

# Neural Network Approximation: Three Hidden Layers Are Enough

Zuowei Shen<sup>a</sup>, Haizhao Yang<sup>b</sup>, Shijun Zhang<sup>a</sup>

<sup>a</sup>*Department of Mathematics, National University of Singapore*

<sup>b</sup>*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 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](mailto:matzuows@nus.edu.sg) (Zuowei Shen), [haizhao@purdue.edu](mailto:haizhao@purdue.edu) (Haizhao Yang), [zhangshijun@u.nus.edu](mailto: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 \(2020b\)](#); [Petersen and Voigtlaender \(2018\)](#); [Yarotsky \(2018\)](#); [Gühring et al. \(2019\)](#); [Yarotsky and Zhevnerchuk \(2019\)](#) and the reference therein. Obviously, the curse of dimensionality exists in ReLU networks for these generic functions and, therefore, ReLU networks would need to be exponentially large in  $d$  to maintain a reasonably good approximation accuracy. The curse could be lessened when target function spaces are smaller. To name a few, [Poggio et al. \(2017\)](#); [Barron and Klusowski \(2018\)](#); [E et al. \(2019\)](#); [Montanelli et al. \(2020\)](#); [Chen et al. \(2019a\)](#); [Hutzenthaler et al. \(2020\)](#) and reference therein for ReLU networks. The limitation of ReLU networks motivated the work in [Shen et al. \(2020\)](#) to introduce Floor-ReLU networks built with either a Floor ( $\lfloor x \rfloor$ ) or ReLU ( $\max\{0, x\}$ ) activation function in each neuron. It was shown

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

The most important message of Shen et al. (2020) (and probably also of Yarotsky and Zhevnerchuk (2019)) is that the combination of simple activation functions can create super approximation power. In the Floor-ReLU networks mentioned above, the power of depth is fully reflected in the approximation rate  $3\lambda d^{\alpha/2} N^{-\alpha\sqrt{L}}$  that is root-exponential in depth. However, the power of width is much weaker and the approximation rate is polynomial in width if depth is fixed. This seems to be inconsistent with recent development of network optimization theory Jacot et al. (2018); Du et al. (2019); Mei et al. (2018); Wu et al. (2018); Chen et al. (2019b); Lu et al. (2020); Luo and Yang (2020), where larger width instead of depth can ease the challenge of highly 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  $\sigma_1$ ,  $\sigma_2$ , and  $\sigma_3$  in FLES networks (see Figure 1 for an illustration):

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

where

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

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

$$69 \quad |\phi(\mathbf{x}) - f(\mathbf{x})| \leq 2\omega_f(2\sqrt{d})2^{-N} + \omega_f(2\sqrt{d}2^{-N}),$$

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

$$72 \quad \phi(\mathbf{x}) = 2\omega_f(2\sqrt{d}) \sum_{j=1}^N 2^{-j} \sigma_3 \left( a_j \cdot \sigma_2 \left( 1 + \sum_{i=1}^d 2^{(i-1)N} \sigma_1(2^{N-1} x_i) \right) \right) + f(\mathbf{0}) - \omega_f(2\sqrt{d}).$$

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

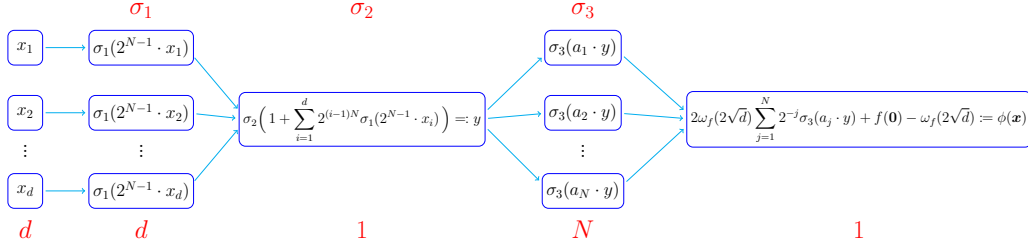


Figure 1: An illustration of the desired three-hidden-layer network in Theorem 1.1 for any  $\mathbf{x} = (x_1, x_2, \dots, x_d) \in \mathbb{R}$ . Each of the red functions “ $\sigma_1$ ”, “ $\sigma_2$ ”, and “ $\sigma_3$ ” above the network is the activation function of the corresponding hidden layer. The number of neurons in each hidden layer is indicated by the red number below it.

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

$$82 \quad |f(\mathbf{x}) - f(\mathbf{y})| \leq \lambda \|\mathbf{x} - \mathbf{y}\|_2^\alpha, \quad \text{for any } \mathbf{x}, \mathbf{y} \in [0, 1]^d, \quad (1)$$

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

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

$$91 \quad |\phi(\mathbf{x}) - f(\mathbf{x})| \leq 3\lambda(2\sqrt{d})^\alpha 2^{-\alpha N}, \quad \text{for any } \mathbf{x} = (x_1, x_2, \dots, x_d) \in [0, 1]^d,$$

92 *where  $\phi$  is defined by a formula in  $a_1, a_2, \dots, a_N$  as follows.*

$$93 \quad \phi(\mathbf{x}) = 2\omega_f(2\sqrt{d}) \sum_{j=1}^N 2^{-j} \sigma_3 \left( a_j \cdot \sigma_2 \left( 1 + \sum_{i=1}^d 2^{(i-1)N} \sigma_1(2^{N-1} x_i) \right) \right) + f(\mathbf{0}) - \omega_f(2\sqrt{d}).$$

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

105 The rest of this paper is organized as follows. In Section 2, we discuss  
 106 the application scope of our theory, study the connection between the ap-  
 107 proximation error and the Vapnik–Chervonenkis (VC) dimension, establish  
 108 Corollary 2.3 to extend our analysis to general continuous functions on a  
 109 bounded set, and compare related works in the literature. We will prove  
 110 Theorem 1.1 and Corollary 2.3 in Section 3. In Section 4, we explore alter-  
 111 native continuous activation functions other than  $\sigma_1$ ,  $\sigma_2$ , and  $\sigma_3$  for super  
 112 approximation power. Finally, we conclude this paper in Section 5.

## 113 2. Discussion

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

### 116 2.1. Application scope of our theory in machine learning

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

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

$$120 \quad R_{\mathcal{D}}(\boldsymbol{\theta}) := \mathbb{E}_{\mathbf{x} \sim U(\mathcal{X})} [\ell(\phi(\mathbf{x}; \boldsymbol{\theta}), f(\mathbf{x}))]$$

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

$$124 \quad R_{\mathcal{D}}(\boldsymbol{\theta}) = \int_{[0,1]^d} \frac{1}{2} |\phi(\mathbf{x}; \boldsymbol{\theta}) - f(\mathbf{x})|^2 d\mathbf{x}.$$

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

$$126 \quad \boldsymbol{\theta}_{\mathcal{D}} := \arg \min_{\boldsymbol{\theta}} R_{\mathcal{D}}(\boldsymbol{\theta}).$$

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

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

$$130 \quad R_{\mathcal{S}}(\boldsymbol{\theta}) := \frac{1}{n} \sum_{i=1}^n \ell(\phi(\mathbf{x}_i; \boldsymbol{\theta}), f(\mathbf{x}_i)).$$

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

$$133 \quad \boldsymbol{\theta}_{\mathcal{S}} = \arg \min_{\boldsymbol{\theta}} R_{\mathcal{S}}(\boldsymbol{\theta}), \tag{2}$$

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

$$139 \quad R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}}) = \mathbb{E}_{\mathbf{x} \sim U(\mathcal{X})} [\ell(\phi(\mathbf{x}; \boldsymbol{\theta}_{\mathcal{N}}), f(\mathbf{x}))] \stackrel{e.g.}{=} \int_{[0,1]^d} \frac{1}{2} |\phi(\mathbf{x}; \boldsymbol{\theta}_{\mathcal{N}}) - f(\mathbf{x})|^2 d\mathbf{x},$$

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

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

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

147 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  
 148  $\boldsymbol{\theta}_{\mathcal{S}}$  is a global minimizer of  $R_{\mathcal{S}}(\boldsymbol{\theta})$ . Constructive approximation provides  
 149 an upper bound of  $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$  in terms of the network size, e.g., in terms of  
 150 the network width and depth, or in terms of the number of parameters.  
 151 The second term of Equation (3) is bounded by the optimization error of the  
 152 numerical algorithm applied to solve the empirical loss minimization problem  
 153 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,  
 154 but not  $\boldsymbol{\theta}_{\mathcal{N}} - \boldsymbol{\theta}_{\mathcal{S}}$ . The study of the bounds for the third and fourth terms is  
 155 referred to as the generalization error analysis of neural networks. See Figure  
 156 2 for the intuitions of these three errors.

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

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

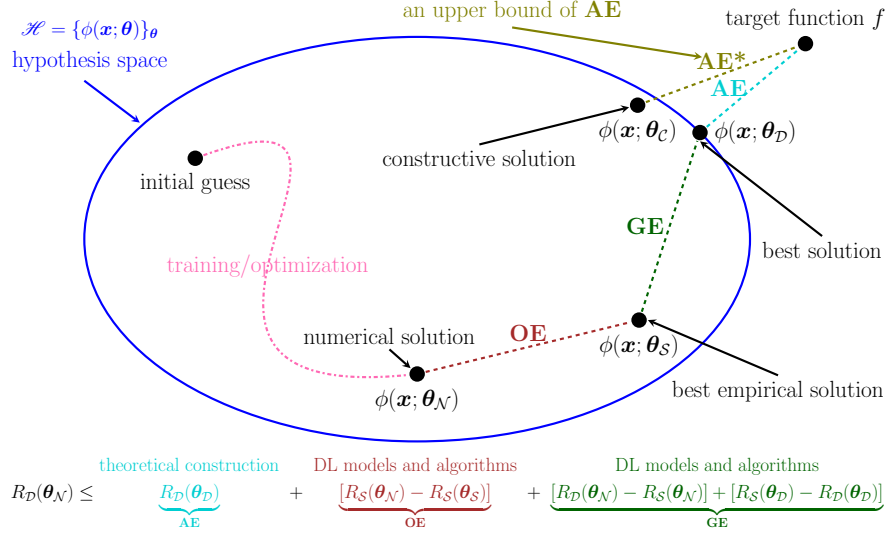


Figure 2: The intuitions of the approximation error (AE), the optimization error (OE), and the generalization error (GE). DL is short of deep learning. One needs to control AE, OE, and GE in order to bound the discrepancy between the target function  $f$  and the numerical solution  $\phi(\mathbf{x};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

177 networks are two other separate future directions.

## 178 2.2. Connection between approximation error and VC-dimension

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

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

$$185 \quad \left| \left\{ [h(\mathbf{x}_1), h(\mathbf{x}_2), \dots, h(\mathbf{x}_m)]^T \in \{0, 1\}^m : h \in H \right\} \right| = 2^m,$$

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

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

$$189 \quad \Pi_H(m) := \max_{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m \in \mathcal{X}} \left| \left\{ [h(\mathbf{x}_1), h(\mathbf{x}_2), \dots, h(\mathbf{x}_m)]^T \in \{0, 1\}^m : h \in H \right\} \right|.$$



190 **Definition 2.1** (VC-dimension). Let  $H$  be a class of functions from  $\mathcal{X}$  to  
 191  $\{0, 1\}$ . The VC-dimension of  $H$ , denoted by  $\text{VCDim}(H)$ , is the size of the  
 192 largest shattered set, namely,

$$193 \quad \text{VCDim}(H) := \sup \{m \in \mathbb{N}^+ : \Pi_H(m) = 2^m\}.$$

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

$$196 \quad \mathcal{T}(t) := \begin{cases} 1, & t \geq 0, \\ 0, & t < 0 \end{cases} \quad \text{and} \quad \mathcal{T} \circ \mathcal{F} := \{\mathcal{T} \circ f : f \in \mathcal{F}\}.$$

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

200 As shown in Yarotsky (2018, 2017); Shen et al. (2019, 2020); Lu et al.  
 201 (2020); Shen et al. (2020); Zhang (2020), VC-dimension essentially deter-  
 202 mines the lower bound of the approximation errors of networks. For sim-  
 203 plicity, we use  $\text{Hölder}([0, 1]^d, \alpha, \lambda)$  as an example, where  $\text{Hölder}([0, 1]^d, \alpha, \lambda)$   
 204 denotes the space of Hölder continuous functions of order  $\alpha \in (0, 1]$  and a  
 205 Hölder constant  $\lambda > 0$ . Without loss of generality, we assume  $\lambda = 1$ . Theorem  
 206 2.2 below shows that the best possible approximation error of functions in  
 207  $\text{Hölder}([0, 1]^d, \alpha, 1)$  approximated by functions in  $\mathcal{F}$  is bounded by a formula  
 208 characterized by  $\text{VCDim}(\mathcal{F})$ .

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

$$211 \quad \inf_{\phi \in \mathcal{F}} \|\phi - f\|_{L^\infty([0, 1]^d)} \leq \varepsilon, \quad \text{for any } f \in \text{Hölder}([0, 1]^d, \alpha, 1). \quad (4)$$

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

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

$$217 \quad \mathcal{E}_{\alpha, d}(\mathcal{F}) := \sup_{f \in \text{Hölder}([0, 1]^d, \alpha, 1)} \left( \inf_{\phi \in \mathcal{F}} \|\phi - f\|_{L^\infty([0, 1]^d)} \right).$$

218 Then, Theorem 2.2 implies that

$$219 \quad \text{VCDim}(\mathcal{F})^{-\alpha/d}/9 \leq \mathcal{E}_{\alpha,d}(\mathcal{F}), \quad (5)$$

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

$$225 \quad \text{VCDim}(\mathcal{F}_{N,L}) \leq \mathcal{O}(N^2 L \cdot L \cdot \ln(N^2 L)) \leq \mathcal{O}(N^2 L^2 \ln(NL)),$$

226 where  $\mathcal{F}_{N,L}$  is the space consisting of all functions implemented by ReLU  
 227 networks with width  $N$  and depth  $L$ . It is shown in Section 4.4.1 of Zhang  
 228 (2020) that

$$229 \quad C_1(\alpha, d) \cdot \left(N^2 L^2 \ln(NL)\right)^{-\alpha/d} \leq \mathcal{E}_{\alpha,d}(\mathcal{F}_{N,L}) \leq C_2(\alpha, d) \cdot \left(N^2 L^2\right)^{-\alpha/d},$$

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

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

$$237 \quad \text{VCDim}\left(\left\{\phi : \phi(x) = \cos(ax), a \in \mathbb{R}\right\}\right) = \infty.$$

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

248 *2.3. Further interpretation of our theory*

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

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

$$261 \quad \omega_f^E(r) := \sup \{ |f(\mathbf{x}) - f(\mathbf{y})| : \|\mathbf{x} - \mathbf{y}\|_2 \leq r, \mathbf{x}, \mathbf{y} \in E \}, \quad \text{for any } r \geq 0.$$

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

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

$$266 \quad |\phi(\mathbf{x}) - f(\mathbf{x})| \leq 2\omega_f^E(3R\sqrt{d})2^{-N} + \omega_f^E(3R\sqrt{d}2^{-N}),$$

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

$$269 \quad \phi(\mathbf{x}) = 2\omega_f^E(3R\sqrt{d}) \sum_{j=1}^N 2^{-j} \sigma_3 \left( a_j \cdot \sigma_2 \left( 1 + \sum_{i=1}^d 2^{(i-1)N} \sigma_1(2^N \frac{x_i + R}{3R}) \right) \right) + C_f,$$

270 where  $C_f$  is a constant determined by  $f$ .

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

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

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

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

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

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

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

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

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

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

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

#### 2.4. Kolmogorov-Arnold Superposition Theorem

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

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

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

There has been an active research line to develop more practical network approximation based on KST Kůrková (1991, 1992); Maiorov and Pinkus (1999); Guliyev and Ismailov (2018); Montanelli and Yang (2020); Igel'nik and Parikh (2003); Schmidt-Hieber (2020a) by relaxing the exact representation to network approximation with an  $\varepsilon$ -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 \(2020a\)](#), resulting in cubic-spline networks or deep ReLU networks to approximate  $f \in C([0, 1]^d)$ . But the approximation bounds in these works still suffer from the curse of dimensionality unless  $f$  has simple outer functions in the KST. It is still an open problem to characterize the class of functions with a moderate outer function in KST.

To the best of our knowledge, the most successful construction of neural networks with  $f$ -independent activation functions conquering the curse of dimensionality is in [Maiorov and Pinkus \(1999\)](#); [Guliyev and Ismailov \(2018\)](#), where a two-hidden-layer network with  $\mathcal{O}(d)$  neurons can approximate  $f \in C([0, 1]^d)$  within an arbitrary error  $\varepsilon$ . 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  $\varrho$  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  $\delta$ . 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  $\varepsilon$ . Unfortunately, the construction of the parameters of this magic network relies on the evaluation of the outer and inner functions of KST, which is not computationally feasible even if computation with arbitrary

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

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

precision is allowed.

We would like to remark that, though the approximation rate of FLES networks in this paper is relatively worse than the approximation rate in 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.<sup>1</sup> As shown in Table 1, there exists a trade-off between the complexity of activation functions and the network size when the approximation error is fixed. A key advantage of our FLES networks is to use simple and explicit activation functions to attain an exponential convergence rate.

## 2.5. Discussion on the literature

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

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

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



413 following form were considered:

$$414 \quad f(\mathbf{x}) = \int_{\tilde{\Omega}} a(\mathbf{w}) K(\mathbf{w} \cdot \mathbf{x}) d\mu(\mathbf{w}), \quad (7)$$

415 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  
 416 integral representation is equivalent to the expectation of a high-dimensional  
 417 random function when  $\mathbf{w}$  is treated as a random variable. By the law of  
 418 large number theory, the average of  $N$  samples of the integrand leads to  
 419 an approximation of  $f(\mathbf{x})$  with an approximation error bounded by  $\frac{C_f \sqrt{\mu(\Omega)}}{\sqrt{N}}$   
 420 measured in  $L^2(\Omega, \mu)$  (Equation (6) of [Barron \(1993\)](#)), where  $\mathcal{O}(N)$  is the  
 421 total number of parameters in the network,  $C_f$  is a  $d$ -dimensional integral  
 422 with an integrand related to  $f$ , and  $\mu(\Omega)$  is the Lebesgue measure of  $\Omega$ .  
 423 As discussed in [Barron \(1993\)](#),  $\mu(\Omega)$  and  $C_f$  would be exponential in  $d$  and  
 424 standard smoothness properties of  $f$  alone are not enough to remove the  
 425 exponential dependence of  $C_f$  on  $d$ . Therefore, the curse of dimensionality  
 426 exists in the whole approximation error while the curse does not exist in the  
 427 approximation rate in  $N$ .

428 Linear approximation is an efficient approximation tool for smooth func-  
 429 tions that computes the approximant of a target function via a linear projec-  
 430 tion to a Hilbert space or a Banach space as the approximant space. Typical  
 431 examples include approximation via orthogonal polynomials, Fourier series  
 432 expansion, etc. Inspired by the seminal work in [Yarotsky \(2017\)](#), where deep  
 433 ReLU networks were constructed to approximate polynomials with exponen-  
 434 tial convergence, subsequent works in [E and Wang \(2018\)](#); [Opschoor et al. \(2019\)](#);  
 435 [Montanelli and Du \(2019\)](#); [Chen and Wu \(2019\)](#); [Montanelli et al. \(2020\)](#);  
 436 [Yarotsky and Zhevnerchuk \(2019\)](#); [Lu et al. \(2020\)](#); [Montanelli and Yang \(2020\)](#);  
 437 [Yang and Wang \(2020\)](#) have constructed deep ReLU networks to  
 438 approximate various smooth function spaces. The main idea of these works is  
 439 to approximate smooth functions via (piecewise) polynomial approximation  
 440 first and then construct deep ReLU networks to approximate the ensemble  
 441 of polynomials. But the curse of dimensionality exists since polynomial ap-  
 442 proximation cannot avoid the curse. Finally, a different approach is used in  
 443 [Li et al. \(2019\)](#). The authors of [Li et al. \(2019\)](#) use a dynamic system based  
 444 approach to obtain a universal approximation property of residual networks.

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



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

### 3. Theoretical Analysis

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

#### 3.1. Notations

The main notations of this paper are listed as follows.

- Vectors and matrices are denoted in a bold font. Standard vectorization is adopted in the matrix and vector computation. For example, adding a scalar and a vector means adding the scalar to each entry of the vector.
- Let  $\mathbb{Z}$ ,  $\mathbb{N}$ , and  $\mathbb{N}^+$  denote the set of integers, natural numbers, all positive integers, respectively, i.e.,  $\mathbb{Z} = \{0, 1, 2, \dots\} \cup \{-1, -2, -3, \dots\}$ ,  $\mathbb{N} = \{0, 1, 2, \dots\}$ , and  $\mathbb{N}^+ = \{1, 2, 3, \dots\}$ .

- 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 defined by

$$\|\mathbf{x}\|_p := (|x_1|^p + |x_2|^p + \dots + |x_d|^p)^{1/p}.$$

- Let  $\sigma : \mathbb{R} \rightarrow \mathbb{R}$  denote the rectified linear unit (ReLU), i.e.  $\sigma(x) = \max\{0, x\}$ . With a slight abuse of notation, we define  $\sigma : \mathbb{R}^d \rightarrow \mathbb{R}^d$  as
- $$\sigma(\mathbf{x}) = \begin{bmatrix} \max\{0, x_1\} \\ \vdots \\ \max\{0, x_d\} \end{bmatrix} \text{ for any } \mathbf{x} = (x_1, \dots, x_d) \in \mathbb{R}^d.$$

- 481 • The floor function (Floor) is defined as  $\lfloor x \rfloor := \max\{n : n \leq x, n \in \mathbb{Z}\}$  for  
482 any  $x \in \mathbb{R}$ .
- 483 • For  $\theta \in [0, 1)$ , suppose its binary representation is  $\theta = \sum_{\ell=1}^{\infty} \theta_{\ell} 2^{-\ell}$  with  
484  $\theta_{\ell} \in \{0, 1\}$ , we introduce a special notation  $\text{bin}0.\theta_1\theta_2\cdots\theta_L$  to denote the  
485  $L$ -term binary representation of  $\theta$ , i.e.,  $\text{bin}0.\theta_1\theta_2\cdots\theta_L := \sum_{\ell=1}^L \theta_{\ell} 2^{-\ell}$ .
- 486 • The expression “a network with width  $N$  and depth  $L$ ” means
  - 487 – The maximum width of this network for all **hidden** layers is no  
488 more than  $N$ .
  - 489 – The number of **hidden** layers of this network is no more than  $L$ .

### 490 3.2. Proof of Theorem 1.1 and Corollary 2.3

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

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

$$496 |\phi(\mathbf{x}) - f(\mathbf{x})| \leq 2\omega_f(\sqrt{d})2^{-N} + \omega_f(\sqrt{d}2^{-N}),$$

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

$$499 \phi(\mathbf{x}) = 2\omega_f(\sqrt{d}) \sum_{j=1}^N 2^{-j} \sigma_3 \left( a_j \cdot \sigma_2 \left( 1 + \sum_{i=1}^d 2^{(i-1)N} \sigma_1(2^N x_i) \right) \right) + f(\mathbf{0}) - \omega_f(\sqrt{d}).$$

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

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

503 *Proof of Theorem 1.1.* Given any  $f \in C([0, 1]^d)$ , by Lemma 4.2 of Shen et al.  
504 (2020) via setting  $E = [0, 1]^d$  and  $S = \mathbb{R}^d$ , there exists  $g \in C(\mathbb{R}^d)$  such that

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

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

$$509 \quad |\tilde{\phi}(\mathbf{x}) - \tilde{g}(\mathbf{x})| \leq 2\omega_g^S(\sqrt{d})2^{-N} + \omega_g^S(\sqrt{d}2^{-N}), \quad \text{for any } \mathbf{x} \in [0, 1]^d, \quad (8)$$

510 where

$$511 \quad \tilde{\phi}(\mathbf{x}) = 2\omega_g^S(\sqrt{d}) \sum_{j=1}^N 2^{-j} \sigma_3 \left( a_j \cdot \sigma_2 \left( 1 + \sum_{i=1}^d 2^{(i-1)N} \sigma_1(2^N x_i) \right) \right) + \tilde{g}(\mathbf{0}) - \omega_g^S(\sqrt{d}).$$

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

$$513 \quad \omega_g^S(r) = \omega_g^S(2r) = \omega_f^E(2r) = \omega_f(2r), \quad \text{for any } r \geq 0.$$

514 Define  $\phi(\mathbf{x}) := \tilde{\phi}(2\mathbf{x})$  for any  $\mathbf{x} \in \mathbb{R}^d$ . Then by Equation (8), for any  $\mathbf{x} \in$   
 515  $[0, 1]^d = E$ , we have  $\frac{\mathbf{x}}{2} \in [0, \frac{1}{2}]^d \subseteq [0, 1]^d$ , implying

$$\begin{aligned} |\phi(\mathbf{x}) - f(\mathbf{x})| &= |\phi(\mathbf{x}) - g(\mathbf{x})| = |\tilde{\phi}(\frac{\mathbf{x}}{2}) - \tilde{g}(\frac{\mathbf{x}}{2})| \\ 516 \quad &\leq 2\omega_g^S(\sqrt{d})2^{-N} + \omega_g^S(\sqrt{d}2^{-N}) \\ &= 2\omega_f(2\sqrt{d})2^{-N} + \omega_f(2\sqrt{d}2^{-N}), \end{aligned}$$

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

$$518 \quad 2\omega_f(2\sqrt{d}) \sum_{j=1}^N 2^{-j} \sigma_3 \left( a_j \cdot \sigma_2 \left( 1 + \sum_{i=1}^d 2^{(i-1)N} \sigma_1(2^{N-1} x_i) \right) \right) + f(\mathbf{0}) - \omega_f(2\sqrt{d}).$$

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

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

521 *Proof of Corollary 2.3.* Given any  $f \in C(E)$ , by Lemma 4.2 of Shen et al.  
 522 (2020) via setting  $S = \mathbb{R}^d$ , there exists  $g \in C(\mathbb{R}^d)$  such that

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

525 Define

$$526 \quad \tilde{g}(\mathbf{x}) := g(3R\mathbf{x} - R), \quad \text{for any } \mathbf{x} \in \mathbb{R}^d.$$

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 that

$$|\tilde{\phi}(\mathbf{x}) - \tilde{g}(\mathbf{x})| \leq 2\omega_g^S(\sqrt{d})2^{-N} + \omega_g^S(\sqrt{d}2^{-N}), \quad \text{for any } \mathbf{x} \in [0, 1]^d, \quad (9)$$

where

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

Recall 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

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

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

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

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

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

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

### 3.3. Proof of Theorem 3.1

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

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

- 549 2. Construct a vector-valued function  $\Phi_1 : \mathbb{R}^d \rightarrow \mathbb{R}^d$  projecting the whole  
550 cube  $Q_\beta$  to the index  $\beta$ , i.e.,  $\Phi_1(\mathbf{x}) = \beta$  for all  $\mathbf{x} \in Q_\beta$  and each  
551  $\beta \in \{0, 1, \dots, J-1\}^d$ .
- 552 3. Construct a linear function  $\phi_2 : \mathbb{R}^d \rightarrow \mathbb{R}$  bijectively mapping  $\beta \in$   
553  $\{0, 1, \dots, J-1\}^d$  to  $\phi_2(\beta) \in \{1, 2, \dots, J^d\}$ .
- 554 4. Construct a function  $\phi_3 : \mathbb{R} \rightarrow \mathbb{R}$  mapping  $\phi_2(\beta) \in \{1, 2, \dots, J^d\}$  approx-  
555 imately to  $\tilde{f}(\mathbf{x}_\beta)$ , i.e.,  $\phi_3(\phi_2(\beta)) \approx \tilde{f}(\mathbf{x}_\beta)$  for each  $\beta \in \{0, 1, \dots, J-1\}^d$ .
- 556 5. Define  $\tilde{\phi} := \phi_3 \circ \phi_2 \circ \Phi_1$ . Then  $\tilde{\phi}$  is a piecewise constant function mapping  
557  $\mathbf{x} \in Q_\beta$  to  $\phi_3(\phi_2(\beta)) \approx \tilde{f}(\mathbf{x}_\beta)$  for each  $\beta \in \{0, 1, \dots, J-1\}^d$ , implying  
558  $\tilde{\phi} \approx \tilde{f}$ . Finally, re-scale and shift  $\tilde{\phi}$  to obtain the final function  $\phi$   
559 approximating  $f$  well.

560 Recall that

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

562 where

563 
$$\mathcal{T}(x) := \mathbb{1}_{x \geq 0} = \begin{cases} 1, & x \geq 0, \\ 0, & x < 0, \end{cases} \quad \text{for any } x \in \mathbb{R}.$$

564 Step 1 and 5 are straightforward. To implement Step 2, we introduce  $\sigma_1$   
565 since it can help to significantly simplify the construction of the vector-valued  
566 projecting function  $\Phi_1$ . The implementation of Step 3 is based on the  $J$ -ary  
567 representation, namely, let  $\phi_2(\mathbf{x}) := 1 + \sum_{i=1}^d J^{i-1} x_i$ . The most technical step  
568 above is Step 4, which is essentially a point fitting problem. Solving such  
569 a problem eventually relies on the bit extraction technique in [Shen et al.](#)  
570 [\(2020\)](#); [Lu et al. \(2020\)](#); [Shen et al. \(2020\)](#); [Harvey et al. \(2017\)](#); [Bartlett](#)  
571 [et al. \(1998\)](#); [Zhang \(2020\)](#); [Yarotsky \(2018\)](#). To extract sufficient many bits  
572 with a limited neuron budget, we introduce two powerful activation functions  
573  $\sigma_2$  and  $\sigma_3$ , as shown in the proposition below.

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

576 
$$\sigma_3(2^k \cdot a) = \theta_k, \quad \text{for any } k \in \{1, 2, \dots, K\},$$

577 where

578 
$$a = \sum_{j=1}^K 2^{-j-1} \cdot \theta_j \in [0, \frac{1}{2}).$$

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

$$580 \quad 0 \leq \sum_{j=1}^K 2^{-j-1} \cdot \theta_j \leq \sum_{j=1}^K 2^{-j-1} < \frac{1}{2},$$

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

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

$$583 \quad 2^k \cdot a = 2^k \cdot \sum_{j=1}^K 2^{-j-1} \cdot \theta_j = \underbrace{\sum_{j=1}^{k-1} 2^{k-j-1} \cdot \theta_j}_{\text{an integer}} + \overbrace{\frac{1}{2}\theta_k}^{0 \text{ or } \frac{1}{2}} + \underbrace{\sum_{j=k+1}^K 2^{k-j-1} \cdot \theta_j}_{\text{in } [0, \frac{1}{2})}. \quad (10)$$

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

$$587 \quad 0 \leq \sum_{j=k+1}^K 2^{k-j-1} \cdot \theta_j \leq \sum_{j=k+1}^K 2^{k-j-1} < \frac{1}{2}$$

588 Therefore, by Equation (10), we have

$$589 \quad 2^k \cdot a \in \bigcup_{n \in \mathbb{N}} [n, n + \frac{1}{2}), \text{ if } \theta_k = 0 \quad \text{and} \quad 2^k \cdot a \in \bigcup_{n \in \mathbb{N}} [n + \frac{1}{2}, n + 1), \text{ if } \theta_k = 1. \quad (11)$$

590 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  
 591 that

$$592 \quad \sigma_3(x) = 0 \text{ if } x \in \bigcup_{n \in \mathbb{N}} [n, n + \frac{1}{2}) \quad \text{and} \quad \sigma_3(x) = 1 \text{ if } x \in \bigcup_{n \in \mathbb{N}} [n + \frac{1}{2}, n + 1).$$

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

$$594 \quad 2^k \cdot a \in \bigcup_{n \in \mathbb{N}} [n, n + \frac{1}{2}) \implies \sigma_3(2^k \cdot a) = 0 = \theta_k.$$

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

$$596 \quad 2^k \cdot a \in \bigcup_{n \in \mathbb{N}} [n + \frac{1}{2}, n + 1) \implies \sigma_3(2^k \cdot a) = 1 = \theta_k.$$

597 Since  $k \in \{1, 2, \dots, K\}$  is arbitrary, we have  $\sigma_3(2^k \cdot a) = \theta_k$  for any  $k \in$   
 598  $\{1, 2, \dots, K\}$ . So we finish the proof.  $\square$

---

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

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

$$601 \quad \{f : f(x) = \sigma_3(a \cdot x), \text{ for } a \in \mathbb{R}\}$$

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

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

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

608 **Step 1:** Set up.

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

$$612 \quad \tilde{f} := \frac{f - f(\mathbf{0}) + \omega_f(\sqrt{d})}{2\omega_f(\sqrt{d})}. \quad (12)$$

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

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

$$616 \quad Q_\beta := \left\{ \mathbf{x} = (x_1, x_2, \dots, x_d) : x_i \in \left[\frac{\beta_i}{J}, \frac{\beta_i+1}{J}\right) \text{ for } i = 1, 2, \dots, d \right\},$$

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

618 **Step 2:** Construct  $\Phi_1$  mapping  $\mathbf{x} \in Q_\beta$  to  $\beta$  for each  $\beta \in \{0, 1, \dots, J-1\}^d$ .

619 Define

$$620 \quad \Phi_1(\mathbf{x}) := \left( \sigma_1(Jx_1), \sigma_1(Jx_2), \dots, \sigma_1(Jx_d) \right) = \left( \lfloor Jx_1 \rfloor, \lfloor Jx_2 \rfloor, \dots, \lfloor Jx_d \rfloor \right),$$

621 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

$$623 \quad \Phi_1(\mathbf{x}) = \left( \lfloor Jx_1 \rfloor, \lfloor Jx_2 \rfloor, \dots, \lfloor Jx_d \rfloor \right) = (\beta_1, \beta_2, \dots, \beta_d) = \beta. \quad (13)$$

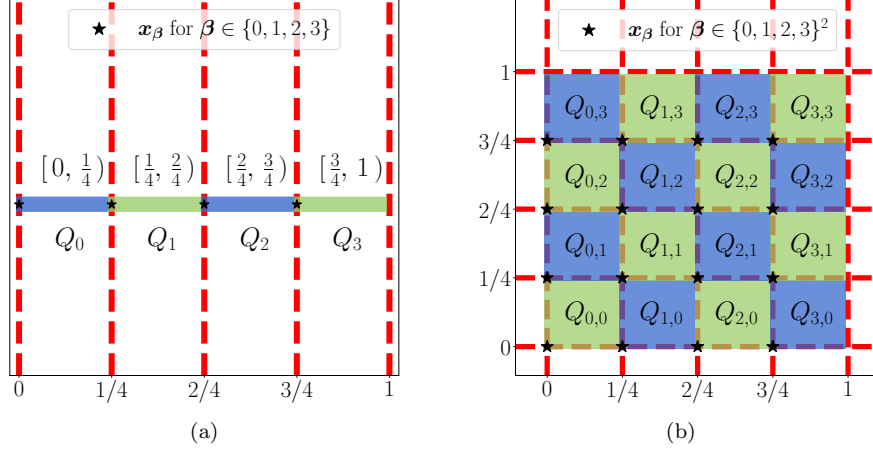


Figure 3: Illustrations of  $Q_\beta$  and  $\mathbf{x}_\beta$  for any  $\beta \in \{0, 1, \dots, J-1\}^d$ . (a)  $J = 4$ ,  $d = 1$ . (b)  $J = 4$ ,  $d = 2$ .

624 **Step 3:** Construct  $\phi_2$  bijectively mapping  $\beta \in \{0, 1, \dots, J-1\}^d$  to  $\phi_2(\beta) \in$   
625  $\{1, 2, \dots, J^d\}$ .

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

$$627 \quad \phi_2(\mathbf{x}) := 1 + \sum_{i=1}^d J^{i-1} x_i, \quad \text{for each } \mathbf{x} = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d.$$

628 Then  $\phi_2$  is a bijection from  $\{0, 1, \dots, J-1\}^d$  to  $\{1, 2, \dots, J^d\}$ .

629 **Step 4:** Construct  $\phi_3$  mapping  $\phi_2(\beta) \in \{1, 2, \dots, J^d\}$  approximately to  $\tilde{f}(\mathbf{x}_\beta)$ .  
630

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

$$633 \quad \xi_k := \tilde{f}(\mathbf{x}_\beta) \in [0, 1], \quad \text{for any } k \in \{1, 2, \dots, J^d\} \text{ with } k = \phi_2(\beta). \quad (14)$$

634 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

$$635 \quad |\xi_k - \text{bin} 0.\theta_{k,1}\theta_{k,2}\dots\theta_{k,N}| \leq 2^{-N}. \quad (15)$$

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

$$638 \quad \sigma_3(2^k \cdot a_j) = \theta_{k,j}, \quad \text{for any } k \in \{1, 2, \dots, J^d\}.$$



639 Define

$$640 \quad \phi_3(x) := \sum_{j=1}^N 2^{-j} \sigma_3(a_j \cdot \sigma_2(x)) = \sum_{j=1}^N 2^{-j} \sigma_3(2^x \cdot a_j), \quad \text{for any } x \in \mathbb{R}.$$

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

$$642 \quad \phi_3(k) = \sum_{j=1}^N 2^{-j} \sigma_3(2^k \cdot a_j) = \sum_{j=1}^N 2^{-j} \cdot \theta_{k,j} = \text{bin} 0.\theta_{k,1}\theta_{k,2}\dots\theta_{k,N}. \quad (16)$$

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

645 Define  $\tilde{\phi} := \phi_3 \circ \phi_2 \circ \Phi_1$ , by Equation (13), (14), (15), and (16), we have,  
646 for any  $\mathbf{x} \in Q_\beta$  and each  $\beta \in \{0, 1, \dots, J-1\}^d$  with  $k = \phi_2(\beta)$ ,

$$\begin{aligned} |\tilde{\phi}(\mathbf{x}) - \tilde{f}(\mathbf{x})| &= |\phi_3 \circ \phi_2 \circ \Phi_1(\mathbf{x}) - \tilde{f}(\mathbf{x}_\beta)| + |\tilde{f}(\mathbf{x}_\beta) - \tilde{f}(\mathbf{x})| \\ &\leq |\phi_3 \circ \phi_2(\beta) - \tilde{f}(\mathbf{x}_\beta)| + \omega_{\tilde{f}}\left(\frac{\sqrt{d}}{J}\right) \\ 647 \quad &\leq |\phi_3(k) - \xi_k| + \omega_{\tilde{f}}\left(\frac{\sqrt{d}}{J}\right) \\ &\leq |\text{bin} 0.\theta_{k,1}\theta_{k,2}\dots\theta_{k,N} - \xi_k| + \omega_{\tilde{f}}\left(\frac{\sqrt{d}}{J}\right) \leq 2^{-N} + \omega_{\tilde{f}}\left(\frac{\sqrt{d}}{J}\right). \end{aligned}$$

648 Finally, define  $\phi := 2\omega_f(\sqrt{d})\tilde{\phi} + f(\mathbf{0}) - \omega_f(\sqrt{d})$ . Equation (12) implies  
649  $\omega_f(r) = 2\omega_f(\sqrt{d})\omega_{\tilde{f}}(r)$  for any  $r \geq 0$ , deducing

$$\begin{aligned} |\phi(\mathbf{x}) - f(\mathbf{x})| &= 2\omega_f(\sqrt{d})|\tilde{\phi}(\mathbf{x}) - \tilde{f}(\mathbf{x})| \leq 2\omega_f(\sqrt{d})(2^{-N} + \omega_{\tilde{f}}\left(\frac{\sqrt{d}}{J}\right)) \\ 650 \quad &= 2\omega_f(\sqrt{d})2^{-N} + \omega_f\left(\frac{\sqrt{d}}{J}\right) \\ &= 2\omega_f(\sqrt{d})2^{-N} + \omega_f(\sqrt{d}2^{-N}), \end{aligned}$$

651 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-  
652 tions of  $\Phi_1$ ,  $\phi_2$ , and  $\phi_3$  that

$$\begin{aligned} \phi(\mathbf{x}) &= 2\omega_f(\sqrt{d})\phi_3 \circ \phi_2 \circ \Phi_1(\mathbf{x}) + f(\mathbf{0}) - \omega_f(\sqrt{d}) \\ &= 2\omega_f(\sqrt{d})\phi_3\left(1 + \sum_{i=1}^d J^{i-1}\sigma_1(Jx_i)\right) + f(\mathbf{0}) - \omega_f(\sqrt{d}) \\ 653 \quad &= 2\omega_f(\sqrt{d})\sum_{j=1}^N 2^{-j}\sigma_3\left(a_j \cdot \sigma_2\left(1 + \sum_{i=1}^d 2^{(i-1)N}\sigma_1(2^N x_i)\right)\right) + f(\mathbf{0}) - \omega_f(\sqrt{d}). \end{aligned}$$

654 So we finish the proof. □

#### 655 4. Approximation with continuous activation functions

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

$$662 \quad \varrho_{1,\delta}(x) := \begin{cases} n-1, & x \in [n-1, n-\delta] \text{ for any } n \in \mathbb{Z}, \\ (x-n+\delta)/\delta, & x \in (n-\delta, n] \text{ for any } n \in \mathbb{Z}. \end{cases}$$

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

$$665 \quad \varrho_2(x) := 3^x, \quad \text{and} \quad \varrho_3(x) := \tilde{\mathcal{T}}(\cos(2\pi x)), \quad \text{for any } x \in \mathbb{R},$$

666 where

$$667 \quad \tilde{\mathcal{T}}(x) := \begin{cases} 0, & x \in (\cos(\frac{4\pi}{9}), \infty), \\ 1 - x/\cos(\frac{4\pi}{9}), & x \in [0, \cos(\frac{4\pi}{9})], \\ 1, & x \in (-\infty, 0), \end{cases} \quad \text{for any } x \in \mathbb{R},$$

668 is a continuous piecewise linear function.  $\varrho_2$  plays the same role of  $\sigma_2(x) = 2^x$   
 669 and  $\varrho_3$  is essentially a “continuous version” of  $\sigma_3$  in FLES.

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

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

$$675 \quad \|\phi - f\|_{L^p([0,1]^d)}^p \leq \left(2\omega_f(\sqrt{d})2^{-N} + \omega_f(\sqrt{d}2^{-N})\right)^p + 2d\delta(|f(\mathbf{0})| + \omega_f(\sqrt{d}))^p,$$

676 where  $\phi$  is defined by a formula in  $a_1, a_2, \dots, a_N$  as follows

$$677 \quad \phi(\mathbf{x}) = 2\omega_f(\sqrt{d}) \sum_{j=1}^N 2^{-j} \varrho_3\left(a_j \cdot \varrho_2\left(1 + \sum_{i=1}^d 2^{(i-1)N} \varrho_{1,\delta}(2^N x_i)\right)\right) + f(\mathbf{0}) - \omega_f(\sqrt{d}).$$

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

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

To prove Theorem 4.1, we need the following proposition.

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

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

where

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

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

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

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

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

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

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

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

701 and

$$702 \quad \cos(2\pi x) \in [-1, \cos(\frac{2\pi}{3})] \subseteq [-1, 0], \quad \text{for any } x \in \bigcup_{n \in \mathbb{N}} [n + \frac{1}{3}, n + \frac{5}{9}).$$

703 Recall that

$$704 \quad \tilde{\mathcal{T}}(x) := \begin{cases} 0, & x \in (\cos(\frac{4\pi}{9}), \infty), \\ 1 - x/\cos(\frac{4\pi}{9}), & x \in [0, \cos(\frac{4\pi}{9})], \\ 1, & x \in (-\infty, 0), \end{cases} \quad \text{for any } x \in \mathbb{R}.$$

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

$$706 \quad 3^k \cdot a \in \bigcup_{n \in \mathbb{N}} [n, n + \frac{2}{9}) \implies \varrho_3(3^k \cdot a) = \tilde{\mathcal{T}}(\cos(2\pi \cdot 3^k \cdot a)) = 0 = \theta_k.$$

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

$$708 \quad 3^k \cdot a \in \bigcup_{n \in \mathbb{N}} [n + \frac{1}{3}, n + \frac{5}{9}) \implies \varrho_3(3^k \cdot a) = \tilde{\mathcal{T}}(\cos(2\pi \cdot 3^k \cdot a)) = 1 = \theta_k.$$

709 Since  $k \in \{1, 2, \dots, K\}$  is arbitrary, we have  $\varrho_3(3^k \cdot a) = \theta_k$  for any  $k \in$   
710  $\{1, 2, \dots, K\}$ . So we finish the proof.  $\square$

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

$$714 \quad \Lambda([0, 1]^d, J, \delta) := \bigcup_{i=1}^d \left\{ \mathbf{x} = (x_1, \dots, x_d) \in [0, 1]^d : x_i \in \bigcup_{j=1}^{J-1} [\frac{j-\delta}{J}, \frac{j}{J}] \right\}. \quad (18)$$

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

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

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

719 **Step 1:** Set up.

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

$$723 \quad \tilde{f} := \frac{f - f(\mathbf{0}) + \omega_f(\sqrt{d})}{2\omega_f(\sqrt{d})}. \quad (19)$$

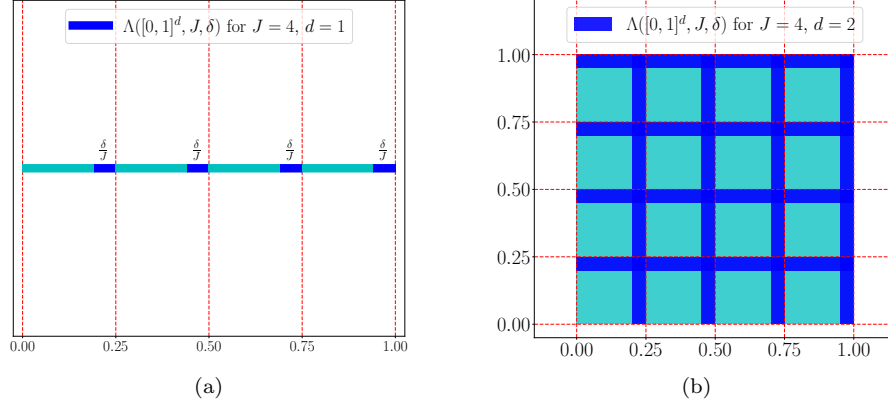


Figure 4: Illustrations of  $\Lambda([0, 1]^d, J, \delta)$ . (a)  $J = 4$ ,  $d = 1$ . (b)  $J = 4$ ,  $d = 2$ .

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

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

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

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

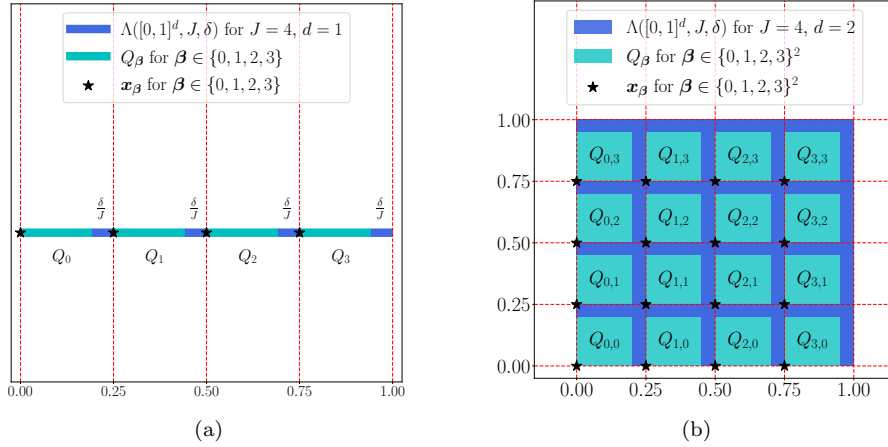


Figure 5: Illustrations of  $\Lambda([0, 1]^d, J, \delta)$ ,  $Q_\beta$ , and  $\mathbf{x}_\beta$  for any  $\beta \in \{0, 1, \dots, J-1\}^d$ . (a)  $J = 4$ ,  $d = 1$ . (b)  $J = 4$ ,  $d = 2$ .

**Step 2:** Construct  $\Phi_1$  mapping  $\mathbf{x} \in Q_\beta$  to  $\beta$  for each  $\beta \in \{0, 1, \dots, J-1\}^d$ .

730 Define

$$731 \quad \Phi_1(\mathbf{x}) := \left( \varrho_{1,\delta}(Jx_1), \varrho_{1,\delta}(Jx_2), \dots, \varrho_{1,\delta}(Jx_d) \right),$$

732 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

$$734 \quad \Phi_1(\mathbf{x}) = \left( \varrho_{1,\delta}(Jx_1), \varrho_{1,\delta}(Jx_2), \dots, \varrho_{1,\delta}(Jx_d) \right) = (\beta_1, \beta_2, \dots, \beta_d) = \beta. \quad (20)$$

735 **Step 3:** Construct  $\phi_2$  bijectively mapping  $\beta \in \{0, 1, \dots, J-1\}^d$  to  $\phi_2(\beta) \in$   
736  $\{1, 2, \dots, J^d\}$ .

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

$$738 \quad \phi_2(\mathbf{x}) := 1 + \sum_{i=1}^d J^{i-1} x_i, \quad \text{for each } \mathbf{x} = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d.$$

739 Then  $\phi_2$  is a bijection from  $\{0, 1, \dots, J-1\}^d$  to  $\{1, 2, \dots, J^d\}$ .

740 **Step 4:** Construct  $\phi_3$  mapping  $\phi_2(\beta) \in \{1, 2, \dots, J^d\}$  approximately to  $\tilde{f}(\mathbf{x}_\beta)$ .

741

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

$$744 \quad \xi_k := \tilde{f}(\mathbf{x}_\beta) \in [0, 1], \quad \text{for any } k \in \{1, 2, \dots, J^d\} \text{ with } k = \phi_2(\beta). \quad (21)$$

745 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

$$746 \quad |\xi_k - \text{bin } 0.\theta_{k,1}\theta_{k,2}\dots\theta_{k,N}| \leq 2^{-N}. \quad (22)$$

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

$$749 \quad \varrho_3(3^k \cdot a_j) = \theta_{k,j}, \quad \text{for any } k \in \{1, 2, \dots, J^d\}.$$

750 Define

$$751 \quad \phi_3(x) := \sum_{j=1}^N 2^{-j} \varrho_3(a_j \cdot \varrho_2(x)) = \sum_{j=1}^N 2^{-j} \varrho_3(3^x \cdot a_j), \quad \text{for any } x \in \mathbb{R}.$$

752 We get

$$753 \quad \varrho_3(x) \in [0, 1], \quad \text{for any } x \in \mathbb{R} \implies \phi_3(x) \in [0, 1], \quad \text{for any } x \in \mathbb{R}, \quad (23)$$

754 and

$$755 \quad \phi_3(k) = \sum_{j=1}^N 2^{-j} \varrho_3(3^k \cdot a_j) = \sum_{j=1}^N 2^{-j} \cdot \theta_{k,j} = \text{bin}0.\theta_{k,1}\theta_{k,2}\cdots\theta_{k,N}, \quad (24)$$

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

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

759 Define  $\tilde{\phi} := \phi_3 \circ \phi_2 \circ \Phi_1$ , by Equation (20), (21), (22), and (24), we have,  
760 for any  $\mathbf{x} \in Q_{\beta}$  and each  $\beta \in \{0, 1, \dots, J-1\}^d$  with  $k = \phi_2(\beta)$ ,

$$\begin{aligned} |\tilde{\phi}(\mathbf{x}) - \tilde{f}(\mathbf{x})| &= |\phi_3 \circ \phi_2 \circ \Phi_1(\mathbf{x}) - \tilde{f}(\mathbf{x}_{\beta})| + |\tilde{f}(\mathbf{x}_{\beta}) - \tilde{f}(\mathbf{x})| \\ &\leq |\phi_3 \circ \phi_2(\beta) - \tilde{f}(\mathbf{x}_{\beta})| + \omega_{\tilde{f}}(\frac{\sqrt{d}}{J}) \\ 761 \quad &\leq |\phi_3(k) - \xi_k| + \omega_{\tilde{f}}(\frac{\sqrt{d}}{J}) \\ &\leq |\text{bin}0.\theta_{k,1}\theta_{k,2}\cdots\theta_{k,N} - \xi_k| + \omega_{\tilde{f}}(\frac{\sqrt{d}}{J}) \leq 2^{-N} + \omega_{\tilde{f}}(\frac{\sqrt{d}}{J}). \end{aligned}$$

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

$$\begin{aligned} |\phi(\mathbf{x}) - f(\mathbf{x})| &= 2\omega_f(\sqrt{d})|\tilde{\phi}(\mathbf{x}) - \tilde{f}(\mathbf{x})| \leq 2\omega_f(\sqrt{d})(2^{-N} + \omega_{\tilde{f}}(\frac{\sqrt{d}}{J})) \\ 764 \quad &= 2\omega_f(\sqrt{d})2^{-N} + \omega_f(\frac{\sqrt{d}}{J}), \end{aligned}$$

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

$$766 \quad \phi := 2\omega_f(\sqrt{d})\phi_3 \circ \phi_2 \circ \Phi_1 + f(\mathbf{0}) - \omega_f(\sqrt{d}),$$

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

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

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

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

771 By the definitions of  $\Phi_1$ ,  $\phi_2$ , and  $\phi_3$ , we have

$$\begin{aligned}
\phi(\mathbf{x}) &= 2\omega_f(\sqrt{d})\phi_3 \circ \phi_2 \circ \Phi_1(\mathbf{x}) + f(\mathbf{0}) - \omega_f(\sqrt{d}) \\
&= 2\omega_f(\sqrt{d})\phi_3 \left( 1 + \sum_{i=1}^d J^{i-1} \varrho_{1,\delta}(Jx_i) \right) + f(\mathbf{0}) - \omega_f(\sqrt{d}) \\
772 &= 2\omega_f(\sqrt{d}) \sum_{j=1}^N 2^{-j} \varrho_3 \left( a_j \cdot \varrho_2 \left( 1 + \sum_{i=1}^d 2^{(i-1)N} \varrho_{1,\delta}(2^N x_i) \right) \right) + f(\mathbf{0}) - \omega_f(\sqrt{d}).
\end{aligned}$$

773 So we finish the proof.  $\square$

## 774 5. Conclusion

775 This paper has introduced a theoretical framework to show that three  
776 hidden layers are enough for neural network approximation to achieve expo-  
777 nential convergence and avoid the curse of dimensionality for approxim-  
778 ating functions as general as (Hölder) continuous functions. The key idea  
779 is to leverage the power of multiple simple activation functions: the Floor  
780 function ( $\lfloor x \rfloor$ ), the exponential function ( $2^x$ ), the step function ( $\mathbb{1}_{x \geq 0}$ ), or  
781 their compositions. This new class of networks is called the FLES network.  
782 Given a Lipschitz continuous function  $f$  on  $[0, 1]^d$ , it was shown by construc-  
783 tion that FLES networks with width  $\max\{d, N\}$  and three hidden layers  
784 admit a uniform approximation rate  $6\lambda\sqrt{d}2^{-N}$ , where  $\lambda$  is the Lipschitz con-  
785 stant of  $f$ . More generally for an arbitrary continuous function  $f$  on  $[0, 1]^d$   
786 with a modulus of continuity  $\omega_f(\cdot)$ , the constructive approximation rate is  
787  $2\omega_f(2\sqrt{d})2^{-N} + \omega_f(2\sqrt{d}2^{-N})$ . We also extend such a result to general con-  
788 tinuous functions on a bounded set  $E \subseteq \mathbb{R}^d$ . The results in this paper provide  
789 a theoretical lower bound of the power of FLES networks. Whether or not



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

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

## References

- Arnold, V.I., 1957. On functions of three variables. Dokl. Akad. Nauk SSSR, 679–681.
- Barron, A.R., 1993. Universal approximation bounds for superpositions of a sigmoidal function. IEEE Transactions on Information Theory 39, 930–945. doi:[10.1109/18.256500](https://doi.org/10.1109/18.256500).
- Barron, A.R., Klusowski, J.M., 2018. Approximation and estimation for high-dimensional deep learning networks. [arXiv:1809.03090](https://arxiv.org/abs/1809.03090).
- Bartlett, P., Maierov, V., Meir, R., 1998. Almost linear VC-dimension bounds for piecewise polynomial networks. Neural Computation 10, 217–3.
- Bengio, Y., Léonard, N., Courville, A., 2013. Estimating or propagating gradients through stochastic neurons for conditional computation [arXiv:1308.3432](https://arxiv.org/abs/1308.3432).
- Boo, Y., Shin, S., Sung, W., 2020. Quantized neural networks: Characterization and holistic optimization [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.
- Carrillo, J.A.T., Jin, S., Li, L., Zhu, Y., 2019. A consensus-based global optimization method for high dimensional machine learning problems. [arXiv:1909.09249](https://arxiv.org/abs/1909.09249).
- Chen, L., Wu, C., 2019. A note on the expressive power of deep rectified linear unit networks in high-dimensional spaces. Mathematical Methods in the Applied Sciences 42, 3400–3404. doi:[10.1002/mma.5575](https://doi.org/10.1002/mma.5575).

- 820 Chen, M., Jiang, H., Liao, W., Zhao, T., 2019a. Efficient approximation  
821 of deep ReLU networks for functions on low dimensional manifolds, in:  
822 Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.,  
823 Garnett, R. (Eds.), Advances in Neural Information Processing Systems  
824 32. Curran Associates, Inc., pp. 8174–8184.
- 825 Chen, Z., Cao, Y., Zou, D., Gu, Q., 2019b. How much over-parameterization  
826 is sufficient to learn deep ReLU networks? CoRR arXiv:1911.12360. URL:  
827 <https://arxiv.org/abs/1911.12360>.
- 828 Du, S.S., Zhai, X., Póczos, B., Singh, A., 2019. Gradient descent prov-  
829 ably optimizes over-parameterized neural networks, in: International Con-  
830 ference on Learning Representations. URL: [https://openreview.net/](https://openreview.net/forum?id=S1eK3i09YQ)  
831 [forum?id=S1eK3i09YQ](https://openreview.net/forum?id=S1eK3i09YQ).
- 832 E, W., Ma, C., Wu, L., 2019. A priori estimates of the population risk for  
833 two-layer neural networks. Communications in Mathematical Sciences 17,  
834 1407 – 1425.
- 835 E, W., Wang, Q., 2018. Exponential convergence of the deep neural network  
836 approximation for analytic functions. CoRR abs/1807.00297. URL: [http:](http://arxiv.org/abs/1807.00297)  
837 [/arxiv.org/abs/1807.00297](http://arxiv.org/abs/1807.00297), [arXiv:1807.00297](https://arxiv.org/abs/1807.00297).
- 838 Gühring, I., Kutyniok, G., Petersen, P., 2019. Error bounds for approxima-  
839 tions with deep ReLU neural networks in  $W^{s,p}$  norms. [arXiv:1902.07896](https://arxiv.org/abs/1902.07896).
- 840 Guliyev, N.J., Ismailov, V.E., 2018. Approximation capability of two hidden  
841 layer feedforward neural networks with fixed weights. Neurocomputing  
842 316, 262 – 269. doi:[10.1016/j.neucom.2018.07.075](https://doi.org/10.1016/j.neucom.2018.07.075).
- 843 Harvey, N., Liaw, C., Mehrabian, A., 2017. Nearly-tight VC-dimension  
844 bounds for piecewise linear neural networks, in: Kale, S., Shamir, O.  
845 (Eds.), Proceedings of the 2017 Conference on Learning Theory, PMLR,  
846 Amsterdam, Netherlands. pp. 1064–1068. URL: [http://proceedings.](http://proceedings.mlr.press/v65/harvey17a.html)  
847 [mlr.press/v65/harvey17a.html](http://proceedings.mlr.press/v65/harvey17a.html).
- 848 Holland, J.H., 1992. Genetic algorithms. Scientific American 267, 66–73.  
849 URL: <http://www.jstor.org/stable/24939139>.

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

853 Hutzenthaler, M., Jentzen, A., Wurstemberger, v.W., 2020. Overcoming the  
854 curse of dimensionality in the approximative pricing of financial derivatives  
855 with default risks. *Electron. J. Probab.* 25, 73 pp. doi:[10.1214/20-EJP423](https://doi.org/10.1214/20-EJP423).

856 Igel'nik, B., Parikh, N., 2003. Kolmogorov's spline network. *IEEE Transac-*  
857 *tions on Neural Networks* 14, 725–733.

858 Jacot, A., Gabriel, F., Hongler, C., 2018. Neural tangent kernel: Conver-  
859 gence and generalization in neural networks, in: Bengio, S., Wallach, H.,  
860 Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.), *Ad-*  
861 *vances in Neural Information Processing Systems*. Curran Associates, Inc.,  
862 volume 31, pp. 8571–8580. URL: [https://proceedings.neurips.cc/](https://proceedings.neurips.cc/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf)  
863 [paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf](https://proceedings.neurips.cc/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf).

864 Kennedy, J., Eberhart, R., 1995. Particle swarm optimization, in: *Pro-*  
865 *ceedings of ICNN'95 - International Conference on Neural Networks*, pp.  
866 1942–1948 vol.4.

867 Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated  
868 annealing. *Science* 220, 671–680. doi:[10.1126/science.220.4598.671](https://doi.org/10.1126/science.220.4598.671).

869 Kolmogorov, A.N., 1956. On the representation of continuous functions of  
870 several variables by superposition of continuous functions of a smaller num-  
871 ber of variables. *Dokl. Akad. Nauk SSSR* , 179–182.

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

875 Kůrková, V., 1991. Kolmogorov's theorem is relevant. *Neural Computation*  
876 3, 617–622. doi:[10.1162/neco.1991.3.4.617](https://doi.org/10.1162/neco.1991.3.4.617).

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

879 Li, Q., Lin, T., Shen, Z., 2019. Deep learning via dynamical systems: An  
880 approximation perspective. arXiv e-prints [arXiv:1912.10382](https://arxiv.org/abs/1912.10382).

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

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

886 Lu, Y., Ma, C., Lu, Y., Lu, J., Ying, L., 2020. A mean-field analy-  
887 sis of deep resnet and beyond: Towards provable optimization via over-  
888 parameterization from depth. CoRR abs/2003.05508. URL: <https://arxiv.org/abs/2003.05508>, [arXiv:2003.05508](https://arxiv.org/abs/2003.05508).

890 Luo, T., Yang, H., 2020. Two-layer neural networks for par-  
891 tial differential equations: Optimization and generalization theory  
892 [arXiv:abs/2006.15733](https://arxiv.org/abs/2006.15733).

893 Maiorov, V., Pinkus, A., 1999. Lower bounds for approximation by MLP neu-  
894 ral networks. Neurocomputing 25, 81 – 91. doi:[10.1016/S0925-2312\(98\)](https://doi.org/10.1016/S0925-2312(98)00111-8)  
895 [00111-8](https://doi.org/10.1016/S0925-2312(98)00111-8).

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

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

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

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

908 Nelder, J., Mead, R., 1965. A simplex method for function minimization.  
909 Comput. J. 7, 308–313. doi:[10.1093/comjnl/7.4.308](https://doi.org/10.1093/comjnl/7.4.308).

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

912 <https://math.ethz.ch/sam/research/reports.html?id=839>.

- 913 Petersen, P., Voigtlaender, F., 2018. Optimal approximation of piecewise  
 914 smooth functions using deep ReLU neural networks. *Neural Networks*  
 915 108, 296 – 330. doi:[10.1016/j.neunet.2018.08.019](https://doi.org/10.1016/j.neunet.2018.08.019).
- 916 Pinnau, R., Totzeck, C., Tse, O., Martin, S., 2017. A consensus-based  
 917 model for global optimization and its mean-field limit. *Mathematical*  
 918 *Models and Methods in Applied Sciences* 27, 183–204. doi:[10.1142/  
 919 S0218202517400061](https://doi.org/10.1142/S0218202517400061).
- 920 Poggio, T., Mhaskar, H.N., Rosasco, L., Miranda, B., Liao, Q., 2017. Why  
 921 and when can deep—but not shallow—networks avoid the curse of dimen-  
 922 sionality: A review. *International Journal of Automation and Computing*  
 923 14, 503–519.
- 924 Schmidt-Hieber, J., 2020a. The kolmogorov-arnold representation theorem  
 925 revisited. [arXiv:2007.15884](https://arxiv.org/abs/2007.15884).
- 926 Schmidt-Hieber, J., 2020b. Nonparametric regression using deep neural net-  
 927 works with ReLU activation function. *Annals of Statistics* 48, 1875–1897.  
 928 URL: <https://projecteuclid.org/euclid.aos/1597370649>.
- 929 Shen, Z., Yang, H., Zhang, S., 2019. Nonlinear approximation via composi-  
 930 tions. *Neural Networks* 119, 74 – 84. doi:[10.1016/j.neunet.2019.07.011](https://doi.org/10.1016/j.neunet.2019.07.011).
- 931 Shen, Z., Yang, H., Zhang, S., 2020. Deep network approximation charac-  
 932 terized by number of neurons. *Communications in Computational Physics*  
 933 28, 1768–1811. doi:[10.4208/cicp.0A-2020-0149](https://doi.org/10.4208/cicp.0A-2020-0149).
- 934 Shen, Z., Yang, H., Zhang, S., 2020. Deep network with approximation error  
 935 being reciprocal of width to power of square root of depth. *arXiv e-prints*  
 936 [arXiv:2006.12231](https://arxiv.org/abs/2006.12231).
- 937 Wang, P., Hu, Q., Zhang, Y., Zhang, C., Liu, Y., Cheng, J., 2018. Two-step  
 938 quantization for low-bit neural networks, in: 2018 IEEE/CVF Conference  
 939 on Computer Vision and Pattern Recognition, pp. 4376–4384.
- 940 Wu, L., Ma, C., E, W., 2018. How sgd selects the global minima in over-  
 941 parameterized learning: A dynamical stability perspective, in: Bengio, S.,  
 942 Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R.  
 943 (Eds.), *Advances in Neural Information Processing Systems* 31. Curran  
 944 Associates, Inc., pp. 8279–8288.

- 945 Yang, Y., Wang, Y., 2020. Approximation in shift-invariant spaces with deep  
946 ReLU neural networks. arXiv e-prints [arXiv:2005.11949](https://arxiv.org/abs/2005.11949).
- 947 Yarotsky, D., 2017. Error bounds for approximations with deep ReLU net-  
948 works. *Neural Networks* 94, 103 – 114. doi:[10.1016/j.neunet.2017.07.](https://doi.org/10.1016/j.neunet.2017.07.002)  
949 [002](https://doi.org/10.1016/j.neunet.2017.07.002).
- 950 Yarotsky, D., 2018. Optimal approximation of continuous functions by very  
951 deep ReLU networks, in: Bubeck, S., Perchet, V., Rigollet, P. (Eds.),  
952 *Proceedings of the 31st Conference On Learning Theory*, PMLR. pp. 639–  
953 649. URL: <http://proceedings.mlr.press/v75/yarotsky18a.html>.
- 954 Yarotsky, D., Zhevnerchuk, A., 2019. The phase diagram of approximation  
955 rates for deep neural networks. arXiv e-prints [arXiv:1906.09477](https://arxiv.org/abs/1906.09477).
- 956 Yin, P., Lyu, J., Zhang, S., Osher, S., Qi, Y., Xin, J., 2019. Understand-  
957 ing straight-through estimator in training activation quantized neural nets  
958 [arXiv:1903.05662](https://arxiv.org/abs/1903.05662).
- 959 Zhang, S., 2020. Deep neural network approximation via function com-  
960 positions. PhD Thesis, National University of Singapore URL: [https:](https://scholarbank.nus.edu.sg/handle/10635/186064)  
961 [//scholarbank.nus.edu.sg/handle/10635/186064](https://scholarbank.nus.edu.sg/handle/10635/186064).