

767 Define $\tilde{\phi} := \phi_3 \circ \phi_2 \circ \Phi_1$, by Equation (16), (17), (18), and (20), we have,
768 for any $\mathbf{x} \in Q_\beta$ and each $\beta \in \{0, 1, \dots, J-1\}^d$ with $k = \phi_2(\beta)$,

$$\begin{aligned} |\tilde{\phi}(\mathbf{x}) - \tilde{f}(\mathbf{x})| &\leq |\phi_3 \circ \phi_2 \circ \Phi_1(\mathbf{x}) - \tilde{f}(\mathbf{x}_\beta)| + |\tilde{f}(\mathbf{x}_\beta) - \tilde{f}(\mathbf{x})| \\ 769 \quad &\leq |\phi_3 \circ \phi_2(\beta) - \tilde{f}(\mathbf{x}_\beta)| + \omega_{\tilde{f}}\left(\frac{\sqrt{d}}{J}\right) \leq |\phi_3(k) - \xi_k| + \omega_{\tilde{f}}\left(\frac{\sqrt{d}}{J}\right) \\ &\leq |\text{bin}0.\theta_{k,1}\theta_{k,2}\cdots\theta_{k,N} - \xi_k| + \omega_{\tilde{f}}\left(\frac{\sqrt{d}}{J}\right) \leq 2^{-N} + \omega_{\tilde{f}}\left(\frac{\sqrt{d}}{J}\right). \end{aligned}$$

770 Finally, define $\phi := 2\omega_f(\sqrt{d})\tilde{\phi} + f(\mathbf{0}) - \omega_f(\sqrt{d})$. Equation (15) implies
771 $\omega_f(r) = 2\omega_f(\sqrt{d})\omega_{\tilde{f}}(r)$ for any $r \geq 0$, deducing

$$\begin{aligned} |\phi(\mathbf{x}) - f(\mathbf{x})| &= 2\omega_f(\sqrt{d})|\tilde{\phi}(\mathbf{x}) - \tilde{f}(\mathbf{x})| \leq 2\omega_f(\sqrt{d})(2^{-N} + \omega_{\tilde{f}}\left(\frac{\sqrt{d}}{J}\right)) \\ 772 \quad &= 2\omega_f(\sqrt{d})2^{-N} + \omega_f\left(\frac{\sqrt{d}}{J}\right), \end{aligned}$$

773 for any $\mathbf{x} \in \bigcup_{\beta \in \{0,1,\dots,J-1\}^d} Q_\beta$. By Equation (19) and the definition of

$$774 \quad \phi = 2\omega_f(\sqrt{d})\phi_3 \circ \phi_2 \circ \Phi_1 + f(\mathbf{0}) - \omega_f(\sqrt{d}),$$

775 we have $\|\phi\|_{L^\infty(\mathbb{R}^d)} \leq |f(\mathbf{0})| + \omega_f(\sqrt{d})$. Let $\mu(\cdot)$ denote the Lebesgue measure.

776 Note that $\|f\|_{L^\infty([0,1]^d)} \leq |f(\mathbf{0})| + \omega_f(\sqrt{d})$. It follows from $\mu(\Lambda([0,1]^d, J, \delta)) \leq$

777 $Jd\frac{\delta}{J} = d\delta$ that

$$\begin{aligned}
& \|\phi - f\|_{L^p([0,1]^d)}^p = \int_{[0,1]^d} |\phi(\mathbf{x}) - f(\mathbf{x})|^p d\mathbf{x} \\
& = \sum_{\beta \in \{0,1,\dots,J-1\}^d} \int_{Q_\beta} |\phi(\mathbf{x}) - f(\mathbf{x})|^p d\mathbf{x} + \int_{\Lambda([0,1]^d, J, \delta)} |\phi(\mathbf{x}) - f(\mathbf{x})|^p d\mathbf{x} \\
778 & \leq \sum_{\beta \in \{0,1,\dots,J-1\}^d} \mu(Q_\beta) \left(2\omega_f(\sqrt{d})2^{-N} + \omega_f\left(\frac{\sqrt{d}}{J}\right) \right)^p + (2|f(\mathbf{0})| + 2\omega_f(\sqrt{d}))^p d\delta \\
& \leq \left(2\omega_f(\sqrt{d})2^{-N} + \omega_f(\sqrt{d}2^{-N}) \right)^p + 2^p d\delta (|f(\mathbf{0})| + \omega_f(\sqrt{d}))^p.
\end{aligned}$$

779 By the definitions of Φ_1 , ϕ_2 , and ϕ_3 , we have

$$\begin{aligned}
\phi(\mathbf{x}) &= 2\omega_f(\sqrt{d})\phi_3 \circ \phi_2 \circ \Phi_1(\mathbf{x}) + f(\mathbf{0}) - \omega_f(\sqrt{d}) \\
&= 2\omega_f(\sqrt{d})\phi_3 \left(1 + \sum_{i=1}^d J^{i-1} \varrho_{1,\delta}(Jx_i) \right) + f(\mathbf{0}) - \omega_f(\sqrt{d}) \\
780 &= 2\omega_f(\sqrt{d}) \sum_{j=1}^N 2^{-j} \varrho_3 \left(a_j \cdot \varrho_2 \left(1 + \sum_{i=1}^d 2^{(i-1)N} \varrho_{1,\delta}(2^N x_i) \right) \right) + f(\mathbf{0}) - \omega_f(\sqrt{d}).
\end{aligned}$$

781 So we finish the proof. \square

782 5. Conclusion

783 This paper has introduced a theoretical framework to show that three
784 hidden layers are enough for neural network approximation to achieve expo-
785 nential convergence and avoid the curse of dimensionality for approxim-
786 ating functions as general as (Hölder) continuous functions. The key idea
787 is to leverage the power of multiple simple activation functions: the floor
788 function ($\lfloor x \rfloor$), the exponential function (2^x), the step function ($\mathbb{1}_{x \geq 0}$), or
789 their compositions. This new class of networks is called the FLES network.
790 Given a Lipschitz continuous function f on $[0,1]^d$, it was shown by con-
791 struction that FLES networks with width $\max\{d, N\}$ and three hidden lay-
792 ers admit a uniform approximation rate $6\lambda\sqrt{d}2^{-N}$, where λ is the Lipschitz
793 constant of f . More generally for an arbitrary continuous function f on
794 $[0,1]^d$ with a modulus of continuity $\omega_f(\cdot)$, the constructive approximation
795 rate is $2\omega_f(2\sqrt{d})2^{-N} + \omega_f(2\sqrt{d}2^{-N})$. We also extend such a result to gen-
796 eral bounded 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 114, 679–681. URL: <http://mi.mathnet.ru/dan22002>.
- 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](https://doi.org/10.1109/18.256500).
- Barron, A.R., Klusowski, J.M., 2018. Approximation and estimation for high-dimensional deep learning networks. arXiv e-prints [arXiv:1809.03090](https://arxiv.org/abs/1809.03090).
- Bartlett, P., Maiorov, V., Meir, R., 1998. Almost linear VC-dimension bounds for piecewise polynomial networks. Neural Computation 10, 2159–2173. doi:[10.1162/089976698300017016](https://doi.org/10.1162/089976698300017016).
- Bengio, Y., Léonard, N., Courville, A., 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv e-print [arXiv:1308.3432](https://arxiv.org/abs/1308.3432).
- Boo, Y., Shin, S., Sung, W., 2020. Quantized neural networks: Characterization and holistic optimization. arXiv e-print [arXiv:2006.00530](https://arxiv.org/abs/2006.00530).
- Braun, J., Griebel, M., 2009. On a constructive proof of Kolmogorov’s superposition theorem. Constructive Approximation 30, 653–675. doi:[10.1007/s00365-009-9054-2](https://doi.org/10.1007/s00365-009-9054-2).
- Carrillo, J.A.T., Jin, S., Li, L., Zhu, Y., 2019. A consensus-based global optimization method for high dimensional machine learning problems. arXiv e-print [arXiv:1909.09249](https://arxiv.org/abs/1909.09249).

- 827 Chen, L., Wu, C., 2019. A note on the expressive power of deep rectified
828 linear unit networks in high-dimensional spaces. *Mathematical Methods*
829 *in the Applied Sciences* 42, 3400–3404. doi:[10.1002/mma.5575](https://doi.org/10.1002/mma.5575).
- 830 Chen, M., Jiang, H., Liao, W., Zhao, T., 2019a. Efficient approximation
831 of deep ReLU networks for functions on low dimensional manifolds, in:
832 Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.,
833 Garnett, R. (Eds.), *Advances in Neural Information Processing Systems*
834 32. Curran Associates, Inc., pp. 8174–8184.
- 835 Chen, Z., Cao, Y., Zou, D., Gu, Q., 2019b. How much over-parameterization
836 is sufficient to learn deep ReLU networks? CoRR arXiv:1911.12360. URL:
837 <https://arxiv.org/abs/1911.12360>.
- 838 Du, S.S., Zhai, X., Póczos, B., Singh, A., 2019. Gradient descent prov-
839 ably optimizes over-parameterized neural networks, in: *International Con-*
840 *ference on Learning Representations*. URL: [https://openreview.net/](https://openreview.net/forum?id=S1eK3i09YQ)
841 [forum?id=S1eK3i09YQ](https://openreview.net/forum?id=S1eK3i09YQ).
- 842 E, W., Ma, C., Wu, L., 2019. A priori estimates of the population risk for
843 two-layer neural networks. *Communications in Mathematical Sciences* 17,
844 1407–1425. doi:[10.4310/CMS.2019.v17.n5.a11](https://doi.org/10.4310/CMS.2019.v17.n5.a11).
- 845 E, W., Wang, Q., 2018. Exponential convergence of the deep neural network
846 approximation for analytic functions. CoRR abs/1807.00297. URL: [http:](http://arxiv.org/abs/1807.00297)
847 [//arxiv.org/abs/1807.00297](http://arxiv.org/abs/1807.00297), [arXiv:1807.00297](https://arxiv.org/abs/1807.00297).
- 848 Gühring, I., Kutyniok, G., Petersen, P., 2019. Error bounds for approxi-
849 mations with deep ReLU neural networks in $W^{s,p}$ norms. arXiv e-prints
850 [arXiv:1902.07896](https://arxiv.org/abs/1902.07896).
- 851 Guliyev, N.J., Ismailov, V.E., 2018. Approximation capability of two hidden
852 layer feedforward neural networks with fixed weights. *Neurocomputing*
853 316, 262–269. doi:[10.1016/j.neucom.2018.07.075](https://doi.org/10.1016/j.neucom.2018.07.075).
- 854 Harvey, N., Liaw, C., Mehrabian, A., 2017. Nearly-tight VC-dimension
855 bounds for piecewise linear neural networks, in: Kale, S., Shamir, O.
856 (Eds.), *Proceedings of the 2017 Conference on Learning Theory*, PMLR,
857 Amsterdam, Netherlands. pp. 1064–1068. URL: [http://proceedings.](http://proceedings.mlr.press/v65/harvey17a.html)
858 [mlr.press/v65/harvey17a.html](http://proceedings.mlr.press/v65/harvey17a.html).

- 859 Holland, J.H., 1992. Genetic algorithms. *Scientific American* 267, 66–73.
860 URL: <http://www.jstor.org/stable/24939139>.
- 861 Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y., 2017.
862 Quantized neural networks: Training neural networks with low precision
863 weights and activations. *J. Mach. Learn. Res.* 18, 6869–6898.
- 864 Hutzenthaler, M., Jentzen, A., Wurstemberger, v.W., 2020. Overcoming the
865 curse of dimensionality in the approximative pricing of financial derivatives
866 with default risks. *Electron. J. Probab.* 25, 73 pp. doi:[10.1214/20-EJP423](https://doi.org/10.1214/20-EJP423).
- 867 Igel'nik, B., Parikh, N., 2003. Kolmogorov's spline network. *IEEE Transac-*
868 *tions on Neural Networks* 14, 725–733. doi:[10.1109/TNN.2003.813830](https://doi.org/10.1109/TNN.2003.813830).
- 869 Jacot, A., Gabriel, F., Hongler, C., 2018. Neural tangent kernel: Conver-
870 gence and generalization in neural networks, in: Bengio, S., Wallach, H.,
871 Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.), *Ad-*
872 *vances in Neural Information Processing Systems*. Curran Associates, Inc.,
873 volume 31, pp. 8571–8580. URL: [https://proceedings.neurips.cc/
874 paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf](https://proceedings.neurips.cc/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf).
- 875 Kennedy, J., Eberhart, R., 1995. Particle swarm optimization, in: *Pro-*
876 *ceedings of ICNN'95 - International Conference on Neural Networks*, pp.
877 1942–1948 vol.4. doi:[10.1109/ICNN.1995.488968](https://doi.org/10.1109/ICNN.1995.488968).
- 878 Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated
879 annealing. *Science* 220, 671–680. doi:[10.1126/science.220.4598.671](https://doi.org/10.1126/science.220.4598.671).
- 880 Kolmogorov, A.N., 1956. On the representation of continuous functions of
881 several variables by superposition of continuous functions of a smaller num-
882 ber of variables. *Dokl. Akad. Nauk SSSR* 108, 179–182. doi:[10.1007/
883 978-3-642-01742-1_5](https://doi.org/10.1007/978-3-642-01742-1_5).
- 884 Kolmogorov, A.N., 1957. On the representation of continuous functions of
885 several variables by superposition of continuous functions of one variable
886 and addition. *Dokl. Akad. Nauk SSSR* 114, 953–956. URL: [http://mi.
887 mathnet.ru/dan22050](http://mi.mathnet.ru/dan22050).
- 888 Kůrková, V., 1991. Kolmogorov's theorem is relevant. *Neural Computation*
889 3, 617–622. doi:[10.1162/neco.1991.3.4.617](https://doi.org/10.1162/neco.1991.3.4.617).

890 Kůrková, V., 1992. Kolmogorov’s theorem and multilayer neural networks.
891 Neural Networks 5, 501–506. doi:[10.1016/0893-6080\(92\)90012-8](https://doi.org/10.1016/0893-6080(92)90012-8).

892 Li, Q., Lin, T., Shen, Z., to appear. Deep learning via dynamical systems:
893 An approximation perspective. Journal of European Mathematical Society
894 .

895 Lin, Y., Lei, M., Niu, L., 2019. Optimization strategies in quantized neural
896 networks: A review, in: 2019 International Conference on Data Mining
897 Workshops (ICDMW), pp. 385–390. doi:[10.1109/ICDMW.2019.00063](https://doi.org/10.1109/ICDMW.2019.00063).

898 Lu, J., Shen, Z., Yang, H., Zhang, S., 2020. Deep network approximation for
899 smooth functions. arXiv e-prints [arXiv:2001.03040](https://arxiv.org/abs/2001.03040).

900 Lu, Y., Ma, C., Lu, Y., Lu, J., Ying, L., 2020. A mean-field analysis of deep
901 resnet and beyond: Towards provable optimization via overparameteriza-
902 tion from depth. CoRR abs/2003.05508. [arXiv:2003.05508](https://arxiv.org/abs/2003.05508).

903 Luo, T., Yang, H., 2020. Two-Layer Neural Networks for Partial Differen-
904 tial Equations: Optimization and Generalization Theory. arXiv e-prints
905 [arXiv:2006.15733](https://arxiv.org/abs/2006.15733).

906 Maierov, V., Pinkus, A., 1999. Lower bounds for approximation by MLP neu-
907 ral networks. Neurocomputing 25, 81–91. doi:[10.1016/S0925-2312\(98\)](https://doi.org/10.1016/S0925-2312(98)00111-8)
908 [00111-8](https://doi.org/10.1016/S0925-2312(98)00111-8).

909 Mei, S., Montanari, A., Nguyen, P.M., 2018. A mean field view of the land-
910 scape of two-layer neural networks. Proceedings of the National Academy
911 of Sciences 115, E7665–E7671. doi:[10.1073/pnas.1806579115](https://doi.org/10.1073/pnas.1806579115).

912 Montanelli, H., Du, Q., 2019. New error bounds for deep ReLU networks
913 using sparse grids. SIAM Journal on Mathematics of Data Science 1, 78–
914 92. doi:[10.1137/18M1189336](https://doi.org/10.1137/18M1189336).

915 Montanelli, H., Yang, H., 2020. Error bounds for deep ReLU networks using
916 the Kolmogorov-Arnold superposition theorem. Neural Networks 129, 1–6.
917 doi:[10.1016/j.neunet.2019.12.013](https://doi.org/10.1016/j.neunet.2019.12.013).

918 Montanelli, H., Yang, H., Du, Q., 2020. Deep ReLU networks overcome the
919 curse of dimensionality for bandlimited functions. Journal of Computa-
920 tional Mathematics .

- 921 Nelder, J., Mead, R., 1965. A simplex method for function minimization.
922 Comput. J. 7, 308–313. doi:[10.1093/comjnl/7.4.308](https://doi.org/10.1093/comjnl/7.4.308).
- 923 Opschoor, J.A., Schwab, C., Zech, J., 2019. Exponential ReLU DNN
924 expression of holomorphic maps in high dimension. Technical Report.
925 Seminar for Applied Mathematics, ETH Zürich. Zurich. URL: <https://math.ethz.ch/sam/research/reports.html?id=839>.
926
- 927 Petersen, P., Voigtlaender, F., 2018. Optimal approximation of piecewise
928 smooth functions using deep ReLU neural networks. Neural Networks
929 108, 296–330. doi:[10.1016/j.neunet.2018.08.019](https://doi.org/10.1016/j.neunet.2018.08.019).
- 930 Pinnau, R., Totzeck, C., Tse, O., Martin, S., 2017. A consensus-based
931 model for global optimization and its mean-field limit. Mathematical
932 Models and Methods in Applied Sciences 27, 183–204. doi:[10.1142/S0218202517400061](https://doi.org/10.1142/S0218202517400061).
933
- 934 Poggio, T., Mhaskar, H.N., Rosasco, L., Miranda, B., Liao, Q., 2017. Why
935 and when can deep—but not shallow—networks avoid the curse of dimen-
936 sionality: A review. International Journal of Automation and Computing
937 14, 503–519. doi:[10.1007/s11633-017-1054-2](https://doi.org/10.1007/s11633-017-1054-2).
- 938 Schmidt-Hieber, J., 2020. Nonparametric regression using deep neural net-
939 works with ReLU activation function. Annals of Statistics 48, 1875–1897.
940 URL: <https://projecteuclid.org/euclid.aos/1597370649>.
- 941 Schmidt-Hieber, J., 2021. The Kolmogorov–Arnold representation theorem
942 revisited. Neural Networks 137, 119–126. doi:[10.1016/j.neunet.2021.01.020](https://doi.org/10.1016/j.neunet.2021.01.020).
943
- 944 Shen, Z., Yang, H., Zhang, S., 2019. Nonlinear approximation via composi-
945 tions. Neural Networks 119, 74–84. doi:[10.1016/j.neunet.2019.07.011](https://doi.org/10.1016/j.neunet.2019.07.011).
- 946 Shen, Z., Yang, H., Zhang, S., 2020. Deep network approximation charac-
947 terized by number of neurons. Communications in Computational Physics
948 28, 1768–1811. doi:[10.4208/cicp.0A-2020-0149](https://doi.org/10.4208/cicp.0A-2020-0149).
- 949 Shen, Z., Yang, H., Zhang, S., 2021. Deep network with approximation
950 error being reciprocal of width to power of square root of depth. Neural
951 Computation 33, 1005–1036. doi:[10.1162/neco_a_01364](https://doi.org/10.1162/neco_a_01364).

952 Shen, Z., Yang, H., Zhang, S., to appear. Optimal approximation rate of relu
 953 networks in terms of width and depth. *Journal de Mathématiques Pures*
 954 *et Appliquées* .

955 Wang, P., Hu, Q., Zhang, Y., Zhang, C., Liu, Y., Cheng, J., 2018. Two-step
 956 quantization for low-bit neural networks, in: 2018 IEEE/CVF Conference
 957 on Computer Vision and Pattern Recognition, pp. 4376–4384. doi:[10.
 958 1109/CVPR.2018.00460](https://doi.org/10.1109/CVPR.2018.00460).

959 Wu, L., Ma, C., E, W., 2018. How sgd selects the global minima in over-
 960 parameterized learning: A dynamical stability perspective, in: Bengio, S.,
 961 Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R.
 962 (Eds.), *Advances in Neural Information Processing Systems* 31. Curran
 963 Associates, Inc., pp. 8279–8288. URL: [https://papers.nips.cc/paper/
 964 2018/hash/6651526b6fb8f29a00507de6a49ce30f-Abstract.html](https://papers.nips.cc/paper/2018/hash/6651526b6fb8f29a00507de6a49ce30f-Abstract.html).

965 Yang, Y., Wang, Y., 2020. Approximation in shift-invariant spaces with deep
 966 ReLU neural networks. *arXiv e-prints* [arXiv:2005.11949](https://arxiv.org/abs/2005.11949).

967 Yarotsky, D., 2017. Error bounds for approximations with deep ReLU net-
 968 works. *Neural Networks* 94, 103–114. doi:[10.1016/j.neunet.2017.07.
 969 002](https://doi.org/10.1016/j.neunet.2017.07.002).

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

974 Yarotsky, D., Zhevnerchuk, A., 2020. The phase diagram of
 975 approximation rates for deep neural networks 33, 13005–13015.
 976 URL: [https://proceedings.neurips.cc/paper_files/paper/2020/
 977 hash/979a3f14bae523dc5101c52120c535e9-Abstract.html](https://proceedings.neurips.cc/paper_files/paper/2020/hash/979a3f14bae523dc5101c52120c535e9-Abstract.html).

978 Yin, P., Lyu, J., Zhang, S., Osher, S.J., Qi, Y., Xin, J., 2019. Understand-
 979 ing straight-through estimator in training activation quantized neural nets
 980 URL: <https://openreview.net/forum?id=Skh4jRcKQ>.

981 Zhang, S., 2020. Deep neural network approximation via function com-
 982 positions. PhD Thesis, National University of Singapore URL: [https:
 983 //scholarbank.nus.edu.sg/handle/10635/186064](https://scholarbank.nus.edu.sg/handle/10635/186064).