Neural Networks 119 (2019) 74–84

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Nonlinear approximation via compositions

Zuowei Shen, Haizhao Yang [*,1], Shijun Zhang

Department of Mathematics, National University of Singapore, Singapore

A R T I C L E I N F O

Article history:
Received 19 February 2019
Received in revised form 14 May 2019
Accepted 10 July 2019
Available online 2 August 2019

Keywords:
Deep neural networks
ReLU activation function
Nonlinear approximation
Function composition
Hölder continuity
Parallel computing

A B S T R A C T

Given a function dictionary $\mathcal{D}$ and an approximation budget $N \in \mathbb{N}$, nonlinear approximation seeks the linear combination of the best N terms $\{T_n\}_{1\le n\le N} \subseteq \mathcal{D}$ to approximate a given function f with the minimum approximation error

$$\varepsilon_{L,f} := \min_{\{g_n\}\subseteq\mathbb{R},\{T_n\}\subseteq\mathcal{D}} \|f(\boldsymbol{x}) - \sum_{n=1}^{N} g_n T_n(\boldsymbol{x})\|.$$

Motivated by recent success of deep learning, we propose dictionaries with functions in a form of compositions, i.e.,

$$T(\boldsymbol{x}) = T^{(L)} \circ T^{(L-1)} \circ \cdots \circ T^{(1)}(\boldsymbol{x})$$

for all $T \in \mathcal{D}$, and implement T using ReLU feed-forward neural networks (FNNs) with L hidden layers. We further quantify the improvement of the best N-term approximation rate in terms of N when L is increased from 1 to 2 or 3 to show the power of compositions. In the case when $L > 3$, our analysis shows that increasing L cannot improve the approximation rate in terms of N.

In particular, for any function f on $[0, 1]$, regardless of its smoothness and even the continuity, if f can be approximated using a dictionary when $L = 1$ with the best N-term approximation rate $\varepsilon_{L,f} = \mathcal{O}(N^{-\eta})$, we show that dictionaries with $L = 2$ can improve the best N-term approximation rate to $\varepsilon_{L,f} = \mathcal{O}(N^{-2\eta})$. We also show that for Hölder continuous functions of order α on $[0, 1]^d$, the application of a dictionary with $L = 3$ in nonlinear approximation can achieve an essentially tight best N-term approximation rate $\varepsilon_{L,f} = \mathcal{O}(N^{-2\alpha/d})$. Finally, we show that dictionaries consisting of wide FNNs with a few hidden layers are more attractive in terms of computational efficiency than dictionaries with narrow and very deep FNNs for approximating Hölder continuous functions if the number of computer cores is larger than N in parallel computing.

1. Introduction

For non-smooth and high-dimensional function approximation, a favorable technique popularized in recent decades is the nonlinear approximation (DeVore, 1998) 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 (Daubechies, 1992), dictionary learning (Tariyal, Majumdar, Singh, & Vatsa, 2016), data compression and denoising (Jiang, 1996; Joutsensalo, 1994), adaptive pursuit (Davis, 1994; Ohlsson, Yang, Dong, & Sastry, 2013), compressed sensing (Candes & Wakin, 2008; Donoho, 2006)).

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 nth coordinate being T_n, and g is a linear map from $\mathbb{R}^N$ to $\mathbb{R}$ with the nth coordinate being $g_n \in \mathbb{R}$. The nonlinear approximation seeks g and T such that

$$\{\{T_n\}, \{g_n\}\} = \arg\min_{\{g_n\}\subseteq\mathbb{R},\{T_n\}\subseteq\mathcal{D}} \|f(\boldsymbol{x}) - \sum_{n=1}^{N} g_n T_n(\boldsymbol{x})\|_*, \tag{1.2}$$

[*] Corresponding author.
E-mail addresses: matzuows@nus.edu.sg (Z. Shen), haizhao@nus.edu.sg (H. Yang), zhangshijun@u.nus.edu (S. Zhang).
[1] H. Yang was supported by the startup of the Department of Mathematics at the National University of Singapore.

https://doi.org/10.1016/j.neunet.2019.07.011
0893-6080/© 2019 Elsevier Ltd. All rights reserved.

which is also called the best 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 (Chen & Donoho, 1994; Mallat & Zhang, 1993), adaptive splines (DeVore, 1998; Petrushev, 2003), radial basis functions (Devore & Ron, 2010; Hangelbroek & Ron, 2010; Xie & Cao, 2013), sigmoidal neural networks (Barron, 1993; Costarelli & Sambucini, 2017; Costarelli & Vinti, 2017, 2018; Cybenko, 1989; Hornik, Stinchcombe, & White, 1989; Lewicki & Marino, 2004; Llanas & Sainz, 2006), etc. For functions in Besov spaces with smoothness s, Devore and Ron (2010) and Hangelbroek and Ron (2010) constructed an $\mathcal{O}(N^{-s/d})$[2] approximation that is almost optimal (Lin, Liu, Rong, & Xu, 2014) and the smoothness cannot be reduced generally (Hangelbroek & Ron, 2010). For Hölder continuous functions of order 1 on $[0,1]^d$, Xie and Cao (2013) 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

ReLU FNNs have been proved to be a powerful tool in many fields from various points of view (Anthony & Bartlett, 2009; Bartlett, Maiorov, & Meir, 1998; Bianchini & Scarselli, 2014; Harvey, Liaw, & Mehrabian, 2017; Kearns & Schapire, 1994; Montufar, Pascanu, Cho, & Bengio, 2014; Petersen & Voigtlaender, 2018; Sakurai, 1999), which motivates us to tackle the open problem above via function compositions in the nonlinear approximation using deep ReLU FNNs, i.e.,

$$f(\boldsymbol{x}) \approx g \circ T^{(L)} \circ T^{(L-1)} \circ \cdots \circ T^{(1)}(\boldsymbol{x}), \tag{1.3}$$

where $T^{(i)}(\boldsymbol{x}) = \sigma(\boldsymbol{W}_i\boldsymbol{x} + \boldsymbol{b}_i)$ with $\boldsymbol{W}_i \in \mathbb{R}^{N_i \times N_{i-1}}$, $\boldsymbol{b}_i \in \mathbb{R}^{N_i}$ for $i = 1, \ldots, L$, σ is the ReLU activation function, and f is a Hölder continuous function. For the convenience of analysis, we consider $N_i = N$ for $i = 1, \ldots, L$. Let $\mathcal{D}_L$ be the dictionary consisting of ReLU FNNs $g \circ T^{(L)} \circ T^{(L-1)} \circ \cdots \circ T^{(1)}(\boldsymbol{x})$ with width N and depth L. To identify the optimal FNN to approximate $f(\boldsymbol{x})$, it is sufficient to solve the following optimization problem

$$\phi^* = \arg\min_{\phi \in \mathcal{D}_L} \|f - \phi\|_*. \tag{1.4}$$

The fundamental limit of nonlinear approximation via the proposed dictionary is essentially determined by the approximation power of function compositions in (1.3), which gives a performance guarantee of the minimizer in (1.4). Since function compositions are implemented via ReLU FNNs, the remaining problem is to quantify the approximation capacity of deep ReLU FNNs, especially their ability to improve the best N-term approximation rate in N for any fixed L defined as

$$\varepsilon_{L,f}(N) = \min_{\phi \in \mathcal{D}_L} \|f - \phi\|_*. \tag{1.5}$$

[2] 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.

Function compositions can significantly enrich the dictionary of nonlinear approximation and this idea was not considered in the literature previously due to the expensive computation of function compositions in solving the minimization problem in (1.4). Fortunately, recent development of efficient algorithms for optimization with compositions (e.g., backpropagation techniques (Fukushima, 1980; Rumelhart, McClelland, Group, & University of California, 1986; Werbos, 1975) and parallel computing techniques (Cireşan, Meier, Masci, Gambardella, & Schmidhuber, 2011; Scherer, Müller, & Behnke, 2010)) makes it possible to explore the proposed dictionary in this paper. Furthermore, with advanced optimization algorithms (Duchi, Hazan, & Singer, 2011; Johnson & Zhang, 2013; Kingma & Ba, 2014), good local minima of (1.4) can be identified efficiently (Kawaguchi, 2016; Kawaguchi & Bengio, 2018; Nguyen & Hein, 2017).

1.2. Related work and contribution

The main goal in the remaining article is to quantify the best N-term approximation rate $\varepsilon_{L,f}(N)$ defined in (1.5) for ReLU FNNs in the dictionary $\mathcal{D}_L$ with a fixed depth L when f is a Hölder continuous function. This topic is related to several existing approximation theories in the literature, but none of these existing works can be applied to answer the problem addressed in this paper.

First of all, this paper identifies explicit formulas for the best N-term approximation rate

$$\varepsilon_{L,f}(N) \le \begin{cases} 2\nu N^{-2\alpha}, & \text{when } L \ge 2 \text{ and } d = 1, \\ 2(2\sqrt{d})^{\alpha}\nu N^{-2\alpha/d}, & \text{when } L \ge 3 \text{ and } d > 1, \end{cases} \tag{1.6}$$

for any $N \in \mathbb{N}^+$ and Hölder continuous function f of order α with a constant ν, while existing theories (Liang & Srikant, 2016; Lu, Pu, Wang, Hu, & Wang, 2017; Montanelli & Du, 2017; Montanelli, Yang, & Du, 2019; Petersen & Voigtlaender, 2018; Suzuki, 2019; Weinan & Wang, 2018; Yarotsky, 2017, 2018) can only provide implicit formulas in the sense that the approximation error contains an unknown prefactor and work only for sufficiently large N or L larger than some unknown numbers. For example, the approximation rate in Yarotsky (2018) via a narrow and deep ReLU FNN is $c(d)L^{-2\alpha/d}$ with $c(d)$ unknown and for L larger than a sufficiently large unknown number $\mathcal{L}$; the approximation rate in Yarotsky (2018) via a wide and shallow ReLU FNN is $c(d)N^{-\alpha/d}$ with $c(d)$ unknown and for N larger than a sufficiently large unknown number $\mathcal{N}$. For another example, given an approximation error ε, Petersen and Voigtlaender (2018) proved the existence of a ReLU FNN with a constant but still unknown number of layers approximating a C^{β} function within the target error. Similarly, given the ε error, Montanelli and Du (2017), Montanelli et al. (2019) and Weinan and Wang (2018) estimate the scaling of the network size in ε and the scaling contains unknown prefactors. Given an arbitrary L and N, no existing work can provide an explicit formula for the approximation error to guide practical network design, e.g., to guarantee whether the network is large enough to meet the accuracy requirement. This paper provides such formulas for the first time and in fact the bound in these formulas is asymptotically tight as we shall prove later.

Second, our target functions are Hölder continuous, while most of existing works aim for a smaller function space with certain smoothness, e.g. functions in $C^{\alpha}([0,1]^d)$ with $\alpha \ge 1$ (Liang & Srikant, 2016; Lu et al., 2017; Weinan & Wang, 2018; Yarotsky, 2017), band-limited functions (Montanelli et al., 2019), Korobov spaces (Montanelli & Du, 2017), or Besov spaces (Suzuki, 2019). To the best of our knowledge, there is only one existing article (Yarotsky, 2018) concerning the approximation power of deep ReLU FNNs for $C([0,1]^d)$. However, the conclusion of Yarotsky (2018) only works for ReLU FNNs with a fixed width $2d + 10$

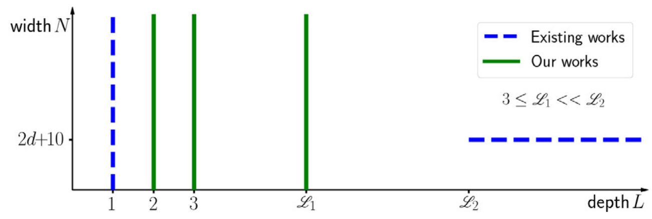

Fig. 1. A comparison of existing works and our contribution on the approximation power of ReLU FNNs for Hölder continuous functions of order α. Existing results in two cases: (1) $\mathcal{O}(N^{-\alpha/d})$ approximation rate for ReLU FNNs with depth $L = 1$ and width N; (2) $\mathcal{O}(L^{-2\alpha/d})$ approximation rate for ReLU FNNs with depth L larger than a sufficiently large unknown number $\mathcal{L}_2$ and width $2d + 10$. Our contribution: $\mathcal{O}(N^{-2\alpha/d})$ approximation rate for ReLU FNNs width depth $L \ge 3$ and width N in the case of $d > 1$ ($L \ge 2$ in the case of $d = 1$).

and a sufficiently large L, instead of a fixed L and an arbitrary N as required in the nonlinear approximation (see Fig. 1 for the comparison of the conclusion of Yarotsky (2018) and this paper).

As we can see in Fig. 1, the improvement of the best N-term approximation rate in terms of N when L is increased from 1 to 2 or 3 is significant, which shows the power of depth in ReLU FNNs. However, in the case when $L > 3$, our analysis shows that increasing L cannot improve the approximation rate in terms of N. As an interesting corollary of our analysis, for any function f on [0, 1], regardless of its smoothness and even the continuity, if f can be approximated using functions in $\mathcal{D}_1$ with the best N-term approximation rate $\varepsilon_{L,f} = \mathcal{O}(N^{-\eta})$, we show that functions in $\mathcal{D}_2$ can improve the best N-term approximation rate to $\varepsilon_{L,f} = \mathcal{O}(N^{-2\eta})$. Extending this conclusion for a general d dimensional function is challenging and we leave it as future work.

From the point of view of analysis techniques, this paper introduces new analysis methods merely based on the structure of FNNs, while existing works (Liang & Srikant, 2016; Lu et al., 2017; Montanelli & Du, 2017; Montanelli et al., 2019; Petersen & Voigtlaender, 2018; Suzuki, 2019; Weinan & Wang, 2018; Yarotsky, 2017, 2018) rely on constructing FNNs to approximate traditional basis in approximation theory, e.g., polynomials, splines, and sparse grids, which are used to approximate smooth functions.

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 are 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 a depth that is asymptotically $\mathcal{O}(1)$ relative to the width.

1.3. 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 jth column of $\boldsymbol{A}$; $\boldsymbol{A}(i,:)$ is the ith 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 ith element of $\boldsymbol{v}$. $\boldsymbol{v} = [v_1, \ldots, v_n]^T = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}$ 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 $\lfloor \xi \rfloor := \max\{i : i \le \xi,\ i \in \mathbb{N}\}$ and $\lceil \xi \rceil := \min\{i : i \ge \xi,\ i \in \mathbb{N}\}$.
- 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 $f(\boldsymbol{n}) \le Cg(\boldsymbol{n})$ 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 endpoints of each linear piece are called "breakpoints" 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 $\{\mathrm{c}_i\}_{1\le i\le m}$, 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, NN(#input $= 2$; 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 $[n, n, \ldots, n] \in \mathbb{N}^L$. For example,

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

- For $\phi \in \mathrm{NN}(\#\mathrm{input} = d;\ \mathrm{widthvec} = [N_1, N_2, \ldots, N_L])$, 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 ith linear transform in ϕ, respectively, i.e., $\boldsymbol{h}_i := \boldsymbol{W}_i\tilde{\boldsymbol{h}}_{i-1} + \boldsymbol{b}_i$ for $i = 1, 2, \ldots, L+1$ and $\tilde{\boldsymbol{h}}_i = \sigma(\boldsymbol{h}_i)$ for $i = 1, 2, \ldots, L$.

2.2. Lemmas

Let us study the properties of ReLU FNNs with only one hidden layer to warm up in Lemma 2.1. 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 $\boldsymbol{W}_1 = [1, 1, \ldots, 1]^T \in \mathbb{R}^{N\times 1}$, 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 $[x_0, x_N]$.[3]

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

Part (1) in Lemma 2.1 follows by setting $\boldsymbol{b}_1 = [-x_0, -x_1, \ldots, -x_{N-1}]^T$. 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.

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 $i = 0, 1, \ldots, m(n+1)$, 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 $i = 0, 1, \ldots, m(n+1)$;

(2) ϕ is linear on each interval $[x_{i-1}, x_i]$ for $i \notin \{(n+1)j : j = 1, 2, \ldots, m\}$;

(3) $\sup_{x\in[x_0, x_{m(n+1)}]} |\phi(x)| \leq 3 \max_{i\in\{0,1,\ldots,m(n+1)\}} y_i \prod_{k=1}^{n} \left(1 + \frac{\max\{x_{j(n+1)+n} - x_{j(n+1)+k-1} : j=0,1,\ldots,m-1\}}{\min\{x_{j(n+1)+k} - x_{j(n+1)+k-1} : j=0,1,\ldots,m-1\}}\right)$.

Proof. For simplicity of notation, we define $I_0(m, n) := \{0, 1, \ldots, m(n+1)\}$, $I_1(m, n) := \{j(n+1) : j = 1, 2, \ldots, m\}$, and $I_2(m, n) := I_0(m, n)\backslash I_1(m, n)$ 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

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

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 $\boldsymbol{g} = [g_0, g_1^+, g_1^-, \ldots, g_n^+, g_n^-]^T$, then $\{g_0, g_1^+, g_1^-, \ldots, g_n^+, g_n^-\} \subseteq \mathrm{NN}(\#\mathrm{input} = 1;\ \mathrm{widthvec} = [2m])$. 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, $\exists\ \boldsymbol{W}_1 = [1, 1, \ldots, 1]^T \in \mathbb{R}^{2m\times 1}$ and $\boldsymbol{b}_1 \in \mathbb{R}^{2m}$ such that sub-networks in $\{g_0, g_1^+, g_1^-, \ldots, g_n^+, g_n^-\}$ have the same set of break points: $\left\{x_i : i \in I_1(m, n) \cup \big(I_1(m, n) - 1\big) \cup \{0\}\right\}$, 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 $\boldsymbol{g} = [g_0, g_1^+, g_1^-, \ldots, g_n^+, g_n^-]^T$ 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 $\{g_0, g_1^+, g_1^-, \ldots, g_n^+, g_n^-\}$ 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 $\{g_0, g_1^+, g_1^-, \ldots, g_n^+, g_n^-\}$. 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 $f_1 := f_0 - \tilde{g}_0$, where $\tilde{g}_0 = \sigma(g_0) = g_0$ as shown in Eq. (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 Fig. 2(a) for an illustration of f_0, f_1, and g_0.

Step 2.2: Mathematical induction.

For each $k \in \{1, 2, \ldots, n\}$, 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 Fig. 2(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.

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 $\left\{g_k^+(x_i) : i \in (I_1(m, n) - n - 1) \cup (I_1(m, n) - 1) \cup \{m(n+1)\}\right\}$ 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 $\ell = 0, 1, \ldots, k-1$, and $\tilde{g}_k^+$ is linear on each interval $[x_{j(n+1)+\ell}, x_{j(n+1)+\ell+1}]$ for $\ell = 0, 1, \ldots, n-1$.
- 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:

[3] 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.

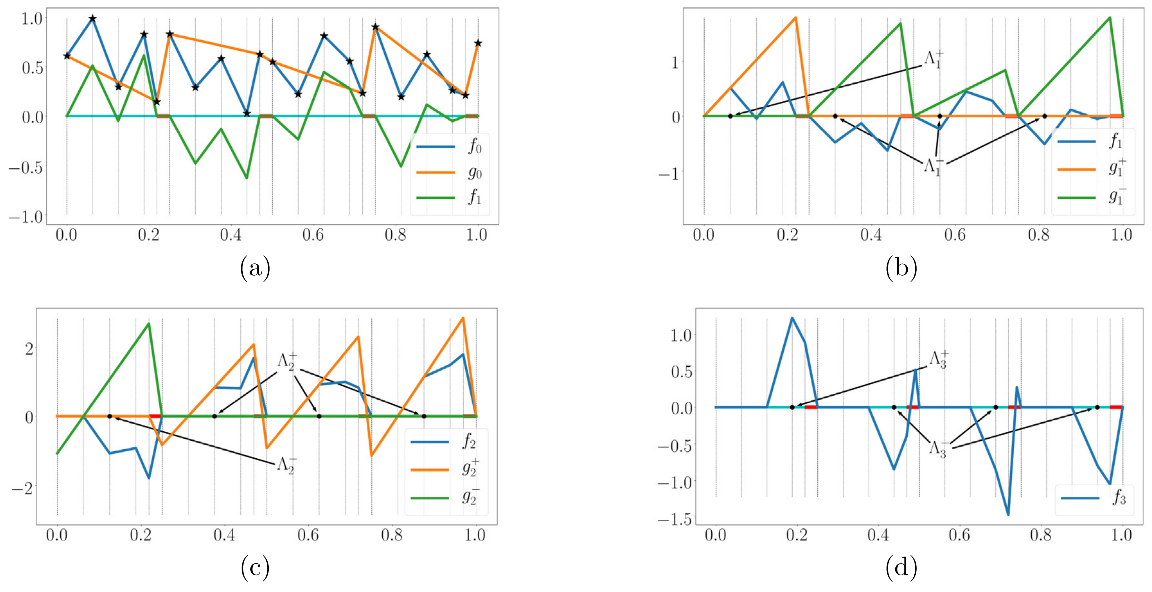

Fig. 2. Illustrations of the proof of Lemma 2.2, especially Step 2 of the proof, when $m = n = 4$, with the "don't-care" region in red. (a) Given samples $\{(x_i, y_i) : i = 0, 1, \ldots, m(n+1)\}$ marked with "star" signs, suppose $f_0(x)$ is a CPL function fitting the samples, construct g_0 such that $f_1 = f_0 - \sigma(g_0)$ 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. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- 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 $\ell = 0, 1, \ldots, k-1$, and $\tilde{g}_k^-(x)$ is linear on each interval $[x_{j(n+1)+\ell}, x_{j(n+1)+\ell+1}]$ for $\ell = 0, 1, \ldots, n-1$.
- 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 $\ell = 0, 1, \ldots, n-1$, $j \in \Lambda_k^+ \cup \Lambda_k^- = \{0, 1, \ldots, m-1\}$. 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 Fig. 2(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 $\Lambda_k^+ \cup \Lambda_k^- = \{0, 1, \ldots, m-1\}$, 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 Fig. 2(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 Fig. 2, 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 $\boldsymbol{g} = [g_0, g_1^+, g_1^-, \ldots, g_n^+, g_n^-]^T$ 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 $\boldsymbol{W}_3 = [1, 1, -1, 1, -1, \ldots, 1, -1] \in \mathbb{R}^{1\times(2n+1)}$ 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 $k \in \{1, 2, \ldots, n\}$, it holds that

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

and

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

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

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

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

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

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

$$\begin{aligned}&\sup_{x\in[x_0,x_{m(n+1)}]} |f_{n+1}(x)| \\ &\le 2 \max_{i\in I_0(m,n)} y_i \prod_{k=1}^{n} \left(\frac{\max\{x_{j(n+1)+n} - x_{j(n+1)+k-1} : j=0,1,\ldots,m-1\}}{\min\{x_{j(n+1)+k} - x_{j(n+1)+k-1} : j=0,1,\ldots,m-1\}} + 1\right).\end{aligned}$$

Hence

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

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 $f \in Lip(\nu, \alpha, d)$ with $\alpha \in (0, 1]$, we have*[4]*:*

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

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

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

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

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

Step 1: The case $d = 1$.

Given any $f \in \text{Lip}(\nu, \alpha, d)$ 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 $X = \{\frac{i}{N^2} : i = 0, 1, \ldots, N^2\} \cup \{\frac{i}{N} - \delta : i = 1, 2, \ldots, N\}$, 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 $\{(x_i, \bar{f}(x_i)) : i \in \{0, 1, \ldots, N(N + 1)\}\}$, there exists $\phi \in \text{NN}(\#\text{input} = 1;\ \text{widthvec} = [2N, 2N + 1])$ such that

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

It follows that

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

Define $H_0 = \cup_{i\in\{(N+1)j:j=1,2,\ldots,N\}}[x_{i-1}, x_i]$, 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 $\bar{f} \in \text{Lip}(1, \alpha, d)$ 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, $\phi - 1 \in \text{NN}(\#\text{input} = 1;\ \text{widthvec} = [2N, 2N + 1])$ 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 Fig. 3 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 $\psi_0 \in \text{NN}(\#\text{input} = 1;\ \text{widthvec} = [2n])$ such that

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

Define the projection map[5] ψ by

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

Note that $\psi \in \text{NN}(\#\text{input} = 1;\ \text{widthvec} = [2dn])$. Given $f \in \text{Lip}(\nu, \alpha, d)$, 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

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

[4] 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 $\mu([0, 1]^d) = 1$.

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

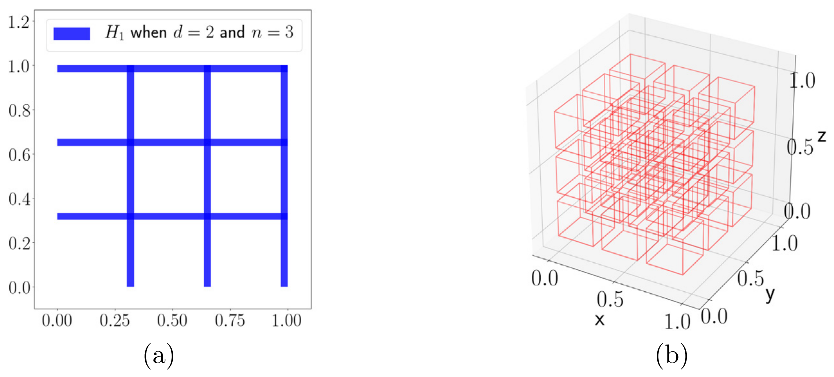

Fig. 3. An illustration of H_1 and n^d small non-overlapping sub-cubes that H_1 separates when $n = 3$. (a) When $d = 2$, H_1 in blue separates $[0, 1]^2$ into $n^d = 9$ sub-cubes. (b) When $d = 3$, H_1 (no color) separates $[0, 1]^3$ into $n^d = 27$ sub-cubes in red.

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

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

and

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

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

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

Define $H_1 = \cup_{j=1}^{d} \big\{\boldsymbol{x} = [x_1, x_2, \ldots, x_d]^T \in [0, 1]^d : x_j \in \cup_{i=1}^{n}[\frac{i}{n} - \delta, \frac{i}{n}]\big\}$, which separates the d-dimensional cube into n^d important sub-cubes as illustrated in Fig. 3. To index these d-dimensional smaller sub-cubes, define $Q_{\boldsymbol{\theta}} = \big\{\boldsymbol{x} = [x_1, x_2, \ldots, x_d]^T \in [0, 1]^d : x_i \in [\frac{\theta_i}{n}, \frac{\theta_i+1}{n} - \delta],\ i = 1, 2, \ldots, d\big\}$ for each d-dimensional index $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_d]^T \in \{0, 1, \ldots, n-1\}^d$. By (3.3), (3.4), and the definition of ψ_0, for any $\boldsymbol{x} = [x_1, x_2, \ldots, x_d]^T \in Q_{\boldsymbol{\theta}}$, 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 $\mu(H_1) \le dn\delta$, $[0, 1]^d = \cup_{\boldsymbol{\theta}\in\{0,1,\ldots,n-1\}^d} Q_{\boldsymbol{\theta}} \cup H_1$, (3.6), and (3.7), we have

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

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

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

Note that $f - (\phi - \sqrt{d}) = \bar{f} - \phi$. Hence, $\phi - \sqrt{d} \in$ NN(#input $= d$; 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,

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

and

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

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 $f \in \text{Lip}(\nu, \alpha, d)$ 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 NN(#input $= 1$; widthvec $= [2m, 2n + 1]$) contains 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.

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

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

The proof of Theorem 3.3 relies on the nearly-tight VC-dimension bounds of ReLU FNNs given in Harvey et al. (2017) and is similar to Theorem 4 of Yarotsky (2017). Hence, we only sketch out its proof and a complete proof can be found in Zhang (2019).

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 $C > 0$, $\forall f \in Lip(\nu, \alpha, d)$ with $\alpha \in (0, 1]$, then $\exists N_0 > 0$, for all $N \geq N_0$, there exists $\phi \in NN(\#\text{input} = d;\ \text{maxwidth} \leq N;\ \#\text{layer} \leq L)$ such that*

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

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 $\nu \neq 1$, the proof is similar by rescaling $f \in \text{Lip}(\nu, \alpha, d)$ and FNNs with ν. Let us denote the VC dimension of a function set $\mathcal{F}$ by $\text{VCDim}(\mathcal{F})$. By Harvey et al. (2017), there exists $C_1 > 0$ such that

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

By Eq. (3), we have

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

One can estimate a lower bound of

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

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

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

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 $N \geq N_0$, $CPL(CN^{2+\rho})$ is not contained in the closure of $NN(\#\text{input} = 1;\ \text{maxwidth} \leq N;\ \#\text{layer} \leq L)$ 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 (Kumar, 2002) 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

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

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

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

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 Kumar (2002) for a detailed introduction).

In real applications, a most frequently asked question would be: given a target function $f \in \text{Lip}(\nu, \alpha, d)$, 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 (Yarotsky, 2018) and this manuscript both utilize a nearly uniform width. Combining the theory in Yarotsky (2018) and ours, we summarize several statistics of ReLU FNNs in parallel computing in Tables 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 limit 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 $m \in [1,(2d+10)^2]$	$\mathcal{O}(N^2/m+t_s\ln m+\frac{t_wN}{\sqrt{m}}\ln m)$	$\mathcal{O}(L(N^2/m+t_s\ln m+\frac{t_wN}{\sqrt{m}}\ln m))$	$\mathcal{O}(N(d^2/m+t_s\ln m+\frac{t_wd}{\sqrt{m}}\ln m))$
Running time for $m \in ((2d+10)^2, N^2]$	$\mathcal{O}(N^2/m+t_s\ln m+\frac{t_wN}{\sqrt{m}}\ln m)$	$\mathcal{O}(L(N^2/m+t_s\ln m+\frac{t_wN}{\sqrt{m}}\ln m))$	$\mathcal{O}(N\ln d)$
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 Filip, Javeed, and Trefethen (2018) 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. 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 (Kingma & Ba, 2014) 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".[6] 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 high 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 require 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 Tables 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 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 Tables 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 high 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.

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

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 20 000 training steps with training batch size 10 000, and the unit is second(s).

N	Depth	Width	Test error	Improvement ratio	#parameter	Time
32	2	32	8.06×10^{-2}	–	1 153	3.09×10^{1}
32	4	32	3.98×10^{-4}	–	3 265	3.82×10^{1}
32	8	32	1.50×10^{-5}	–	7 489	5.60×10^{1}
32	32	12	1.29×10^{-3}	–	4 873	1.27×10^{2}
64	2	64	2.51×10^{-2}	3.21	4 353	3.45×10^{1}
64	4	64	4.27×10^{-5}	9.32	12 673	5.00×10^{1}
64	8	64	2.01×10^{-6}	7.46	29 313	7.91×10^{1}
64	64	12	1.16×10^{-1}	0.01	9 865	2.37×10^{2}
128	2	128	2.04×10^{-3}	12.3	16 897	5.03×10^{1}
128	4	128	1.05×10^{-5}	4.07	49 921	8.21×10^{1}
128	8	128	1.47×10^{-6}	1.37	115 969	1.41×10^{2}
128	128	12	3.17×10^{-1}	0.37	19 849	4.47×10^{2}

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 20 000 training steps with training batch size 10 000, and the unit is second(s).

#parameter	Depth	Width	Test error	Improvement ratio	Time
5 038	2	69	1.13×10^{-2}	–	3.84×10^{1}
5 041	4	40	1.65×10^{-4}	–	3.80×10^{1}
4 993	8	26	1.69×10^{-5}	–	5.07×10^{1}
5 029	33	12	4.77×10^{-3}	–	1.28×10^{2}
9 997	2	98	4.69×10^{-3}	2.41	4.40×10^{1}
10 090	4	57	7.69×10^{-5}	2.14	4.67×10^{1}
9 954	8	37	7.43×10^{-6}	2.27	5.92×10^{1}
10 021	65	12	2.80×10^{-1}	0.02	2.31×10^{2}
19 878	2	139	1.43×10^{-3}	3.28	5.18×10^{1}
20 170	4	81	2.30×10^{-5}	3.34	6.26×10^{1}
20 194	8	53	2.97×10^{-6}	2.50	7.08×10^{1}
20 005	129	12	3.17×10^{-1}	0.88	4.30×10^{2}

5. Conclusions

We studied the approximation and computation efficiency of nonlinear approximation via compositions, especially when the dictionary $\mathcal{D}_L$ consists of deep ReLU FNNs with width N and depth L. Our main goal is to quantify the best N-term approximation rate $\varepsilon_{L,f}(N)$ for the dictionary $\mathcal{D}_L$ when f is a Hölder continuous function. This topic is related to several existing approximation theories in the literature, but they cannot be applied to answer our problem. By introducing new analysis techniques that are merely based on the FNN structure instead of traditional approximation basis functions used in existing work, we have established a new theory to address our problem.

In particular, for any function f on $[0, 1]$, regardless of its smoothness and even the continuity, if f can be approximated using a dictionary when $L = 1$ with the best N-term approximation rate $\varepsilon_{L,f} = \mathcal{O}(N^{-\eta})$, we showed that dictionaries with $L = 2$ can improve the best N-term approximation rate to $\varepsilon_{L,f} = \mathcal{O}(N^{-2\eta})$. We also showed that for Hölder continuous functions of order α on $[0, 1]^d$, the application of a dictionary with $L = 3$ in nonlinear approximation can achieve an essentially tight best N-term approximation rate $\varepsilon_{L,f} = \mathcal{O}(N^{-2\alpha/d})$, and increasing L further cannot improve the approximation rate in N. Finally, we showed that dictionaries consisting of wide FNNs with a few hidden layers are more attractive in terms of computational efficiency than dictionaries with narrow and very deep FNNs for approximating Hölder continuous functions if the number of computer cores is larger than N in parallel computing.

Our results were based on the L^1-norm in the analysis of constructive approximations in Section 3.1; while we used L^∞-norm in the unachievable approximation rate in Section 3.2. In fact, we can define a new norm that is weaker than the L^∞-norm and stronger than the L^1-norm such that all theorems in this paper hold in the same norm. The analysis based on the new norm can be found in Zhang (2019) and our future work, and it shows that the approximation rate in this paper is tight in the same norm. Finally, although the current result is only valid for Hölder continuous functions, it is easy to generalize it to general continuous functions by introducing the moduli of continuity, which is also left as future work.

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 (The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg) (n.d.), 2019) for extra GPU resource. We would like to acknowledge the editor and reviewers for their help in improving this manuscript.

References

Anthony, M., & Bartlett, P. L. (2009). *Neural network learning: Theoretical foundations* (1st ed.). New York, NY, USA: Cambridge University Press.
Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, *39*(3), 930–945. http://dx.doi.org/10.1109/18.256500.
Bartlett, P., Maiorov, V., & Meir, R. (1998). Almost linear VC dimension bounds for piecewise polynomial networks. *Neural Computation*, *10*, 2159-2173.
Bianchini, M., & Scarselli, F. (2014). On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks Learing Systems*, *25*(8), 1553–1565. http://dx.doi.org/10.1109/TNNLS.2013.2293637.
Candes, E. J., & Wakin, M. B. (2008). An introduction to compressive sampling. *IEEE Signal Processing Magazine*, *25*(2), 21–30. http://dx.doi.org/10.1109/MSP.2007.914731.
Chen, S., & Donoho, D. (1994). Basis pursuit. In *Proceedings of 1994 28th Asilomar conference on signals, systems and computers, Vol. 1* (pp. 41–44). http://dx.doi.org/10.1109/ACSSC.1994.471413.
Cireşan, D. C., Meier, U., Masci, J., Gambardella, L. M., & Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. In *IJCAI'11, Proceedings of the twenty-second international joint conference on artificial intelligence - volume volume two* (pp. 1237–1242). AAAI Press, Retrieved from http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-210.
Costarelli, D., & Sambucini, A. R. (2017). Saturation classes for max-product neural network operators activated by sigmoidal functions. *Results in Mathematics*, *72*(3), 1555–1569. http://dx.doi.org/10.1007/s00025-017-0692-6.

Costarelli, D., & Vinti, G. (2017). Convergence for a family of neural network operators in orlicz spaces. *Mathematische Nachrichten*, *290*(2–3), 226–235, Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1002/mana.201600006.

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

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

Daubechies, I. (1992). *Ten lectures on wavelets*. Society for Industrial and Applied Mathematics, Retrieved from https://epubs.siam.org/doi/abs/10.1137/1.9781611970104.

Davis, G. (1994). *Adaptive nonlinear approximations*.

DeVore, R. A. (1998). Nonlinear approximation. *Acta Numerica*, *7*, 51–150. http://dx.doi.org/10.1017/S0962492900002816.

Devore, R., & Ron, A. (2010). Approximation using scattered shifts of a multivariate function. *Transactions of the American Mathematical Society*, *362*(12), 6205–6229, Retrieved from http://www.jstor.org/stable/40997201.

Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on Information Theory*, *52*(4), 1289–1306. http://dx.doi.org/10.1109/TIT.2006.871582.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*, 2121–2159, Retrieved from http://dl.acm.org/citation.cfm?id=1953048.2021068.

Filip, S.-I., Javeed, A., & Trefethen, L. N. (2018). Smooth random functions, random odes, and Gaussian processes. *SIAM Review*, (in press) Retrieved from https://hal.inria.fr/hal-01944992.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, [ISSN: 1432-0770] *36*(4), 193–202, Retrieved from https://doi.org/10.1007/BF00344251.

Hangelbroek, T., & Ron, A. (2010). Nonlinear approximation using Gaussian kernels. *Journal of Functional Analysis*, *259*(1), 203–219, Retrieved from http://www.sciencedirect.com/science/article/pii/S0022123610000467.

Harvey, N., Liaw, C., & Mehrabian, A. (2017). Nearly-tight VC-dimension bounds for piecewise linear neural networks. In S. Kale, & O. Shamir (Eds.), *Proceedings of machine learning research*: *Vol. 65*, *Proceedings of the 2017 conference on learning theory* (pp. 1064–1068). Amsterdam, Netherlands: PMLR, Retrieved from http://proceedings.mlr.press/v65/harvey17a.html.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366, Retrieved from http://www.sciencedirect.com/science/article/pii/0893608089900208.

Jiang, J. (1996). Design of neural networks for lossless data compression. *Optimization and Engineering*, *35*, 35–35–7, Retrieved from https://doi.org/10.1117/1.600614.

Johnson, R., & Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS'13, Proceedings of the 26th international conference on neural information processing systems - Volume 1* (pp. 315–323). USA: Curran Associates Inc., Retrieved from http://dl.acm.org/citation.cfm?id=2999611.2999647.

Joutsensalo, J. (1994). 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* (pp. 637–640). http://dx.doi.org/10.1109/ICNN.1994.374249.

Kawaguchi, K. (2016). Deep learning without poor local minima. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems, Vol. 29* (pp. 586–594). Curran Associates, Inc., Retrieved from http://papers.nips.cc/paper/6112-deep-learning-without-poor-local-minima.pdf.

Kawaguchi, K., & Bengio, Y. (2018). *Depth with Nonlinearity Creates No Bad Local Minima in ResNets*. Retrieved from https://arxiv.org/abs/1810.09038.

Kearns, M. J., & Schapire, R. E. (1994). Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences*, *48*(3), 464–497, Retrieved from http://dx.doi.org/10.1016/S0022-0000(05)80062-5.

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization, CoRR abs/1412.6980, Retrieved from http://arxiv.org/abs/1412.6980.

Kumar, V. (2002). *Introduction to parallel computing* (2nd ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc..

Lewicki, G., & Marino, G. (2004). Approximation of functions of finite variation by superpositions of a sigmoidal function. *Applied Mathematics Letters*, *17*(10), 1147–1152, Retrieved from http://www.sciencedirect.com/science/article/pii/S089396590481694X.

Liang, S., & Srikant, R. (2016). Why Deep Neural Networks? CoRR abs/1610.04161. Retrieved from http://arxiv.org/abs/1610.04161.

Lin, S., Liu, X., Rong, Y., & Xu, Z. (2014). Almost optimal estimates for approximation and learning by radial basis function networks. *Machine Learning*, *95*(2), 147–164. http://dx.doi.org/10.1007/s10994-013-5406-z, Retrieved from https://doi.org/10.1007/s10994-013-5406-z.

Llanas, B., & Sainz, F. (2006). Constructive approximate interpolation by neural networks. *Journal of Computational and Applied Mathematics*, *188*(2), 283–308, Retrieved from http://www.sciencedirect.com/science/article/pii/S0377042705002566.

Lu, Z., Pu, H., Wang, F., Hu, Z., & Wang, L. (2017). The Expressive Power of Neural Networks: A View from the Width, CoRR abs/1709.02540, Retrieved from http://arxiv.org/abs/1709.02540.

Mallat, S. G., & Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, *41*(12), 3397–3415. http://dx.doi.org/10.1109/78.258082.

Montanelli, H., & Du, Q. (2017). *New error bounds for deep networks using sparse grids*. Retrieved from https://arxiv.org/abs/1712.08688.

Montanelli, H., Yang, H., & Du, Q. (2019). *Deep ReLU networks overcome the curse of dimensionality for bandlimited functions*. Retrieved from https://arxiv.org/abs/1903.00735v1.

Montufar, G. F., Pascanu, R., Cho, K., & Bengio, Y. (2014). On the number of linear regions of deep neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 27* (pp. 2924–2932). Curran Associates, Inc., Retrieved from http://papers.nips.cc/paper/5422-on-the-number-of-linear-regions-of-deep-neural-networks.pdf.

Nguyen, Q. N., & Hein, M. (2017). The loss surface of deep and wide neural networks, CoRR abs/1704.08045, Retrieved from http://arxiv.org/abs/1704.08045.

Ohlsson, H., Yang, A. Y., Dong, R., & Sastry, S. S. (2013). Nonlinear basis pursuit. In *2013 Asilomar conference on signals, systems and computers* (pp. 115–119). http://dx.doi.org/10.1109/ACSSC.2013.6810285.

Petersen, P., & Voigtlaender, F. (2018). Optimal approximation of piecewise smooth functions using deep relu neural networks. *Neural Networks*, *108*, 296–330, Retrieved from http://www.sciencedirect.com/science/article/pii/S0893608018302454.

Petrushev, P. (2003). Multivariate n-term rational and piecewise polynomial approximation. *Journal of Approximation Theory*, *121*(1), 158–197. http://dx.doi.org/10.1016/S0021-9045(02)00060-6, Retrieved from http://www.sciencedirect.com/science/article/pii/S0021904502000606.

Rumelhart, D., McClelland, J., Group, P. R., & University of California, S. D. P. R. G. (1986). *A bradford book*: *Vol. 2*, *Psychological and biological models*. MIT Press, Retrieved from https://books.google.com.sg/books?id=davmLgzusB8C.

Sakurai, A. (1999). Tight bounds for the VC-dimension of piecewise polynomial networks. In *Advances in neural information processing systems* (pp. 323–329). Neural information processing systems foundation.

Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In K. Diamantaras, W. Duch, & L. S. Iliadis (Eds.), *Artificial neural networks – ICANN 2010* (pp. 92–101). Berlin, Heidelberg: Springer Berlin Heidelberg.

Suzuki, T. (2019). 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*. Retrieved from https://openreview.net/forum?id=H1ebTsActm.

Tariyal, S., Majumdar, A., Singh, R., & Vatsa, M. 2016. Greedy Deep Dictionary Learning, CoRR, abs/1602.00203, Retrieved from http://arxiv.org/abs/1602.00203.

The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg) (n.d.) (2019).

Weinan, E., & Wang, Q. (2018). Exponential Convergence of the Deep Neural Network Approximation for Analytic Functions, CoRR abs/1807.00297. Retrieved from http://arxiv.org/abs/1807.00297.

Werbos, P. (1975). Beyond regression: New tools for prediction and analysis in the behavioral sciences. Harvard University, Retrieved from https://books.google.com.sg/books?id=z81XmgEACAAJ.

Xie, T. F., & Cao, F. L. (2013). The rate of approximation of Gaussian radial basis neural networks in continuous function space. *Acta Mathematica Sinica, English Series*, *29*(2), 295–302, Retrieved from https://doi.org/10.1007/s10114-012-1369-4.

Yarotsky, D. (2017). Error bounds for approximations with deep relu networks. *Neural Networks*, *94*, 103–114, Retrieved from http://www.sciencedirect.com/science/article/pii/S0893608017301545.

Yarotsky, D. (2018). Optimal approximation of continuous functions by very deep relu networks. In S. Bubeck, V. Perchet, & P. Rigollet (Eds.), *Proceedings of machine learning research*: *Vol. 75*, *Proceedings of the 31st conference on learning theory* (pp. 639–649). PMLR, Retrieved from http://proceedings.mlr.press/v75/yarotsky18a.html.

Zhang, S. (2019). *Approximation Theories for Deep Neural Networks via Function Compositions* (Ph.D. Thesis), the National University of Singapore, (n.d.) submitted for publication.