NONLINEAR APPROXIMATION VIA COMPOSITIONS[*]

ZUOWEI SHEN[†], HAIZHAO YANG[‡], AND SHIJUN ZHANG[§]

Abstract. We study the advantage and optimal implementation of function compositions in nonlinear approximation when compositions are implemented using multi-layer feed-forward neural networks (FNNs) with ReLU activation functions, especially in modern computing architecture, via the N-term approximation rate together with parallel efficiency for the first time. First, for any function f on $[0,1]$, regardless of its smoothness and even the continuity, if f can be approximated via nonlinear approximation using one-hidden-layer ReLU FNNs with an approximation rate $\mathcal{O}(N^{-\eta})$, we quantitatively show that function compositions can improve the approximation rate to $\mathcal{O}(N^{-2\eta})$. Second, for Hölder continuous functions of order α with a uniform Lipchitz constant ν on a d-dimensional cube, we show that the N-term approximation via ReLU FNNs with two or three function compositions can achieve an essentially tight approximation rate $\mathcal{O}(N^{-2\alpha/d})$. Considering the computational efficiency per training iteration in parallel computing, FNNs with $\mathcal{O}(1)$ hidden layers are an optimal choice for approximating Hölder continuous functions if computing resources are enough.

Key words. Deep Neural Networks, ReLU Activation Function, Nonlinear Approximation, Function Composition, Hölder Continuity, Parallel Computing.

1. Introduction.

For non-smooth and high-dimensional function approximation, a favorable technique popularized in recent decades is the nonlinear approximation [16] that does not limit the approximants to come from linear spaces, obtaining sparser representation, cheaper computation, and more robust estimation, and therein emerged the bloom of many breakthroughs in applied mathematics and computer science (e.g., wavelet analysis [13], dictionary learning [55], data compression and denoising [27, 29], adaptive pursuit [14, 45], compressed sensing [17, 6]).

Typically, nonlinear approximation is a two-stage algorithm that designs a good redundant nonlinear dictionary, $\mathcal{D}$, in its first stage, and identifies the optimal approximant as a linear combination of N elements of $\mathcal{D}$ in the second stage:

$$f(\boldsymbol{x}) \approx g \circ T(\boldsymbol{x}) := \sum_{n=1}^{N} g_n T_n(\boldsymbol{x}), \tag{1.1}$$

where $f(\boldsymbol{x})$ is the target function in a Hilbert space $\mathcal{H}$ associated with a norm denoted as $\|\cdot\|_*$, $\{T_n\} \subseteq \mathcal{D} \subseteq \mathcal{H}$, T is a nonlinear map from $\mathbb{R}^d$ to $\mathbb{R}^N$ with the n-th coordinate being T_n, and g is a linear map from $\mathbb{R}^N$ to $\mathbb{R}$ with the n-th coordinate being $g_n \in \mathbb{R}$. The nonlinear approximation seeks g and T such that

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

which is also called the N-term approximation. One remarkable approach of nonlinear approximation is based on one-hidden-layer neural networks that give simple and elegant bases of the form $T(\boldsymbol{x}) = \sigma(\boldsymbol{W}\boldsymbol{x}+\boldsymbol{b})$, where $\boldsymbol{W}\boldsymbol{x}+\boldsymbol{b}$ is a linear transform in $\boldsymbol{x}$ with the transformation matrix $\boldsymbol{W}$ (named as the weight matrix) and a shifting vector $\boldsymbol{b}$ (called bias), and σ is a nonlinear function (called the activation function). The approximation

$$f(\boldsymbol{x}) \approx \sum_{n=1}^{N} g_n T_n(\boldsymbol{x}) = \sum_{n=1}^{N} g_n \sigma(\boldsymbol{W}_n \boldsymbol{x} + \boldsymbol{b}_n)$$

includes wavelets pursuits [41, 7], adaptive splines [16, 47], radial basis functions [15, 21, 58], sigmoidal neural networks [39, 36, 10, 9, 11], etc. For functions in Besov spaces with smoothness s, [15, 21] constructed an

[*]

Funding: H. Yang was supported by the startup of the Department of Mathematics at the National University of Singapore.

[†]Department of Mathematics, National University of Singapore (matzuows@nus.edu.sg).
[‡]Department of Mathematics, National University of Singapore (haizhao@nus.edu.sg).
[§]Department of Mathematics, National University of Singapore (zhangshijun@u.nus.edu).

$\mathcal{O}(N^{-s/d})$[1] approximation that is almost optimal [38] and the smoothness cannot be reduced generally [21]. For Hölder continuous functions of order 1 on $[0,1]^d$, [58] essentially constructed an $\mathcal{O}(N^{-\frac{1}{2d}})$ approximation, which is far from the lower bound $\mathcal{O}(N^{-2/d})$ as we shall prove in this paper. Achieving the optimal approximation rate of general continuous functions in constructive approximation, especially in high dimensional spaces, remains an unsolved challenging problem.

1.1. Problem Statement. We try to tackle the open problem above via function compositions in the nonlinear approximation. i.e.,

$$f(\boldsymbol{x}) \approx g \circ T^{(L)} \circ T^{(L-1)} \circ \cdots \circ T^{(1)}(\boldsymbol{x}), \tag{1.2}$$

where $T^{(i)}$ is a nonlinear map from $\mathbb{R}^{N_{i-1}}$ to $\mathbb{R}^{N_i}$ for $i = 1, \dots, L$ with $N_0 = d$ and $N_L = N$. Function compositions enrich the bases of dictionaries in nonlinear approximation and hence could accelerate the convergence rate of the N-term approximation. In terms of numerical computation, there exist efficient algorithms to implement the nonlinear approximation via compositions, e.g., multi-layer neural networks, which has been popularized nowadays with the invention of effective neural network techniques [56, 20, 49], the development of high-performance computing [52, 8], and the design of new neural architectures [35, 24, 26, 61, 57].

More specifically, given a set of parameters $\boldsymbol{\theta}$ containing all weight matrices and bias vectors, an FNN $\phi(\boldsymbol{x};\boldsymbol{\theta})$ with L hidden layers (the i-th one of which has N_i neurons) to implement the nonlinear approximation via L function compositions in (1.2) can be defined recursively as $\boldsymbol{h}_i := \boldsymbol{W}_i \tilde{\boldsymbol{h}}_{i-1} + \boldsymbol{b}_i$, for $i = 1, \dots, L+1$, and $\tilde{\boldsymbol{h}}_i = \sigma(\boldsymbol{h}_i)$, for $i = 1, \dots, L$, where $\boldsymbol{W}_i \in \mathbb{R}^{N_i \times N_{i-1}}$ and $\boldsymbol{b}_i \in \mathbb{R}^{N_i}$ are the weight matrix and the bias vector in the i-th linear transform in ϕ, respectively, assuming $N_0 = d$, $N_{L+1} = 1$, $\tilde{\boldsymbol{h}}_0 = \boldsymbol{x}$, and $\phi(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{h}_{L+1}$. To identify the optimal FNN to approximate $f(\boldsymbol{x})$, it is sufficient to solve the following optimization problem

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}\big(f(\boldsymbol{x}), \phi(\boldsymbol{x};\boldsymbol{\theta})\big), \tag{1.3}$$

where $\mathcal{L}(\cdot,\cdot)$ is an appropriate loss function quantifying the difference between $f(\boldsymbol{x})$ and $\phi(\boldsymbol{x};\boldsymbol{\theta})$. With advanced optimization algorithms [18, 28, 33], good local minima can be identified efficiently [30, 44, 31].

1.2. Contribution. An immediate question is how to quantify the power of function compositions in nonlinear approximation, if any, and what are the upper and lower bounds of the advantage. Our first result is to show that, for an arbitrary function f on $[0,1]$, if f can be approximated via nonlinear approximation using one-hidden-layer ReLU FNNs with an approximation rate $\mathcal{O}(N^{-\eta})$ for any $\eta > 0$, we show that dictionaries with two function compositions via two-hidden-layer (one composition) ReLU FNNs can improve the approximation rate to $\mathcal{O}(N^{-2\eta})$. Second, for Hölder continuous functions of order α with a Lipchitz constant ν on a d-dimensional cube, we show that the N-term approximation via ReLU FNNs with two or three hidden layers can achieve the approximation rate $\mathcal{O}(\nu N^{-2\alpha/d})$. Instead of constructing FNNs to approximate traditional approximation tools like polynomials and splines as in existing literature [50, 37, 48, 59, 60, 42, 22, 40, 46, 53, 54] on the approximation theory of deep neural networks, this paper adopts a novel perspective and provides new analysis methods merely based on the structure of FNNs and is able to analyze the approximation capacity of FNNs with $\mathcal{O}(1)$ (i.e., a small constant relative to the width N) layers. More importantly, the approximation rate $\mathcal{O}(\nu N^{-2\alpha/d})$ is **non-asymptotic** (**quantitative**) and valid for all positive integers N with an explicit formula to specify its constant prefactor.

The fact that composing functions once or twice can improve the approximation rate from $\mathcal{O}(N^{-\eta})$ to $\mathcal{O}(N^{-2\eta})$ raises an interesting and widely studied conjecture: whether it is possible to improve the approximation rate to $\mathcal{O}(N^{-L\eta})$ via L times function compositions. We show that if the depth of the composition is $L = \mathcal{O}(1)$, a nearly tight approximation rate is $\mathcal{O}(\nu N^{-2\alpha/d})$, which implies that adding one more layer cannot improve the approximation rate when N is large, although it can improve the approximation accuracy due to the increase of parameters in the approximant.

[1] In this paper, we use the big $\mathcal{O}(\cdot)$ notation when we only care about the scaling in terms of the variables inside $(\cdot)$ and the prefactor outside $(\cdot)$ is independent of these variables.

Finally, we analyze the approximation efficiency of neural networks in parallel computing, a very important point of view that was not paid attention to in the literature. In most applications, the efficiency of deep learning computation highly relies on parallel computation. We show that a narrow and very deep neural network is inefficient, if its approximation rate is not exponentially better than wide and shallower networks. Hence, neural networks with $\mathcal{O}(1)$ layers is more attractive in modern computational platforms, considering the computational efficiency per training iteration in parallel computing platforms. Our conclusion does not conflict with the current state-of-the-art deep learning research, since most of these successful deep neural networks have depth that is asymptotically $\mathcal{O}(1)$ relative to the width.

1.3. Related work. This paper focuses on the power of compositions in nonlinear approximation and the ReLU FNN is one possible choice to implement the nonlinear approximation. Our analysis on the lower bound of the achievable approximation rate (or equivalently the upper bound of N in the N-term approximation for a given accuracy) is quantitative and proved by construction; while the result for the upper bound of unachievable approximation rate (or equivalently the lower bound of N in the N-term approximation for a given accuracy) is asymptotic. To the best of our knowledge, quantitative analysis in this paper is new; related works in the literature are asymptotic and they can be summarized as follows.

The topic in this paper is related to the study of the expressiveness of neural networks from many other points of views, e.g., in terms of combinatorics [43], topology [5], Vapnik-Chervonenkis (VC) dimension [4, 51, 23] and fat-shattering dimension [32, 2], information theory [46], classical approximation theory [12, 25, 3] etc. Particularly in approximation theory, previous research focused on the design of Sigmoid activation functions [39, 36, 10, 9, 11], while recent works emphasized the power of depth (or equivalently the power of function compositions) in ReLU FNNs and explained why increasing depth could lead to larger approximation capacity [50, 37, 48, 59, 60, 42, 22, 40, 46, 53, 54].

Approximation theories in [37, 48, 59, 46, 42, 54] aim for target functions with stronger smoothness, e.g. functions in $C^\alpha([0,1]^d)$ with $\alpha \geq 1$, Korobov spaces, or Besev spaces, and show that deep FNNs have better approximation capacity than shallow FNNs, especially in the case when the target functions are polynomials. A common idea utilized in these works is to construct FNNs to approximate traditional basis in approximation theory, e.g., polynomials, splines, and sparse grids, which are used to approximate smooth functions. Shallow FNNs therein have minimum depth asymptotically depending on the accuracy ε or dimension d.

A more closely related work that proves the optimal rate of approximation of general continuous functions by ReLU FNNs in terms of the number of network weights, W, and the modulus of continuity of the function was presented in [60]. In particular, if we define the modulus of continuity $\omega_f(r)$ of $f:[0,1]^d \to \mathbb{R}$ by

$$\omega_f(r) \coloneqq \max\{|f(\boldsymbol{x}) - f(\boldsymbol{y})| : \boldsymbol{x}, \boldsymbol{y} \in [0,1]^d, |\boldsymbol{x} - \boldsymbol{y}| \leq r\},$$

then [60] proved that the optimal approximation rate of $f \in C([0,1]^d)$ in the L^∞-norm by a deep ReLU FNN is $\mathcal{O}\big(\omega_f(\mathcal{O}(W^{-2/d}))\big)$, when the FNN has width $\mathcal{O}(d)$, depth $\mathcal{O}(W)$, i.e., the FNN is very deep and narrow.

1.4. Organization. The rest of the paper is organized as follows. Section 2 summarizes the notations throughout this paper. Section 3 presents the main theorems while Section 4 shows numerical tests in parallel computing to support the claims in this paper. Finally, Section 5 concludes this paper with a short discussion.

2. Preliminaries. For the purpose of convenience, we present notations and elementary lemmas used throughout this paper as follows.

2.1. Notations.

- Matrices are denoted by bold uppercase letters, e.g., $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ is a real matrix of size $m \times n$, and $\boldsymbol{A}^T$ denotes the transpose of $\boldsymbol{A}$. Correspondingly, $\boldsymbol{A}(i,j)$ is the (i,j)-th entry of $\boldsymbol{A}$; $\boldsymbol{A}(:,j)$ is the j-th column of $\boldsymbol{A}$; $\boldsymbol{A}(i,:)$ is the i-th row of $\boldsymbol{A}$.
- Vectors are denoted as bold lowercase letters, e.g., $\boldsymbol{v} \in \mathbb{R}^n$ is a column vector of size n and $\boldsymbol{v}(i)$ is the i-th element of $\boldsymbol{v}$. $\boldsymbol{v} = [v_1, \cdots, v_n]^T = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}$ are vectors consisting of numbers $\{v_i\}$ with $\boldsymbol{v}(i) = v_i$.

- The Lebesgue measure is denoted as $\mu(\cdot)$.
- The set difference of A and B is denoted by $A\backslash B$. A^c denotes $[0,1]^d\backslash A$ for any $A\subseteq[0,1]^d$.
- For a set of numbers A, and a number x, $A-x:=\{y-x:y\in A\}$.
- For any $\xi\in\mathbb{R}$, let $\lfloor\xi\rfloor:=\max\{i:i\leq\xi,\ i\in\mathbb{N}\}$ and $\lceil\xi\rceil:=\min\{i:i\geq\xi,\ i\in\mathbb{N}\}$.
- Assume $\boldsymbol{n}\in\mathbb{N}^n$, then $f(\boldsymbol{n})=\mathcal{O}(g(\boldsymbol{n}))$ means that there exists positive C independent of $\boldsymbol{n}$, f, and g such that $f(\boldsymbol{n})\leq Cg(\boldsymbol{n})$ when $\boldsymbol{n}(i)$ goes to $+\infty$ for all i.
- Define $\mathrm{Lip}(\nu,\alpha,d)$ as the class of functions g defined on $[0,1]^d$ satisfying the uniformly Lipchitz property of order α with a Lipchitz constant $\nu>0$.
- Let $\mathrm{CPL}(N)$ be the set of continuous piecewise linear functions with $N-1$ pieces mapping $[0,1]$ to $\mathbb{R}$. The end points of each linear piece are called "break points" in this paper.
- Let $\sigma:\mathbb{R}\to\mathbb{R}$ denote the rectified linear unit (ReLU), i.e. $\sigma(x)=\max\{0,x\}$.
- We will use NN as a ReLU neural network for short and use Python-type notations to specify a class of NN's, e.g., $\mathrm{NN}(\mathrm{c}_1;\ \mathrm{c}_2;\ \cdots;\ \mathrm{c}_m)$ is a set of ReLU FNN's satisfying m conditions given by $\{\mathrm{c}_i\}_{1\leq i\leq m}$, each of which may specify the number of inputs (#input), the total number of nodes in all hidden layers (#node), the number of hidden layers (#layer), the total number of parameters (#parameter), and the width in each hidden layer (widthvec), the maximum width of all hidden layers (maxwidth) etc. For example, $\mathrm{NN}(\#\mathrm{input}=2;\ \mathrm{widthvec}=[100,100])$ is a set of NN's ϕ satisfying:
  - ϕ maps from $\mathbb{R}^2$ to $\mathbb{R}$.
  - ϕ has two hidden layers and the number of nodes in each hidden layer is 100.
- $[n]^L$ is short for $[n,n,\cdots,n]\in\mathbb{N}^L$. For example,

$$\mathrm{NN}(\#\mathrm{input}=d;\ \mathrm{widthvec}=[100,100,100])=\mathrm{NN}(\#\mathrm{input}=d;\ \mathrm{widthvec}=[100]^3).$$

- For $\phi\in\mathrm{NN}(\#\mathrm{input}=d;\ \mathrm{widthvec}=[N_1,N_2,\cdots,N_L])$, if we define $N_0=d$ and $N_{L+1}=1$, then the architecture of ϕ can be briefly described as follows:

$$\boldsymbol{x}=\tilde{\boldsymbol{h}}_0\xrightarrow{\boldsymbol{W}_1,\ \boldsymbol{b}_1}\boldsymbol{h}_1\xrightarrow{\sigma}\tilde{\boldsymbol{h}}_1\ \cdots\ \xrightarrow{\boldsymbol{W}_L,\ \boldsymbol{b}_L}\boldsymbol{h}_L\xrightarrow{\sigma}\tilde{\boldsymbol{h}}_L\xrightarrow{\boldsymbol{W}_{L+1},\ \boldsymbol{b}_{L+1}}\phi(\boldsymbol{x})=\boldsymbol{h}_{L+1},$$

  where $\boldsymbol{W}_i\in\mathbb{R}^{N_i\times N_{i-1}}$ and $\boldsymbol{b}_i\in\mathbb{R}^{N_i}$ are the weight matrix and the bias vector in the i-th linear transform in ϕ, respectively, i.e., $\boldsymbol{h}_i:=\boldsymbol{W}_i\tilde{\boldsymbol{h}}_{i-1}+\boldsymbol{b}_i$ for $i=1,2,\cdots,L+1$ and $\tilde{\boldsymbol{h}}_i=\sigma(\boldsymbol{h}_i)$ for $i=1,2,\cdots,L$.

2.2. Lemmas. Let us study the properties of ReLU FNNs with only one hidden layer to warm up in Lemma 2.1 below. It indicates that $\mathrm{CPL}(N+1)=\mathrm{NN}(\#\mathrm{input}=1;\ \mathrm{widthvec}=[N])$ for any $N\in\mathbb{N}^+$.

LEMMA 2.1. *Suppose* $\phi\in\mathrm{NN}(\#\mathrm{input}=1;\ \mathrm{widthvec}=[N])$ *with an architecture:*

$$x\xrightarrow{\boldsymbol{W}_1,\ \boldsymbol{b}_1}\boldsymbol{h}\xrightarrow{\sigma}\tilde{\boldsymbol{h}}\xrightarrow{\boldsymbol{W}_2,\ \boldsymbol{b}_2}\phi(x).$$

Then ϕ is a continuous piecewise linear function. Let $\boldsymbol{W}_1=[1,1,\cdots,1]^T\in\mathbb{R}^{N\times1}$, then we have:

(1) Given a sequence of strictly increasing numbers x_0, $x_1,\cdots$, x_N, there exists $\boldsymbol{b}_1\in\mathbb{R}^N$ independent of $\boldsymbol{W}_2$ and $\boldsymbol{b}_2$ such that the break points of ϕ are exactly x_0, $\cdots$, x_N on the interval $[x_0,x_N]$[②]*.*

(2) Suppose $\{x_i\}_{i\in\{0,1,\cdots,N\}}$ and $\boldsymbol{b}_1$ are given in (1). Given any sequence $\{y_i\}_{i\in\{0,1,\cdots,N\}}$, there exist $\boldsymbol{W}_2$ and $\boldsymbol{b}_2$ such that $\phi(x_i)=y_i$ for $i=0,1,\cdots,N$ and ϕ is linear on $[x_i,x_{i+1}]$ for $i=0,1,\cdots,N-1$.

Part (1) in Lemma 2.1 follows by setting $\boldsymbol{b}_1=[-x_0,-x_1,\cdots,-x_{N-1}]^T$. The existence in Part (2) is equivalent to the existence of a solution of linear equations, which is left for the reader. Next, we study the properties of ReLU FNNs with two hidden layers. In fact, we can show that the closure of $\mathrm{NN}(\#\mathrm{input}=1;\ \mathrm{widthvec}=[2m,2n+1])$ contains $\mathrm{CPL}(mn+1)$ for any $m,n\in\mathbb{N}^+$, where the closure is in the sense of L^p-norm for any $p\in[1,\infty)$. The proof of this property relies on the following lemma.

[②] We only consider the interval $[x_0,x_N]$ and hence x_0 and x_N are treated as break points. $\phi(x)$ might not have a real break point in a small open neighborhood of x_0 or x_N.

LEMMA 2.2. *For any $m,n \in \mathbb{N}^+$, given any $m(n+1)+1$ samples $(x_i, y_i) \in \mathbb{R}^2$ with $x_0 < x_1 < x_2 < \cdots < x_{m(n+1)}$ and $y_i \geq 0$ for $i = 0,1,\cdots,m(n+1)$, there exists $\phi \in \mathrm{NN}(\#\mathrm{input} = 1; \mathrm{widthvec} = [2m, 2n+1])$ satisfying the following conditions:*
(1) $\phi(x_i) = y_i$ for $i = 0,1,\cdots,m(n+1)$;
(2) ϕ is linear on each interval $[x_{i-1}, x_i]$ for $i \notin \{(n+1)j : j = 1,2,\cdots,m\}$;
(3) $\sup_{x\in[x_0,\, x_{m(n+1)}]}|\phi(x)| \leq 3\max_{i\in\{0,1,\cdots,m(n+1)\}} y_i \prod_{k=1}^n \Big(1 + \frac{\max\{x_{j(n+1)+n} - x_{j(n+1)+k-1} : j=0,1,\cdots,m-1\}}{\min\{x_{j(n+1)+k} - x_{j(n+1)+k-1} : j=0,1,\cdots,m-1\}}\Big)$.

Proof. For simplicity of notation, we define $I_0(m,n) \coloneqq \{0,1,\cdots,m(n+1)\}$, $I_1(m,n) \coloneqq \{j(n+1) : j = 1,2,\cdots,m\}$, and $I_2(m,n) \coloneqq I_0(m,n)\backslash I_1(m,n)$ for any $m,n \in \mathbb{N}^+$. Since $\phi \in \mathrm{NN}(\#\mathrm{input} = 1; \mathrm{widthvec} = [2m, 2n+1])$, the architecture of ϕ is

$$x \xrightarrow{\boldsymbol{W}_1,\ \boldsymbol{b}_1} \boldsymbol{h} \xrightarrow{\ \sigma\ } \tilde{\boldsymbol{h}} \xrightarrow{\boldsymbol{W}_2,\ \boldsymbol{b}_2} \boldsymbol{g} \xrightarrow{\ \sigma\ } \tilde{\boldsymbol{g}} \xrightarrow{\boldsymbol{W}_3,\ \boldsymbol{b}_3} \phi(x). \tag{2.1}$$

Note that $\boldsymbol{g}$ maps $x \in \mathbb{R}$ to $\boldsymbol{g}(x) \in \mathbb{R}^{2n+1}$ and hence each entry of $\boldsymbol{g}(x)$ itself is a sub-network with one hidden layer. Denote $\boldsymbol{g} = [g_0, g_1^+, g_1^-, \cdots, g_n^+, g_n^-]^T$, then $\{g_0, g_1^+, g_1^-, \cdots, g_n^+, g_n^-\} \subseteq \mathrm{NN}(\#\mathrm{input} = 1; \mathrm{widthvec} = [2m])$. Our proof of Lemma 2.2 is mainly based on the repeated applications of Lemma 2.1 to determine parameters of $\phi(x)$ such that Conditions (1) to (3) hold.

Step 1: Determine $\boldsymbol{W}_1$ and $\boldsymbol{b}_1$.

By Lemma 2.1, $\exists\ \boldsymbol{W}_1 = [1,1,\cdots,1]^T \in \mathbb{R}^{2m\times 1}$ and $\boldsymbol{b}_1 \in \mathbb{R}^{2m}$ such that sub-networks in $\{g_0, g_1^+, g_1^-, \cdots, g_n^+, g_n^-\}$ have the same set of break points: $\Big\{x_i : i \in I_1(m,n) \cup \big(I_1(m,n) - 1\big) \cup \{0\}\Big\}$, no matter what $\boldsymbol{W}_2$ and $\boldsymbol{b}_2$ are.

Step 2: Determine $\boldsymbol{W}_2$ and $\boldsymbol{b}_2$.

This is the key step of the proof. Our ultimate goal is to set up $\boldsymbol{g} = [g_0, g_1^+, g_1^-, \cdots, g_n^+, g_n^-]^T$ such that, after a nonlinear activation function, there exists a linear combination in the last step of our network (specified by $\boldsymbol{W}_3$ and $\boldsymbol{b}_3$ as shown in (2.1)) that can generate a desired $\phi(x)$ matching the sample points $\{(x_i, y_i)\}_{0\leq i\leq m(n+1)}$. In the previous step, we have determined the break points of $\{g_0, g_1^+, g_1^-, \cdots, g_n^+, g_n^-\}$ by setting up $\boldsymbol{W}_1$ and $\boldsymbol{b}_1$; in this step, we will identify $\boldsymbol{W}_2 \in \mathbb{R}^{(2n+1)\times 2m}$ and $\boldsymbol{b}_2 \in \mathbb{R}^{2n+1}$ to fully determine $\{g_0, g_1^+, g_1^-, \cdots, g_n^+, g_n^-\}$. This will be conducted in two sub-steps.

Step 2.1: Set up.

Suppose $f_0(x)$ is a continuous piecewise linear function defined on $[0,1]$ fitting the given samples $f_0(x_i) = y_i$ for $i \in I_0(m,n)$, and f_0 is linear between any two adjacent points of $\{x_i : i \in I_0(m,n)\}$. We are able to choose $\boldsymbol{W}_2(1,:)$ and $\boldsymbol{b}_2(1)$ such that $g_0(x_i) = f_0(x_i)$ for $i \in I_1(m,n) \cup (I_1(m,n)-1) \cup \{0\}$ by Lemma 2.1, since there are $2m+1$ points in $I_1(m,n) \cup (I_1(m,n)-1) \cup \{0\}$. Define $f_1 \coloneqq f_0 - \tilde{g}_0$, where $\tilde{g}_0 = \sigma(g_0) = g_0$ as shown in Equation (2.1), since g_0 is positive by the construction of Lemma 2.1. Then we have $f_1(x_i) = f_0(x_i) - \tilde{g}_0(x_i) = 0$ for $i \in (I_1(m,n) - n - 1) \cup \{m(n+1)\}$. See Figure 1 (a) for an illustration of f_0, f_1, and g_0.

Step 2.2: Mathematical induction.

For each $k \in \{1,2,\cdots,n\}$, given f_k, we determine $\boldsymbol{W}_2(2k,:)$, $\boldsymbol{b}_2(2k)$, $\boldsymbol{W}_2(2k+1,:)$, and $\boldsymbol{b}_2(2k+1)$, to completely specify g_k^+ and g_k^-, which in turn can determine f_{k+1}. Hence, it is only enough to show how to proceed with an arbitrary k, since the initialization of the induction, f_1, has been constructed in Step 2.1. See Figure 1 (b)-(d) for the illustration of the first two induction steps. We recursively rely on the fact of f_k that

- $f_k(x_i) = 0$ for $i \in \cup_{\ell=0}^{k-1}(I_1(m,n) - n - 1 + \ell) \cup \{m(n+1)\}$,
- f_k is linear on each interval $[x_{i-1}, x_i]$ for $i \in I_2(m,n)\backslash\{0\}$,

to construct f_{k+1} satisfying similar conditions as follows:

- $f_{k+1}(x_i) = 0$ for $i \in \cup_{\ell=0}^{k}(I_1(m,n) - n - 1 + \ell) \cup \{m(n+1)\}$,
- f_{k+1} is linear on each interval $[x_{i-1}, x_i]$ for $i \in I_2(m,n)\backslash\{0\}$.

The induction process for $\boldsymbol{W}_2(2k,:)$, $\boldsymbol{b}_2(2k)$, $\boldsymbol{W}_2(2k+1,:)$, $\boldsymbol{b}_2(2k+1)$, and f_{k+1} can be divided into four parts.

Step 2.2.1: Define index sets.

Let $\Lambda_k^+ = \{j : f_k(x_{j(n+1)+k}) \geq 0,\ 0 \leq j < m\}$ and $\Lambda_k^- = \{j : f_k(x_{j(n+1)+k}) < 0,\ 0 \leq j < m\}$. The cardinality of $\Lambda_k^+ \cup \Lambda_k^-$ is m. We will use Λ_k^+ and Λ_k^- to generate $2m+1$ samples to determine CPL functions $g_k^+(x)$ and $g_k^-(x)$ in the next step.

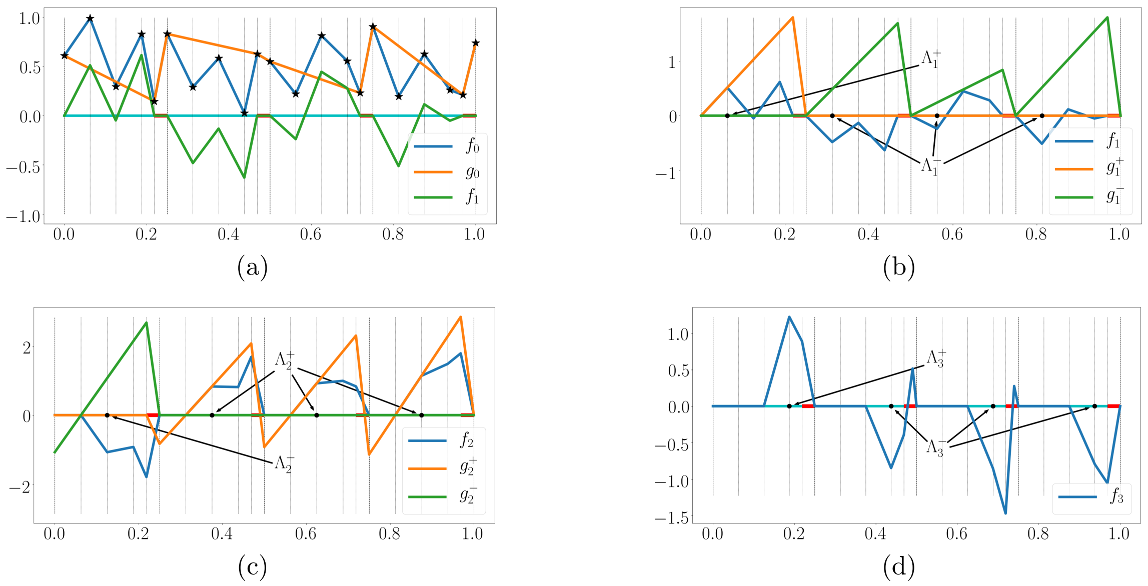


FIG. 1. *Illustrations of the proof of Lemma 2.2, especially Step 2 of the proof, when $m = n = 4$, with the "don't-care" region in red. (a) Given samples $\{(x_i, y_i) : i = 0, 1, \cdots, m(n+1)\}$ marked with "star" signs, suppose $f_0(x)$ is a CPL function fitting the samples, construct g_0 such that $f_1 = f_0 - \sigma(g_0)$ is closer to 0 than f_0 in the L^∞ sense. (b) Construct g_1^+ and g_1^- such that $f_2 = f_1 - \sigma(g_1^+) + \sigma(g_1^-)$ is closer to 0 than f_1 in the L^∞ sense in a subset of the "important" region. (c) Construct g_2^+ and g_2^- such that $f_3 = f_2 - \sigma(g_2^+) + \sigma(g_2^-)$ is closer to 0 than f_2 in the L^∞ sense in a larger subset of the "important" region. (d) The visualization of f_3, which is 0 in the "important" areas that have been processed and may remain large near the "don't-care" region. f_k will decay quickly outside the "don't-care" region as k increases.*

Step 2.2.2: Determine $\boldsymbol{W}_2(2k,:)$ and $\boldsymbol{b}_2(2k)$.

By Lemma 2.1, we can choose $\boldsymbol{W}_2(2k,:)$ and $\boldsymbol{b}_2(2k)$ to fully determine $g_k^+(x)$ such that each $g_k^+(x_i)$ matches a specific value for $i \in (I_1(m,n) - n - 1) \cup (I_1(m,n) - 1) \cup \{m(n+1)\}$. The values of $\{g_k^+(x_i) : i \in (I_1(m,n) - n - 1) \cup (I_1(m,n) - 1) \cup \{m(n+1)\}\}$ are specified as:

- If $j \in \Lambda_k^+$, specify the values of $g_k^+(x_{j(n+1)})$ and $g_k^+(x_{j(n+1)+n})$ such that $g_k^+(x_{j(n+1)+k-1}) = 0$ and $g_k^+(x_{j(n+1)+k}) = f_k(x_{j(n+1)+k})$. The existence of these values fulfilling the requirements above comes from the fact that $g_k^+(x)$ is linear on the interval $[x_{j(n+1)}, x_{j(n+1)+n}]$ and $g_k^+(x)$ only depends on the values of $g_k^+(x_{j(n+1)+k-1})$ and $g_k^+(x_{j(n+1)+k})$ on $[x_{j(n+1)}, x_{j(n+1)+n}]$. Now it is easy to verify that $\tilde{g}_k^+(x) := \sigma(g_k^+(x))$ satisfies $\tilde{g}_k^+(x_{j(n+1)+k}) = f_k(x_{j(n+1)+k}) \geq 0$ and $\tilde{g}_k^+(x_{j(n+1)+\ell}) = 0$ for $\ell = 0, 1, \cdots, k-1$, and $\tilde{g}_k^+$ is linear on each interval $[x_{j(n+1)+\ell}, x_{j(n+1)+\ell+1}]$ for $\ell = 0, 1, \cdots, n-1$.
- If $j \in \Lambda_k^-$, let $g_k^+(x_{j(n+1)}) = g_k^+(x_{j(n+1)+n}) = 0$. Then $\tilde{g}_k^+(x) = 0$ on the interval $[x_{j(n+1)}, x_{j(n+1)+n}]$.
- Finally, specify the value of $g_k^+(x)$ at $x = x_{m(n+1)}$ as 0.

Step 2.2.3: Determine $\boldsymbol{W}_2(2k+1,:)$ and $\boldsymbol{b}_2(2k+1)$.

Similarly, we choose $\boldsymbol{W}_2(2k+1,:)$ and $\boldsymbol{b}_2(2k+1)$ such that $g_k^-(x)$ matches specific values as follows:

- If $j \in \Lambda_k^-$, specify the values of $g_k^-(x_{j(n+1)})$ and $g_k^-(x_{j(n+1)+n})$ such that $g_k^-(x_{j(n+1)+k-1}) = 0$ and $g_k^-(x_{j(n+1)+k}) = -f_k(x_{j(n+1)+k})$. Then $\tilde{g}_k^-(x) := \sigma(g_k^-(x))$ satisfies $\tilde{g}_k^-(x_{j(n+1)+k}) = -f_k(x_{j(n+1)+k}) > 0$ and $\tilde{g}_k^-(x_{j(n+1)+\ell}) = 0$ for $\ell = 0, 1, \cdots, k-1$, and $\tilde{g}_k^-(x)$ is linear on each interval $[x_{j(n+1)+\ell}, x_{j(n+1)+\ell+1}]$ for $\ell = 0, 1, \cdots, n-1$.

- If $j \in \Lambda_k^+$, let $g_k^-(x_{j(n+1)}) = g_k^-(x_{j(n+)+n}) = 0$. Then $\tilde{g}_k^-(x) = 0$ on the interval $[x_{j(n+1)}, x_{j(n+1)+n}]$.
- Finally, specify the value of $g_k^-(x)$ at $x = x_{m(n+1)}$ as 0.

Step 2.2.4: Construct f_{k+1} from g_k^+ and g_k^-.

For the sake of clarity, the properties of g_k^+ and g_k^- constructed in Step 2.2.3 are summarized below:
(1) $f_k(x_i) = \tilde{g}_k^+(x_i) = \tilde{g}_k^-(x_i) = 0$ for $i \in \cup_{\ell=0}^{k-1}(I_1(m,n) - n - 1 + \ell) \cup \{m(n+1)\}$;
(2) If $j \in \Lambda_k^+$, $\tilde{g}_k^+(x_{j(n+1)+k}) = f_k(x_{j(n+1)+k}) \geq 0$ and $\tilde{g}_k^-(x_{j(n+1)+k}) = 0$;
(3) If $j \in \Lambda_k^-$, $\tilde{g}_k^-(x_{j(n+1)+k}) = -f_k(x_{j(n+1)+k}) > 0$ and $\tilde{g}_k^+(x_{j(n+1)+k}) = 0$;
(4) $\tilde{g}_k^+$ and $\tilde{g}_k^-$ are linear on each interval $[x_{j(n+1)+\ell}, x_{j(n+1)+\ell+1}]$ for $\ell = 0, 1, \cdots, n-1$, $j \in \Lambda_k^+ \cup \Lambda_k^- = \{0, 1, \cdots, m-1\}$. In other words, $\tilde{g}_k^+$ and $\tilde{g}_k^-$ are linear on each interval $[x_{i-1}, x_i]$ for $i \in I_2(m,n) \backslash \{0\}$.

See Figure 1 (a)-(c) for the illustration of g_0, g_1^+, g_1^-, g_2^+, and g_2^-, and to verify their properties as listed above.

Note that $\Lambda_k^+ \cup \Lambda_k^- = \{0, 1, \cdots, m-1\}$, so $f_k(x_i) - \tilde{g}_k^+(x_i) + \tilde{g}_k^-(x_i) = 0$ for $i \in \cup_{\ell=0}^{k}(I_1(m,n) - n - 1 + \ell) \cup \{m(n+1)\}$. Now we define $f_{k+1} := f_k - \tilde{g}_k^+ + \tilde{g}_k^-$, then

- $f_{k+1}(x_i) = 0$ for $i \in \cup_{\ell=0}^{k}(I_1(m,n) - n - 1 + \ell) \cup \{m(n+1)\}$;
- f_{k+1} is linear on each interval $[x_{i-1}, x_i]$ for $i \in I_2(m,n) \backslash \{0\}$.

See Figure 1 (b)-(d) for the illustration of f_1, f_2, and f_3, and to verify their properties as listed just above. This finishes the mathematical induction process. As we can imagine based on Figure 1, when k increases, the support of f_k shrinks to the “don't-care” region.

Step 3: Determine $\boldsymbol{W}_3$ and $\boldsymbol{b}_3$.

With the special vector function $\boldsymbol{g} = [g_0, g_1^+, g_1^-, \cdots, g_n^+, g_n^-]^T$ constructed in Step 2, we are able to specify $\boldsymbol{W}_3$ and $\boldsymbol{b}_3$ to generate a desired $\phi(x)$ with a well-controlled L^∞-norm matching the samples $\{(x_i, y_i)\}_{0 \leq i \leq m(n+1)}$.

In fact, we can simply set $\boldsymbol{W}_3 = [1, 1, -1, 1, -1, \cdots, 1, -1] \in \mathbb{R}^{1 \times (2n+1)}$ and $\boldsymbol{b}_3 = 0$, which finishes the construction of $\phi(x)$. The rest of the proof is to verify the properties of $\phi(x)$. Note that $\phi = \tilde{g}_0 + \sum_{l=1}^n \tilde{g}_l^+ - \sum_{l=1}^n \tilde{g}_l^-$. By the mathematical induction, we have:

- $f_{n+1} = f_0 - \tilde{g}_0 - \sum_{\ell=1}^n \tilde{g}_\ell^+ + \sum_{\ell=1}^n \tilde{g}_\ell^-$;
- $f_{n+1}(x_i) = 0$ for $i \in \cup_{\ell=0}^{n}(I_1(m,n) - n - 1 + \ell) \cup \{m(n+1)\} = I_0(m,n)$;
- f_{n+1} is linear on each interval $[x_{i-1}, x_i]$ for $i \in I_2(m,n) \backslash \{0\}$.

Hence, $\phi = \tilde{g}_0 + \sum_{\ell=1}^n \tilde{g}_\ell^+ - \sum_{\ell=1}^n \tilde{g}_\ell^- = f_0 - f_{n+1}$. Then ϕ satisfies Conditions (1) and (2) of this lemma. It remains to check that ϕ satisfies Condition (3).

By the definition of f_1, we have

$$\sup_{x \in [x_0, x_{m(n+1)}]} |f_1(x)| \leq 2 \max\{y_i : i \in I_0(m,n)\}. \tag{2.2}$$

By the induction process in Step 2, for $k \in \{1, 2, \cdots, n\}$, it holds that

$$\sup_{x \in [x_0, x_{m(n+1)}]} |\tilde{g}_k^+(x)| \leq \frac{\max\{x_{j(n+1)+n} - x_{j(n+1)+k-1} : j = 0, 1, \cdots, m-1\}}{\min\{x_{j(n+1)+k} - x_{j(n+1)+k-1} : j = 0, 1, \cdots, m-1\}} \sup_{x \in [x_0, x_{m(n+1)}]} |f_k(x)| \tag{2.3}$$

and

$$\sup_{x \in [x_0, x_{m(n+1)}]} |\tilde{g}_k^-(x)| \leq \frac{\max\{x_{j(n+1)+n} - x_{j(n+1)+k-1} : j = 0, 1, \cdots, m-1\}}{\min\{x_{j(n+1)+k} - x_{j(n+1)+k-1} : j = 0, 1, \cdots, m-1\}} \sup_{x \in [x_0, x_{m(n+1)}]} |f_k(x)|. \tag{2.4}$$

Since either $\tilde{g}_k^+(x)$ or $\tilde{g}_k^-(x)$ is equal to 0 on $[0,1]$, we have

$$\sup_{x \in [x_0, x_{m(n+1)}]} |\tilde{g}_k^+(x) - \tilde{g}_k^-(x)| \leq \frac{\max\{x_{j(n+1)+n} - x_{j(n+1)+k-1} : j = 0, 1, \cdots, m-1\}}{\min\{x_{j(n+1)+k} - x_{j(n+1)+k-1} : j = 0, 1, \cdots, m-1\}} \sup_{x \in [x_0, x_{m(n+1)}]} |f_k(x)|. \tag{2.5}$$

Note that $f_{k+1} = f_k - \tilde{g}_k^+ + \tilde{g}_k^-$, which means

$$\begin{aligned}\sup_{x\in[x_0,\,x_{m(n+1)}]} |f_{k+1}(x)| &\le \sup_{x\in[x_0,\,x_{m(n+1)}]} |\tilde{g}_k^+(x) - \tilde{g}_k^-(x)| + \sup_{x\in[x_0,\,x_{m(n+1)}]} |f_k(x)| \\ &\le \Big(\tfrac{\max\{x_{j(n+1)+n}-x_{j(n+1)+k-1}:j=0,1,\cdots,m-1\}}{\min\{x_{j(n+1)+k}-x_{j(n+1)+k-1}:j=0,1,\cdots,m-1\}} + 1\Big) \sup_{x\in[x_0,\,x_{m(n+1)}]} |f_k(x)|\end{aligned} \tag{2.6}$$

for $k \in \{1, 2, \cdots, n\}$. Then we have

$$\sup_{x\in[x_0,\,x_{m(n+1)}]} |f_{n+1}(x)| \le 2 \max_{i\in I_0(m,n)} y_i \prod_{k=1}^{n} \Big(\tfrac{\max\{x_{j(n+1)+n}-x_{j(n+1)+k-1}:j=0,1,\cdots,m-1\}}{\min\{x_{j(n+1)+k}-x_{j(n+1)+k-1}:j=0,1,\cdots,m-1\}} + 1\Big).$$

Hence

$$\begin{aligned}\sup_{x\in[x_0,\,x_{m(n+1)}]} |\phi(x)| &\le \sup_{x\in[x_0,\,x_{m(n+1)}]} |f_0(x)| + \sup_{x\in[x_0,\,x_{m(n+1)}]} |f_{n+1}(x)| \\ &\le 3 \max_{i\in I_0(m,n)} y_i \prod_{k=1}^{n} \Big(\tfrac{\max\{x_{j(n+1)+n}-x_{j(n+1)+k-1}:j=0,1,\cdots,m-1\}}{\min\{x_{j(n+1)+k}-x_{j(n+1)+k-1}:j=0,1,\cdots,m-1\}} + 1\Big).\end{aligned}$$

So, we finish the proof. □

3. Main Results.

We present our main results in this section. First, we quantitatively prove an achievable approximation rate in the N-term nonlinear approximation by construction, i.e., the lower bound of the approximation rate. Second, we show a lower bound of the approximation rate asymptotically, i.e., no approximant exists asymptotically following the approximation rate. Finally, we discuss the efficiency of the nonlinear approximation considering the approximation rate and parallel computing in FNNs together.

3.1. Quantitative Achievable Approximation Rate.

THEOREM 3.1. *For any $N \in \mathbb{N}^+$ and $f \in \mathrm{Lip}(\nu, \alpha, d)$ with $\alpha \in (0, 1]$, we have*[3]*:*

(1) If $d = 1$, $\exists\ \phi \in \mathrm{NN}(\#\text{input} = 1;\ \text{widthvec} = [2N, 2N+1])$ such that

$$\|\phi - f\|_{L^1([0,1])} \le 2\nu N^{-2\alpha}, \qquad \text{for any} \quad N \in \mathbb{N}^+;$$

(2) If $d > 1$, $\exists\ \phi \in \mathrm{NN}(\#\text{input} = d;\ \text{widthvec} = [2d\lfloor N^{2/d}\rfloor, 2N+2, 2N+3])$ such that

$$\|\phi - f\|_{L^1([0,1]^d)} \le 2(2\sqrt{d})^{\alpha}\nu N^{-2\alpha/d}, \qquad \text{for any} \quad N \in \mathbb{N}^+.$$

Proof. Without loss of generality, we assume $f(0) = 0$ and $\nu = 1$.

Step 1: The case $d = 1$.

Given any $f \in \mathrm{Lip}(\nu, \alpha, d)$ and $N \in \mathbb{N}^+$, we know $|f(x)| \le 1$ for any $x \in [0, 1]$ since $f(0) = 0$ and $\nu = 1$. Set $\bar{f} = f + 1 \ge 0$, then $0 \le \bar{f}(x) \le 2$ for any $x \in [0, 1]$. Let $X = \{\frac{i}{N^2} : i = 0, 1, \cdots, N^2\} \cup \{\frac{i}{N} - \delta : i = 1, 2, \cdots, N\}$, where δ is a sufficiently small positive number depending on N, and satisfying (3.2). Let us order X as $x_0 < x_1 < \cdots < x_{N(N+1)}$. By Lemma 2.2, given the set of samples $\{(x_i, \bar{f}(x_i)) : i \in \{0, 1, \cdots, N(N+1)\}\}$, there exists $\phi \in \mathrm{NN}(\#\text{input} = 1;\ \text{widthvec} = [2N, 2N+1])$ such that

- $\phi(x_i) = \bar{f}(x_i)$ for $i = 0, 1, \cdots, N(N+1)$;
- ϕ is linear on each interval $[x_{i-1}, x_i]$ for $i \notin \{(N+1)j : j = 1, 2, \cdots, N\}$;
- ϕ has an upper bound estimation: $\sup\{\phi(x) : x \in [0, 1]\} \le 6(N+1)!$.

It follows that

$$|\bar{f}(x) - \phi(x)| \le (x_i - x_{i-1})^{\alpha} \le N^{-2\alpha}, \quad \text{if } x \in [x_{i-1}, x_i], \quad \text{for } i \notin \{(N+1)j : j = 1, 2, \cdots, N\}.$$

[3]It is easy to generalize the results in Theorem 3.1 and Corollary 3.2 from L^1 to L^p-norm for $p \in [1, \infty)$ since $\mu([0, 1]^d) = 1$.

Define $H_0 = \cup_{i\in\{(N+1)j:j=1,2,\cdots,N\}}[x_{i-1},x_i]$, then

$$|\bar{f}(x)-\phi(x)| \le N^{-2\alpha}, \quad \text{for any } x\in[0,1]\backslash H_0, \tag{3.1}$$

by the fact that $\bar{f}\in \mathrm{Lip}(1,\alpha,d)$ and points in X are equispaced. By $\mu(H_0)\le N\delta$, it follows that

$$\begin{aligned}\|\bar{f}-\phi\|_{L^1([0,1])} &= \int_{H_0}|\bar{f}(x)-\phi(x)|dx + \int_{[0,1]\backslash H_0}|\bar{f}(x)-\phi(x)|dx \\ &\le N\delta(2+6(N+1)!) + N^2(N^{-2\alpha})N^{-2} \le 2N^{-2\alpha},\end{aligned}$$

where the last inequality comes from the fact δ is small enough satisfying

$$N\delta(2+6(N+1)!) \le N^{-2\alpha}. \tag{3.2}$$

Note that $f-(\phi-1) = f+1-\phi = \bar{f}-\phi$. Hence, $\phi-1\in \mathrm{NN}(\#\text{input}=1;\ \text{widthvec}=[2N,2N+1])$ and $\|f-(\phi-1)\|\le 2N^{-2\alpha}$. So, we finish the proof for the case $d=1$.

Step 2: The case $d>1$.

The main idea is to project the d-dimensional problem into a one-dimensional one and use the results proved above. For any $N\in\mathbb{N}^+$, let $n=\lfloor N^{2/d}\rfloor$ and δ be a sufficiently small positive number depending on N and d, and satisfying (3.8). We will divide the d-dimensional cube into n^d small non-overlapping sub-cubes (see Figure 2 for an illustration when $d=3$ and $n=3$), each of which is associated with a representative point, e.g., a vertex of the sub-cube. Due to the continuity, the target function f can be represented by their values at the representative points. We project these representatives to one-dimensional samples via a ReLU FNN ψ and construct a ReLU FNN $\bar{\phi}$ to fit them. Finally, the ReLU FNN ϕ on the d-dimensional space approximating f can be constructed by $\phi=\bar{\phi}\circ\psi$. The precise construction can be found below.

By Lemma 2.1, there exists $\psi_0\in \mathrm{NN}(\#\text{input}=1;\ \text{widthvec}=[2n])$ such that

- $\psi_0(1)=n-1$, and $\psi_0(\frac{i}{n})=\psi_0(\frac{i+1}{n}-\delta)=i$ for $i=0,1,\cdots,n-1$;
- ψ_0 is linear between any two adjacent points of $\{\frac{i}{n}: i=0,1,\cdots,n\}\cup\{\frac{i}{n}-\delta: i=1,2,\cdots,n\}$.

Define the projection map[4] ψ by

$$\psi(\boldsymbol{x}) = \sum_{i=1}^{d}\tfrac{1}{n^i}\psi_0(x_i), \quad \text{for } \boldsymbol{x}=[x_1,x_2,\cdots,x_d]^T\in[0,1]^d. \tag{3.3}$$

Note that $\psi\in \mathrm{NN}(\#\text{input}=1;\ \text{widthvec}=[2dn])$. Given $f\in\mathrm{Lip}(\nu,\alpha,d)$, then $|f(\boldsymbol{x})|\le\sqrt{d}$ for $\boldsymbol{x}\in[0,1]^d$ since $f(0)=0$, $\nu=1$, and $\alpha\in(0,1]$. Define $\bar{f}=f+\sqrt{d}$, then $0\le\bar{f}(\boldsymbol{x})\le 2\sqrt{d}$ for $\boldsymbol{x}\in[0,1]^d$. Hence, we have

$$\left\{\Big(\sum_{i=1}^{d}\tfrac{\theta_i}{n^i},\ \bar{f}(\tfrac{\boldsymbol{\theta}}{n})\Big) : \boldsymbol{\theta}=[\theta_1,\theta_2,\cdots,\theta_d]^T\in\{0,1,\cdots,n-1\}^d\right\}\cup\{(1,0)\}$$

as a set of n^d+1 samples of a one-dimensional function. By Lemma 2.1, $\exists\ \bar{\phi}\in \mathrm{NN}(\#\text{input}=1;\ \text{widthvec}=[2\lceil n^{d/2}\rceil, 2\lceil n^{d/2}\rceil+1])$ such that

$$\bar{\phi}\Big(\sum_{i=1}^{d}\tfrac{\theta_i}{n^i}\Big) = \bar{f}\left(\tfrac{\boldsymbol{\theta}}{n}\right), \quad \text{for } \boldsymbol{\theta}=[\theta_1,\theta_2,\cdots,\theta_d]^T\in\{0,1,\cdots,n-1\}^d, \tag{3.4}$$

and

$$\sup_{t\in[0,1]}|\bar{\phi}(t)| \le 6\sqrt{d}\left(\lceil n^{d/2}\rceil+1\right)!. \tag{3.5}$$

[4]The idea constructing such ψ comes from the binary representation.

Since the range of ψ on $[0,1]^d$ is a subset of $[0,1]$, $\exists\ \phi \in \mathrm{NN}\big(\#\text{input} = d;\ \text{widthvec} = [2nd, 2\lceil n^{d/2}\rceil, 2\lceil n^{d/2}\rceil + 1]\big)$ defined via $\phi(\boldsymbol{x}) = \bar{\phi}\circ\psi(\boldsymbol{x})$ for $\boldsymbol{x} \in [0,1]^d$ such that

$$\sup_{\boldsymbol{x}\in[0,1]^d} |\phi(\boldsymbol{x})| \le 6\sqrt{d}\big(\lceil n^{d/2}\rceil + 1\big)!. \tag{3.6}$$

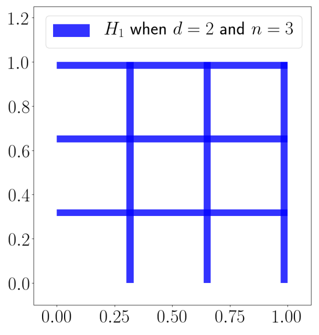
Define $H_1 = \cup_{j=1}^d \big\{\boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T \in [0,1]^d : x_j \in \cup_{i=1}^n [\frac{i}{n} - \delta, \frac{i}{n}]\big\}$, which separates the d-dimensional cube into n^d important sub-cubes as illustrated in Figure 2. To index these d-dimensional smaller sub-cubes, define $Q_{\boldsymbol{\theta}} = \big\{\boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T \in [0,1]^d : x_i \in [\frac{\theta_i}{n}, \frac{\theta_i+1}{n} - \delta],\ i = 1, 2, \cdots, d\big\}$ for each d-dimensional index $\boldsymbol{\theta} = [\theta_1, \theta_2, \cdots, \theta_d]^T \in \{0, 1, \cdots, n-1\}^d$. By (3.3), (3.4), and the definition of ψ_0, for any $\boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T \in Q_{\boldsymbol{\theta}}$, we have $\phi(\boldsymbol{x}) = \bar{\phi}(\psi(\boldsymbol{x})) = \bar{\phi}\Big(\sum_{i=1}^d \frac{1}{n^i}\psi_0(x_i)\Big) = \bar{\phi}\Big(\sum_{i=1}^d \frac{1}{n^i}\theta_i\Big) = \bar{f}\left(\frac{\boldsymbol{\theta}}{n}\right)$. Then

$$|\bar{f}(\boldsymbol{x}) - \phi(\boldsymbol{x})| = |\bar{f}(\boldsymbol{x}) - \bar{f}\left(\tfrac{\boldsymbol{\theta}}{n}\right)| \le (\sqrt{d}/n)^\alpha, \quad \text{for any } \boldsymbol{x} \in Q_{\boldsymbol{\theta}}. \tag{3.7}$$

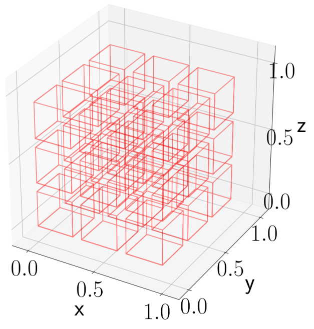
Because $\mu(H_1) \le dn\delta$, $[0,1]^d = \cup_{\boldsymbol{\theta}\in\{0,1,\cdots,n-1\}^d} Q_{\boldsymbol{\theta}} \cup H_1$, (3.6), and (3.7), we have

$$\begin{aligned}
\|\bar{f} - \phi\|_{L^1([0,1]^d)} &= \int_{H_1} |\bar{f} - \phi| d\boldsymbol{x} + \int_{[0,1]^d\setminus H_1} |\bar{f} - \phi| d\boldsymbol{x} \\
&\le \mu(H_1)\Big(2\sqrt{d} + 6\sqrt{d}\big(\lceil n^{d/2}\rceil + 1\big)!\Big) + \sum_{\boldsymbol{\theta}\in\{0,1,\cdots,n-1\}^d} \int_{Q_{\boldsymbol{\theta}}} |\bar{f} - \phi| d\boldsymbol{x} \\
&\le 2n\delta d\sqrt{d}\Big(1 + 3\big(\lceil n^{d/2}\rceil + 1\big)!\Big) + \sum_{\boldsymbol{\theta}\in\{0,1,\cdots,n-1\}^d} (\sqrt{d}/n)^\alpha \mu(Q_{\boldsymbol{\theta}}) \\
&\le 2d^{\alpha/2} n^{-\alpha},
\end{aligned}$$

where the last inequality comes from the fact that δ is small enough such that



(a)



(b)

FIG. 2. *An illustration of H_1 and n^d small non-overlapping sub-cubes that H_1 separates when $n = 3$. (a) When $d = 2$, H_1 in blue separates $[0,1]^2$ into $n^d = 9$ sub-cubes. (b) When $d = 3$, H_1 (no color) separates $[0,1]^3$ into $n^d = 27$ sub-cubes in red.*

$$2n\delta d\sqrt{d}\Big(1 + 3\big(\lceil n^{d/2}\rceil + 1\big)!\Big) \le d^{\alpha/2} n^{-\alpha}. \tag{3.8}$$

Note that $f - (\phi - \sqrt{d}) = \bar{f} - \phi$. Hence, $\phi - \sqrt{d} \in \mathrm{NN}(\#\text{input} = d;\ \text{widthvec} = [2nd, 2\lceil n^{d/2}\rceil, 2\lceil n^{d/2}\rceil + 1])$ and $\|f - (\phi - \sqrt{d})\|_{L^1([0,1]^d)} \le 2d^{\alpha/2} n^{-\alpha}$. Since $n = \lfloor N^{2/d}\rfloor$, we have $\lceil n^{d/2}\rceil \le N + 1$. Therefore,

$$\phi - \sqrt{d} \in \mathrm{NN}(\#\text{input} = d;\ \text{widthvec} = [2d\lfloor N^{2/d}\rfloor, 2N + 2, 2N + 3])$$

and

$$\|f-(\phi-\sqrt{d})\|_{L^1([0,1]^d)} \le 2d^{\alpha/2}n^{-\alpha} = 2d^{\alpha/2}\lfloor N^{2/d}\rfloor^{-\alpha} \le 2d^{\alpha/2}(N^{2/d}/2)^{-\alpha} = 2(2\sqrt{d})^{\alpha}N^{-2\alpha/d},$$

where the second inequality comes from the fact $\lfloor x\rfloor \ge \frac{x}{2}$ for $x \in [1,\infty)$. So, we finish the proof when $d>1$. □

Theorem 3.1 shows that for $f \in \mathrm{Lip}(\nu,\alpha,d)$ ReLU FNNs with two or three function compositions can achieve the approximation rate $\mathcal{O}(\nu N^{-2\alpha/d})$. Following the same proof as in Theorem 3.1, we can show that:

Corollary 3.2. *$\forall\ m,n \in \mathbb{N}^+$, the closure of* $\mathrm{NN}(\#\text{input}=1;\ \text{widthvec}=[2m,2n+1])$ *contains* $\mathrm{CPL}(mn+1)$ *in the sense of L^1-norm.*

An immediate implication of Corollary 3.2 is that, for any function f on $[0,1]$, if f can be approximated via one-hidden-layer ReLU FNNs with an approximation rate $\mathcal{O}(N^{-\eta})$ for any $\eta>0$, the rate can be improved to $\mathcal{O}(N^{-2\eta})$ via one more composition.

3.2. Asymptotic Unachievable Approximation Rate. In Section 3.1, we have analyzed the approximation capacity of ReLU FNNs in the nonlinear approximation for general continuous functions by construction. In this section, we will show that the construction in Section 3.1 is essentially and asymptotically tight via showing the approximation lower bound in Theorem 3.3 below.

Theorem 3.3. *For any $L \in \mathbb{N}^+$, $\rho>0$, and $C>0$, $\exists\ f \in \mathrm{Lip}(\nu,\alpha,d)$ with $\alpha\in(0,1]$, for all $N_0>0$, there exists $N \ge N_0$ such that*

$$\inf_{\phi\in\mathrm{NN}(\#\text{input}=d;\ \text{maxwidth}\le N;\ \#\text{layer}\le L)} \|\phi(\boldsymbol{x})-f(\boldsymbol{x})\|_{L^\infty([0,1]^d)} \ge C\nu N^{-(2\alpha/d+\rho)}.$$

The proof of Theorem 3.3 relies on the nearly-tight VC-dimension bounds of ReLU FNNs given in [23] and is similar to Theorem 4 of [59]. Hence, we only sketch out its proof and a complete proof can be found in [62].

Proof. We will prove this theorem by contradiction. Assuming that Theorem 3.3 is not true, we can show the following claim, which will lead to a contradiction in the end.

Claim 3.4. *There exist $L\in\mathbb{N}^+$, $\rho>0$, and $C>0$, $\forall\ f\in\mathrm{Lip}(\nu,\alpha,d)$ with $\alpha\in(0,1]$, then $\exists\ N_0>0$, for all $N\ge N_0$, there exists $\phi\in\mathrm{NN}(\#\text{input}=d;\ \text{maxwidth}\le N;\ \#\text{layer}\le L)$ such that*

$$\|f-\phi\|_{L^\infty([0,1]^d)} \le C\nu N^{-(2\alpha/d+\rho)}.$$

If this claim is true, then we have a better approximation rate. So we need to disproof this claim in order to prove Theorem 3.3.

Without loss of generality, we assume $\nu=1$; in the case of $\nu\ne1$, the proof is similar by rescaling $f\in\mathrm{Lip}(\nu,\alpha,d)$ and FNNs with ν. Let us denote the VC dimension of a function set $\mathcal{F}$ by $\mathrm{VCDim}(\mathcal{F})$. By [23], there exists $C_1>0$ such that

$$\mathrm{VCDim}\big(\mathrm{NN}(\#\text{input}=d;\ \#\text{parameter}\le W;\ \#\text{layer}\le L)\big) \le C_1 W L \ln W.$$

By Equation (3), we have

$$\begin{aligned}
&\mathrm{VCDim}\big(\mathrm{NN}(\#\text{input}=d;\ \text{maxwidth}\le N;\ \#\text{layer}\le L)\big)\\
&\le \mathrm{VCDim}\left(\mathrm{NN}(\#\text{input}=d;\ \#\text{parameter}\le (LN+d+2)(N+1);\ \#\text{layer}\le L)\right)\\
&\le C_1\big((LN+d+2)(N+1)\big)L\ln\big((LN+d+2)(N+1)\big).
\end{aligned}$$

One can estimate a lower bound of

$$\mathrm{VCDim}\big(\mathrm{NN}(\#\text{input}=d;\ \text{maxwidth}\le N;\ \#\text{layer}\le L)\big)$$

using Claim 3.4, and this lower bound can be $b_\ell \coloneqq \lfloor N^{2/d+\rho/(2\alpha)} \rfloor^d$, which is asymptotically larger than

$$b_u \coloneqq C_1\big((LN+d+2)(N+1)\big)L\ln\big((LN+d+2)(N+1)\big) = \mathcal{O}(N^2\ln N)$$

in (3), leading to a contradiction that disproves the assumption that "Theorem 3.3 is not true". □

Theorem 3.1 shows that the N-term approximation rate via two or three-hidden-layer ReLU FNNs can achieve $\mathcal{O}(N^{-2\alpha/d})$, while Theorem 3.3 shows that the rate cannot be improved to $\mathcal{O}(N^{-(2\alpha/d+\rho)})$ for any $\rho > 0$. It was conjectured in the literature that function compositions can improve the approximation capacity exponentially. For general continuous functions, Theorem 3.3 shows that this conjecture is not true, i.e., if the depth of the composition is $L = \mathcal{O}(1)$, the approximation rate cannot be better than $\mathcal{O}(N^{-2\alpha/d})$, not to mention $\mathcal{O}(N^{-L\alpha/d})$, which implies that adding one more layer cannot improve the approximation rate when N is large and $L > 2$.

Following the same proof as in Theorem 3.3, we have the following corollary, which shows that the result in Corollary 3.2 cannot be improved.

Corollary 3.5. *$\forall\ \rho > 0$, $C > 0$ and $L \in \mathbb{N}^+$, $\exists\ N_0(\rho, C, L)$ such that for any integer $N \ge N_0$, $\mathrm{CPL}(CN^{2+\rho})$ is not contained in the closure of $\mathrm{NN}(\#\text{input} = 1;\ \text{maxwidth} \le N;\ \#\text{layer} \le L)$ in the sense of L^∞-norm.*

3.3. Approximation and Computation Efficiency in Parallel Computing. In this section, we will discuss the efficiency of the N-term approximation via ReLU FNNs in parallel computing. This is of more practical interest than the optimal approximation rate purely based on the number of parameters of the nonlinear approximation, since it is impractical to use FNNs without parallel computing in real applications. Without loss of generality, we assume $\nu = 1$, $N \gg 1$, and $d \gg 1$.

Let us summarize standard statistics of the time and memory complexity in parallel computing [34] in one training iteration of ReLU FNNs with $\mathcal{O}(N)$ width and $\mathcal{O}(L)$ depth using m computing cores and $\mathcal{O}(1)$ training data samples per iteration. Let $T_s(N,L,m)$ and $T_d(N,L,m)$ denote the time complexity in shared and distributed memory parallel computing, respectively. Similarly, $M_s(N,L,m)$ and $M_d(N,L,m)$ are the memory complexity for shared and distributed memory, respectively. $M_s(N,L,m)$ is the total memory requirement; while $M_d(N,L,m)$ is the memory requirement per computing core. Then

$$T_s(N,L,m) = \begin{cases} \mathcal{O}\big(L(N^2/m + \ln\frac{m}{N})\big), & m \in [1, N^2], \\ \mathcal{O}(L\ln N), & m \in (N^2, \infty); \end{cases} \tag{3.9}$$

$$T_d(N,L,m) = \begin{cases} \mathcal{O}\big(L(N^2/m + t_s\ln m + \frac{t_w N}{\sqrt{m}}\ln m)\big), & m \in [1, N^2], \\ \mathcal{O}(L\ln N), & m \in (N^2, \infty); \end{cases} \tag{3.10}$$

$$M_s(N,L,m) = \mathcal{O}(LN^2), \quad \text{for all } m \in \mathbb{N}^+; \tag{3.11}$$

and

$$M_d(N,L,m) = \mathcal{O}(LN^2/m + 1), \quad \text{for all } m \in \mathbb{N}^+, \tag{3.12}$$

where t_s and t_w are the "start-up time" and "per-word transfer time" in the data communication between different computing cores, respectively (see [34] for a detailed introduction).

In real applications, a most frequently asked question would be: given a target function $f \in \mathrm{Lip}(\nu,\alpha,d)$, a target approximation accuracy ε, and a certain amount of computational resources, e.g., m computer processors, assuming the computer memory is enough, what is a good choice of FNN architecture we should use to reduce the running time of our computers? Certainly, the answer depends on the number of processors m and ideally we hope to increase m by a factor of r to reduce the time (and memory in the distributed environment) complexity by the same factor r, which is the scalability of parallel computing.

We answer the question raised just above using FNN architectures that almost have a uniform width, since the optimal approximation theory of very deep FNNs [60] and this manuscript both utilize a nearly uniform width. Combining the theory in [60] and ours, we summarize several statistics of ReLU FNNs in parallel computing in Table 1 and 2 when FNNs nearly have the same approximation accuracy. For shared memory parallel computing, from Table 1 we see that: if computing resources are enough, shallower FNNs with $\mathcal{O}(1)$ hidden layers require less and even exponentially less running time than very deep FNNs; if computing resources are limited, shallower FNNs might not be applicable or are slower, and hence very deep FNNs are a good choice. For distributed memory parallel computing, the conclusion is almost the same by Table 2, except that the memory limitation is not an issue for shallower FNNs if the number of processors is large enough. In sum, if the approximation rate of very deep FNNs is not exponentially better than shallower FNNs, very deep FNNs are less efficient than shallower FNNs theoretically if computing resources are enough.

Table 1
The comparison of approximation and computation efficiency of different ReLU FNNs in shared memory parallel computing with m processors when FNNs nearly have the same approximation accuracy. The analysis is asymptotic in d and N and is optimal up to a log factor; "running time" in this table is the time spent on each training step with $\mathcal{O}(1)$ training samples.

	NN(width $= [2d\lfloor N^{2/d}\rfloor, 2N, 2N]$)	NN(width $= [N]^L$)	NN(width $= [2d+10]^N$)
accuracy ε	$\mathcal{O}(\sqrt{d}N^{-2\alpha/d})$	$\mathcal{O}(N^{-2\alpha/d})$	$\mathcal{O}(C(d)N^{-2\alpha/d})$
number of weights	$\mathcal{O}(N^2)$	$\mathcal{O}(LN^2)$	$\mathcal{O}(d^2N)$
number of nodes	$\mathcal{O}(N)$	$\mathcal{O}(LN)$	$\mathcal{O}(dN)$
running time for $m \in [1,(2d+10)^2]$	$\mathcal{O}(N^2/m)$	$\mathcal{O}(LN^2/m)$	$\mathcal{O}(N(d^2/m + \ln\frac{m}{d}))$
running time for $m \in ((2d+10)^2, N^2]$	$\mathcal{O}(N^2/m + \ln\frac{m}{N})$	$\mathcal{O}(L(N^2/m + \ln\frac{m}{N}))$	$\mathcal{O}(N\ln d)$
running time for $m \in (N^2,\infty)$	$\mathcal{O}(\ln N)$	$\mathcal{O}(L\ln N)$	$\mathcal{O}(N\ln d)$
total memory	$\mathcal{O}(N^2)$	$\mathcal{O}(LN^2)$	$\mathcal{O}(d^2N)$

Table 2
The comparison of approximation and computation efficiency of different ReLU FNNs in distributed memory parallel computing with m processors when FNNs nearly have the same approximation accuracy. The analysis is asymptotic in d and N and is optimal up to a log factor; "running time" in this table is the time spent on each training step with $\mathcal{O}(1)$ training samples.

	NN(width $= [2d\lfloor N^{2/d}\rfloor, 2N, 2N]$)	NN(width $= [N]^L$)	NN(width $= [2d+10]^N$)
accuracy ε	$\mathcal{O}(\sqrt{d}N^{-2\alpha/d})$	$\mathcal{O}(N^{-2\alpha/d})$	$\mathcal{O}(C(d)N^{-2\alpha/d})$
number of weights	$\mathcal{O}(N^2)$	$\mathcal{O}(LN^2)$	$\mathcal{O}(d^2N)$
number of nodes	$\mathcal{O}(N)$	$\mathcal{O}(LN)$	$\mathcal{O}(dN)$
running time for $m \in [1,(2d+10)^2]$	$\mathcal{O}(N^2/m + t_s\ln m + \frac{t_w N}{\sqrt{m}}\ln m)$	$\mathcal{O}(L(N^2/m + t_s\ln m + \frac{t_w N}{\sqrt{m}}\ln m))$	$\mathcal{O}(N(d^2/m + t_s\ln m + \frac{t_w d}{\sqrt{m}}\ln m))$
running time for $m \in ((2d+10)^2, N^2]$	$\mathcal{O}(N^2/m + t_s\ln m + \frac{t_w N}{\sqrt{m}}\ln m)$	$\mathcal{O}(L(N^2/m + t_s\ln m + \frac{t_w N}{\sqrt{m}}\ln m))$	$\mathcal{O}(N\ln d)$
running time for $m \in (N^2,\infty)$	$\mathcal{O}(\ln N)$	$\mathcal{O}(L\ln N)$	$\mathcal{O}(N\ln d)$
memory per processor	$\mathcal{O}(N^2/m+1)$	$\mathcal{O}(LN^2/m+1)$	$\mathcal{O}(d^2N/m+1)$

4. Numerical Experiments.

In this section, we provide two sets of numerical experiments to compare different ReLU FNNs using shared memory GPU parallel computing. The numerical results for distributed memory parallel computing would be similar. All numerical tests were conducted using Tensorflow and an NVIDIA P6000 GPU with 3840 CUDA parallel-processing cores.

Since it is difficult to generate target functions $f \in \mathrm{Lip}(\nu,\alpha,d)$ with fixed ν and α, we cannot directly verify the nonlinear approximation rate, but we are able to observe numerical evidence close to our theoretical conclusions. Furthermore, we are able to verify the running time estimates in Section 3.3 and show that, to achieve the same theoretical approximation rate, shallow FNNs are more efficient than very deep FNNs.

In our numerical tests, we generate 50 random smooth functions as our target functions using the algorithm in [19] with a wavelength parameter $\lambda = 0.1$ and an amplitude parameter $\sqrt{(2/\lambda)}$ therein. These target functions are uniformly sampled with 20000 ordered points $\{x_i\}$ in $[0,1]$ to form a data set. The training data set consists of samples with odd indices i's, while the test data set consists of samples with even indices i's.

Table 3

Comparison between NN(#input = 1; widthvec = $[N]^L$) *and* NN(#input = 1; widthvec = $[12]^N$) *for* $N = 32, 64, 128$ *and* $L = 2, 4, 8$. *"Time" in this table is the total running time spent on* 20000 *training steps with training batch size* 10000*, and the unit is second(s).*

N	depth	width	test error	improvement ratio	#parameter	time
32	2	32	8.06×10^{-2}	–	1153	3.09×10^{1}
32	4	32	3.98×10^{-4}	–	3265	3.82×10^{1}
32	8	32	1.50×10^{-5}	–	7489	5.60×10^{1}
32	32	12	1.29×10^{-3}	–	4873	1.27×10^{2}
64	2	64	2.51×10^{-2}	3.21	4353	3.45×10^{1}
64	4	64	4.27×10^{-5}	9.32	12673	5.00×10^{1}
64	8	64	2.01×10^{-6}	7.46	29313	7.91×10^{1}
64	64	12	1.16×10^{-1}	0.01	9865	2.37×10^{2}
128	2	128	2.04×10^{-3}	12.3	16897	5.03×10^{1}
128	4	128	1.05×10^{-5}	4.07	49921	8.21×10^{1}
128	8	128	1.47×10^{-6}	1.37	115969	1.41×10^{2}
128	128	12	3.17×10^{-1}	0.37	19849	4.47×10^{2}

The loss function is defined as the mean square error between the target function and the FNN approximant evaluated on training sample points. The ADAM algorithm [33] with a decreasing learning rate from 0.005 to 0.0005, a batch size 10000, and a maximum number of epochs 20000, is applied to minimize the mean square error. The minimization is randomly initialized by the "normal initialization method"[⑤]. The test error is defined as the mean square error evaluated on test sample points. The training and test data sets are essentially the same in our numerical test since we aim at studying the approximation power of FNNs instead of the generalization capacity of FNNs. Note that due to the highly non-convexity of the optimization problem, there might be chances such that the minimizers we found are bad local minimizers. Hence, we compute the average test error of the best 40 tests among the total 50 tests of each architecture.

To observe numerical phenomena in terms of N-term nonlinear approximation, in the first set of numerical experiments, we use two types of FNNs to obtain approximants: the first type has $L = \mathcal{O}(1)$ layers with different sizes of width N; the second type has a fixed width $N = 12$ with different numbers of layers L. Numerical results are summarized in Table 3. To observe numerical phenomena in terms of the number of parameters in FNNs, in the second set of numerical experiments, we use FNNs with the same number of parameters but different sizes of width N and different numbers of layers L. Numerical results are summarized in Table 4.

By the last columns of Table 3, we verified that as long as the number of computing cores m is larger than or equal to N^2, the running time per iteration of FNNs with $\mathcal{O}(1)$ layers is $\mathcal{O}(\ln N)$, while the running time per iteration of FNNs with $\mathcal{O}(N)$ layers and $\mathcal{O}(1)$ width is $\mathcal{O}(N)$. By the last columns of Table 4, we see that when the number of parameters is the same, very deep FNNs requires much more running time per iteration than shallower FNNs and the difference becomes more significant when the number of parameters increases. Hence, very deep FNNs are much less efficient than shallower FNNs in parallel computing.

Besides, by Table 3 and 4, the test error of very deep FNNs cannot be improved if the depth is increased and the error even becomes larger when depth is larger. However, when the number of layers is fixed, increasing width can reduce the test error. More quantitatively, we define the *improvement ratio* of an FNN with width N and depth L in Table 3 as the ratio of the test error of an FNN in NN(#input = 1; widthvec = $[N/2]^L$) (or NN(#input = 1; widthvec = $[N]^{L/2}$)) over the test error of the current FNN in NN(#input = 1; widthvec = $[N]^L$). Similarly, the improvement ratio of an FNN with a number of parameters W in Table 4 is defined as

[⑤] See https://medium.com/prateekvishnu/xavier-and-he-normal-he-et-al-initialization-8e3d7a087528.

the ratio of the test error of an FNN with the same type of architecture and a number of parameters $W/2$ over the test error of the current FNN. According to the improvement ratio in Table 3 and 4, when $L = \mathcal{O}(1)$, the numerical approximation rate in terms of N is in a range between 2 to 4. Due to the highly non-convexity of the deep learning optimization and the difficulty to generate target functions of the same class with a fixed order α and constant ν, we cannot accurately verify the approximation rate. But the statistics of the improvement ratio can roughly reflect the approximation rate and the numerical results stand in line with our theoretical analysis.

Table 4
Comparison between shallow FNNs and deep FNNs when the total number of parameters (#parameter) is fixed. "Time" in this table is the total running time spent on 20000 *training steps with training batch size* 10000*, and the unit is second(s).*

#parameter	depth	width	test error	improvement ratio	time
5038	2	69	1.13×10^{-2}	–	3.84×10^{1}
5041	4	40	1.65×10^{-4}	–	3.80×10^{1}
4993	8	26	1.69×10^{-5}	–	5.07×10^{1}
5029	33	12	4.77×10^{-3}	–	1.28×10^{2}
9997	2	98	4.69×10^{-3}	2.41	4.40×10^{1}
10090	4	57	7.69×10^{-5}	2.14	4.67×10^{1}
9954	8	37	7.43×10^{-6}	2.27	5.92×10^{1}
10021	65	12	2.80×10^{-1}	0.02	2.31×10^{2}
19878	2	139	1.43×10^{-3}	3.28	5.18×10^{1}
20170	4	81	2.30×10^{-5}	3.34	6.26×10^{1}
20194	8	53	2.97×10^{-6}	2.50	7.08×10^{1}
20005	129	12	3.17×10^{-1}	0.88	4.30×10^{2}

5. Conclusions.

We study the approximation and computation efficiency of function compositions in nonlinear approximation, especially when the composition is implemented using ReLU FNNs in parallel computing. New analysis techniques have been proposed to quantified the advantages of function compositions in accelerating the approximation rate, and in achieving the optimal approximation rate with $\mathcal{O}(1)$ hidden layers for Hölder continuous functions. Moreover, for any function f on $[0,1]$, regardless of its smoothness and even the continuity, if f can be approximated via nonlinear approximation using one-hidden-layer ReLU FNNs with an approximation rate $\mathcal{O}(N^{-\eta})$ for any $\eta > 0$, we show that two-hidden-layer ReLU FNNs can improve the approximation rate to $\mathcal{O}(N^{-2\eta})$. Finally, considering the computational efficiency in parallel computing, FNNs with $\mathcal{O}(1)$ hidden layers are a better choice for approximating Hölder continuous functions if computing resources are enough. Our discussion provides a new point of view for the debate of "deep vs. shallow" in the literature of deep learning research. We would like to conclude our paper with a simple message that depth is good but a very deep ReLU FNN could be less efficient that a relative shallower FNN with $\mathcal{O}(1)$ hidden layers when we consider the approximation rate and parallel computing at the same time.

In the analysis of constructive approximations in Section 3.1, our results were based on the L^1-norm; while in the unachievable approximation rate in Section 3.2, we used L^∞-norm. In fact, the approximation error in Section 3.1 can be controlled uniformly well if we exclude a "don't-care" region independent of the target function with an arbitrarily small measure. This observation motivates us to define a new norm denoted as $L^{1,\infty}$-norm, which is weaker than the L^∞-norm and stronger than the L^1-norm, such that all theorems in this paper hold in the same $L^{1,\infty}$-norm.

In particular, we define a shrinking function $\lambda(x) \coloneqq \frac{2^{-x}}{x}$ for any $x \in [1,\infty)$, and correspondingly the shrinking region $\Omega(k,d)$ that gradually shrinks to a set of rational points in $[0,1]^d$ as k increases via

$$\Omega(k,d) = \cup_{\ell=k}^{\infty} \left(\cup_{i=1}^{d} \{ \boldsymbol{x} = [x_1, \cdots, x_d]^T \in \mathbb{R}^d : x_i \in \mathcal{S}(\ell) \} \right),$$

where $\mathcal{S}(\ell) = \cup_{j=1}^{\ell} [j/\ell - \lambda(\ell), j/\ell]$. Intuitively, the shrinking region $\Omega(k,d)$ can be understood as the "don't-care" region when k is sufficiently large. Finally, the $L^{1,\infty}$-norm can be defined based on the "don't-care" region via

$$\|f\|_{L^{1,\infty}([0,1]^d)} \coloneqq \sup \left\{ \tfrac{1}{\ln k} \|f\|_{L^\infty(\Omega(k,d)^c)} : k \ge 3,\, k \in \mathbb{N} \right\} + \|f\|_{L^1([0,1]^d)},$$

for any $f \in L^1([0,1]^d) \cap L^\infty([0,1]^d)$. Apparently, $\|\cdot\|_{L^{1,\infty}([0,1]^d)}$ is a norm satisfying

$$\|f\|_{L^1([0,1]^d)} \le \|f\|_{L^{1,\infty}([0,1]^d)} \le 2\|f\|_{L^\infty([0,1]^d)}, \quad \text{for any } f \in L^1([0,1]^d) \cap L^\infty([0,1]^d).$$

The analysis based on the $L^{1,\infty}$-norm can be found in [62]. It shows that Theorem 3.1 and 3.3 are still true in the sense of the $L^{1,\infty}$-norm. Hence, the analysis of the approximation rate in this paper is tight in the $L^{1,\infty}$-norm.

Acknowledgments. H. Yang thanks the Department of Mathematics at the National University of Singapore for the startup grant, the Ministry of Education in Singapore for the grant MOE2018-T2-2-147, NVIDIA Corporation for the donation of the Titan Xp GPU, and NSCC [1] for extra GPU recource. We would like to acknowledge the editor and reviewers for their help in improving this manuscript.

REFERENCES

[1] *The computational work for this article was partially performed on resources of the national supercomputing centre, singapore (https://www.nscc.sg).*

[2] M. Anthony and P. L. Bartlett, *Neural Network Learning: Theoretical Foundations*, Cambridge University Press, New York, NY, USA, 1st ed., 2009.

[3] A. R. Barron, *Universal approximation bounds for superpositions of a sigmoidal function*, IEEE Transactions on Information Theory, 39 (1993), pp. 930–945, https://doi.org/10.1109/18.256500.

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

[5] M. Bianchini and F. Scarselli, *On the complexity of neural network classifiers: A comparison between shallow and deep architectures*, IEEE Transactions on Neural Networks and Learning Systems, 25 (2014), pp. 1553–1565, https://doi.org/10.1109/TNNLS.2013.2293637.

[6] E. J. Candes and M. B. Wakin, *An introduction to compressive sampling*, IEEE Signal Processing Magazine, 25 (2008), pp. 21–30, https://doi.org/10.1109/MSP.2007.914731.

[7] S. Chen and D. Donoho, *Basis pursuit*, in Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers, vol. 1, Oct 1994, pp. 41–44 vol.1, https://doi.org/10.1109/ACSSC.1994.471413.

[8] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, *Flexible, high performance convolutional neural networks for image classification*, in Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two, IJCAI'11, AAAI Press, 2011, pp. 1237–1242, http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-210.

[9] D. Costarelli and A. R. Sambucini, *Saturation classes for max-product neural network operators activated by sigmoidal functions*, Results in Mathematics, 72 (2017), pp. 1555 – 1569, https://doi.org/10.1007/s00025-017-0692-6.

[10] D. Costarelli and G. Vinti, *Convergence for a family of neural network operators in orlicz spaces*, Mathematische Nachrichten, 290 (2017), pp. 226–235, https://onlinelibrary.wiley.com/doi/abs/10.1002/mana.201600006.

[11] D. Costarelli and G. Vinti, *Approximation results in orlicz spaces for sequences of kantorovich max-product neural network operators*, Results in Mathematics, 73 (2018), pp. 1 – 15, https://doi.org/10.1007/s00025-018-0799-4.

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

[13] I. Daubechies, *Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematics, 1992, https://epubs.siam.org/doi/abs/10.1137/1.9781611970104.

[14] G. Davis, *Adaptive nonlinear approximations*, 1994.

[15] R. DEVORE and A. RON, *Approximation using scattered shifts of a multivariate function*, Transactions of the American Mathematical Society, 362 (2010), pp. 6205–6229, http://www.jstor.org/stable/40997201.

[16] R. A. DeVore, *Nonlinear approximation*, Acta Numerica, 7 (1998), p. 51–150, https://doi.org/10.1017/S0962492900002816.

[17] D. L. Donoho, *Compressed sensing*, IEEE Transactions on Information Theory, 52 (2006), pp. 1289–1306, https://doi.org/10.1109/TIT.2006.871582.

[18] J. Duchi, E. Hazan, and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization*, J. Mach. Learn. Res., 12 (2011), pp. 2121–2159, http://dl.acm.org/citation.cfm?id=1953048.2021068.

[19] S.-I. Filip, A. Javeed, and L. N. Trefethen, *Smooth random functions, random ODEs, and Gaussian processes.* To appear in SIAM Review., Dec. 2018, https://hal.inria.fr/hal-01944992.

[20] K. Fukushima, *Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position*, Biological Cybernetics, 36 (1980), pp. 193–202, https://doi.org/10.1007/BF00344251.

[21] T. Hangelbroek and A. Ron, *Nonlinear approximation using gaussian kernels*, Journal of Functional Analysis, 259 (2010), pp. 203 – 219, http://www.sciencedirect.com/science/article/pii/S0022123610000467.

[22] B. Hanin and M. Sellke, *Approximating continuous functions by ReLU nets of minimal width*, (2017), https://arxiv.org/abs/1710.11278.

[23] N. Harvey, C. Liaw, and A. Mehrabian, *Nearly-tight VC-dimension bounds for piecewise linear neural networks*, in Proceedings of the 2017 Conference on Learning Theory, S. Kale and O. Shamir, eds., vol. 65 of Proceedings of Machine Learning Research, Amsterdam, Netherlands, 07–10 Jul 2017, PMLR, pp. 1064–1068, http://proceedings.mlr.press/v65/harvey17a.html.

[24] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016, pp. 770–778, https://doi.org/10.1109/CVPR.2016.90.

[25] K. Hornik, M. Stinchcombe, and H. White, *Multilayer feedforward networks are universal approximators*, Neural Networks, 2 (1989), pp. 359 – 366, http://www.sciencedirect.com/science/article/pii/0893608089900208.

[26] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, *Densely connected convolutional networks*, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2017), pp. 2261–2269.

[27] J. Jiang, *Design of neural networks for lossless data compression*, Optical Engineering, 35 (1996), pp. 35 – 35 – 7, https://doi.org/10.1117/1.600614.

[28] R. Johnson and T. Zhang, *Accelerating stochastic gradient descent using predictive variance reduction*, in Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1, NIPS'13, USA, 2013, Curran Associates Inc., pp. 315–323, http://dl.acm.org/citation.cfm?id=2999611.2999647.

[29] J. Joutsensalo, *Nonlinear data compression and representation by combining self-organizing map and subspace rule*, in Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94), vol. 2, June 1994, pp. 637–640 vol.2, https://doi.org/10.1109/ICNN.1994.374249.

[30] K. Kawaguchi, *Deep learning without poor local minima*, in Advances in Neural Information Processing Systems 29, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds., Curran Associates, Inc., 2016, pp. 586–594, http://papers.nips.cc/paper/6112-deep-learning-without-poor-local-minima.pdf.

[31] K. Kawaguchi and Y. Bengio, *Depth with nonlinearity creates no bad local minima in resnets*, (2018), https://arxiv.org/abs/1810.09038, https://arxiv.org/abs/1810.09038.

[32] M. J. Kearns and R. E. Schapire, *Efficient distribution-free learning of probabilistic concepts*, J. Comput. Syst. Sci., 48 (1994), pp. 464–497, http://dx.doi.org/10.1016/S0022-0000(05)80062-5.

[33] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, CoRR, abs/1412.6980 (2014), http://arxiv.org/abs/1412.6980, https://arxiv.org/abs/1412.6980.

[34] V. Kumar, *Introduction to Parallel Computing*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd ed., 2002.

[35] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, *Object recognition with gradient-based learning*, in Shape, Contour and Grouping in Computer Vision, London, UK, UK, 1999, Springer-Verlag, pp. 319–, http://dl.acm.org/citation.cfm?id=646469.691875.

[36] G. Lewicki and G. Marino, *Approximation of functions of finite variation by superpositions of a sigmoidal function*, Applied Mathematics Letters, 17 (2004), pp. 1147 – 1152, http://www.sciencedirect.com/science/article/pii/S089396590481694X.

[37] S. Liang and R. Srikant, *Why deep neural networks?*, CoRR, abs/1610.04161 (2016), http://arxiv.org/abs/1610.04161, https://arxiv.org/abs/1610.04161.

[38] S. Lin, X. Liu, Y. Rong, and Z. Xu, *Almost optimal estimates for approximation and learning by radial basis function networks*, Machine Learning, 95 (2014), pp. 147–164, https://doi.org/10.1007/s10994-013-5406-z, https://doi.org/10.1007/s10994-013-5406-z.

[39] B. Llanas and F. Sainz, *Constructive approximate interpolation by neural networks*, Journal of Computational and Applied Mathematics, 188 (2006), pp. 283 – 308, http://www.sciencedirect.com/science/article/pii/S0377042705002566.

[40] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, *The expressive power of neural networks: A view from the width*, vol. abs/1709.02540, 2017, http://arxiv.org/abs/1709.02540, https://arxiv.org/abs/1709.02540.

[41] S. G. Mallat and Z. Zhang, *Matching pursuits with time-frequency dictionaries*, IEEE Transactions on Signal Processing, 41 (1993), pp. 3397–3415, https://doi.org/10.1109/78.258082.

[42] H. Montanelli and Q. Du, *New error bounds for deep networks using sparse grids*, (2017), https://arxiv.org/abs/1712.08688.

[43] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, *On the number of linear regions of deep neural networks*, in Advances in Neural Information Processing Systems 27, Z. Ghahramani, M. Welling, C. Cortes, N. D.

Lawrence, and K. Q. Weinberger, eds., Curran Associates, Inc., 2014, pp. 2924–2932, http://papers.nips.cc/paper/5422-on-the-number-of-linear-regions-of-deep-neural-networks.pdf.

[44] Q. N. Nguyen and M. Hein, *The loss surface of deep and wide neural networks*, CoRR, abs/1704.08045 (2017), http://arxiv.org/abs/1704.08045, https://arxiv.org/abs/1704.08045.

[45] H. Ohlsson, A. Y. Yang, R. Dong, and S. S. Sastry, *Nonlinear basis pursuit*, in 2013 Asilomar Conference on Signals, Systems and Computers, Nov 2013, pp. 115–119, https://doi.org/10.1109/ACSSC.2013.6810285.

[46] P. Petersen and F. Voigtlaender, *Optimal approximation of piecewise smooth functions using deep ReLU neural networks*, Neural Networks, 108 (2018), pp. 296 – 330, http://www.sciencedirect.com/science/article/pii/S0893608018302454.

[47] P. Petrushev, *Multivariate n-term rational and piecewise polynomial approximation*, Journal of Approximation Theory, 121 (2003), pp. 158 – 197, https://doi.org/https://doi.org/10.1016/S0021-9045(02)00060-6, http://www.sciencedirect.com/science/article/pii/S0021904502000606.

[48] D. Rolnick and M. Tegmark, *The power of deeper networks for expressing natural functions*, CoRR, abs/1705.05502 (2017), http://arxiv.org/abs/1705.05502, https://arxiv.org/abs/1705.05502.

[49] D. Rumelhart, J. McClelland, P. R. Group, and S. D. P. R. G. University of California, *Psychological and Biological Models*, no. v. 2 in A Bradford Book, MIT Press, 1986, https://books.google.com.sg/books?id=davmLgzusB8C.

[50] I. Safran and O. Shamir, *Depth-width tradeoffs in approximating natural functions with neural networks*, in Proceedings of the 34th International Conference on Machine Learning, D. Precup and Y. W. Teh, eds., vol. 70 of Proceedings of Machine Learning Research, International Convention Centre, Sydney, Australia, 06–11 Aug 2017, PMLR, pp. 2979–2987, http://proceedings.mlr.press/v70/safran17a.html.

[51] A. Sakurai, *Tight bounds for the VC-dimension of piecewise polynomial networks*, in Advances in Neural Information Processing Systems, Neural information processing systems foundation, 1999, pp. 323–329.

[52] D. Scherer, A. Müller, and S. Behnke, *Evaluation of pooling operations in convolutional architectures for object recognition*, in Artificial Neural Networks – ICANN 2010, K. Diamantaras, W. Duch, and L. S. Iliadis, eds., Berlin, Heidelberg, 2010, Springer Berlin Heidelberg, pp. 92–101.

[53] J. Schmidt-Hieber, *Nonparametric regression using deep neural networks with ReLU activation function*, (2017), https://arxiv.org/abs/1708.06633.

[54] T. Suzuki, *Adaptivity of deep reLU network for learning in besov and mixed smooth besov spaces: optimal rate and curse of dimensionality*, in International Conference on Learning Representations, 2019, https://openreview.net/forum?id=H1ebTsActm.

[55] S. Tariyal, A. Majumdar, R. Singh, and M. Vatsa, *Greedy deep dictionary learning*, CoRR, abs/1602.00203 (2016), http://arxiv.org/abs/1602.00203, https://arxiv.org/abs/1602.00203.

[56] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Harvard University, 1975, https://books.google.com.sg/books?id=z81XmgEACAAJ.

[57] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, *Aggregated residual transformations for deep neural networks*, CoRR, abs/1611.05431 (2016), http://arxiv.org/abs/1611.05431, https://arxiv.org/abs/1611.05431.

[58] T. F. Xie and F. L. Cao, *The rate of approximation of gaussian radial basis neural networks in continuous function space*, Acta Mathematica Sinica, English Series, 29 (2013), pp. 295–302, https://doi.org/10.1007/s10114-012-1369-4.

[59] D. Yarotsky, *Error bounds for approximations with deep ReLU networks*, Neural Networks, 94 (2017), pp. 103 – 114, http://www.sciencedirect.com/science/article/pii/S0893608017301545.

[60] D. Yarotsky, *Optimal approximation of continuous functions by very deep ReLU networks*, in Proceedings of the 31st Conference On Learning Theory, S. Bubeck, V. Perchet, and P. Rigollet, eds., vol. 75 of Proceedings of Machine Learning Research, PMLR, 06–09 Jul 2018, pp. 639–649, http://proceedings.mlr.press/v75/yarotsky18a.html.

[61] S. Zagoruyko and N. Komodakis, *Wide residual networks*, CoRR, abs/1605.07146 (2016).

[62] S. Zhang, *Approximation theories for deep neural networks via function compositions*, Ph.D. Thesis submitted to the National University of Singapore.