# Neural Network Approximation: Three Hidden Layers Are Enough

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

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

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

**Keywords:** Exponential Convergence, Curse of Dimensionality, Deep Neural Network, Floor-Exponential-Step Activation Function, Continuous Function.

## 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 activation functions 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 a width $\max\{d, N\}$ and three hidden layers can uniformly approximate a Hölder function $f$ on $[0,1]^d$ with an exponential approximation rate $3\lambda d^{\alpha/2} 2^{-\alpha N}$, where $\alpha \in (0,1]$ and $\lambda$ 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 $\omega_f(\sqrt{d}\, 2^{-N}) + 2\omega_f(\sqrt{d}) 2^{-N}$. As a consequence, this new class of networks overcomes the curse of dimensionality in approximation power when the variation of $\omega_f(r)$ as $r \to 0$ is moderate (e.g., $\omega_f(r) \lesssim r^\alpha$ for Hölder continuous functions), since the major term to be concerned in our approximation rate is essentially $\sqrt{d}$ times a function of $N$ independent of $d$ within the modulus of continuity.

# 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 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 $\mathcal{O}(\|f\|_{C^s}N^{-2s/d}L^{-2s/d})$ [34, 21], respectively. For results in terms of the number of nonzero parameters, the reader is referred to [38, 31, 30, 39, 12, 40] 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 samller. To name a few, [3, 10, 28, 7, 15] and reference therein for ReLU networks. The limitation of ReLU networks motivated the work in [35] 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 [35] 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 [35] (and probably also of [40]) 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 [17, 9, 25, 36, 8, 22, 23], where larger width instead of depth can ease the challenge of highly noncovex optimization. The mystery of the power of width and depth remains and it motivates us to demonstrate that width can also enable super approximation power when armed with appropriate activation functions. In particular, we explore the Floor function, the exponential function ($2^x$), the step function ($\mathbb{1}_{x\geq 0}$), or their compositions as activation functions to build fully-connected feed-forward neural networks (FNNs). 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 $\max\{N,d\}$ and three hidden layers can uniformly approximate a continuous function $f$ on $[0,1]^d$ with an exponential approximation rate $2\omega_f(\sqrt{d})2^{-N} + \omega_f(\sqrt{d}\,2^{-N})$, where $\omega_f(\cdot)$ is the modulus of continuity defined as

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

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

$$\mathcal{T}(x) := \mathbb{1}_{x\ge 0} = \left\{\begin{smallmatrix} 1, & x\ge 0, \\ 0, & x<0, \end{smallmatrix}\right. \qquad \text{for any } x \in \mathbb{R}.$$

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

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

*for any $\boldsymbol{x} = (x_1, \cdots, x_d) \in [0,1)^d$, where*

$$\phi(\boldsymbol{x}) = 2\omega_f(\sqrt{d}) \sum_{j=1}^{N} 2^{-j} \sigma_3\Bigg( a_j \sigma_2\Big( 1 + \sum_{i=1}^{d} 2^{(i-1)N} \sigma_1(2^N x_i) \Big) \Bigg) + f(\boldsymbol{0}) - \omega_f(\sqrt{d}) \tag{1.1}$$

*can be implemented by an FNN with activation functions $\sigma_1$, $\sigma_2$, and $\sigma_3$, width $\max\{N, d\}$, three hidden layers, and $2(d+N+1)$ nonzero parameters.*

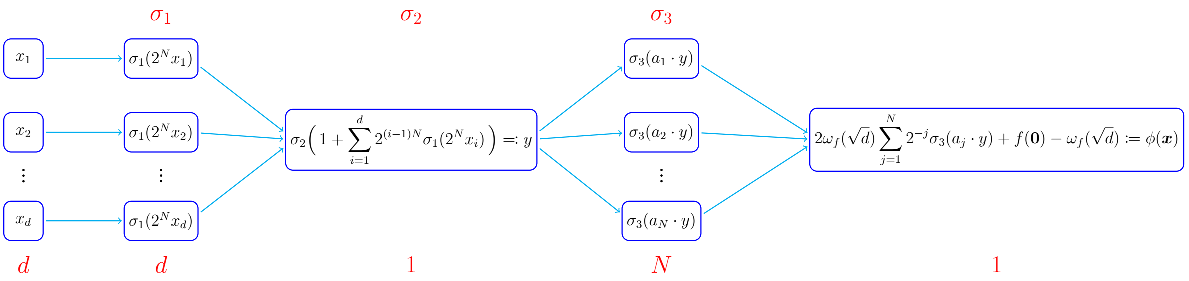


Figure 1: An illustration of the desired three-hidden-layer network in Theorem 1.1 for any $\boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}$. Each of the red functions "$\sigma_1$", "$\sigma_2$", and "$\sigma_3$" above the network is the activation function of the corresponding hidden layer.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

In Theorem 1.1, $[0,1)^d$ and $[0,1]^d$ can be unified and generalized to $[0,M]^d$ for any $M > 0$ at the price of more complicated constants in the approximation rate as we shall see in a corollary later. The rate in $\omega_f(\sqrt{d}\,2^{-N})$ implicitly depends on $N$ through the modulus of continuity of $f$, while the rate in $2\omega_f(\sqrt{d})2^{-N}$ is explicit in $N$. Simplifying the implicit approximation rate to make it explicitly depending on $N$ is challenging in general. However, if $f$ is a Hölder continuous function on $[0,1]^d$ of order $\alpha \in (0,1]$ with a constant $\lambda$, i.e., $f(\boldsymbol{x})$ satisfying

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \le \lambda \|\boldsymbol{x} - \boldsymbol{y}\|_2^\alpha, \quad \text{for any } \boldsymbol{x}, \boldsymbol{y} \in [0,1]^d, \tag{1.2}$$

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

**Corollary 1.2.** *Given any $N \in \mathbb{N}^+$ and a Hölder continuous function $f$ on $[0,1]^d$ of order $\alpha$ with a constant $\lambda$, there exists a function $\phi$ implemented by a three-hidden-layer FNN with activation functions $\sigma_1$, $\sigma_2$, and $\sigma_3$, width* $\max\{d,N\}$ *and* $2(d+N+1)$ *nonzero parameters such that*

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le 3\lambda d^{\alpha/2}2^{-\alpha N}, \quad \text{for any } \boldsymbol{x} \in [0,1)^d.$$

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

The rest of this paper is organized as follows. In Section 2, we discuss the application scope of our theory and compare related works in the literature. We will prove Theorem 1.1 and its corollaries in Section 3. Finally, we conclude this paper in Section 4.

# 2 Discussion

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

## 2.1 Application scope of our theory in machine learning

In supervised learning, an unknown target function $f(\boldsymbol{x})$ defined on a domain $\Omega$ is learned through its finitely many samples $\{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i=1}^n$. If neural networks are applied in supervised learning, the following optimization problem is solved to identify a neural network $\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{S}})$ with $\boldsymbol{\theta}_{\mathcal{S}}$ as the set of parameters to infer $f(\boldsymbol{x})$ for unseen data samples $\boldsymbol{x}$:

$$\boldsymbol{\theta}_{\mathcal{S}} = \arg\min_{\boldsymbol{\theta}} R_{\mathcal{S}}(\boldsymbol{\theta}) := \arg\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{\{\boldsymbol{x}_i\}_{i=1}^n} \ell\big(\phi(\boldsymbol{x}_i;\boldsymbol{\theta}), f(\boldsymbol{x}_i)\big) \tag{2.1}$$

with a loss function typically taken as $\ell(y,y') = \frac{1}{2}|y-y'|^2$. The inference error is usually measured by $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{S}})$, where

$$R_{\mathcal{D}}(\boldsymbol{\theta}) := \mathrm{E}_{\boldsymbol{x}\sim U(\Omega)}\left[\ell(\phi(\boldsymbol{x};\boldsymbol{\theta}), f(\boldsymbol{x}))\right],$$

where the expectation is taken with an unknown data distribution $U(\Omega)$ over $\Omega$. In the analysis, $U(\Omega)$ is assumed to be known, e.g, a uniform distribution for simplicity.

Note that the best neural network to infer $f(\boldsymbol{x})$ is $\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{D}})$ with $\boldsymbol{\theta}_{\mathcal{D}}$ given by

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

The best possible inference error is $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$. In real applications, $U(\Omega)$ is unknown and only finitely many samples from this distribution are available. Hence, the empirical loss $R_{\mathcal{S}}(\boldsymbol{\theta})$ is minimized hoping to obtain $\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{S}})$, instead of minimizing the population loss $R_{\mathcal{D}}(\boldsymbol{\theta})$ to obtain $\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{D}})$. In practice, a numerical optimization method to solve (2.1) may result in a numerical solution (denoted as $\boldsymbol{\theta}_{\mathcal{N}}$) that may not be a global minimizer $\boldsymbol{\theta}_{\mathcal{S}}$. Therefore, the actually learned neural network to infer $f(\boldsymbol{x})$ is $\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{N}})$ and the corresponding inference error is measured by $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}})$.

By the discussion just above, it is crucial to quantify $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}})$ to see how good the learned neural network $\phi(\boldsymbol{x};\boldsymbol{\theta}_{\mathcal{N}})$ is, since $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}})$ is the expected inference error over all possible data samples. Note that

$$\begin{aligned} R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}}) &= [R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{N}})] + [R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{N}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{S}})] + [R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{S}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{D}})] \\ &\quad + [R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{D}}) - R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})] + R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}}) \\ &\le R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}}) + [R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{N}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{S}})] \\ &\quad + [R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{N}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{N}})] + [R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{D}}) - R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})], \end{aligned} \tag{2.2}$$

where the inequality comes from the fact that $[R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{S}}) - R_{\mathcal{S}}(\boldsymbol{\theta}_{\mathcal{D}})] \le 0$ since $\boldsymbol{\theta}_{\mathcal{S}}$ is a global minimizer of $R_{\mathcal{S}}(\boldsymbol{\theta})$. The constructive approximation established in this paper and in the literature provides an upper bound of $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$ in terms of the network size, e.g., in terms of the network width and depth, or in terms of the number of parameters. The second term of (2.2) is bounded by the optimization error of the numerical algorithm applied to solve the empirical loss minimization problem in (2.1). If the numerical algorithm is able to find a global minimizer, the second term is equal to zero. The theoretical guarantee of the convergence of an optimization algorithm to a global minimizer $\boldsymbol{\theta}_{\mathcal{S}}$ and the characterization of the convergence belong to the optimization analysis of neural networks. The third and fourth term of (2.2) are usually bounded in terms of the sample size $n$ and a certain norm of the corresponding set of parameters $\boldsymbol{\theta}_{\mathcal{N}}$ and $\boldsymbol{\theta}_{\mathcal{D}}$, respectively. The study of the bounds for the third and fourth terms is referred to as the generalization error analysis of neural networks.

Theorem 1.1 and its corollaries provide an upper bound of $R_{\mathcal{D}}(\boldsymbol{\theta}_{\mathcal{D}})$. This bound only depends on the given budget of neurons and layers of FLES networks. Hence, this bound is independent of the empirical loss minimization in (2.1) and the optimization algorithm used to compute the numerical solution of (2.1). In other words, Theorem 1.1 and its corollaries quantify the approximation power of FLES networks with a given size. Designing efficient optimization algorithms and analyzing the generalization bounds for FLESU networks are two other separate future directions.

## 2.2 Further interpretation of our theory

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

First, we can generalize Theorem 1.1 to the function space $C([0,M]^d)$ for any $M > 0$ in the following corollary with the modulus of continuity $\omega_f^{[0,M]^d}$ defined as follows. For an arbitrary set $E \subseteq \mathbb{R}^d$, $\omega_f^E(r)$ is defined via $\omega_f^E(r) \coloneqq \sup\big\{|f(\boldsymbol{x}) - f(\boldsymbol{y})| : \|\boldsymbol{x}-\boldsymbol{y}\|_2 \le r,\ \boldsymbol{x},\boldsymbol{y} \in E\big\}$, for any $r \ge 0$. As defined earlier, $\omega_f(r)$ is short of $\omega_f^{[0,1]^d}(r)$. The proof of this corollary will be presented in Section 3.2.

**Corollary 2.1.** *Given any $N \in \mathbb{N}^+$ and a continuous function $f$ on $[0,M]^d$ for any $M > 0$, there exist $a_1, a_2, \cdots, a_N \in [0, \frac{1}{2})$ such that*

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le 2\omega_f^{[0,M]^d}(2M\sqrt{d})2^{-N} + \omega_f^{[0,M]^d}(2M\sqrt{d}\,2^{-N}),$$

*for any $\boldsymbol{x} = (x_1, \cdots, x_d) \in [0,M]^d$, where*

$$\phi(\boldsymbol{x}) = 2\omega_f^{[0,M]^d}(2M\sqrt{d}) \sum_{j=1}^{N} 2^{-j}\sigma_3\bigg(a_j\sigma_2\Big(1 + \sum_{i=1}^{d} 2^{(i-1)N}\sigma_1\big(\tfrac{2^N x_i}{2M}\big)\Big)\bigg) + f(\boldsymbol{0}) - \omega_f^{[0,M]^d}(2M\sqrt{d})$$

*can be implemented by a three-hidden-layer FNN with activation functions $\sigma_1$, $\sigma_2$, and $\sigma_3$, width $\max\{N,d\}$, and $2(d+N+1)$ nonzero parameters.*

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

Second, most interesting continuous functions in practice have a good modulus of continuity such that there is no implicit curse of dimensionality hidden in $\omega_f(\cdot)$. For example, we have discussed the case of Hölder continuous functions previously. We would like to remark that the class of Hölder continuous functions implicitly depends on $d$ through its definition in (1.2), but this dependence is moderate since the $\ell^2$- norm in (1.2) 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 a small $r > 0$, which leads to an approximation rate

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

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 a small $r > 0$, which leads to an approximation rate

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

The power $\frac{1}{d}$ 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 a small $r > 0$, which results in the approximation rate

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

which achieves the exponential convergence and avoids the curse of dimensionality when we use very deep networks with a fixed width. But if we fix depth, there is no exponential convergence and the curse exists. 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.

## 2.3 Kolmogorov-Arnold Superposition Theorem

A closely related research topic is the Kolmogorov-Arnold representation theorem (KST) [18, 1, 19] and its approximation in a form of modern neural networks. Our FLES networks admit super approximation power with a fixed number of layers for continuous functions and the KST exactly represent continuous functions using two hidden layers and $\mathcal{O}(d)$ neurons. More specifically, given any $f \in C([0,1]^d)$, the KST shows that there exist continuous functions $\phi_q : \mathbb{R} \to \mathbb{R}$ and $\psi_{q,p} : \mathbb{R} \to \mathbb{R}$ such that

$$f(\boldsymbol{x}) = \sum_{q=0}^{2d} \phi_q\left( \sum_{p=1}^{d} \psi_{q,p}(x_p) \right), \quad \text{for any } \boldsymbol{x} = (x_1, \cdots, x_d) \in [0,1]^d. \tag{2.3}$$

Note that the activation functions $\{\phi_q\}$ (also called outer functions) of the neural network in Equation (2.3) 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 [5].

There has been an active research line to develop more practical network approximation based on KST [20, 24, 13, 27, 16, 32] 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 ideas 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 a desired accuracy. Using this idea, the seminal work in [20] 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 [16, 27, 32], 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 [24, 13], where a two-hidden-layer network with $\mathcal{O}(d)$ neurons can approximate $f \in$

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 |
|---|---|---|---|---|---|
| [18, 1, 19] | 2 | $2d+1$ | $f$-dependent | 0 | original KST |
| [24, 13] | 2 | $\mathcal{O}(d)$ | $f$-independent | arbitrary error $\varepsilon$ | based on KST |
| [33] | 3 | $\mathcal{O}(dN)$ | ReLU | $\mathcal{O}(N^{-2/d})$ | 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 |

$C([0,1]^d)$ within an arbitrary error $\varepsilon$. Let us briefly summerize their main ideas to obtain such an exicting 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, devide 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 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 [24, 13], 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 avoid the curse of dimensionality. For a large dimension $d$, the width parameter of our FLES network can be chosen as $N = d$, which leads to a FLES network of size $\mathcal{O}(d)$ with an approximation accuracy $\mathcal{O}(2^{-d})$ for Lipschitz continuous functions. $\mathcal{O}(2^{-d})$ is sufficiently attractive. In practice, when $d$ is very large, $N$ could be much smaller than $d$ and our approximation rate is still attractive.

Finally, we list several KST-related results in Table 1 for a quick comparison.

## 2.4 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 [2], its variants or generalization [3, 10, 6, 28], and related references therein, $d$-dimensional functions of the following form were considered:

$$f(\boldsymbol{x}) = \int_{\widetilde{\Omega}} a(\boldsymbol{w}) K(\boldsymbol{w} \cdot \boldsymbol{x}) d\mu(\boldsymbol{w}), \tag{2.4}$$

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

Linear approximation is an efficient approximation tool for smooth functions that computes the approximant of a target function via a linear projection to a Hilbert space or a Banach space as the approximant space. Typical examples include approximation via orthogonal polynomials, Fourier series expansion, etc. Inspired by the seminal work in [38], where deep ReLU networks were constructed to approximate polynomials with exponential convergence, subsequent works in [11, 29, 26, 6, 28, 40, 21, 27, 37] have constructed deep ReLU networks to approximate various smooth function spaces. The main idea of these works is to approximate smooth functions via (piecewise) polynomial approximation first and then construct deep ReLU networks to approximate the ensemble of polynomials. If the approximation rate of polynomials to approximate the target function is exponential, then the approximation rate of deep ReLU networks to approximate the target function is also exponential. But the curse of dimensionality exists since polynomial approximation cannot avoid the curse.

The bit extraction proposed in [4] has been a very important technique to develop nearly optimal approximation rates of deep ReLU neural networks [39, 34, 21, 37] and the optimality is based on the nearly optimal VC-dimension bound of ReLU networks in [14]. The bit extraction was also applied in [35, 32] 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 its corollaries in Section 3.2.

## 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, a scalar plus a vector means adding the scalar to each entry of the vector.
- Let $\mathbb{N}^+$ denote the set containing all positive integers, i.e., $\mathbb{N}^+ = \{1, 2, 3, \cdots\}$.
- For any $p \in [1, \infty)$, the $p$-norm of a vector $\boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d$ is defined by

$$\|\boldsymbol{x}\|_p \coloneqq \big(|x_1|^p + |x_2|^p + \cdots + |x_d|^p\big)^{1/p}.$$

- Let $\sigma : \mathbb{R} \to \mathbb{R}$ denote the rectified linear unit (ReLU), i.e. $\sigma(x) = \max\{0, x\}$. With a slight abuse of notation, we define $\sigma : \mathbb{R}^d \to \mathbb{R}^d$ as $\sigma(\boldsymbol{x}) = \begin{bmatrix} \max\{0, x_1\} \\ \vdots \\ \max\{0, x_d\} \end{bmatrix}$ for any $\boldsymbol{x} = (x_1, \cdots, x_d) \in \mathbb{R}^d$.
- The floor function (Floor) is defined as $\lfloor x \rfloor \coloneqq \max\{n : n \le x,\ n \in \mathbb{Z}\}$ for any $x \in \mathbb{R}$.
- For $\theta \in [0, 1)$, suppose its binary representation is $\theta = \sum_{\ell=1}^{\infty} \theta_\ell 2^{-\ell}$ with $\theta_\ell \in \{0, 1\}$, we introduce a special notation $\text{bin}\, 0.\theta_1\theta_2\cdots\theta_L$ to denote the $L$-term binary representation of $\theta$, i.e., $\text{bin}\, 0.\theta_1\theta_2\cdots\theta_L \coloneqq \sum_{\ell=1}^{L} \theta_\ell 2^{-\ell}$.
- The expression "a network with a width $N$ and a depth $L$" means
  - The maximum width of this network for all **hidden** layers is no more than $N$.
  - The number of **hidden** layers of this network is no more than $L$.

## 3.2 Proof of Theorem 1.1 and Corollary 2.1

To prove Theorem 1.1, we first present the proof sketch. Shortly speaking, we construct piecewise constant functions to approximate continuous functions. There are five key steps in our construction.

1. Normalize $f$ as $\widetilde{f}$ satisfying $\widetilde{f}(\boldsymbol{x}) \in [0, 1]$ for any $\boldsymbol{x} \in [0, 1]^d$, divide $[0, 1)^d$ into a set of non-overlapping cubes $\{Q_{\boldsymbol{\beta}}\}_{\boldsymbol{\beta} \in \{0,1,\cdots,J-1\}^d}$, and denote $\boldsymbol{x}_{\boldsymbol{\beta}}$ as the vertex of $Q_{\boldsymbol{\beta}}$ with minimum $\|\cdot\|_1$ norm, where $J$ is an integer determined later. See Figure 2 for the illustrations of $Q_{\boldsymbol{\beta}}$ and $\boldsymbol{x}_{\boldsymbol{\beta}}$.
2. Construct a vector-valued function $\boldsymbol{\Phi}_1 : \mathbb{R}^d \to \mathbb{R}^d$ projecting the whole cube $Q_{\boldsymbol{\beta}}$ to the index $\boldsymbol{\beta}$ for each $\boldsymbol{\beta} \in \{0, 1, \cdots, J-1\}^d$, i.e., $\boldsymbol{\Phi}_1(\boldsymbol{x}) = \boldsymbol{\beta}$ for all $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$.
3. Construct a linear function $\phi_2 : \mathbb{R}^d \to \mathbb{R}$ bijectively mapping $\boldsymbol{\beta} \in \{0, 1, \cdots, J-1\}^d$ to $\phi_2(\boldsymbol{\beta}) \in \{1, 2, \cdots, J^d\}$.

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

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

The most technical step above is Step 4, the realization of which relies on the proposition below.

**Proposition 3.1.** *Given any $K \in \mathbb{N}^+$ and arbitrary $\theta_1, \theta_2, \cdots, \theta_K \in \{0, 1\}$, it holds that*

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

*where*

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

*Proof of Proposition 3.1.* Since $\theta_j \in \{0, 1\}$ for $j \in \{1, 2, \cdots, K\}$, we have

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

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

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

$$2^k \cdot a = 2^k \cdot \sum_{j=1}^{K} 2^{-j-1} \cdot \theta_j = \underbrace{\sum_{j=1}^{k-1} 2^{k-j-1} \cdot \theta_j}_{\text{an integer}} + \overbrace{\tfrac{1}{2}\theta_k}^{0 \text{ or } \frac{1}{2}} + \underbrace{\sum_{j=k+1}^{K} 2^{k-j-1} \cdot \theta_j}_{\text{in } [0, \frac{1}{2})}.[1] \tag{3.1}$$

Clearly, the first term in Equation (3.1) $\sum_{j=1}^{k-1} 2^{k-j-1} \cdot \theta_j$ is a non-negative integer since $\theta_j \in \{0, 1\}$ for any $j \in \{1, 2, \cdots, K\}$. As for the third term in Equation (3.1), we have

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

Therefore, By Equation (3.1), we have

$$2^k \cdot a \in \bigcup_{n \in \mathbb{N}} [n, n + \tfrac{1}{2}), \text{ if } \theta_k = 0, \quad \text{and} \quad 2^k \cdot a \in \bigcup_{n \in \mathbb{N}} [n + \tfrac{1}{2}, n + 1), \text{ if } \theta_k = 1. \tag{3.2}$$

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

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

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

If $\theta_k = 0$, by Equation (3.2), we have

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

Similarly, if $\theta_k = 1$, by Equation (3.2), we have

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

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

We would like to point out that Proposition 3.1 indicates that the VC-dimension of the function space

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

is infinity.

With Proposition 3.1 in hand, we are ready to prove Theorem 1.1.

*Proof of Theorem 1.1.* The proof consists of five steps.

**Step** 1: Set up.

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

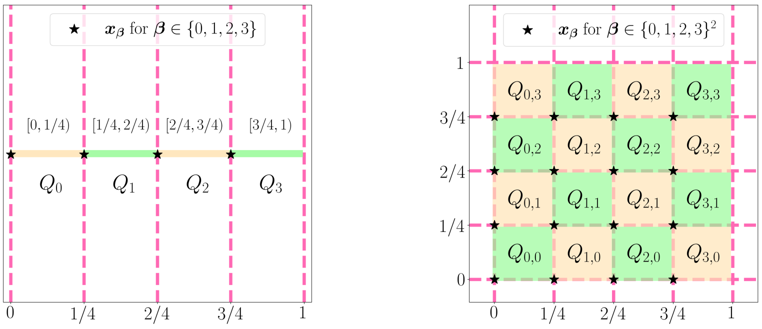
$$\widetilde{f} \coloneqq \big(f - f(\boldsymbol{0}) + \omega_f(\sqrt{d})\big)\big/\big(2\omega_f(\sqrt{d})\big). \tag{3.3}$$

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

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

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

for each $\boldsymbol{\beta} = (\beta_1, \beta_2, \cdots, \beta_d) \in \{0, 1, \cdots, J-1\}^d$. See Figure 2 for illustrations.



(a) (b)

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

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

Define

$$\mathbf{\Phi}_1(\boldsymbol{x}) = \Big( \lfloor Jx_1 \rfloor, \lfloor Jx_2 \rfloor, \cdots, \lfloor Jx_d \rfloor \Big), \quad \text{for any } \boldsymbol{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d.$$

Then, for any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and each $\boldsymbol{\beta} \in \{0, 1, \cdots, J-1\}^d$, we have

$$\mathbf{\Phi}_1(\boldsymbol{x}) = \Big( \lfloor Jx_1 \rfloor, \lfloor Jx_2 \rfloor, \cdots, \lfloor Jx_d \rfloor \Big) = (\beta_1, \beta_2, \cdots, \beta_d) = \boldsymbol{\beta}. \tag{3.4}$$

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

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

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

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

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

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

$$\xi_k \coloneqq \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}}) \in [0, 1], \quad \text{for any } k \in \{1, 2, \cdots, J^d\} \text{ with } k = \phi_2(\boldsymbol{\beta}). \tag{3.5}$$

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

$$|\xi_k - \text{bin}\, 0.\theta_{k,1}\theta_{k,2}\cdots\theta_{k,N}| \le 2^{-N}. \tag{3.6}$$

For each $j \in \{1, 2, \cdots, N\}$, by Proposition 3.1 (set $K = J^d$ therein), there exists $a_j \in [0, \frac{1}{2})$ such that

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

Define

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

We get

$$\phi_3(k) = \sum_{j=1}^{N} 2^{-j} \sigma_3(2^k \cdot a_j) = \sum_{j=1}^{N} 2^{-j} \cdot \theta_{k,j} = \text{bin}\, 0.\theta_{k,1}\theta_{k,2}\cdots\theta_{k,N}. \tag{3.7}$$

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

Define $\widetilde{\phi} \coloneqq \phi_3 \circ \phi_2 \circ \mathbf{\Phi}_1$, by Equation (3.4), (3.5), (3.6), and (3.7), we have, for any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and each $\boldsymbol{\beta} \in \{0, 1, \cdots, J-1\}^d$ with $k = \phi_2(\boldsymbol{\beta})$,

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

Finally, define $\phi \coloneqq 2\omega_f(\sqrt{d})\widetilde{\phi} + f(\mathbf{0}) - \omega_f(\sqrt{d})$. Equation (3.3) implies $\omega_f(r) = 2\omega_f(\sqrt{d})\omega_{\widetilde{f}}(r)$ for any $r \ge 0$, deducing

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

for any $\boldsymbol{x} \in \bigcup_{\boldsymbol{\beta} \in \{0,1,\cdots,J-1\}^d} Q_{\boldsymbol{\beta}} = [0,1)^d$. It follows from $J = 2^N$ and the definitions of $\boldsymbol{\Phi}_1$, $\phi_2$, and $\phi_3$ that

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

So we finish the proof. □

Finally, we present the proof of Corollary 2.1 below.

*Proof of Corollary 2.1.* Given any $f \in C([0,M]^d)$, by Lemma 4.2 of [34] (set $E = [0,M]^d$ and $S = [0,2M)^d$ therein), there exists $g \in C([0,2M)^d)$ such that

- $f(\boldsymbol{x}) = g(\boldsymbol{x})$ for any $\boldsymbol{x} \in [0,M]^d$;
- $\omega_f^{[0,M]^d}(r) = \omega_g^{[0,2M)^d}(r)$ for any $r \ge 0$.

Set $\widetilde{g}(\boldsymbol{x}) = g(2M\boldsymbol{x})$ for any $\boldsymbol{x} \in [0,1)^d$, then $\widetilde{g} \in C([0,1)^d)$. By applying Theorem 1.1 to $\widetilde{g}$, there exist $a_1, a_2, \cdots, a_N \in [0,\frac{1}{2})$ such that

$$|\phi(2M\boldsymbol{x}) - g(2M\boldsymbol{x})| = |\widetilde{\phi}(\boldsymbol{x}) - \widetilde{g}(\boldsymbol{x})| \le 2\omega_{\widetilde{g}}^{[0,1)^d}(\sqrt{d})2^{-N} + \omega_{\widetilde{g}}^{[0,1)^d}(\sqrt{d}\,2^{-N}), \tag{3.8}$$

for any $\boldsymbol{x} \in [0,1)^d$, where $\phi(2M\boldsymbol{x}) = \widetilde{\phi}(\boldsymbol{x})$ and

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

Note that $\omega_{\widetilde{g}}^{[0,1)^d}(r) = \omega_g^{[0,2M)^d}(2Mr) = \omega_f^{[0,M]^d}(2Mr)$ for any $r \ge 0$. Then by Equation (3.8), for any $\boldsymbol{x} \in [0,M]^d \subseteq [0,2M)^d$, we have

$$\begin{aligned} |\phi(\boldsymbol{x}) - f(\boldsymbol{x})| = |\phi(\boldsymbol{x}) - g(\boldsymbol{x})| &= |\widetilde{\phi}(\tfrac{\boldsymbol{x}}{2M} - \widetilde{g}(\tfrac{\boldsymbol{x}}{2M})| \\ &\le 2\omega_{\widetilde{g}}^{[0,1)^d}(\sqrt{d})2^{-N} + \omega_{\widetilde{g}}^{[0,1)^d}(\sqrt{d}\,2^{-N}) \\ &= 2\omega_f^{[0,M]^d}(2M\sqrt{d})2^{-N} + \omega_f^{[0,M]^d}(2M\sqrt{d}\,2^{-N}), \end{aligned}$$

where

$$\phi(\boldsymbol{x}) = 2\omega_f^{[0,M]^d}(2M\sqrt{d})\sum_{j=1}^{N} 2^{-j}\sigma_3\Bigg(a_j\sigma_2\Big(1 + \sum_{i=1}^{d} 2^{(i-1)N}\sigma_1(\tfrac{2^N x_i}{2M})\Big)\Bigg) + f(\mathbf{0}) - \omega_f^{[0,M]^d}(2M\sqrt{d})$$

With the discussion above, we have proved Corollary 2.1. □

# 4 Conclusion

This paper has introduced a theoretical framework to show that three hidden layers are enough for neural network approximation to achieve exponential convergence and avoid the curse of dimensionality for approximating functions as general as (Hölder) continuous functions. The key idea is to leverage the power of multiple simple activation functions: the Floor function ($\lfloor x \rfloor$), the exponential function ($2^x$), the step function ($\mathbb{1}_{x\geq 0}$), or their compositions. This new class of networks is called the FLES network. Given a Lipschitz continuous function $f$ on $[0,1]^d$, it was shown by construction that FLES networks with width $\max\{d, N\}$ and three hidden layers admit a uniform approximation rate $3\lambda\sqrt{d}\,2^{-N}$, where $\lambda$ is the Lipschitz constant of $f$. More generally for an arbitrary continuous function $f$ on $[0,1]^d$ with a modulus of continuity $\omega_f(\cdot)$, the constructive approximation rate is $\omega_f(\sqrt{d}\,2^{-N}) + 2\omega_f(\sqrt{d})2^{-N}$. The results in this paper provide a theoretical lower bound of the power of FLES networks. Whether or not this bound is achievable in actual computation relies on advanced algorithm design as a separate line of research.

**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

[1] V. I. Arnold. On functions of three variables. *Dokl. Akad. Nauk SSSR*, pages 679–681, 1957.

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

[3] A. R. Barron and J. M. Klusowski. Approximation and estimation for high-dimensional deep learning networks, 2018.

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

[5] J. Braun and M. Griebel. On a constructive proof of kolmogorov's superposition theorem. *Constructive Approximation*, 30:653–675, 2009.

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

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

[8] Z. Chen, Y. Cao, D. Zou, and Q. Gu. How much over-parameterization is sufficient to learn deep ReLU networks? *CoRR*, arXiv:1911.12360, 2019.

[9] S. S. Du, X. Zhai, B. Poczos, and A. Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

[10] W. E, C. Ma, and L. Wu. A priori estimates of the population risk for two-layer neural networks. *Communications in Mathematical Sciences*, 17(5):1407 – 1425, 2019.

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

[12] I. Gühring, G. Kutyniok, and P. Petersen. Error bounds for approximations with deep relu neural networks in $w^{s,p}$ norms, 2019.

[13] N. J. Guliyev and V. E. Ismailov. Approximation capability of two hidden layer feedforward neural networks with fixed weights. *Neurocomputing*, 316:262 – 269, 2018.

[14] N. Harvey, C. Liaw, and A. Mehrabian. Nearly-tight VC-dimension bounds for piecewise linear neural networks. In S. Kale and O. Shamir, editors, *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 1064–1068, Amsterdam, Netherlands, 07–10 Jul 2017. PMLR.

[15] M. Hutzenthaler, A. Jentzen, and v. W. Wurstemberger. Overcoming the curse of dimensionality in the approximative pricing of financial derivatives with default risks. *Electron. J. Probab.*, 25:73 pp., 2020.

[16] B. Igelnik and N. Parikh. Kolmogorov's spline network. *IEEE Transactions on Neural Networks*, 14(4):725–733, 2003.

[17] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8571–8580. Curran Associates, Inc., 2018.

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

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

[20] V. Kůrková. Kolmogorov's theorem and multilayer neural networks. *Neural Networks*, 5(3):501 – 506, 1992.

[21] J. Lu, Z. Shen, H. Yang, and S. Zhang. Deep network approximation for smooth functions. *arXiv e-prints*, page arXiv:2001.03040, Jan. 2020.

[22] Y. Lu, C. Ma, Y. Lu, J. Lu, and L. Ying. A mean-field analysis of deep resnet and beyond: Towards provable optimization via overparameterization from depth. *CoRR*, abs/2003.05508, 2020.

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

[24] V. Maiorov and A. Pinkus. Lower bounds for approximation by MLP neural networks. *Neurocomputing*, 25(1):81 – 91, 1999.

[25] S. Mei, A. Montanari, and P.-M. Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

[26] H. Montanelli and Q. Du. New error bounds for deep ReLU networks using sparse grids. *SIAM Journal on Mathematics of Data Science*, 1(1):78–92, 2019.

[27] H. Montanelli and H. Yang. Error bounds for deep ReLU networks using the Kolmogorov-Arnold superposition theorem. *Neural Networks*, 129:1 – 6, 2020.

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

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

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

[31] J. Schmidt-Hieber. Nonparametric regression using deep neural networks with ReLU activation function. 2017.

[32] J. Schmidt-Hieber. The kolmogorov-arnold representation theorem revisited, 2020.

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

[34] Z. Shen, H. Yang, and S. Zhang. Deep network approximation characterized by number of neurons. *Communications in Computational Physics*, 2020.

[35] Z. Shen, H. Yang, and S. Zhang. Deep network with approximation error being reciprocal of width to power of square root of depth. *arXiv:2006.12231*, 2020.

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

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

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

[39] D. Yarotsky. Optimal approximation of continuous functions by very deep ReLU networks. In S. Bubeck, V. Perchet, and P. Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 639–649. PMLR, 06–09 Jul 2018.

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