Neural Network Architecture Beyond Width and Depth

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

Haizhao Yang
Department of Mathematics
University of Maryland, College Park
hzyang@umd.edu

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

Abstract

This paper proposes a new neural network architecture by introducing an additional dimension called height beyond width and depth. Neural network architectures with height, width, and depth as hyper-parameters are called three-dimensional architectures. It is shown that neural networks with three-dimensional architectures are significantly more expressive than the ones with two-dimensional architectures (those with only width and depth as hyper-parameters), e.g., standard fully connected networks. The new network architecture is constructed recursively via a nested structure, and hence we call a network with the new architecture nested network (NestNet). A NestNet of height s is built with each hidden neuron activated by a NestNet of height $\le s-1$. When $s=1$, a NestNet degenerates to a standard network with a two-dimensional architecture. It is proved by construction that height-s ReLU NestNets with $\mathcal{O}(n)$ parameters can approximate 1-Lipschitz continuous functions on $[0,1]^d$ with an error $\mathcal{O}(n^{-(s+1)/d})$, while the optimal approximation error of standard ReLU networks with $\mathcal{O}(n)$ parameters is $\mathcal{O}(n^{-2/d})$. Furthermore, such a result is extended to generic continuous functions on $[0,1]^d$ with the approximation error characterized by the modulus of continuity. Finally, we use numerical experimentation to show the advantages of the super-approximation power of ReLU NestNets.

1 Introduction

In this paper, we design a new neural network architecture by introducing one more dimension, called height, in addition to width and depth in the characterization of dimensions of neural networks. We call neural network architectures with height, width, and depth as hyper-parameters three-dimensional architectures. It is proved by construction that neural networks with three-dimensional architectures improve the approximation power significantly, compared to standard networks with two-dimensional architectures (those with only width and depth as hyper-parameters). The approximation power of standard neural networks has been widely studied in recent years. The optimality of the approximation of standard fully-connected rectified linear unit (ReLU) networks (e.g., see [35, 40, 49, 52]) implies limited room for further improvements. This motivates us to design a new neural network architecture by introducing an additional dimension of height beyond width and depth.

* Corresponding author.

We will focus on the ReLU ($\max\{0,x\}$) activation function and use it to demonstrate our ideas. Our new network architecture is constructed recursively via a nested structure, and hence we call a neural network with the new architecture nested network (**NestNet**). A NestNet of height s is built with each hidden neuron activated by a NestNet of height $\le s-1$. In the case of $s=1$, a NestNet degenerates to a standard network with a two-dimensional architecture. Let us use a simple example to explain the height of a NestNet. We say a network is activated by $\varrho_1,\cdots,\varrho_r$ if each hidden neuron of this network is activated by one of $\varrho_1,\cdots,\varrho_r$. Here, $\varrho_1,\cdots,\varrho_r$ are trainable functions mapping $\mathbb{R}$ to $\mathbb{R}$. Then, a network of height $s\ge 2$ can be regarded as a $(\varrho_1,\cdots,\varrho_r)$-activated network, where $\varrho_1,\cdots,\varrho_r$ are (realized by) networks of height $\le s-1$. See an example of a height-2 network in Figure 1. The network therein can be regarded as a (ϱ_1,ϱ_2)-activated network, where ϱ_1 and ϱ_2 are (realized by) networks of height 1 (i.e., standard networks). The number of parameters in the network of Figure 1 is the sum of the numbers of parameters in $\mathcal{L}_0,\mathcal{L}_1,\mathcal{L}_2$ and ϱ_1,ϱ_2.

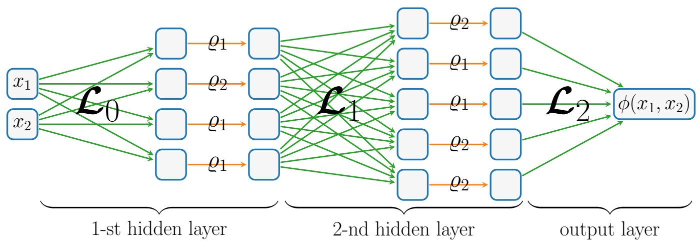


Figure 1: An example of a network of height 2, where ϱ_1 and ϱ_2 are (realized by) networks of height 1 (i.e., standard networks). Here, $\mathcal{L}_0$, $\mathcal{L}_1$ and $\mathcal{L}_2$ are affine linear maps.

We remark that a NestNet can be regarded as a sufficiently large standard network by expanding all of its sub-network activation functions. We propose the nested network architecture since it shares the parameters via repetitions of sub-network activation functions. In other words, a NestNet can provide a special parameter-sharing scheme. This is the key reason why the NestNet has much better approximation power than the standard network. If we regard the network in Figure 1 as a NestNet of height 2, then the number of parameters is the sum of the numbers of parameters in $\mathcal{L}_0,\mathcal{L}_1,\mathcal{L}_2$ and ϱ_1,ϱ_2. However, if we expand the network in Figure 1 to a large standard network, then the number of parameters in ϱ_1 and ϱ_2 will be added many times for computing the total number of parameters.

Next, let us discuss our new network architecture from the perspective of hyper-parameters. We call the network architecture with only width as a hyper-parameter one-dimensional architecture. Its depth and height are both equal to one. Neural networks with this type of architecture are generally called shallow networks. See an example in Figure 2(a). We call the network architecture with only width and depth as hyper-parameters two-dimensional architecture. Its height is equal to one. Neural networks with this type of architecture are generally called deep networks. See an example in Figure 2(b). We call the network architecture with height, width, and depth as hyper-parameters three-dimensional architecture, which is proposed in this paper. Neural networks with this type of architecture are called NestNets. See an example in Figure 2(c). One may refer to Table 1 for the approximation power of networks with these three types of architectures discussed above.

Table 1: Comparison for the approximation error of 1-Lipschitz continuous functions on $[0,1]^d$ approximated by ReLU NestNets and standard ReLU networks.

	dimension(s)	#parameters	approximation error	remark	reference
one-hidden-layer network	width varies (depth = height = 1)	$\mathcal{O}(n)$	n^{-1} for $d=1$	linear combination	
deep network	width and depth vary (height = 1)	$\mathcal{O}(n)$	$n^{-2/d}$	composition	[35, 40, 49, 52]
NestNet of height s	width, depth, and height vary	$\mathcal{O}(n)$	$n^{-(s+1)/d}$	nested composition	this paper

Our main contributions are summarized as follows. We first propose a three-dimensional neural network architecture by introducing one more dimension called height beyond width and depth. We show that neural networks with three-dimensional architectures are significantly more expressive than standard networks. In particular, we prove that height-s ReLU NestNets with $\mathcal{O}(n)$ parameters can approximate 1-Lipschitz continuous functions on $[0,1]^d$ with an error $\mathcal{O}(n^{-(s+1)/d})$, which is much better than the optimal error $\mathcal{O}(n^{-2/d})$ of standard ReLU networks with $\mathcal{O}(n)$ parameters. In

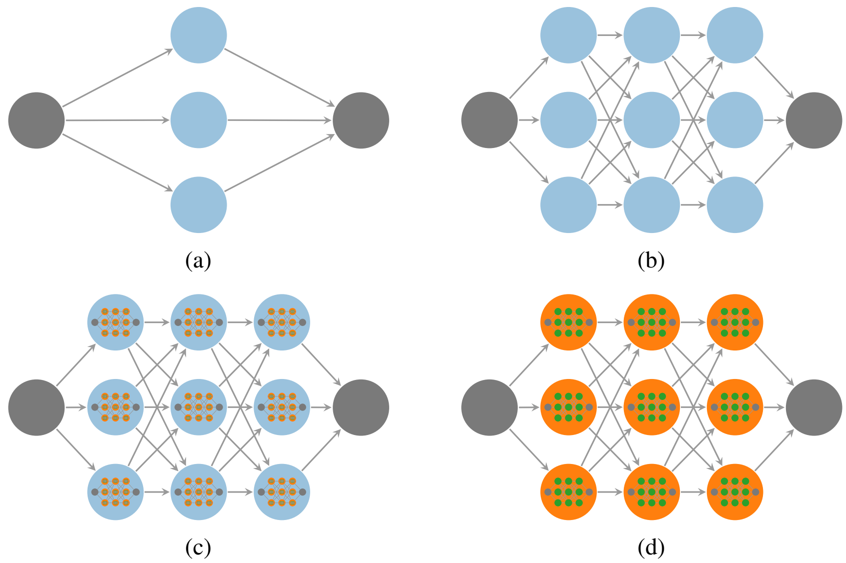


Figure 2: Illustrations of neural networks with one-, two-, and three-dimensional architectures. (a) One-dimensional case (width = 3, depth = height = 1). (b) Two-dimensional case (width = depth = 3, height = 1). (c) Three-dimensional case (width = depth = height = 3). (d) Zoom-in of an activation function of the network in (c). The network in (d) can also be regarded as a network of height 2.

the case of $s + 1 \geq d$, the approximation error is bounded by $\mathcal{O}(n^{-(s+1)/d}) \leq \mathcal{O}(n^{-1})$, which means we overcome the curse of dimensionality. Furthermore, we extend our result to generic continuous functions with the approximation error characterized by the modulus of continuity. See Theorem 2.1 and Corollary 2.2 for more details. Finally, we conduct simple experiments to show the numerical advantages of the super-approximation power of ReLU NestNets.

The rest of this paper is organized as follows. In Section 2, we present the main results, provide the ideas of proving them, and discuss related work. The detailed proofs of the main results are placed in the appendix. Next, we conduct experiments to show the advantages of the super-approximation power of ReLU NestNets in Section 3. Finally, Section 4 concludes this paper with a short discussion.

2 Main results and related work

In this section, we first present our main results and discuss the proof ideas. The detailed proofs of the main results are placed in the appendix. Next, we discuss related work from multiple perspectives.

2.1 Main results

We use $\mathcal{NN}_s\{n\}$ for $n, s \in \mathbb{N}$ to denote the set of functions realized by height-s ReLU NestNets with as most n parameters. We will give the mathematical definition of $\mathcal{NN}_s\{n\}$. We first discuss some notations regarding affine linear maps. We use $\mathscr{L}$ to denote the set of all affine linear maps, i.e.,

$$\mathscr{L} \coloneqq \Big\{ \boldsymbol{\mathcal{L}} : \boldsymbol{\mathcal{L}}(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b},\ \boldsymbol{W} \in \mathbb{R}^{d_2 \times d_1},\ \boldsymbol{b} \in \mathbb{R}^{d_2},\ d_1, d_2 \in \mathbb{N}^+ \Big\}.$$

Let $\#\boldsymbol{\mathcal{L}}$ denote the number of parameters in $\boldsymbol{\mathcal{L}} \in \mathscr{L}$, i.e.,

$$\#\boldsymbol{\mathcal{L}} = (d_1 + 1)d_2 \quad \text{if } \boldsymbol{\mathcal{L}}(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b} \quad \text{for } \boldsymbol{W} \in \mathbb{R}^{d_2 \times d_1} \text{ and } \boldsymbol{b} \in \mathbb{R}^{d_2}.$$

We use $\vec{g} = (\varrho_1, \cdots, \varrho_k)$ to denote an activation function vector, where $\varrho_i : \mathbb{R} \to \mathbb{R}$ is an activation function for each $i \in \{1, \cdots, k\}$. When $\vec{g} = (\varrho_1, \cdots, \varrho_k)$ is applied to a vector input $\boldsymbol{x} = (x_1, \cdots, x_k)$,

$$\vec{g}(\boldsymbol{x}) = \Big(\varrho_1(x_1), \cdots, \varrho_k(x_k) \Big) \quad \text{for any } \boldsymbol{x} = (x_1, \cdots, x_k) \in \mathbb{R}^k.$$

Let $\text{set}(\vec{g})$ denote the function set containing all entries (functions) in $\vec{g}$. For example, if $\vec{g} = (\varrho_1, \varrho_2, \varrho_3, \varrho_2, \varrho_1)$, then $\text{set}(\vec{g}) = \{\varrho_1, \varrho_2, \varrho_3\}$.

To define $\mathcal{NN}_s\{n\}$ for $n, s \in \mathbb{N}$ recursively, we first consider the degenerate case. Define

$$\mathcal{NN}_0\{n\} \coloneqq \big\{\mathrm{id}_{\mathbb{R}},\, \mathrm{ReLU}\big\} \eqqcolon \mathcal{NN}_s\{0\} \quad \text{for } n, s \in \mathbb{N},$$

where $\mathrm{id}_{\mathbb{R}} : \mathbb{R} \to \mathbb{R}$ is the identity map. That is, we regard the identity map and ReLU as height-0 ReLU NestNets with n parameters or as height-s ReLU NestNets with 0 parameters.

Next, let us present the recursive step. For $n, s \in \mathbb{N}^+$, a (vector-valued) function $\boldsymbol{\phi} \in \mathcal{NN}_s\{n\}$ has the following form:

$$\boldsymbol{\phi} = \boldsymbol{\mathcal{L}}_m \circ \vec{g}_m \circ \ \cdots \ \circ \boldsymbol{\mathcal{L}}_1 \circ \vec{g}_1 \circ \boldsymbol{\mathcal{L}}_0, \tag{1}$$

where $\boldsymbol{\mathcal{L}}_0, \cdots, \boldsymbol{\mathcal{L}}_m \in \mathscr{L}$ are affine linear maps. Moreover, Equation (1) satisfies the following two conditions:

- Condition on activation functions:

$$\bigcup_{i=1}^{m} \mathrm{set}(\vec{g}_i) = \{\varrho_1, \cdots, \varrho_r\} \quad \text{and} \quad \varrho_j \in \bigcup_{i=0}^{s-1} \mathcal{NN}_i\{n_j\} \quad \text{for } j = 1, \cdots, r. \tag{2}$$

  Here, $\vec{g}_i$ is an activation function vector for each $i \in \{1, \cdots, m\}$. All entries in $\vec{g}_1, \cdots, \vec{g}_m$ form an activation function set $\{\varrho_1, \cdots, \varrho_r\}$. For each $j \in \{1, \cdots, r\}$, ϱ_j can be realized by a height-i NestNet with $\le n_j$ parameters for some $i = i_j \le s-1$. This condition means each hidden neuron is activated by a NestNet of lower height.

- Condition on the number of parameters:

$$\sum_{i=0}^{m} \#\boldsymbol{\mathcal{L}}_i \ + \ \sum_{j=1}^{r} n_j \le n. \tag{3}$$

  This condition means the total number of parameters is no more than n. The total number of parameters is calculated by adding two parts. The first one is the number of parameters in affine linear maps $\boldsymbol{\mathcal{L}}_0, \cdots, \boldsymbol{\mathcal{L}}_m$. The other part is the number of parameters in the activation set $\{\varrho_1, \cdots, \varrho_r\}$ formed by the entries in activation function vectors $\vec{g}_1, \cdots, \vec{g}_m$.

Then, with two conditions in Equations (2) and (3), we can define $\mathcal{NN}_s\{n\}$ for $n, s \in \mathbb{N}^+$ as follows:

$$\boldsymbol{\mathcal{NN}}_s\{n\} \coloneqq \Bigg\{\boldsymbol{\phi} : \boldsymbol{\phi} = \boldsymbol{\mathcal{L}}_m \circ \vec{g}_m \circ \cdots \circ \boldsymbol{\mathcal{L}}_1 \circ \vec{g}_1 \circ \boldsymbol{\mathcal{L}}_0, \ \ \boldsymbol{\mathcal{L}}_0, \cdots, \boldsymbol{\mathcal{L}}_m \in \mathscr{L}, \ \bigcup_{i=1}^{m} \mathrm{set}(\vec{g}_i) = \{\varrho_1, \cdots, \varrho_r\}, \\ \varrho_j \in \bigcup_{i=0}^{s-1} \boldsymbol{\mathcal{NN}}_i\{n_j\} \ \text{ for } j = 1, \cdots, r, \ \ \sum_{i=0}^{m} \#\boldsymbol{\mathcal{L}}_i \ + \ \sum_{j=1}^{r} n_j \le n\Bigg\}.$$

We remark that, in the definition above, m can be equal to 0. In this case, the function $\boldsymbol{\phi}$ degenerates to an affine linear map.

In the NestNet example in Figure 1, the function ϕ therein is in $\bigcup_{n\in\mathbb{N}} \mathcal{NN}_2\{n\}$ and the activation function vectors $\vec{g}_1$ and $\vec{g}_2$ can be represented as

$$\vec{g}_1 = \big(\varrho_1,\ \varrho_2,\ \varrho_1,\ \varrho_1\big) \quad \text{and} \quad \vec{g}_2 = \big(\varrho_2,\ \varrho_1,\ \varrho_1,\ \varrho_2,\ \varrho_2\big).$$

Moreover, the activation function set containing all entries in $\vec{g}_1$ and $\vec{g}_2$ is a subset of $\bigcup_{n\in\mathbb{N}} \mathcal{NN}_1\{n\}$, i.e., $\{\varrho_1, \varrho_2\} \subseteq \bigcup_{n\in\mathbb{N}} \mathcal{NN}_1\{n\}$.

Let $C([0,1]^d)$ denote the set of continuous functions on $[0,1]^d$. By convention, the modulus of continuity of a continuous function $f \in C([0,1]^d)$ is defined as

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

Under these settings, we can find a function in $\mathcal{NN}_s\big\{\mathcal{O}(n)\big\}$ to approximate $f \in C([0,1]^d)$ with an approximation error $\mathcal{O}\big(\omega_f(n^{-(s+1)/d})\big)$, as shown in the main theorem below.

Theorem 2.1. *Given a continuous function $f \in C([0,1]^d)$, for any $n, s \in \mathbb{N}^+$ and $p \in [1, \infty]$, there exists $\phi \in \mathcal{NN}_s\big\{C_{s,d}(n+1)\big\}$ such that*

$$\|\phi - f\|_{L^p([0,1]^d)} \le 7\sqrt{d}\,\omega_f\big(n^{-(s+1)/d}\big),$$

where $C_{s,d} = 10^3 d^2 (s+7)^2$ if $p \in [1, \infty)$ and $C_{s,d} = 10^{d+3} d^2 (s+7)^2$ if $p = \infty$.

We remark that the constant $C_{s,d}$ in Theorem 2.1 is valid for all $n \in \mathbb{N}^+$. As we shall see later, $C_{s,d}$ can be greatly reduced if one only cares about large $n \in \mathbb{N}^+$. Generally, it is challenging to simplify the approximation error in Theorem 2.1 to make it explicitly depend on n due to the complexity of $\omega_f(\cdot)$. However, the approximation error can be simplified to an explicit one depending on n in the case of special target function spaces like Hölder continuous function space. To be exact, if f is a Hölder continuous function on $[0,1]^d$ of order $\alpha \in (0,1]$ with a Hölder constant $\lambda > 0$, then

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

implying $\omega_f(r) \le \lambda r^\alpha$ for any $r \ge 0$. This means we can get an exponentially small approximation error $7\lambda\sqrt{d}\,n^{-(s+1)\alpha/d}$ as shown in Corollary 2.2 below.

Corollary 2.2. *Suppose f is a Hölder continuous function on $[0,1]^d$ of order $\alpha \in (0,1]$ with a Hölder constant $\lambda > 0$. For any $n, s \in \mathbb{N}^+$ and $p \in [1,\infty]$, there exists $\phi \in \mathcal{NN}_s\{C_{s,d}(n+1)\}$ such that*

$$\|\phi - f\|_{L^p([0,1]^d)} \le 7\lambda\sqrt{d}\,n^{-(s+1)\alpha/d},$$

where $C_{s,d} = 10^3 d^2 (s+7)^2$ if $p \in [1,\infty)$ and $C_{s,d} = 10^{d+3} d^2 (s+7)^2$ if $p = \infty$.

In Corollary 2.2, if $\alpha = 1$, i.e., f is a Lipschitz continuous function with a Lipschitz constant $\lambda > 0$, then the approximation error can be further simplified to $7\lambda\sqrt{d}\,n^{-(s+1)/d}$. See Table 1 for the comparison of the approximation error of 1-Lipschitz continuous functions on $[0,1]^d$ approximated by ReLU NestNets and standard ReLU networks.

2.2 Sketch of proving Theorem 2.1

We will discuss how to prove Theorem 2.1. Given a target function $f \in C([0,1]^d)$, the key point is to construct an almost piecewise constant function realized by a ReLU NestNet to approximate f well except for a small region. Then we can get the desired result by dealing with the approximation in this small region. We divide the sketch of proving Theorem 2.1 into three main steps.

1. First, we divide $[0,1]^d$ into a union of cubes $\{Q_{\boldsymbol{\beta}}\}_{\boldsymbol{\beta}\in\{0,1,\cdots,K-1\}^d}$ and a small region Ω with $K = \mathcal{O}(n^{(s+1)/d})$. Each $Q_{\boldsymbol{\beta}}$ is associated with a representative $\boldsymbol{x}_{\boldsymbol{\beta}} \in Q_{\boldsymbol{\beta}}$ for each vector index $\boldsymbol{\beta}$. See Figure 3 for an illustration for $K = 4$ and $d = 2$.
2. Next, we design a vector-valued function $\boldsymbol{\Phi}_1(\boldsymbol{x})$ to map the whole cube $Q_{\boldsymbol{\beta}}$ to its index $\boldsymbol{\beta}$ for each $\boldsymbol{\beta}$. Here, $\boldsymbol{\Phi}_1$ can be defined/constructed via
$$\boldsymbol{\Phi}_1(\boldsymbol{x}) = \big[\phi_1(x_1),\, \phi_1(x_2),\, \cdots,\, \phi_1(x_d)\big]^T,$$
where each one-dimensional function ϕ_1 is a step function outside a small region. We can efficiently construct ReLU NestNets with the desired size to approximate such an almost step function ϕ_1 with sufficiently many "steps" by using the composition architecture of ReLU NestNets. See the appendix for the detailed construction.
3. Finally, we need to construct a function ϕ_2 realized by a ReLU NestNet to map $\boldsymbol{\beta}$ approximately to $f(\boldsymbol{x}_{\boldsymbol{\beta}})$ for each $\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d$. Then we have
$$\phi_2 \circ \boldsymbol{\Phi}_1(\boldsymbol{x}) = \phi_2(\boldsymbol{\beta}) \approx f(\boldsymbol{x}_{\boldsymbol{\beta}}) \approx f(\boldsymbol{x}) \quad \text{for any } \boldsymbol{x} \in Q_{\boldsymbol{\beta}} \text{ and each } \boldsymbol{\beta},$$
implying
$$\phi \coloneqq \phi_2 \circ \boldsymbol{\Phi}_1 \approx f \quad \text{on } [0,1]^d \backslash \Omega.$$
Then, we can get a good approximation on $[0,1]^d$ by using Lemma 3.4 of our previous paper [24] to deal with the approximation inside Ω. We remark that, in the construction of $\phi_2 : \mathbb{R}^d \to \mathbb{R}$, we only need to care about the values of ϕ_2 at a set of K^d points $\{0,1,\cdots,K-1\}^d$. As we shall see later, this is the key point to ease the design of a ReLU NestNet with the desired size to realize ϕ_2.

See Figure 3 for an illustration of the above steps. Observe that in Figure 3, we have

$$\phi(\boldsymbol{x}) = \phi_2 \circ \boldsymbol{\Phi}_1(\boldsymbol{x}) = \phi_2(\boldsymbol{\beta}) \overset{\mathscr{E}_1}{\approx} f(\boldsymbol{x}_{\boldsymbol{\beta}}) \overset{\mathscr{E}_2}{\approx} f(\boldsymbol{x})$$

for any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and each $\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d$. That means $\phi - f$ is bounded by $\mathscr{E}_1 + \mathscr{E}_2$ on $[0,1]^d \backslash \Omega$. For any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and each $\boldsymbol{\beta}$, we have

$$\|\boldsymbol{x}_{\boldsymbol{\beta}} - \boldsymbol{x}\|_2 \le \sqrt{d}/K \quad\Longrightarrow\quad |f(\boldsymbol{x}_{\boldsymbol{\beta}}) - f(\boldsymbol{x})| \le \omega_f(\sqrt{d}/K) \quad\Longrightarrow\quad \mathscr{E}_2 \le \omega_f(\sqrt{d}/K).$$

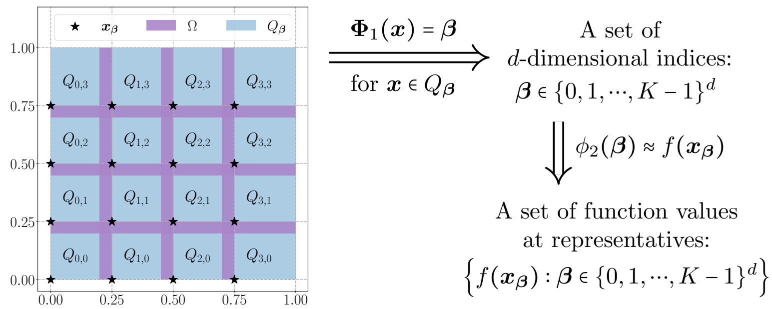


Figure 3: An illustration of the ideas of constructing $\phi = \phi_2 \circ \boldsymbol{\Phi}_1$ to approximate f for $K = 4$ and $d = 2$. Note that $\phi \approx f$ outside Ω since $\phi(\boldsymbol{x}) = \phi_2 \circ \boldsymbol{\Phi}_1(\boldsymbol{x}) = \phi_2(\boldsymbol{\beta}) \approx f(\boldsymbol{x}_{\boldsymbol{\beta}}) \approx f(\boldsymbol{x})$ for any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and each $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$.

The upper bound of $\mathscr{E}_1$ is determined by the construction of $\phi_2 : \mathbb{R}^d \to \mathbb{R}$. As stated previously, we only need to care about the values of ϕ_2 at a set of K^d points $\{0, 1, \cdots, K-1\}^d \subseteq \mathbb{R}^d$, which gives us much freedom to control $\mathscr{E}_1$. As we shall see later, $\mathscr{E}_1$ can be bounded by $\mathcal{O}\big(\omega_f(\sqrt{d}/K)\big)$. Therefore, $\phi - f$ is controlled by $\mathcal{O}\big(\omega_f(\sqrt{d}/K)\big)$ outside Ω, from which we deduce the desired approximation error on $[0,1]^d \backslash \Omega$ since $K = \mathcal{O}(n^{-(s+1)/d})$. Finally, by using Lemma 3.4 of our previous paper [24] to deal with the approximation inside Ω, we can get the desired approximation error on $[0,1]^d$.

2.3 Related work

We first compare our results with existing ones from an approximation perspective. Next, we discuss the parameter-sharing schemes of neural networks. Finally, we connect our NestNet architecture to existing trainable activation functions.

Discussion from an approximation perspective

The study of the approximation power of deep neural networks has become an active topic in recent years. This topic has been extensively studied from many perspectives, e.g., in terms of combinatorics [27], topology [7], information theory [29], fat-shattering dimension [1, 21], Vapnik-Chervonenkis (VC) dimension [6, 14, 32], classical approximation theory [3, 4, 8, 9, 10, 11, 12, 13, 18, 22, 24, 25, 28, 34, 35, 38, 39, 42, 48, 49, 52, 53], etc. To the best of our knowledge, the study of neural network approximation has two main stages: shallow (one-hidden-layer) networks and deep networks.

In the early works of neural network approximation, the approximation power of shallow networks is investigated. In particular, the universal approximation theorem [11, 17, 18], without approximation error estimate, showed that a sufficiently large neural network can approximate a target function in a certain function space arbitrarily well. For one-hidden-layer neural networks of width n and sufficiently smooth functions, an asymptotic approximation error $\mathcal{O}(n^{-1/2})$ in the L^2-norm is proved in [4, 5], leveraging an idea that is similar to Monte Carlo sampling for high-dimensional integrals.

Recently, a large number of works focus on the study of deep neural networks. It is shown in [35, 49, 52] that the optimal approximation error is $\mathcal{O}(n^{-2/d})$ by using ReLU networks with n parameters to approximate 1-Lipschitz continuous functions on $[0,1]^d$. This optimal approximation error follows a natural question: How can we get a better approximation error? Generally, there are two ideas to get better errors. The first one is to consider smaller function spaces, e.g., smooth functions [24, 50] and band-limited functions [26]. The other one is to introduce new networks, e.g., Floor-ReLU networks [36], Floor-Exponential-Step (FLES) networks [37], and (Sin, ReLU, 2^x)-activated networks [20].

This paper proposes a three-dimensional neural network architecture by introducing one more dimension called height beyond width and depth. As shown in Theorem 2.1 and Corollary 2.2, neural networks with three-dimensional architectures are significantly more expressive than the ones with two-dimensional architectures. We will conduct experiments to explore the numerical properties of NestNets in Section 3.

Discussion from a parameter-sharing perspective

As discussed previously, our NestNet architecture can be regarded as a sufficiently large standard network architecture with a specific parameter-sharing scheme. Parameter-sharing schemes are used in neural networks to control the overall number of parameters for reducing memory and communication costs. There are two common parameter-sharing schemes for a neural network. The first scheme is to share parameters in the same layer. A typical network example with this scheme is the convolutional neural network (CNN). In CNN architectures, filters in a CNN layer are shared for all channels, which means the parameters in the filters are shared. The second scheme is to share parameters across different layers of networks, e.g., recurrent neural networks.

In the NestNet architecture, we share parameters via repetitions of sub-network activation functions. Both of parameter-sharing schemes discussed just above are used in the NestNet architecture. The nested architecture of NestNets gives us much freedom to determine how many parameters to share. Beyond parameter-sharing schemes for a neural network, there are also parameter-sharing schemes among different neural networks or models, especially for multi-task learning. One may refer to [30, 33, 44, 45, 46, 51] for more discussion on parameter sharing in neural networks.

Connection to trainable activation functions

The key idea of trainable activation functions is to add a small number of trainable parameters to existing activation functions. Let us present several existing trainable activation functions as follows. A ReLU-like function is introduced in [15] by modifying the negative part of ReLU using a trainable parameter α, i.e., the parametric ReLU (PReLU) is defined as $\mathrm{PReLU}(x) \coloneqq \begin{cases} x & \text{if } x \ge 0 \\ \alpha x & \text{if } x < 0. \end{cases}$ A variant of ELU unit is introduced in [43] by adding two trainable parameters $\beta, \gamma > 0$, i.e., the parametric ELU (PELU) is given by $\mathrm{PELU}(x) \coloneqq \begin{cases} \beta/\gamma & \text{if } x \ge 0 \\ \beta(\exp(x/\gamma) - 1)x & \text{if } x < 0. \end{cases}$ Authors in [31] propose a type of flexible ReLU (FReLU), which is defined via $\mathrm{FReLU}(x) \coloneqq \mathrm{ReLU}(x+\alpha) + \beta$, where α and β are two trainable parameters. One may refer to [2] for a survey of modern trainable activation functions. To the best of our knowledge, most existing trainable activation functions can be regarded as parametric variants of the original activation functions. That is, they are attained via parameterizing the original activation functions with a small number of (typically 1 or 2) trainable parameters.

By contrast, activation functions in our NestNets are much more flexible. They can be (realized by) either complicated or simple sub-NestNets. That is, we can freely determine the number of parameters in the activation functions of NestNets. In other words, in NestNets, we can randomly distribute the parameters in the affine linear maps and activation functions. In short, compared to the networks with existing trainable activation functions, our NestNets are more flexible and have much more freedom in the choice of activation functions.

3 Experimentation

In this section, we will conduct experiments as a proof of concept to explore the numerical properties of ReLU NestNets. It is challenging to tune the hyper-parameters of large NestNets due to their nested architectures. Thus, our experimentation focuses on relatively small NestNets of height 2 and we introduce a simple sub-network activation function ϱ, which is realized by a trainable one-hidden-layer ReLU network of width 3. To be exact, ϱ is given by

$$\varrho(x) = \boldsymbol{w}_1^T \cdot (x\boldsymbol{w}_0 + \boldsymbol{b}_0) + b_1 \quad \text{for any } x \in \mathbb{R}, \tag{4}$$

where $\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{b}_0 \in \mathbb{R}^3$ and $b_1 \in \mathbb{R}$ are trainable parameters. There are 10 parameters in ϱ. The initial settings for ϱ in our experiments are $\boldsymbol{w}_0 = (1,1,1)$, $\boldsymbol{w}_1 = (1,1,-1)$, $\boldsymbol{b}_0 = (-0.2,-0.1,0.0)$, and $b_1 = 0$. We believe that NestNets can achieve good results in some real-world applications if proper optimization algorithms are developed for NestNets. In this paper, we only consider two classification problems: a synthetic classification problem based on the Archimedean spiral in Section 3.1 and an image classification problem corresponding to a standard benchmark dataset Fashion-MNIST [47] in Section 3.2. We remark that a classification function can be continuously extended to $\mathbb{R}^d$ if each class of samples are located in a bounded closed subset of $\mathbb{R}^d$ and these subsets are pairwise disjoint. That means we can apply our theory to classification problems.

3.1 Archimedean spiral

We will design a binary classification experiment by constructing two disjoint sets based on the Archimedean spiral, which can be described by the equation $r = a + b\theta$ in polar coordinates (r, θ) for given $a, b \in \mathbb{R}$. Let us first define two curves (Archimedean spirals) as follows:

$$\widetilde{\mathcal{C}}_i \coloneqq \Big\{(x, y) : x = r_i \cos\theta,\ y = r_i \sin\theta,\ r_i = a_i + b_i\theta,\ \theta \in [0, s\pi]\Big\},$$

for $i = 0, 1$, where $a_0 = 0$, $a_1 = 1$, $b_0 = b_1 = 1/\pi$, and $s = 30$. To simplify the discussion below, we normalize $\widetilde{\mathcal{C}}_i$ as $\mathcal{C}_i \subseteq [0, 1]^2$, where $\mathcal{C}_i$ is defined by

$$\mathcal{C}_i \coloneqq \Big\{(x, y) : x = \tfrac{\widetilde{x}}{2(s+2)} + \tfrac{1}{2},\ y = \tfrac{\widetilde{y}}{2(s+2)} + \tfrac{1}{2},\ (\widetilde{x}, \widetilde{y}) \in \widetilde{\mathcal{C}}_i\Big\},$$

for $i = 0, 1$. Then, we can define the two desired sets as follows:

$$\mathcal{S}_i \coloneqq \Big\{(u, v) : \sqrt{(u - x)^2 + (v - y)^2} \le \varepsilon,\ (x, y) \in \mathcal{C}_i\Big\},$$

for $i = 0, 1$, where $\varepsilon = 0.005$ in our experiments. See an illustration for $\mathcal{S}_0$ and $\mathcal{S}_1$ in Figure 4.

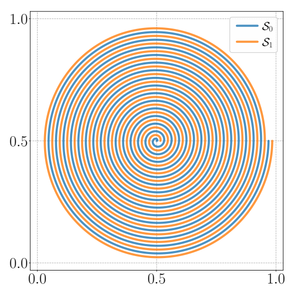


Figure 4: An illustration for $\mathcal{S}_0$ and $\mathcal{S}_1$.

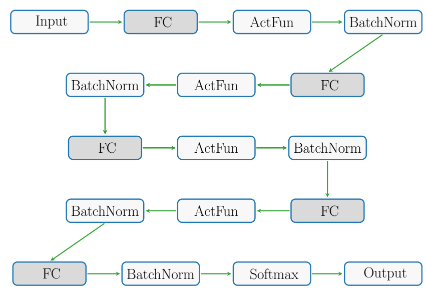


Figure 5: A network architecture illustration.

To explore the numerical performance of NestNets, we design NestNets and standard networks to classify samples in $\mathcal{S}_0 \cup \mathcal{S}_1$. We adopt four-hidden-layer fully connected network architecture of width 20, 35, or 50. To make the optimization more stable, we add the layers of batch normalization [19]. See Figure 5 for an illustration of the full network architecture. In Figure 5, FC and ActFun are short of fully connected layer and activation function, respectively. ActFun is ReLU for standard networks, while for NestNets, ActFun is the learnable sub-network activation function ϱ given in Equation (4).

Before presenting the experiment results, let us present the hyper-parameters for training the networks mentioned above. For each $i \in \{0, 1\}$, we randomly choose 3×10^5 training samples and 3×10^4 test samples in $\mathcal{S}_i$ with label i. Then, we use these 6×10^5 training samples to train the networks and use these 6×10^4 test samples to compute the test accuracy. We use the cross-entropy loss function to evaluate the loss between the networks and the target classification function. The number of epochs and the batch size are set to 500 and 512, respectively. We adopt RAdam [23] as the optimization method. In epochs $5(i - 1) + 1$ to $5i$ for $i = 1, 2, \cdots, 100$, the learning rate is $0.2 \times 0.002 \times 0.9^{i-1}$ for the parameters in ϱ and $0.002 \times 0.9^{i-1}$ for all other parameters. We remark that all training (test) samples are standardized before training, i.e., we rescale the samples to have a mean of 0 and a standard deviation of 1.

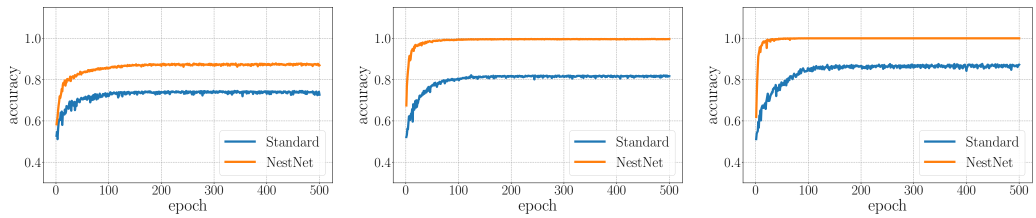
Finally, let us present the experiment results to compare the numerical performances of NestNets and standard networks. We adopt the average of test accuracies in the last 100 epochs as the target test accuracy. As we can see from Table 2 and Figure 6, by adding 10 more parameters (stored in ϱ), NestNets achieve much better test accuracies than standard networks though slightly more training time is required. In an "unfair" comparison, the test accuracy attained by the NestNet with 1.4×10^3 parameters is still better than that of the standard network with 7.9×10^3 parameters. This numerically verifies that the NestNet has much better approximation power than the standard network.

3.2 Fashion-MNIST

We will design convolutional neural network (CNN) architectures activated by ReLU or the sub-network activation function ϱ given in Equation (4) to classify image samples in Fashion-MNIST [47].

Table 2: Test accuracy comparison.

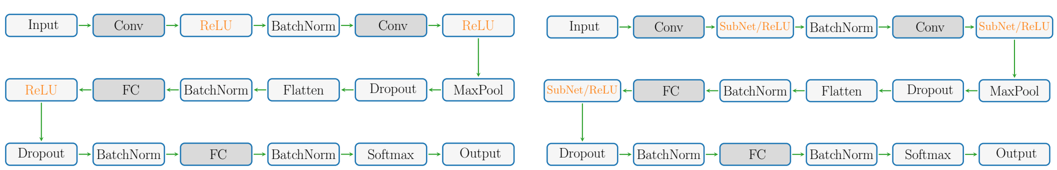
	width	depth	#parameters	activation function	training time	test accuracy
standard network	20	4	1362	ReLU	≈ 2532 s	0.738290
NestNet	20	4	1362 + 10	sub-network (ϱ)	≈ 4016 s	0.873631
standard network	35	4	3957	ReLU	≈ 2595 s	0.816048
NestNet	35	4	3957 + 10	sub-network (ϱ)	≈ 4104 s	0.995962
standard network	50	4	7902	ReLU	≈ 2642 s	0.866118
NestNet	50	4	7902 + 10	sub-network (ϱ)	≈ 4218 s	0.999984



(a) Width = 20. (b) Width = 35. (c) Width = 50.

Figure 6: Test accuracy over epochs.

This dataset consists of a training set of 6×10^4 samples and a test set of 10^4 samples. Each sample is a 28×28 grayscale image, associated with a label from 10 classes. To compare the numerical performances of NestNets and standard networks, we design a standard CNN architecture and a NestNet architecture that is constructed by replacing a few activation functions of a standard CNN network by the sub-network activation function ϱ. For simplicity, we denote the standard CNN and the NestNet as CNN1 and CNN2. To make the optimization more stable, we add the layers of dropout [16, 41] and batch normalization [19]. See illustrations of CNN1 and CNN2 in Figure 7. We present more details of them in Table 3.



(a) CNN1. (b) CNN2.

Figure 7: Illustrations of CNN1 and CNN2. Conv and FC represent convolutional and fully connected layers, respectively. CNN2 is indeed a NestNet of height 2.

Table 3: Details of CNN1 and CNN2.

layers	activation function		output size of each layer	dropout	batch normalization
	CNN1	CNN2			
input $\in \mathbb{R}^{28\times28}$			28×28		
Conv-1: $1 \times (3\times3)$, 12	ReLU	SubNet (ϱ), $1\times(26\times26)$ ReLU, $11\times(26\times26)$	$12 \times (26\times26)$		yes
Conv-2: $12 \times (3\times3)$, 12	ReLU	SubNet (ϱ), $1\times(24\times24)$ ReLU, $11\times(24\times24)$	1728 (MaxPool & Flatten)	0.25	yes
FC-1: 1728, 48	ReLU	SubNet (ϱ), 1 ReLU, 47	48	0.5	yes
FC-2: 48, 10			10 (Softmax)		yes
output $\in \mathbb{R}^{10}$					

Before presenting the numerical results, let us present the hyper-parameters for training two CNN architectures above. We use the cross-entropy loss function to evaluate the loss between the CNNs and the target classification function. The number of epochs and the batch size are set to 500 and 128, respectively. We adopt RAdam [23] as the optimization method and the weight decay of the optimizer is 0.0001. In epochs $5(i-1)+1$ to $5i$ for $i = 1, 2, \cdots, 100$, the learning rate is $0.2 \times 0.002 \times 0.9^{i-1}$

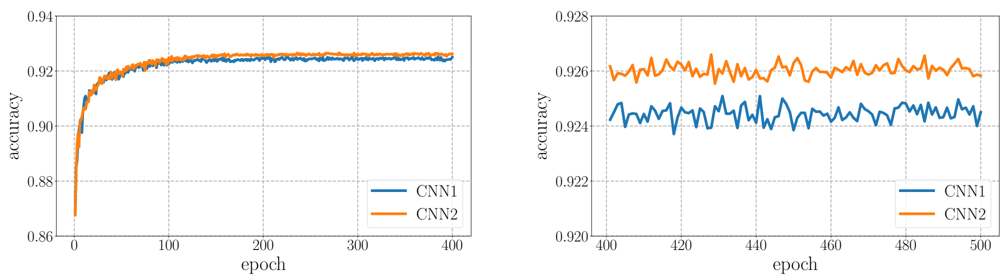
for the parameters in ϱ and $0.002 \times 0.9^{i-1}$ for all other parameters. All training (test) samples in the Fashion-MNIST dataset are standardized in our experiment, i.e., we rescale all training (test) samples to have a mean of 0 and a standard deviation of 1. In the settings above, we repeat the experiment 18 times and discard 3 top-performing and 3 bottom-performing trials by using the average of test accuracy in the last 100 epochs as the performance criterion. For each epoch, we adopt the average of test accuracies in the rest 12 trials as the target test accuracy.

Next, let us present the experiment results to compare the numerical performances of CNN1 and CNN2. The test accuracy comparison of CNN1 and CNN2 is summarized in Table 4.

Table 4: Test accuracy comparison.

	training time	largest accuracy	average of largest 100 accuracies	average accuracy in last 100 epochs
CNN1	≈ 5802 s	0.925290	0.924796	0.924447
CNN2	≈ 7217 s	0.926620	0.926287	0.926032

For each of CNN1 and CNN2, we present the training time, the largest test accuracy, the average of the largest 100 test accuracies, and the average of test accuracies in the last 100 epochs. For an intuitive comparison, we also provide illustrations of the test accuracy over epochs for CNN1 and CNN2 in Figure 8. As we can see from Table 4 and Figure 8, CNN2 performs better than CNN1 though slightly more training time and 10 more parameters are required. This numerically shows that the NestNet is significantly more expressive than the standard network.



(a) Epochs 1-400. (b) Epochs 401-500.

Figure 8: Test accuracy over epochs.

4 Conclusion

This paper proposes a three-dimensional neural network architecture by introducing one more dimension called height beyond width and depth. We show by construction that neural networks with three-dimensional architectures are significantly more expressive than the ones with two-dimensional architectures. We use simple numerical examples to show the advantages of the super-approximation power of ReLU NestNets, which is regarded as a proof of possibility. It would be of great interest to further explore the numerical performance of NestNets to bridge our theoretical results to applications. We believe that NestNets can be further developed and applied to real-world applications.

We remark that our analysis is limited to the ReLU activation function and the (Hölder) continuous function space. It would be interesting to generalize our results to other activation functions (e.g., tanh and sigmoid functions) and other function spaces (e.g, Lebesgue and Sobolev spaces).

Acknowledgments

Z. Shen was supported by Distinguished Professorship of National University of Singapore. H. Yang was partially supported by the US National Science Foundation under award DMS-2244988, DMS-2206333, and the Office of Naval Research Award N00014-23-1-2007.

References

[1] Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[2] Andrea Apicella, Francesco Donnarumma, Francesco Isgrò, and Roberto Prevete. A survey on modern trainable activation functions. *Neural Networks*, 138:14–32, 2021.

[3] Chenglong Bao, Qianxiao Li, Zuowei Shen, Cheng Tai, Lei Wu, and Xueshuang Xiang. Approximation analysis of convolutional neural networks. *Semantic Scholar e-Preprint*, page Corpus ID: 204762668, 2019.

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

[5] Andrew R. Barron and Jason M. Klusowski. Approximation and estimation for high-dimensional deep learning networks. *arXiv e-prints*, page arXiv:1809.03090, September 2018.

[6] Peter Bartlett, Vitaly Maiorov, and Ron Meir. Almost linear VC-dimension bounds for piecewise polynomial networks. *Neural Computation*, 10(8):2159–2173, 1998.

[7] Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems*, 25(8):1553–1565, Aug 2014.

[8] Helmut. Bölcskei, Philipp. Grohs, Gitta. Kutyniok, and Philipp. Petersen. Optimal approximation with sparsely connected deep neural networks. *SIAM Journal on Mathematics of Data Science*, 1(1):8–45, 2019.

[9] Minshuo Chen, Haoming Jiang, Wenjing Liao, and Tuo Zhao. Efficient approximation of deep ReLU networks for functions on low dimensional manifolds. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[10] Charles K. Chui, Shao-Bo Lin, and Ding-Xuan Zhou. Construction of neural networks for realization of localized deep learning. *Frontiers in Applied Mathematics and Statistics*, 4:14, 2018.

[11] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[12] Rémi Gribonval, Gitta Kutyniok, Morten Nielsen, and Felix Voigtlaender. Approximation spaces of deep neural networks. *Constructive Approximation*, 55:259–367, 2022.

[13] Ingo Gühring, Gitta Kutyniok, and Philipp Petersen. Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms. *Analysis and Applications*, 18(05):803–859, 2020.

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

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[16] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[17] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[18] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015.

[20] Yuling Jiao, Yanming Lai, Xiliang Lu, Fengru Wang, Jerry Zhijian Yang, and Yuanyuan Yang. Deep neural networks with ReLU-Sine-Exponential activations break curse of dimensionality on hölder class. *arXiv e-prints*, page arXiv:2103.00542, February 2021.

[21] Michael J. Kearns and Robert E. Schapire. Efficient distribution-free learning of probabilistic concepts. *J. Comput. Syst. Sci.*, 48(3):464–497, June 1994.

[22] Qianxiao Li, Ting Lin, and Zuowei Shen. Deep learning via dynamical systems: An approximation perspective. *Journal of the European Mathematical Society*, to appear.

[23] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2020.

[24] Jianfeng Lu, Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network approximation for smooth functions. *SIAM Journal on Mathematical Analysis*, 53(5):5465–5506, 2021.

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

[26] Hadrien Montanelli, Haizhao Yang, and Qiang Du. Deep ReLU networks overcome the curse of dimensionality for bandlimited functions. *Journal of Computational Mathematics*, 39(6):801–815, 2021.

[27] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2924–2932. Curran Associates, Inc., 2014.

[28] Ryumei Nakada and Masaaki Imaizumi. Adaptive approximation and generalization of deep neural network with intrinsic dimensionality. *Journal of Machine Learning Research*, 21(174):1–38, 2020.

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

[30] Bryan A. Plummer, Nikoli Dryden, Julius Frost, Torsten Hoefler, and Kate Saenko. Neural parameter allocation search. In *International Conference on Learning Representations*, 2022.

[31] Suo Qiu, Xiangmin Xu, and Bolun Cai. Frelu: Flexible rectified linear units for improving convolutional neural networks. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 1223–1228, Los Alamitos, CA, USA, aug 2018. IEEE Computer Society.

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

[33] Pedro Savarese and Michael Maire. Learning implicitly recurrent CNNs through parameter sharing. In *International Conference on Learning Representations*, 2019.

[34] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Nonlinear approximation via compositions. *Neural Networks*, 119:74–84, 2019.

[35] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network approximation characterized by number of neurons. *Communications in Computational Physics*, 28(5):1768–1811, 2020.

[36] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network with approximation error being reciprocal of width to power of square root of depth. *Neural Computation*, 33(4):1005–1036, 03 2021.

[37] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Neural network approximation: Three hidden layers are enough. *Neural Networks*, 141:160–173, 2021.

[38] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network approximation: Achieving arbitrary accuracy with fixed number of neurons. *Journal of Machine Learning Research*, 23(276):1–60, 2022.

[39] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network approximation in terms of intrinsic parameters. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 19909–19934. PMLR, 17–23 Jul 2022.

[40] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Optimal approximation rate of ReLU networks in terms of width and depth. *Journal de Mathématiques Pures et Appliquées*, 157:101–135, 2022.

[41] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[42] Taiji 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.

[43] Ludovic Trottier, Philippe Giguère, and Brahim Chaib-draa. Parametric exponential linear unit for deep convolutional neural networks. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 207–214, 2017.

[44] Matthew Wallingford, Hao Li, Alessandro Achille, Avinash Ravichandran, Charless Fowlkes, Rahul Bhotika, and Stefano Soatto. Task adaptive parameter sharing for multi-task learning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7551–7560, 2022.

[45] Jiaxing Wang, Haoli Bai, Jiaxiang Wu, Xupeng Shi, Junzhou Huang, Irwin King, Michael Lyu, and Jian Cheng. Revisiting parameter sharing for automatic neural channel number search. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5991–6002. Curran Associates, Inc., 2020.

[46] Ze Wang, Xiuyuan Cheng, Guillermo Sapiro, and Qiang Qiu. ACDC: Weight sharing in atom-coefficient decomposed convolution. *arXiv e-prints*, page arXiv:2009.02386, September 2020.

[47] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv e-prints*, page arXiv:1708.07747, August 2017.

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

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

[50] Dmitry Yarotsky and Anton Zhevnerchuk. The phase diagram of approximation rates for deep neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13005–13015. Curran Associates, Inc., 2020.

[51] Lijun Zhang, Qizheng Yang, Xiao Liu, and Hui Guan. Rethinking hard-parameter sharing in multi-domain learning. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 01–06, 2022.

[52] Shijun Zhang. Deep neural network approximation via function compositions. *PhD Thesis, National University of Singapore*, 2020. URL: https://scholarbank.nus.edu.sg/handle/10635/186064.

[53] Ding-Xuan Zhou. Universality of deep convolutional neural networks. *Applied and Computational Harmonic Analysis*, 48(2):787–794, 2020.

Contents of main article and appendix

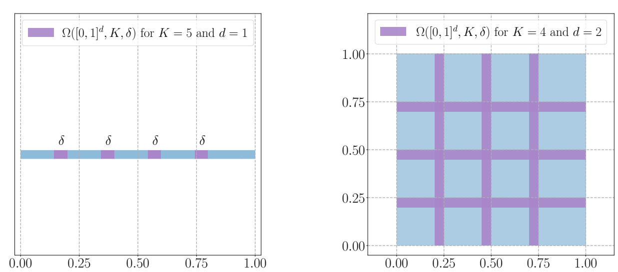
A Proof of main theorem

In this section, we will prove the main theorem, Theorem 2.1, based on an auxiliary theorem, Theorem A.1, which will be proved in Section B. Notations throughout this paper are summarized in Section A.1.

A.1 Notations

Let us summarize all basic notations used in this paper as follows.

- Let $\mathbb{R}$, $\mathbb{Q}$, and $\mathbb{Z}$ denote the set of real numbers, rational numbers, and integers, respectively.
- Let $\mathbb{N}$ and $\mathbb{N}^+$ denote the set of natural numbers and positive natural numbers, respectively. That is, $\mathbb{N}^+ = \{1, 2, 3, \cdots\}$ and $\mathbb{N} = \mathbb{N}^+ \bigcup \{0\}$.
- For any $x \in \mathbb{R}$, let $\lfloor x \rfloor := \max\{n : n \le x,\ n \in \mathbb{Z}\}$ and $\lceil x \rceil := \min\{n : n \ge x,\ n \in \mathbb{Z}\}$.
- Let $\mathbb{1}_S$ be the indicator (characteristic) function of a set S, i.e., $\mathbb{1}_S$ is equal to 1 on S and 0 outside S.
- The set difference of two sets A and B is denoted by $A \backslash B := \{x : x \in A,\ x \notin B\}$.
- Matrices are denoted by bold uppercase letters. For instance, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is a real matrix of size $m \times n$ and $\boldsymbol{A}^T$ denotes the transpose of $\boldsymbol{A}$. Vectors are denoted as bold lowercase letters. For example, $\boldsymbol{v} = [v_1, \cdots, v_d]^T = \begin{bmatrix} v_1 \\ \vdots \\ v_d \end{bmatrix} \in \mathbb{R}^d$ is a column vector.
- For any $p \in [1, \infty)$, the p-norm (or ℓ^p-norm) of a vector $\boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T \in \mathbb{R}^d$ is defined by
$$\|\boldsymbol{x}\|_p = \|\boldsymbol{x}\|_{\ell^p} := \big(|x_1|^p + |x_2|^p + \cdots + |x_d|^p\big)^{1/p}.$$
In the case of $p = \infty$,
$$\|\boldsymbol{x}\|_\infty = \|\boldsymbol{x}\|_{\ell^\infty} := \max\big\{|x_i| : i = 1, 2, \cdots, d\big\}.$$
- By convention, $\sum_{j=n_1}^{n_2} a_j = 0$ if $n_1 > n_2$, no matter what a_j is for each j.
- Given any $K \in \mathbb{N}^+$ and $\delta \in (0, \frac{1}{K})$, define a trifling region $\Omega([0,1]^d, K, \delta)$ of $[0,1]^d$ as
$$\Omega([0,1]^d, K, \delta) := \bigcup_{j=1}^{d} \Big\{ \boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T \in [0,1]^d : x_j \in \bigcup_{k=1}^{K-1} \big(\tfrac{k}{K} - \delta, \tfrac{k}{K}\big) \Big\}. \tag{5}$$
In particular, $\Omega([0,1]^d, K, \delta) = \varnothing$ if $K = 1$. See Figure 9 for two examples of trifling regions.



(a) (b)

Figure 9: Two examples of trifling regions. (a) $K = 5, d = 1$. (b) $K = 4, d = 2$.

- For a continuous piecewise linear function $f(x)$, the x values where the slope changes are typically called **breakpoints**.
- Let $\sigma : \mathbb{R} \to \mathbb{R}$ denote the rectified linear unit (ReLU), i.e. $\sigma(x) = \max\{0, x\}$ for any $x \in \mathbb{R}$. With a slight abuse of notation, we define $\sigma : \mathbb{R}^d \to \mathbb{R}^d$ as $\sigma(\boldsymbol{x}) = \begin{bmatrix} \max\{0, x_1\} \\ \vdots \\ \max\{0, x_d\} \end{bmatrix}$ for any $\boldsymbol{x} = [x_1, \cdots, x_d]^T \in \mathbb{R}^d$.

- Let $\mathcal{NN}_s\{n\}$ for $n, s \in \mathbb{N}^+$ denote the set of functions realized by height-s ReLU NestNets with as most n parameters.
- A function ϕ realized by a ReLU network can be briefly described as follows:

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

  where $\boldsymbol{W}_i \in \mathbb{R}^{N_{i+1}\times N_i}$ and $\boldsymbol{b}_i \in \mathbb{R}^{N_{i+1}}$ are the weight matrix and the bias vector in the i-th affine linear transformation $\boldsymbol{\mathcal{L}}_i$, respectively, i.e.,

$$\boldsymbol{h}_{i+1} = \boldsymbol{W}_i \cdot \widetilde{\boldsymbol{h}}_i + \boldsymbol{b}_i =: \boldsymbol{\mathcal{L}}_i(\widetilde{\boldsymbol{h}}_i) \quad \text{for } i = 0, 1, \cdots, L,$$

  and

$$\widetilde{\boldsymbol{h}}_i = \sigma(\boldsymbol{h}_i) \quad \text{for } i = 1, 2, \cdots, L.$$

  In particular, ϕ can be represented in a form of function compositions as follows

$$\phi = \boldsymbol{\mathcal{L}}_L \circ \sigma \circ \ \cdots \ \circ \boldsymbol{\mathcal{L}}_1 \circ \sigma \circ \boldsymbol{\mathcal{L}}_0,$$

  which has been illustrated in Figure 10.

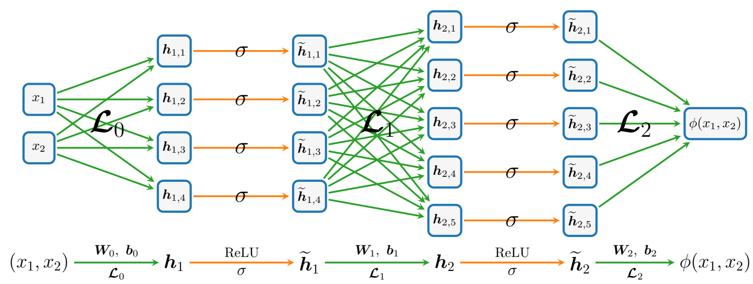


Figure 10: An example of a ReLU network of width 5 and depth 2.

- The expression "a network of width N and depth L" means
  - The number of neurons in each **hidden** layer of this network (architecture) is no more than N.
  - The number of **hidden** layers of this network (architecture) is no more than L.

A.2 Detailed proof of Theorem 2.1

The key point of proving Theorem 2.1 is to construct a piecewise constant function to approximate the target continuous function. However, ReLU NestNets are unable to approximate piecewise constant functions well the continuity of ReLU NestNets. Thus, we introduce the trifling region $\Omega([0,1]^d, K, \delta)$, defined in Equation (5), and use ReLU NestNets to implement piecewise constant functions outside the trifling region. To simplify the proof of Theorem 2.1, we introduce an auxiliary theorem, Theorem A.1 below. It can be regarded as a weaker variant of Theorem 2.1, ignoring the approximation in the trifling region.

Theorem A.1. *Given a continuous function $f \in C([0,1]^d)$, for any $n, s \in \mathbb{N}^+$, there exists $\phi \in \mathcal{NN}_s\big\{355d^2(s+7)^2(2n+1)\big\}$ such that $\|\phi\|_{L^\infty(\mathbb{R}^d)} \le |f(\boldsymbol{0})| + \omega_f(\sqrt{d})$ and*

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le 6\sqrt{d}\,\omega_f\big(n^{-(s+1)/d}\big) \quad \text{for any } \boldsymbol{x} \in [0,1]^d \backslash \Omega([0,1]^d, K, \delta),$$

where $K = \lfloor n^{(s+1)/d} \rfloor$ and δ is an arbitrary number in $(0, \frac{1}{3K}]$.

The proof of Theorem A.1 can be found in Section B. By assuming Theorem A.1 is true, we can easily prove Theorem 2.1 for the case $p \in [1, \infty)$. To prove Theorem 2.1 for the case $p = \infty$, we need to control the approximation error in the trifling region. To this intent, we introduce a theorem to handle the approximation inside the trifling region.

Theorem A.2 (Lemma 3.11 of [52] or Lemma 3.4 of [24]). *Given any $\varepsilon > 0$, $K \in \mathbb{N}^+$, and $\delta \in (0, \frac{1}{3K}]$, assume $f \in C([0,1]^d)$ and $g : \mathbb{R}^d \to \mathbb{R}$ is a general function with*

$$|g(\boldsymbol{x}) - f(\boldsymbol{x})| \le \varepsilon \quad \text{for any } \boldsymbol{x} \in [0,1]^d \backslash \Omega([0,1]^d, K, \delta).$$

Then

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le \varepsilon + d \cdot \omega_f(\delta) \quad \text{for any } \boldsymbol{x} \in [0,1]^d,$$

where $\phi \coloneqq \phi_d$ is defined by induction through $\phi_0 \coloneqq g$ and

$$\phi_{i+1}(\boldsymbol{x}) \coloneqq \mathrm{mid}\big(\phi_i(\boldsymbol{x} - \delta \boldsymbol{e}_{i+1}), \phi_i(\boldsymbol{x}), \phi_i(\boldsymbol{x} + \delta \boldsymbol{e}_{i+1})\big) \quad \text{for } i = 0, 1, \cdots, d-1,$$

where $\{\boldsymbol{e}_i\}_{i=1}^d$ is the standard basis in $\mathbb{R}^d$ and $\mathrm{mid}(\cdot,\cdot,\cdot)$ is the function returning the middle value of three inputs.

Now, let we prove Theorem 2.1 by assuming Theorem A.1 is true, the proof of which can be found in Section B.

Proof of Theorem 2.1. We may assume f is not a constant function since it is a trivial case. Then $\omega_f(r) > 0$ for any $r > 0$. Let us first consider the case $p \in [1, \infty)$. Set $K = \lfloor n^{(s+1)/d} \rfloor$ and choose a sufficiently small $\delta \in (0, \frac{1}{3K}]$ such that

$$\begin{aligned} Kd\delta\big(2|f(\boldsymbol{0})| + 2\omega_f(\sqrt{d})\big)^p &= \lfloor n^{(s+1)/d} \rfloor d\delta\Big(2|f(\boldsymbol{0})| + 2\omega_f(\sqrt{d})\Big)^p \\ &\le \Big(\omega_f(n^{-(s+1)/d})\Big)^p. \end{aligned}$$

By Theorem A.1, there exists

$$\begin{aligned} \phi \in \mathcal{NN}_s\big\{355 d^2 (s+7)^2 (2n+1)\big\} &\subseteq \mathcal{NN}_s\big\{355 d^2 (s+7)^2 \cdot 2(n+1)\big\} \\ &\subseteq \mathcal{NN}_s\big\{10^3 d^2 (s+7)^2 (n+1)\big\} \end{aligned}$$

such that $\|\phi\|_{L^\infty(\mathbb{R}^d)} \le |f(\boldsymbol{0})| + \omega_f(\sqrt{d})$ and

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le 6\sqrt{d}\,\omega_f\big(n^{-(s+1)/d}\big) \quad \text{for any } \boldsymbol{x} \in [0,1]^d \backslash \Omega([0,1]^d, K, \delta).$$

Since $\|f\|_{L^\infty([0,1]^d)} \le |f(\boldsymbol{0})| + \omega_f(\sqrt{d})$ and the Lebesgue measure of $\Omega([0,1]^d, K, \delta)$ is bounded by $Kd\delta$, we have

$$\begin{aligned} \|\phi - f\|_{L^p([0,1]^d)}^p &= \int_{\Omega([0,1]^d, K, \delta)} |\phi(\boldsymbol{x}) - f(\boldsymbol{x})|^p \mathrm{d}\boldsymbol{x} + \int_{[0,1]^d \backslash \Omega([0,1]^d, K, \delta)} |\phi(\boldsymbol{x}) - f(\boldsymbol{x})|^p \mathrm{d}\boldsymbol{x} \\ &\le Kd\delta\Big(2|f(\boldsymbol{0})| + 2\omega_f(\sqrt{d})\Big)^p + \Big(6\sqrt{d}\,\omega_f(n^{-(s+1)/d})\Big)^p \\ &\le \Big(\omega_f\big(n^{-(s+1)/d}\big)\Big)^p + \Big(6\sqrt{d}\,\omega_f(n^{-(s+1)/d})\Big)^p \le \Big(7\sqrt{d}\,\omega_f(n^{-(s+1)/d})\Big)^p. \end{aligned}$$

Hence, we have $\|\phi - f\|_{L^p([0,1]^d)} \le 7\sqrt{d}\,\omega_f\big(n^{-(s+1)/d}\big)$.

Next, let us discuss the case $p = \infty$. Set $K = \lfloor n^{(s+1)/d} \rfloor$ and choose a sufficiently small $\delta \in (0, \frac{1}{3K}]$ such that

$$d \cdot \omega_f(\delta) \le \omega_f\big(n^{-(s+1)/d}\big).$$

By Theorem A.1,

$$\phi_0 \in \mathcal{NN}_s\big\{355 d^2 (s+7)^2 (2n+1)\big\}$$

such that

$$|\phi_0(\boldsymbol{x}) - f(\boldsymbol{x})| \le 6\sqrt{d}\,\omega_f\big(n^{-(s+1)/d}\big) \quad \text{for any } \boldsymbol{x} \in [0,1]^d \backslash \Omega([0,1]^d, K, \delta).$$

By Theorem A.2 with $g = \phi_0$ and $\varepsilon = 6\sqrt{d}\,\omega_f\big(n^{-(s+1)/d}\big)$ therein, we have

$$|\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \le \varepsilon + d \cdot \omega_f(\delta) \le 7\sqrt{d}\,\omega_f\big(n^{-(s+1)/d}\big) \quad \text{for any } \boldsymbol{x} \in [0,1]^d,$$

where $\phi \coloneqq \phi_d$ is defined by induction through

$$\phi_{i+1}(\boldsymbol{x}) \coloneqq \mathrm{mid}\big(\phi_i(\boldsymbol{x} - \delta \boldsymbol{e}_{i+1}), \phi_i(\boldsymbol{x}), \phi_i(\boldsymbol{x} + \delta \boldsymbol{e}_{i+1})\big) \quad \text{for } i = 0, 1, \cdots, d-1,$$

where $\{\boldsymbol{e}_i\}_{i=1}^d$ is the standard basis in $\mathbb{R}^d$ and $\operatorname{mid}(\cdot,\cdot,\cdot)$ is the function returning the middle value of three inputs.

It remains to estimate the number of parameters in the NestNet realizing $\phi = \phi_d$. By Lemma 3.1 of [37], $\operatorname{mid}(\cdot,\cdot,\cdot)$ can be realized by a ReLU network of width 14 and depth 2, and hence with at most $14 \times (14+1) \times (2+1) = 630$ parameters.

By defining a vector-valued function $\boldsymbol{\Phi}_0 : \mathbb{R}^d \to \mathbb{R}^3$ as

$$\boldsymbol{\Phi}_0(\boldsymbol{x}) := \big[\phi_0(\boldsymbol{x} - \delta\boldsymbol{e}_1),\ \phi_0(\boldsymbol{x}),\ \phi_0(\boldsymbol{x} + \delta\boldsymbol{e}_1)\big]^T \quad \text{for any } \boldsymbol{x} \in \mathbb{R}^d,$$

we have $\boldsymbol{\Phi}_0 \in \mathcal{NN}_s\{3^2\big(355d^2(s+7)^2(2n+1)\big)\}$, implying

$$\begin{aligned}\phi_1 = \operatorname{mid}(\cdot,\cdot,\cdot) \circ \boldsymbol{\Phi}_0 &\in \mathcal{NN}_s\Big\{630 + 3^2\Big(355d^2(s+7)^2(2n+1)\Big)\Big\}\\ &\subseteq \mathcal{NN}_s\Big\{10\Big(355d^2(s+7)^2(2n+1)\Big)\Big\}.\end{aligned}$$

Similarly, we have

$$\begin{aligned}\phi = \phi_d \in \mathcal{NN}_s\Big\{10^d\Big(355d^2(s+7)^2(2n+1)\Big)\Big\} &\subseteq \mathcal{NN}_s\Big\{10^d\Big(355d^2(s+7)^2 \cdot 2(n+1)\Big)\Big\}\\ &\subseteq \mathcal{NN}_s\Big\{10^{d+3}d^2(s+7)^2(n+1)\Big\}.\end{aligned}$$

Thus, we finish the proof of Theorem 2.1.

□

B Proof of auxiliary theorem

We will prove the auxiliary theorem, Theorem A.1, in this section. We first present the key ideas in Section B.1. Next, the detailed proof is presented in Section B.2, based on two propositions in Section B.1, the proofs of which can be found in Sections C and D.

B.1 Key ideas of proving Theorem A.1

Our goal is to construct an almost piecewise constant function realized by a ReLU NestNet to approximate the target function $f \in C([0,1]^d)$ well. The construction can be divided into three main steps.

1. First, we divide $[0,1]^d$ into a union of "important" cubes $\{Q_{\boldsymbol{\beta}}\}_{\boldsymbol{\beta}\in\{0,1,\cdots,K-1\}^d}$ and the trifling region $\Omega([0,1]^d, K, \delta)$, where $K = \mathcal{O}(n^{(s+1)/d})$. Each $Q_{\boldsymbol{\beta}}$ is associated with a representative $\boldsymbol{x}_{\boldsymbol{\beta}} \in Q_{\boldsymbol{\beta}}$ for each vector index $\boldsymbol{\beta}$. See Figure 13 for illustrations.
2. Next, we design a vector-valued function $\boldsymbol{\Phi}_1(\boldsymbol{x})$ to map the whole cube $Q_{\boldsymbol{\beta}}$ to its index $\boldsymbol{\beta}$ for each $\boldsymbol{\beta}$. Here, $\boldsymbol{\Phi}_1$ can be defined/constructed via
$$\boldsymbol{\Phi}_1(\boldsymbol{x}) = \big[\phi_1(x_1),\ \phi_1(x_2),\ \cdots,\ \phi_1(x_d)\big]^T,$$
where each one-dimensional function ϕ_1 is a step function outside the trifling region and hence can be realized by a ReLU NestNet.
3. The aim of the final step is essentially to solve a point fitting problem. We will construct a function ϕ_2 realized by a ReLU NestNet to map $\boldsymbol{\beta}$ approximately to $f(\boldsymbol{x}_{\boldsymbol{\beta}})$ for each $\boldsymbol{\beta}$. Then we have
$$\phi_2 \circ \boldsymbol{\Phi}_1(\boldsymbol{x}) = \phi_2(\boldsymbol{\beta}) \approx f(\boldsymbol{x}_{\boldsymbol{\beta}}) \approx f(\boldsymbol{x}) \quad \text{for any } \boldsymbol{x} \in Q_{\boldsymbol{\beta}} \text{ and each } \boldsymbol{\beta},$$
implying
$$\phi := \phi_2 \circ \boldsymbol{\Phi}_1 \approx f \quad \text{on } [0,1]^d \backslash \Omega([0,1]^d, K, \delta).$$
We remark that, in the construction of ϕ_2, we only need to care about the values of ϕ_2 sampled inside the set $\{0,1,\cdots,K-1\}^d$, which is a key point to ease the design of a ReLU NestNet to realize ϕ_2 as we shall see later.

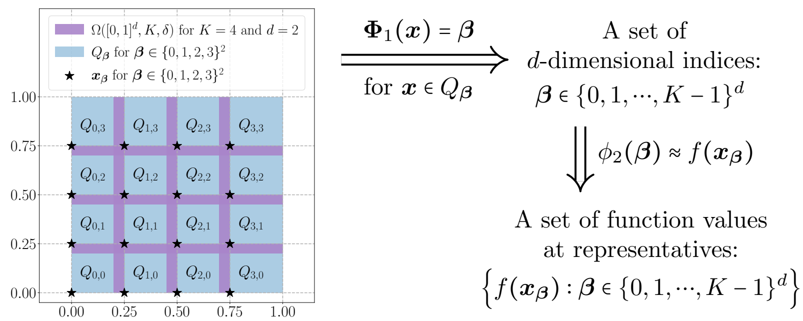


Figure 11: An illustration of the ideas of constructing the desired function $\phi = \phi_2 \circ \boldsymbol{\Phi}_1$. Note that $\phi \approx f$ outside the trifling region since $\phi(\boldsymbol{x}) = \phi_2 \circ \boldsymbol{\Phi}_1(\boldsymbol{x}) = \phi_2(\boldsymbol{\beta}) \approx f(\boldsymbol{x}_{\boldsymbol{\beta}}) \approx f(\boldsymbol{x})$ for any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and each $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$.

Observe that in Figure 11, we have

$$\phi(\boldsymbol{x}) = \phi_2 \circ \boldsymbol{\Phi}_1(\boldsymbol{x}) = \phi_2(\boldsymbol{\beta}) \overset{\mathscr{E}_1}{\approx} f(\boldsymbol{x}_{\boldsymbol{\beta}}) \overset{\mathscr{E}_2}{\approx} f(\boldsymbol{x})$$

for any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and each $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$. That means $\phi - f$ is controlled by $\mathscr{E}_1 + \mathscr{E}_2$ on $[0,1]^d \backslash \Omega([0,1]^d, K, \delta)$. Since $\|\boldsymbol{x} - \boldsymbol{x}_{\boldsymbol{\beta}}\|_2 \le \sqrt{d}/K$ for any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and each $\boldsymbol{\beta}$, $\mathscr{E}_2$ is bounded by $\omega_f(\sqrt{d}/K)$. As we shall see later, $\mathscr{E}_1$ can be bounded by $\mathcal{O}\big(\omega_f(\sqrt{d}/K)\big)$ by applying Proposition B.2. Therefore, $\phi - f$ is controlled by $\mathcal{O}\big(\omega_f(\sqrt{d}/K)\big)$ outside the trifling region, from which we deduce the desired approximation error since $K = \mathcal{O}(n^{-(s+1)/d})$.

Finally, we introduce two propositions to simplify the constructions of $\boldsymbol{\Phi}_1$ and ϕ_2 mentioned above. We first show how to construct a ReLU network to implement a one-dimensional step function ϕ_1 in Proposition B.1 below. Then $\boldsymbol{\Phi}_1$ can be defined via

$$\boldsymbol{\Phi}_1(\boldsymbol{x}) \coloneqq \big[\phi_1(x_1),\, \phi_1(x_2),\, \cdots,\, \phi_1(x_d)\big]^T \quad \text{for any } \boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T \in \mathbb{R}^d.$$

Proposition B.1. *Given any $n, r \in \mathbb{N}^+$, $\delta \in (0,1)$, and $J \in \mathbb{N}^+$ with $J \le 2^{n^r}$, there exists $\phi \in \mathcal{NN}_r\big\{36(r+7)n\big\}$ such that*

$$\phi(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{j=0}^{J-1} [j,\, j+1-\delta]$$

and

$$\phi(x) = J \quad \text{for any } x \in [J,\, J+1].$$

The construction of ϕ_2 is mainly based on Proposition B.2 below, whose proof relies on the bit extraction technique proposed in [6]. As we shall see later, some pre-processing is necessary for meeting the requirements of applying Proposition B.2 to construct ϕ_2.

Proposition B.2. *Given any $\varepsilon > 0$ and $n, s \in \mathbb{N}^+$, assume $y_j \ge 0$ for $j = 0, 1, \cdots, J-1$ are samples with $J \le n^{s+1}$ and*

$$|y_j - y_{j-1}| \le \varepsilon \quad \text{for } j = 1, 2, \cdots, J-1.$$

Then there exists $\phi \in \mathcal{NN}_s\big\{350(s+7)^2(n+1)\big\}$ such that

(i) $|\phi(j) - y_j| \le \varepsilon$ *for* $j = 0, 1, \cdots, J-1$.

(ii) $0 \le \phi(x) \le \max\{y_j : j = 0, 1, \cdots, J-1\}$ *for any* $x \in \mathbb{R}$.

The proofs of these two propositions can be found in Sections C and D. We will give the detailed proof of Theorem A.1 in Section B.2.

B.2 Detailed proof of Theorem A.1

We essentially construct an almost piecewise constant function realized by a ReLU NestNet with at most $\mathcal{O}(n)$ parameters to approximate f. We may assume f is not a constant function since it is a trivial case. Then $\omega_f(r) > 0$ for any $r > 0$. It is clear that $|f(\boldsymbol{x}) - f(\boldsymbol{0})| \le \omega_f(\sqrt{d})$ for any $\boldsymbol{x} \in [0,1]^d$. By defining $\widetilde{f} \coloneqq f - f(\boldsymbol{0}) + \omega_f(\sqrt{d})$, we have $\omega_{\widetilde{f}}(r) = \omega_f(r)$ for any $r \ge 0$ and $0 \le \widetilde{f}(\boldsymbol{x}) \le 2\omega_f(\sqrt{d})$ for any $\boldsymbol{x} \in [0,1]^d$.

Set $K = \lfloor n^{(s+1)/d} \rfloor$ and let δ be an arbitrary number in $(0, \frac{1}{3K}]$. The proof can be divided into four main steps as follows:

1. Divide $[0,1]^d$ into a union of sub-cubes $\{Q_{\boldsymbol{\beta}}\}_{\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d}$ and the trifling region $\Omega([0,1]^d, K, \delta)$, and denote $\boldsymbol{x}_{\boldsymbol{\beta}}$ as the vertex of $Q_{\boldsymbol{\beta}}$ with minimum $\|\cdot\|_1$ norm.
2. Construct a sub-network based on Proposition B.1 to implement a vector function $\boldsymbol{\Phi}_1$ projecting the whole cube $Q_{\boldsymbol{\beta}}$ to the d-dimensional index $\boldsymbol{\beta}$ for each $\boldsymbol{\beta}$, i.e., $\boldsymbol{\Phi}_1(\boldsymbol{x}) = \boldsymbol{\beta}$ for all $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$.
3. Construct a sub-network to implement a function ϕ_2 mapping the index $\boldsymbol{\beta}$ approximately to $\widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}})$. This core step can be further divided into three sub-steps:
   - 3.1. Construct a sub-network to implement ψ_1 bijectively mapping the index set $\{0,1,\cdots,K-1\}^d$ to an auxiliary set $\mathcal{A}_1 \subseteq \left\{\frac{j}{2K^d} : j = 0,1,\cdots,2K^d\right\}$ defined later. See Figure 14 for an illustration.
   - 3.2. Determine a continuous piecewise linear function g with a set of breakpoints $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \{1\}$, where $\mathcal{A}_2 \in \left\{\frac{j}{2K^d} : j = 0,1,\cdots,2K^d\right\}$ is a set defined later. Moreover, g should satisfy two conditions: 1) the values of g at breakpoints in $\mathcal{A}_1$ is given based on $\{\widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}})\}_{\boldsymbol{\beta}}$, i.e., $g \circ \psi_1(\boldsymbol{\beta}) = \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}})$; 2) the values of g at breakpoints in $\mathcal{A}_2 \cup \{1\}$ is defined to reduce the variation of g, which is necessary for applying Proposition B.2.
   - 3.3. Apply Proposition B.2 to construct a sub-network to implement a function ψ_2 approximating g well on $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \{1\}$. Then the desired function ϕ_2 is given by $\phi_2 = \psi_2 \circ \psi_1$ satisfying $\phi_2(\boldsymbol{\beta}) = \psi_2 \circ \psi_1(\boldsymbol{\beta}) \approx g \circ \psi_1(\boldsymbol{\beta}) = \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}})$.
4. Construct the final network to implement the desired function ϕ via $\phi = \phi_2 \circ \boldsymbol{\Phi}_1 + f(\boldsymbol{0}) - \omega_f(\sqrt{d})$. Then we have $\phi_2 \circ \boldsymbol{\Phi}_1(\boldsymbol{x}) = \phi_2(\boldsymbol{\beta}) \approx \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}}) \approx \widetilde{f}(\boldsymbol{x})$ for any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and $\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d$, implying $\phi(\boldsymbol{x}) = \phi_2 \circ \boldsymbol{\Phi}_1(\boldsymbol{x}) + f(\boldsymbol{0}) - \omega_f(\sqrt{d}) \approx \widetilde{f}(\boldsymbol{x}) + f(\boldsymbol{0}) - \omega_f(\sqrt{d}) = f(\boldsymbol{x})$.

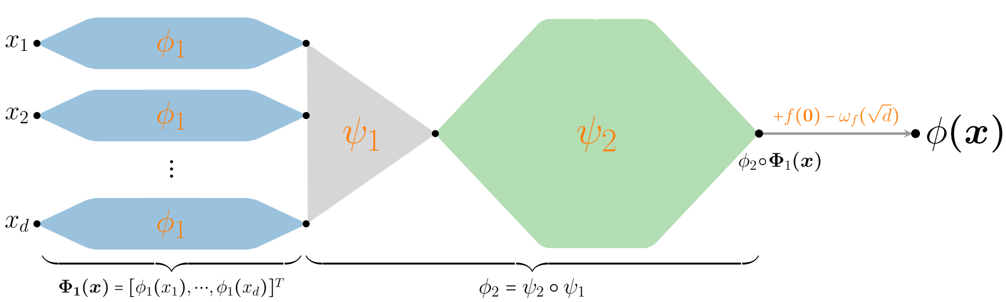


Figure 12: An illustration of the NestNet architecture realizing $\phi = \phi_2 \circ \boldsymbol{\Phi}_1 + f(\boldsymbol{0}) - \omega_f(\sqrt{d})$. Here, ϕ_1 is implemented via Proposition B.1; $\psi_1 : \mathbb{R}^d \to \mathbb{R}$ is an affine linear function; ψ_2 is implemented via Proposition B.2.

See Figure 12 for an illustration of the NestNet architecture realizing $\phi = \phi_2 \circ \boldsymbol{\Phi}_1 + f(\boldsymbol{0}) - \omega_f(\sqrt{d})$. The details of the steps mentioned above can be found below.

Step 1: Divide $[0,1]^d$ into $\{Q_{\boldsymbol{\beta}}\}_{\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d}$ and $\Omega([0,1]^d, K, \delta)$.

Define $\boldsymbol{x}_{\boldsymbol{\beta}} \coloneqq \boldsymbol{\beta}/K$ and

$$Q_{\boldsymbol{\beta}} \coloneqq \Big\{\boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T \in [0,1]^d : x_i \in \big[\tfrac{\beta_i}{K}, \tfrac{\beta_i+1}{K} - \delta \cdot \mathbb{1}_{\{\beta_i \le K-2\}}\big], \quad i = 1, 2, \cdots, d\Big\}$$

for each d-dimensional index $\boldsymbol{\beta} = [\beta_1, \beta_2, \cdots, \beta_d]^T \in \{0, 1, \cdots, K-1\}^d$. Recall that $\Omega([0,1]^d, K, \delta)$ is the trifling region defined in Equation (5). Apparently, $\boldsymbol{x}_{\boldsymbol{\beta}} = \boldsymbol{\beta}/K$ is the vertex of $Q_{\boldsymbol{\beta}}$ with minimum $\|\cdot\|_1$ norm and

$$[0,1]^d = \big(\cup_{\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d} Q_{\boldsymbol{\beta}}\big) \bigcup \Omega([0,1]^d, K, \delta).$$

See Figure 13 for illustrations.

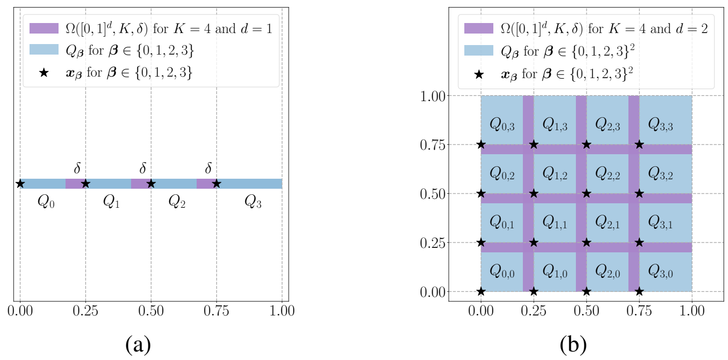


Figure 13: Illustrations of $\Omega([0,1]^d, K, \delta)$, $Q_{\boldsymbol{\beta}}$, and $\boldsymbol{x}_{\boldsymbol{\beta}}$ for $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$. (a) $K = 4$ and $d = 1$. (b) $K = 4$ and $d = 2$.

Step 2: Construct $\boldsymbol{\Phi}_1$ mapping $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ to $\boldsymbol{\beta}$.

Note that

$$K - 1 = \lfloor n^{(s+1)/d} \rfloor - 1 \le n^{s+1} \le \big(n^s\big)^2 \le 4^{(n^s)} = 2^{2(n^s)} \le 2^{(2n)^s} = 2^{\widetilde{n}^s},$$

where $\widetilde{n} = 2n$. By Proposition B.1 with $r = s$ and $J = K - 1 \le 2^{\widetilde{n}^s} = 2^{\widetilde{n}^r}$ therein, there exists

$$\widetilde{\phi}_1 \in \mathcal{NN}_s\big\{36(s+7)\widetilde{n}\big\} = \mathcal{NN}_s\big\{36(s+7)(2n)\big\} = \mathcal{NN}_s\big\{72(s+7)n\big\}$$

such that

$$\widetilde{\phi}_1(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{k=0}^{K-2} [k,\, k+1-\widetilde{\delta}] \text{ with } \widetilde{\delta} = K\delta$$

and

$$\widetilde{\phi}_1(x) = K - 1 \quad \text{for any } x \in [K-1,\, K].$$

Define $\phi_1(x) \coloneqq \widetilde{\phi}_1(Kx)$ for any $x \in \mathbb{R}$. Then, we have $\phi_1 \in \mathcal{NN}_s\big\{72(s+7)n\big\}$ and

$$\phi_1(x) = k \quad \text{if } x \in \big[\tfrac{k}{K}, \tfrac{k+1}{K} - \delta \cdot \mathbb{1}_{\{k \le K-2\}}\big] \quad \text{for } k = 0, 1, \cdots, K-1.$$

It follows that $\phi_1(x_i) = \beta_i$ if $\boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T \in Q_{\boldsymbol{\beta}}$ for each $\boldsymbol{\beta} = [\beta_1, \beta_2, \cdots, \beta_d]^T$.

By defining

$$\boldsymbol{\Phi}_1(\boldsymbol{x}) \coloneqq \big[\phi_1(x_1),\, \phi_1(x_2),\, \cdots,\, \phi_1(x_d)\big]^T \quad \text{for any } \boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T \in \mathbb{R}^d,$$

we have

$$\boldsymbol{\Phi}_1(\boldsymbol{x}) = \boldsymbol{\beta} \quad \text{if } \boldsymbol{x} \in Q_{\boldsymbol{\beta}} \quad \text{for each } \boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d. \tag{6}$$

Step 3: Construct ϕ_2 mapping $\boldsymbol{\beta}$ approximately to $\widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}})$.

The construction of the sub-network implementing ϕ_2 is essentially based on Proposition B.2. To meet the requirements of applying Proposition B.2, we first define two auxiliary sets $\mathcal{A}_1$ and $\mathcal{A}_2$ as

$$\mathcal{A}_1 \coloneqq \Big\{\tfrac{i}{K^{d-1}} + \tfrac{k}{2K^d} : i = 0, 1, \cdots, K^{d-1} - 1 \quad \text{and} \quad k = 0, 1, \cdots, K-1\Big\}$$

and

$$\mathcal{A}_2 \coloneqq \Big\{ \tfrac{i}{K^{d-1}} + \tfrac{K+k}{2K^d} : i = 0, 1, \cdots, K^{d-1}-1 \quad \text{and} \quad k = 0, 1, \cdots, K-1 \Big\}.$$

Clearly,

$$\mathcal{A}_1 \cup \mathcal{A}_2 \cup \{1\} = \big\{ \tfrac{j}{2K^d} : j = 0, 1, \cdots, 2K^d \big\} \quad \text{and} \quad \mathcal{A}_1 \cap \mathcal{A}_2 = \varnothing.$$

See Figure 13 for an illustration of $\mathcal{A}_1$ and $\mathcal{A}_2$. Next, we further divide this step into three sub-steps.

Step 3.1: Construct ψ_1 bijectively mapping $\{0, 1, \cdots, K-1\}^d$ to $\mathcal{A}_1$.

Inspired by the binary representation, we define

$$\psi_1(\boldsymbol{x}) \coloneqq \frac{x_d}{2K^d} + \sum_{i=1}^{d-1} \frac{x_i}{K^i} \quad \text{for any } \boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T \in \mathbb{R}^d. \tag{7}$$

Then ψ_1 is a linear function bijectively mapping the index set $\{0, 1, \cdots, K-1\}^d$ to

$$\begin{aligned} \Big\{ \psi_1(\boldsymbol{\beta}) : \boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d \Big\} &= \Big\{ \tfrac{\beta_d}{2K^d} + \sum_{i=1}^{d-1} \tfrac{\beta_i}{K^i} : \boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d \Big\} \\ = \Big\{ \tfrac{i}{K^{d-1}} + \tfrac{k}{2K^d} : i = 0, 1, \cdots, K^{d-1}-1 \quad \text{and} \quad k = 0, 1, \cdots, K-1 \Big\} &= \mathcal{A}_1. \end{aligned}$$

Step 3.2: Construct g to satisfy $g \circ \psi_1(\boldsymbol{\beta}) = \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}})$ and to meet the requirements of applying Proposition B.2.

Let $g : [0, 1] \to \mathbb{R}$ be a continuous piecewise linear function with a set of breakpoints

$$\big\{ \tfrac{j}{2K^d} : j = 0, 1, \cdots, 2K^d \big\} = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \{1\}.$$

Moreover, the values of g at these breakpoints are assigned as follows:

- At the breakpoint 1, let $g(1) = \widetilde{f}(\mathbf{1})$, where $\mathbf{1} = [1, 1, \cdots, 1]^T \in \mathbb{R}^d$.
- For the breakpoints in $\mathcal{A}_1 = \big\{ \psi_1(\boldsymbol{\beta}) : \boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d \big\}$, we set

$$g\big(\psi_1(\boldsymbol{\beta})\big) = \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}}) \quad \text{for any } \boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d. \tag{8}$$

- The values of g at the breakpoints in $\mathcal{A}_2$ are assigned to reduce the variation of g, which is a requirement of applying Proposition B.2. Recall that

$$\big\{ \tfrac{i}{K^{d-1}} - \tfrac{K+1}{2K^d}, \; \tfrac{i}{K^{d-1}} \big\} \subseteq \mathcal{A}_1 \cup \{1\} \quad \text{for } i = 1, 2, \cdots, K^{d-1},$$

  implying the values of g at $\frac{i}{K^{d-1}} - \frac{K+1}{2K^d}$ and $\frac{i}{K^{d-1}}$ have been assigned in the previous cases for. Thus, the values of g at the breakpoints in $\mathcal{A}_2$ can be successfully assigned by letting g linear on each interval $[\frac{i}{K^{d-1}} - \frac{K+1}{2K^d}, \frac{i}{K^{d-1}}]$ for $i = 1, 2, \cdots, K^{d-1}$ since $\mathcal{A}_2 \subseteq \bigcup_{i=1}^{K^{d-1}} [\frac{i}{K^{d-1}} - \frac{K+1}{2K^d}, \frac{i}{K^{d-1}}]$. See Figure 14 for an illustration.

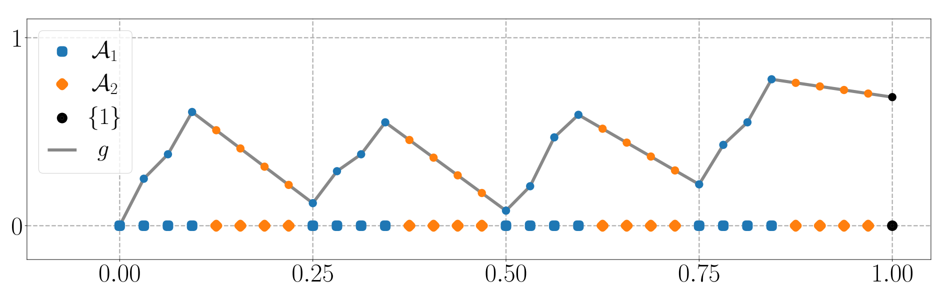


Figure 14: An illustration of $\mathcal{A}_1$, $\mathcal{A}_2$, $\{1\}$, and g for $K = 4$ and $d = 2$.

Apparently, such a function g exists. See Figure 14 for an illustration of g. It is easy to verify that

$$\left|g(\tfrac{j}{2K^d}) - g(\tfrac{j-1}{2K^d})\right| \le \max\left\{\omega_{\widetilde{f}}(\tfrac{\sqrt{d}}{K}),\ \tfrac{\omega_{\widetilde{f}}(\sqrt{d})}{K}\right\} \le \omega_{\widetilde{f}}(\tfrac{\sqrt{d}}{K}) = \omega_f(\tfrac{\sqrt{d}}{K})$$

for $j = 1, 2, \cdots, 2K^d$. Moreover, we have

$$0 \le g(\tfrac{j}{2K^d}) \le 2\omega_f(\sqrt{d}) \quad \text{for } j = 0, 1, \cdots, 2K^d.$$

Step 3.3: Construct ψ_2 approximating g well on $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \{1\}$.

Observe that

$$2K^d = 2\big(\lfloor n^{(s+1)/d} \rfloor\big)^d \le 2n^{s+1} \le (2n)^{s+1} = \widetilde{n}^{s+1}, \quad \text{where } \widetilde{n} = 2n.$$

By Proposition B.2 with $y_j = g(\frac{j}{2K^2})$ and $\varepsilon = \omega_f(\frac{\sqrt{d}}{K}) > 0$ therein, there exists

$$\widetilde{\psi}_2 \in \mathcal{NN}_s\Big\{350(s+7)^2(\widetilde{n}+1)\Big\} = \mathcal{NN}_s\Big\{350(s+7)^2(2n+1)\Big\}$$

such that

$$|\widetilde{\psi}_2(j) - g(\tfrac{j}{2K^d})| \le \omega_f(\tfrac{\sqrt{d}}{K}) \quad \text{for } j = 0, 1, \cdots, 2K^d - 1$$

and

$$0 \le \widetilde{\psi}_2(x) \le \max\left\{g(\tfrac{j}{2K^d}) : j = 0, 1, \cdots, 2K^d - 1\right\} \le 2\omega_f(\sqrt{d}) \quad \text{for any } x \in \mathbb{R}.$$

By defining $\psi_2(x) \coloneqq \widetilde{\psi}_2(2K^d x)$ for any $x \in \mathbb{R}$, we have

$$0 \le \psi_2(x) = \widetilde{\psi}_2(2K^d x) \le 2\omega_f(\sqrt{d}) \quad \text{for any } x \in \mathbb{R} \tag{9}$$

and

$$|\psi_2(\tfrac{j}{2K^d}) - g(\tfrac{j}{2K^d})| = |\widetilde{\psi}_2(j) - g(\tfrac{j}{2K^d})| \le \omega_f(\tfrac{\sqrt{d}}{K}) \quad \text{for } j = 0, 1, \cdots, 2K^d - 1. \tag{10}$$

Let us end Step 3 by defining the desired function ϕ_2 as $\phi_2 \coloneqq \psi_2 \circ \psi_1$. Recall that $\psi_1(\boldsymbol{\beta}) = \mathcal{A}_1 \subseteq \left\{\frac{j}{2K^d} : j = 0, 1, \cdots, 2K^d - 1\right\}$. Then, by Equations (8) and (10), we have

$$\left|\phi_2(\boldsymbol{\beta}) - \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}})\right| = \left|\psi_2(\psi_1(\boldsymbol{\beta})) - g(\psi_1(\boldsymbol{\beta}))\right| \le \omega_f(\tfrac{\sqrt{d}}{K}) \tag{11}$$

for any $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$. Moreover, by Equation (9) and $\phi_2 = \psi_2 \circ \psi_1$, we have

$$0 \le \phi_2(\boldsymbol{x}) = \psi_2\big(\psi(\boldsymbol{x})\big) \le 2\omega_f(\sqrt{d}) \quad \text{for any } \boldsymbol{x} \in \mathbb{R}^d. \tag{12}$$

Step 4: Construct the final network to implement the desired function ϕ.

Define $\phi \coloneqq \phi_2 \circ \boldsymbol{\Phi}_1 + f(\boldsymbol{0}) - \omega_f(\sqrt{d})$. By Equation (12), we have

$$0 \le \phi_2 \circ \boldsymbol{\Phi}_1(\boldsymbol{x}) \le 2\omega_f(\sqrt{d})$$

for any $\boldsymbol{x} \in \mathbb{R}^d$, implying

$$f(\boldsymbol{0}) - \omega_f(\sqrt{d}) \le \phi(\boldsymbol{x}) = \phi_2 \circ \boldsymbol{\Phi}_1(\boldsymbol{x}) + f(\boldsymbol{0}) - \omega_f(\sqrt{d}) \le f(\boldsymbol{0}) + \omega_f(\sqrt{d}).$$

It follows that $\|\phi\|_{L^\infty(\mathbb{R}^d)} \le |f(\boldsymbol{0})| + \omega_f(\sqrt{d})$.

Next, let us estimate the approximation error. Recall that $f = \widetilde{f} + f(\boldsymbol{0}) - \omega_f(\sqrt{d})$ and $\phi = \phi_2 \circ \boldsymbol{\Phi}_1 + f(\boldsymbol{0}) - \omega_f(\sqrt{d})$. By Equations (6) and (11), for any $\boldsymbol{x} \in Q_{\boldsymbol{\beta}}$ and $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$, we have

$$\begin{aligned}
|f(\boldsymbol{x}) - \phi(\boldsymbol{x})| &= \left|\widetilde{f}(\boldsymbol{x}) - \phi_2 \circ \boldsymbol{\Phi}_1(\boldsymbol{x})\right| = |\widetilde{f}(\boldsymbol{x}) - \phi_2(\boldsymbol{\beta})| \\
&\le |\widetilde{f}(\boldsymbol{x}) - \widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}})| + |\widetilde{f}(\boldsymbol{x}_{\boldsymbol{\beta}}) - \phi_2(\boldsymbol{\beta})| \\
&\le \omega_f(\tfrac{\sqrt{d}}{K}) + \omega_f(\tfrac{\sqrt{d}}{K}) \le 2\omega_f\big(2\sqrt{d}\, n^{-(s+1)/d}\big),
\end{aligned}$$

where the last inequality comes from the fact

$$K = \lfloor n^{(s+1)/d} \rfloor \geq n^{(s+1)/d}/2 \quad \text{for } n \in \mathbb{N}^+.$$

Recall the fact $\omega_f(j \cdot r) \leq j \cdot \omega_f(r)$ for any $j \in \mathbb{N}^+$ and $r \in [0, \infty)$. Therefore, for any $\boldsymbol{x} \in \bigcup_{\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d} Q_{\boldsymbol{\beta}} = [0,1]^d \backslash \Omega([0,1]^d, K, \delta)$, we have

$$\begin{aligned} |\phi(\boldsymbol{x}) - f(\boldsymbol{x})| \leq 2\omega_f\Big(2\sqrt{d}\, n^{-(s+1)/d}\Big) &\leq 2\Big\lceil 2\sqrt{d} \Big\rceil \omega_f\Big(n^{-(s+1)/d}\Big) \\ &\leq 6\sqrt{d}\, \omega_f\Big(n^{-(s+1)/d}\Big). \end{aligned}$$

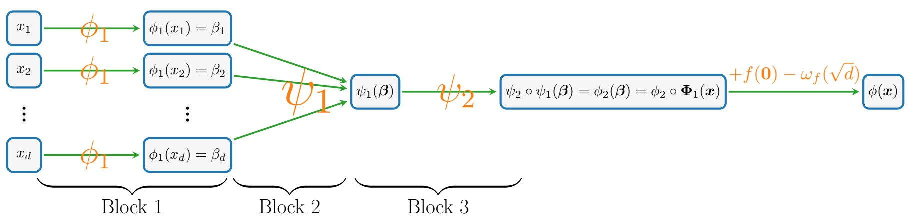


Figure 15: An illustration of the final NestNet realizing $\phi = \phi_2 \circ \boldsymbol{\Phi}_1 + f(\boldsymbol{0}) - \omega_f(\sqrt{d})$ for $\boldsymbol{x} = [x_1, x_2, \cdots, x_d]^T \in Q_{\boldsymbol{\beta}}$ for each $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$.

It remains to estimate the number of parameters in the NestNet realizing ϕ, which is shown in Figure 15. Recall that $\phi_1 \in \mathcal{NN}_s\big\{72(s+7)n\big\}$, ψ_1 is an affine linear map, and $\psi_2 \in \mathcal{NN}_s\big\{350(s+7)^2(2n+1)\big\}$. Therefore, $\phi = \phi_2 \circ \boldsymbol{\Phi}_1 + f(\boldsymbol{0}) - \omega_f(\sqrt{d})$ can be realized by a height-s NestNet with at most

$$\underbrace{d^2\big(72(s+7)n\big)}_{\text{Block 1}} + \underbrace{(d+1)}_{\text{Block 2}} + \underbrace{350(s+7)^2(2n+1)}_{\text{Block 3}} + 1 \;\leq\; 355 d^2 (s+7)^2 (2n+1)$$

parameters, which means we finish the proof of Theorem A.1.

C Proof of Proposition B.1

The key point of proving Proposition B.1 is the composition architecture of neural networks. To simplify the proof, we first establish several lemmas for proving Proposition B.1 in Section C.1. Next, we present the detailed proof of Proposition B.1 in Section C.2 based on the lemmas established in Section C.1.

C.1 Lemmas for proving Proposition B.1

Lemma C.1. *Given any $n, r \in \mathbb{N}^+$ and $\delta \in \big(0, \frac{1}{C(r,n)}\big)$ with $C(r,n) = \prod_{i=1}^r 2^{n^i}$, there exists $\phi \in \mathcal{NN}_r\big\{(12r+68)n\big\}$ such that*

$$\phi(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{\ell=0}^{2^{n^r}-1} \big[\ell,\, \ell + 1 - C(r,n)\cdot\delta\big].$$

We will prove Lemma C.1 by induction. To simplify the proof, we introduce two lemmas for the base case and the induction step.

First, we introduce the following lemma for the base case of proving Lemma C.1.

Lemma C.2. *Given any $n \in \mathbb{N}^+$ and $\delta \in (0,1)$, there exists a function ϕ realized by a ReLU network of width 4 and depth $4n-1$ such that*

$$\phi(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{\ell=0}^{2^n-1} [\ell,\, \ell + 1 - \delta].$$

Proof. Set $\widetilde{\delta} = 2^{-n}\delta$ and define

$$\phi_0(x) \coloneqq \frac{\sigma(x-1+\widetilde{\delta}) - \sigma(x-1)}{\widetilde{\delta}} \quad \text{for } x \in \mathbb{R}.$$

Clearly, ϕ_0 can be realized by a one-hidden-layer ReLU network of width 2. Moreover, we have

$$\phi_0(x) = \frac{\sigma(x-1+\widetilde{\delta}) - \sigma(x-1)}{\widetilde{\delta}} = \frac{0-0}{\widetilde{\delta}} = 0 \quad \text{if } x \in [0, 1-\widetilde{\delta}]$$

and

$$\phi_0(x) = \frac{\sigma(x-1+\widetilde{\delta}) - \sigma(x-1)}{\widetilde{\delta}} = \frac{(x-1+\widetilde{\delta}) - (x-1)}{\widetilde{\delta}} = 1 \quad \text{if } x \in [1, 2-\widetilde{\delta}].$$

By fixing

$$x \in \bigcup_{\ell=0}^{2^n-1} [\ell,\, \ell+1-\delta] = \bigcup_{\ell=0}^{2^n-1} [\ell,\, \ell+1-2^n\widetilde{\delta}],$$

we have $\lfloor x \rfloor \in \{0, 1, \cdots, 2^n-1\}$, implying that $\lfloor x \rfloor$ can be represented as

$$\lfloor x \rfloor = \sum_{i=0}^{n-1} z_i 2^i \quad \text{for } z_0, z_1, \cdots, z_{n-1} \in \{0,1\}.$$

Then, for $j = 0, 1, \cdots, n-1$, we have $\sum_{i=0}^{j} z_i 2^i + 1 \le z_j 2^j + \sum_{i=0}^{j-1} 2^i + 1 \le z_j 2^j + 2^j$, implying

$$\begin{aligned} \tfrac{x - \sum_{i=j+1}^{n-1} z_i 2^i}{2^j} \in \Big[\tfrac{\lfloor x \rfloor - \sum_{i=j+1}^{n-1} z_i 2^i}{2^j},\ \tfrac{\lfloor x \rfloor + 1 - 2^n\widetilde{\delta} - \sum_{i=j+1}^{n-1} z_i 2^i}{2^j}\Big] &= \Big[\tfrac{\sum_{i=0}^{j} z_i 2^i}{2^j},\ \tfrac{\sum_{i=0}^{j} z_i 2^i + 1 - 2^n\widetilde{\delta}}{2^j}\Big] \\ &\subseteq \Big[\tfrac{z_j 2^j}{2^j},\ \tfrac{z_j 2^j + 2^j - 2^n\widetilde{\delta}}{2^j}\Big] \subseteq [z_j,\, z_j + 1 - \widetilde{\delta}]. \end{aligned}$$

It follows that

$$\phi_0\Big(\tfrac{x - \sum_{i=j+1}^{n-1} z_i 2^i}{2^j}\Big) = z_j \quad \text{for } j = 0, 1, \cdots, n-1.$$

Therefore, the desired function ϕ can be realized by the network in Figure 16.

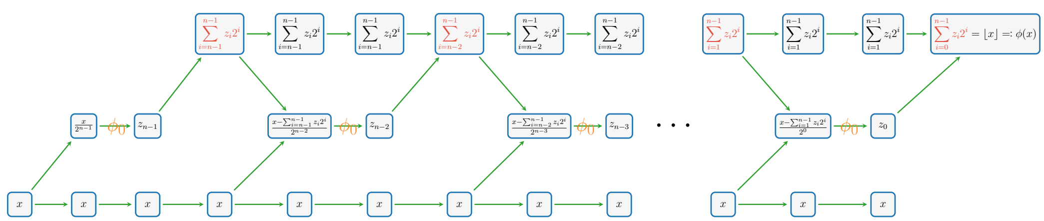


Figure 16: An illustration of the NestNet realizing ϕ. Here, ϕ_0 represent an one-hidden-layer ReLU network of width 2.

Clearly,

$$\phi(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{\ell=0}^{2^n-1} [\ell,\, \ell+1-\delta].$$

Moreover, ϕ can be realized by a ReLU network of width $1+2+1=4$ and depth $(1+1+1)+(1+1+1+1)(n-1) = 4n-1$. Hence, we finish the proof of Lemma C.2. □

Next, we introduce the following lemma for the induction step of proving Lemma C.1.

Lemma C.3. *Given any $n, s, \widehat{n} \in \mathbb{N}^+$ and $\delta \in \big(0, \frac{1}{2n^{s+1}}\big)$, if $g \in \mathcal{NN}_s\{\widehat{n}\}$ satisfying*

$$g(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{\ell=0}^{2^{n^s}-1} [\ell,\, \ell+1-\delta].$$

Then there exists $\phi \in \mathcal{NN}_{s+1}\big\{\widehat{n} + 12n - 7\big\}$ such that

$$\phi(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{\ell=0}^{2^{n^{s+1}}-1} [\ell,\, \ell+1-2^{n^{s+1}}\delta].$$

Proof. By setting $m = 2^{n^s}$, we have $m^n = \left(2^{n^s}\right)^n = 2^{(n^s)n} = 2^{n^{s+1}}$ and

$$g(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{\ell=0}^{m-1} [\ell,\, \ell+1-\delta]. \tag{13}$$

By fixing

$$x \in \bigcup_{\ell=0}^{2^{n^{s+1}}-1} [\ell,\, \ell+1-2^{n^{s+1}}\delta] = \bigcup_{\ell=0}^{m^n-1} [\ell,\, \ell+1-m^n\delta],$$

we have $\lfloor x \rfloor \in \{0,1,\cdots,m^n-1\}$, implying that $\lfloor x \rfloor$ can be represented as

$$\lfloor x \rfloor = \sum_{i=0}^{n-1} z_i m^i \quad \text{for } z_0, z_1, \cdots, z_{n-1} \in \{0,1,\cdots,m-1\}.$$

Then, for $j = 0,1,\cdots,n-1$, we have

$$\sum_{i=0}^{j} z_i m^i + 1 \le z_j m^j + \sum_{i=0}^{j-1} (m-1)m^i + 1 = z_j m^j + m^j,$$

implying

$$\begin{aligned}
\frac{x - \sum_{i=j+1}^{n-1} z_i m^i}{m^j} &\in \left[\frac{\lfloor x \rfloor - \sum_{i=j+1}^{n-1} z_i m^i}{m^j},\ \frac{\lfloor x \rfloor + 1 - m^n\delta - \sum_{i=j+1}^{n-1} z_i m^i}{m^j} \right] \\
&= \left[\frac{\sum_{i=0}^{j} z_i m^i}{m^j},\ \frac{\sum_{i=0}^{j} z_i m^i + 1 - m^n\delta}{m^j} \right] \\
&\subseteq \left[\frac{z_j m^j}{m^j},\ \frac{z_j m^j + m^j - m^n\delta}{m^j} \right] \subseteq \left[z_j,\ z_j + 1 - \delta \right].
\end{aligned}$$

It follows that

$$g\left(\frac{x - \sum_{i=j+1}^{n-1} z_i m^i}{m^j} \right) = z_j \quad \text{for } j = 0,1,\cdots,n-1.$$

Therefore, the desired function ϕ can be realized by the network in Figure 17.

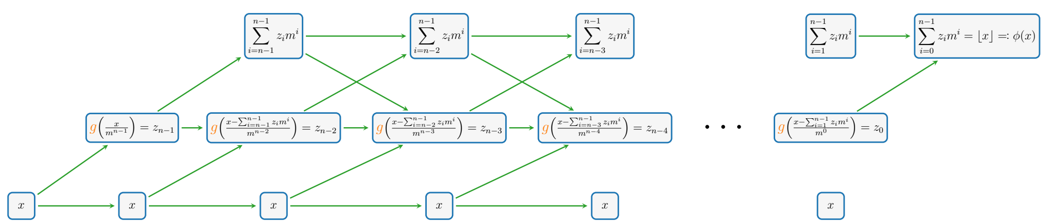


Figure 17: An illustration of the NestNet realizing ϕ. Here, g is regarded as an activation function.

Clearly,

$$\phi(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{\ell=0}^{m^n-1} [\ell,\, \ell+1-m^n\delta] = \bigcup_{\ell=0}^{2^{n^{s+1}}-1} [\ell,\, \ell+1-2^{n^{s+1}}\delta].$$

Moreover, the fact $g \in \mathcal{NN}_s\{\widehat{n}\}$ implies that ϕ can be realized by a height-$(s+1)$ NestNet with at most

$$\underbrace{(1+1)2 + (2+1)3 + (3+1)3(n-2) + (3+1)}_{\text{outer network}} + \underbrace{\widehat{n}}_{g} = \widehat{n} + 12n - 7$$

parameters. Hence, we finish the proof of Lemma C.3. $\square$

With Lemmas C.2 and C.3 in hand, we are ready to prove Lemma C.1.

Proof of Lemma C.1. We will use the mathematical induction to prove Lemma C.1. First, we consider the base case $r = 1$. By Lemma C.2, there exists a function ϕ realized by a ReLU network of width 4 and depth $4n-1$ such that

$$\phi(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{\ell=0}^{2^n-1} [\ell,\, \ell+1-\delta] \subseteq \bigcup_{\ell=0}^{2^n-1} [\ell,\, \ell+1-C(r,n)\cdot\delta] \text{ with } r = 1.$$

Moreover, the network realizing ϕ has at most $(4+1)4\big((4n-1)+1\big) = 80n$ parameters, implying $\phi \in \mathcal{NN}_1\{80n\} \subseteq \mathcal{NN}_1\{(12r+68)n\}$ for $r=1$. Thus, the base case $r=1$ is proved.

Next, assume Lemma C.1 holds for $r = s \in \mathbb{N}^+$. We need to show it is also true for $r = s+1$. By the induction hypothesis, there exists $g \in \mathcal{NN}_s\big\{(12s+68)n\big\}$ such that

$$g(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{\ell=0}^{2^{n^s}-1} [\ell,\, \ell+1-C(s,n)\cdot\delta].$$

By Lemma C.3 with $\widehat{n} = (12s+68)n$ therein and setting $\widehat{\delta} = C(s,n)\cdot\delta$, there exists

$$\phi \in \mathcal{NN}_{s+1}\big\{\widehat{n}+12n-7\big\} \subseteq \mathcal{NN}_{s+1}\big\{(12s+68)n+12n-7\big\} \subseteq \mathcal{NN}_{s+1}\Big\{\big(12(s+1)+68\big)n\Big\}$$

such that

$$\phi(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{\ell=0}^{2^{n^{s+1}}-1} [\ell,\, \ell+1-2^{n^{s+1}}\widehat{\delta}\,].$$

Observe that

$$2^{n^{s+1}}\widehat{\delta} = 2^{n^{s+1}} C(s,n)\cdot\delta = 2^{n^{s+1}}\Big(\prod_{i=1}^{s} 2^{n^i}\Big)\cdot\delta = \Big(\prod_{i=1}^{s+1} 2^{n^i}\Big)\cdot\delta = C(s+1,n)\cdot\delta.$$

It follows that

$$\phi(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{\ell=0}^{2^{n^{s+1}}-1} [\ell,\, \ell+1-C(s+1,n)\cdot\delta].$$

Thus, Lemma C.1 is proved for the case $r = s+1$, which means we finish the induction step. Hence, by the principle of induction, we complete the proof of Lemma C.1. □

C.2 Detailed proof of Proposition B.1

Set $C(r,n) = \prod_{i=1}^{r} 2^{n^i}$ and $\widetilde{\delta} = \frac{\delta}{C(r,n)} \in \big(0,\, \frac{1}{C(r,n)}\big)$. By Lemma C.1, there exists $\phi_0 \in \mathcal{NN}_r\big\{(12r+68)n\big\}$ such that

$$\phi_0(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{\ell=0}^{2^{n^r}-1} [\ell,\, \ell+1-C(r,n)\cdot\widetilde{\delta}\,] = \bigcup_{\ell=0}^{2^{n^r}-1} [\ell,\, \ell+1-\delta].$$

It follows from $J \le 2^{n^r}$ that

$$\phi_0(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{j=0}^{J-1} [j,\, j+1-\delta].$$

Set

$$\widetilde{M} = \max_{x\in[J,J+1]} |\phi_0(x)| \quad \text{and} \quad M = \frac{\widetilde{M}+J}{\delta}.$$

Then, for any $x \in [J,\, J+1]$, we have

$$\phi_0(x) + M\sigma\big(x-(J-\delta)\big) \ge -\widetilde{M} + M\delta = -\widetilde{M} + (\widetilde{M}+J) = J,$$

implying

$$\min\Big\{\phi_0(x) + M\sigma\big(x-(J-\delta)\big),\, J\Big\} = J.$$

Moreover, for any $x \in \bigcup_{j=0}^{J-1} [j,\, j+1-\delta]$, we have $\sigma\big(x-(J-\delta)\big) = 0$, implying

$$\min\Big\{\phi_0(x) + M\sigma\big(x-(J-\delta)\big),\, J\Big\} = \min\Big\{\phi_0(x),\, J\Big\} = \min\Big\{\lfloor x \rfloor,\, J\Big\} = \lfloor x \rfloor.$$

Therefore, by defining

$$\phi(x) \coloneqq \min\Big\{\phi_0(x) + M\sigma\big(x-(J-\delta)\big),\, J\Big\} \quad \text{for any } x \in \bigcup_{j=0}^{J} \big[j,\, j+1-\delta\cdot\mathbb{1}_{\{j\le J-1\}}\big],$$

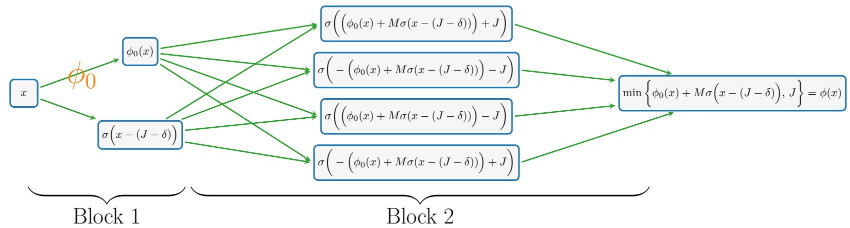


Figure 18: An illustration of the network realizing ϕ for any $x \in \bigcup_{j=0}^{J}\big[j,\, j+1-\delta\cdot\mathbb{1}_{\{j\le J-1\}}\big]$ based on the fact $\min\{a,b\} = \frac{1}{2}\big(\sigma(a+b)-\sigma(-a-b)-\sigma(a-b)-\sigma(-a+b)\big)$.

we have

$$\phi(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{j=0}^{J-1}[j,\, j+1-\delta]$$

and

$$\phi(x) = J \quad \text{for any } x \in [J,\, J+1].$$

Moreover, ϕ can be realized by the network in Figure 18. The fact $\phi_0 \in \mathcal{NN}_r\big\{(12r+68)n\big\}$ implies that ϕ can be realized by a height-r NestNet with at most

$$\underbrace{3\Big((12r+68)n\Big)}_{\text{Block 1}} + \underbrace{(2+1)4+(4+1)}_{\text{Block 2}} \le 36(r+7)n$$

parameters. So we finish the proof of Proposition B.1.

D Proof of Proposition B.2

The key idea of proving Proposition B.2 is the bit extraction technique proposed in [6]. First, we establish several lemmas for proving Proposition B.2 and give their proofs in Section D.1 except for Lemma D.2, the proof of which is placed in Section D.3 since it is complicated. Next, we present the detailed proof of Proposition B.2 in Section D.2 based on the lemmas established in Section D.1.

D.1 Lemmas for proving Proposition B.2

To simplify the proof of Proposition B.2, we establish several lemmas as the intermediate step. We first establish a lemma to show that any continuous piecewise linear functions on $\mathbb{R}$ can be realized by one-hidden-layer ReLU networks.

Lemma D.1. *Given any $p \in \mathbb{N}^+$, any continuous piecewise linear function on $\mathbb{R}$ with at most p breakpoints can be realized by a one-hidden-layer ReLU network of width $p+1$.*

Proof. We will use the mathematical induction to prove Lemma D.1. First, we consider the base case $p = 1$. Suppose $f : \mathbb{R} \to \mathbb{R}$ is a continuous piecewise linear function on $\mathbb{R}$ with at most $p = 1$ breakpoints. Then there exist $a_1, a_2, x_0 \in \mathbb{R}$ such that

$$f(x) = \begin{cases} a_1(x-x_0)+f(x_0) & \text{if } x \ge x_0 \\ a_2(x_0-x)+f(x_0) & \text{if } x < x_0. \end{cases}$$

Thus, $f(x) = a_1\sigma(x-x_0) + a_2\sigma(x_0-x) + f(x_0)$ for any $x \in \mathbb{R}$, implying f can be realized by a one-hidden-layer ReLU network of width $2 = p+1$ for $p = 1$. Hence, Lemma D.1 is proved for the case $p = 1$.

Now, assume Lemma D.1 holds for $p = k \in \mathbb{N}^+$, we would like to show it is also true for $p = k+1$. Suppose $f : \mathbb{R} \to \mathbb{R}$ is a continuous piecewise linear function on with at most $k+1$ breakpoints. We may assume the biggest breakpoint of f is x_0 since it is trivial for the case that f has no breakpoint. Denote the slopes of the linear pieces left and right next to x_0 by a_1 and a_2, respectively. Define

$$\widetilde{f}(x) := f(x) - (a_2-a_1)\sigma(x-x_0) \quad \text{for any } x \in \mathbb{R}.$$

Then $\widetilde{f}$ has at most k breakpoints. By the induction hypothesis, $\widetilde{f}$ can be realized by a one-hidden-layer ReLU network of width $k+1$. Thus, there exist $w_{0,j}, b_{0,j}, w_{1,j}, b_1$ for $j = 1, 2, \cdots, k+1$ such that

$$\widetilde{f}(x) = \sum_{j=1}^{k+1} w_{1,j}\sigma(w_{0,j}x + b_{0,j}) + b_1 \quad \text{for any } x \in \mathbb{R}.$$

Therefore, for any $x \in \mathbb{R}$, we have

$$f(x) = (a_2 - a_1)\sigma(x - x_0) + \widetilde{f}(x) = (a_2 - a_1)\sigma(x - x_0) + \sum_{j=1}^{k+1} w_{1,j}\sigma(w_{0,j}x + b_{0,j}) + b_1,$$

implying f can be realized by a one-hidden-layer ReLU network of width $k+2 = (k+1)+1 = p+1$ for $p = k+1$. Thus, we finish the induction process. Therefore, by the principle of induction, we complete the proof of Lemma D.1. □

Next, we establish a lemma to extract the sum of n^s bits via a height-s NestNet with $\mathcal{O}(n)$ parameters.

Lemma D.2. *Given any $n, s \in \mathbb{N}^+$, there exists $\phi \in \mathcal{NN}_s\{57(s+7)^2(n+1)\}$ such that: For any $\theta_1, \theta_2, \cdots, \theta_{n^s} \in \{0,1\}$, we have*

$$\phi\big(k + \text{bin}\, 0.\theta_1\theta_2\cdots\theta_{n^s}\big) = \sum_{\ell=1}^{k} \theta_\ell \quad \text{for } k = 0, 1, \cdots, n^s. \tag{14}$$

The proof of Lemma D.2 is complicated and hence is placed in Section D.3. Then, based on Lemma D.2, we establish a new lemma, Lemma D.3 below, which is a key intermediate conclusion to prove Proposition B.2.

Lemma D.3. *Given any $n, s \in \mathbb{N}^+$ and $\theta_{i,\ell} \in \{0,1\}$ for $i = 0, 1, \cdots, n-1$ and $\ell = 0, 1, \cdots, m-1$, where $m = n^s$, there exists $\phi \in \mathcal{NN}_s\{58(s+7)^2(n+1)\}$ such that*

$$\phi(j) = \sum_{\ell=0}^{k} \theta_{i,\ell} \quad \text{for } j = 0, 1, \cdots, nm-1,$$

where (i,k) is the unique index pair satisfying $j = im + k$ with $i \in \{0, 1, \cdots, n-1\}$ and $k \in \{0, 1, \cdots, m-1\}$.

Proof. We first construct a network to extract the unique index pair (i,k) from $j \in \{0, 1, \cdots, nm-1\}$ with the following condition

$$j = im + k \quad \text{with } i \in \{0, 1, \cdots, n-1\} \text{ and } k \in \{0, 1, \cdots, m-1\}.$$

There exists a continuous piecewise linear function ϕ_1 with $2n$ breakpoints such that

$$\phi_1(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{\ell=0}^{n-1} [\ell,\, \ell+1-\delta] \text{ with } \delta = \tfrac{1}{2m}.$$

By Lemma D.1, ϕ_1 can be realized by a one-hidden-layer ReLU network of width $2n+1$. Moreover, for any $j \in \{0, 1, \cdots, nm-1\}$, we have

$$\phi_1(\tfrac{j}{m}) = \lfloor \tfrac{j}{m} \rfloor = i \quad \text{and} \quad j - m\phi_1(\tfrac{j}{m}) = j - mi = k,$$

where (i,k) is the unique index pair satisfying $j = im + k$ with $i \in \{0, 1, \cdots, n-1\}$ and $k \in \{0, 1, \cdots, m-1\}$. By defining

$$\mathbf{\Phi}_1(x) := \begin{bmatrix} \phi_1(\frac{x}{m}) \\ x - m\phi_1(\frac{x}{m}) \end{bmatrix} \quad \text{for any } x \geq 0,$$

we have

$$\mathbf{\Phi}_1(j) = \begin{bmatrix} \phi_1(\frac{j}{m}) \\ j - m\phi_1(\frac{j}{m}) \end{bmatrix} = \begin{bmatrix} i \\ k \end{bmatrix} \quad \text{for } j = 0, 1, \cdots, nm-1,$$

where (i,k) is the unique index pair satisfying $j = im+k$ with $i \in \{0, 1, \cdots, n-1\}$ and $k \in \{0, 1, \cdots, m-1\}$. Moreover, $\mathbf{\Phi}_1$ can be realized by a one-hidden-layer ReLU network of width $2(2n+1)+1 = 4n+3$. Hence, the network realizing $\mathbf{\Phi}_1$ has at most $(1+1)(4n+3) + \big((4n+3)+1\big)2 = 16n+14$ parameters.

Define

$$z_i \coloneqq \text{bin}\, 0.\theta_{i,0}\theta_{i,1}\cdots\theta_{i,m-1} \quad \text{for } i = 0, 1, \cdots, n-1.$$

There exists a continuous piecewise linear function $\widetilde{\phi}_2$ with n breakpoints such that

$$\widetilde{\phi}_2(i) = z_i \quad \text{for } i = 0, 1, \cdots, n-1.$$

By Lemma D.1, $\widetilde{\phi}_2$ can be realized by a one-hidden-layer ReLU network of width $n+1$.

By Lemma D.2, there exists $\phi_3 \in \mathcal{NN}_s\big\{57(s+7)^2(n+1)\big\}$ such that: For any $\xi_1, \xi_2, \cdots, \xi_{n^s} \in \{0,1\}$, we have

$$\phi_3\big(k + \text{bin}\, 0.\xi_1\xi_2\cdots\xi_{n^s}\big) = \sum_{\ell=1}^{k} \xi_\ell \quad \text{for } k = 1, 2, \cdots, n^s.$$

It follows from $m = n^s$ that, for any $\xi_0, \xi_1, \cdots, \xi_{m-1} \in \{0,1\}$, we have

$$\phi_3(k + \text{bin}\, 0.\xi_0\xi_1\cdots\xi_{m-1}) = \sum_{\ell=1}^{k} \xi_{\ell-1} = \sum_{\ell=0}^{k-1} \xi_\ell \quad \text{for } k = 1, 2, \cdots, m,$$

implying

$$\phi_3(k + 1 + \text{bin}\, 0.\xi_0\xi_1\cdots\xi_{m-1}) = \sum_{\ell=0}^{k} \xi_\ell \quad \text{for } k = 0, 1, \cdots, m-1.$$

Then, for $i = 0, 1, \cdots, n-1$ and $k = 0, 1, \cdots, m-1$, we have

$$\phi_3\big(k + 1 + \widetilde{\phi}_2(i)\big) = \phi_2(k + 1 + z_i) = \phi_3(k + 1 + \text{bin}\, 0.\theta_{i,0}\theta_{i,1}\cdots\theta_{i,m-1}) = \sum_{\ell=0}^{k} \theta_{i,\ell}.$$

By defining

$$\phi_2(x, y) \coloneqq y + 1 + \widetilde{\phi}_2(x) \quad \text{for any } x, y \in [0, \infty)$$

and $\phi \coloneqq \phi_3 \circ \phi_2 \circ \mathbf{\Phi}_1$, we have

$$\phi(j) = \phi_3 \circ \phi_2 \circ \mathbf{\Phi}_1(j) = \phi_3 \circ \phi_2(i, k) = \phi_3\big(k + 1 + \widetilde{\phi}_2(i)\big) = \sum_{\ell=0}^{k} \theta_{i,\ell}$$

for $j = 0, 1, \cdots, nm-1$, where (i, k) is the unique index pair satisfying $j = im + k$ with $i \in \{0, 1, \cdots, n-1\}$ and $k \in \{0, 1, \cdots, m-1\}$.

It remains to estimate the number of parameters in the NestNet realizing $\phi = \phi_3 \circ \phi_2 \circ \mathbf{\Phi}_1$. Observe that ϕ_2 can be realized by a one-hidden-layer ReLU network of width $(n+1) + 1 = n+2$. Then, the network realizing ϕ_2 has at most $(2+1)(n+2) + \big((n+2)+1\big) = 4n+9$ parameters. Therefore, ϕ can be realized by a height-s NestNet with at most

$$\underbrace{(16n+14)}_{\mathbf{\Phi}_1} + \underbrace{(4n+9)}_{\phi_2} + \underbrace{57(s+7)^2(n+1)}_{\phi_3} \le 58(s+7)^2(n+1)$$

parameters, which means we complete the proof of Lemma D.3. □

D.2 Detailed proof of Proposition B.2

We may assume $J = mn = n^{s+1}$ with $m = n^s$ since we can set $y_{J-1} = y_J = \cdots = y_{mn-1}$ if $J < mn$. Define

$$a_j \coloneqq \lfloor y_j / \varepsilon \rfloor \quad \text{for } j = 0, 1, \cdots, nm-1.$$

Our goal is to construct a function ϕ such that $\phi(j) = a_j\varepsilon$ for $j = 0, 1, \cdots, nm-1$.

For $i = 0, 1, \cdots, n-1$, we define

$$b_{i,\ell} = \begin{cases} 0 & \text{for } \ell = 0 \\ a_{im+\ell} - a_{im+\ell-1} & \text{for } \ell = 1, 2, \cdots, m-1. \end{cases}$$

Since $|y_j - y_{j-1}| \le \varepsilon$ for all j, we have $|a_j - a_{j-1}| \le 1$. It follows that $b_{i,\ell} \in \{-1, 0, 1\}$ for $i = 0, 1, \cdots, n-1$ and $\ell = 0, 1, \cdots, m-1$. Hence, there exist $c_{i,\ell} \in \{0,1\}$ and $d_{i,\ell} \in \{0,1\}$ such that

$$b_{i,\ell} = c_{i,\ell} - d_{i,\ell} \quad \text{for } i = 0, 1, \cdots, n-1 \text{ and } \ell = 0, 1, \cdots, m-1.$$

Since any $j \in \{0, 1, \cdots, nm-1\}$ can be uniquely indexed as $j = im + k$ with $i \in \{0, 1, \cdots, n-1\}$ and $k \in \{0, 1, \cdots, m-1\}$, we have

$$\begin{aligned} a_j = a_{im+k} = a_{im} + \sum_{\ell=1}^{k} \big(a_{im+\ell} - a_{im+\ell-1}\big) = a_{im} + \sum_{\ell=1}^{k} b_{i,\ell} = a_{im} + \sum_{\ell=0}^{k} b_{i,\ell} \\ = a_{im} + \sum_{\ell=0}^{k} c_{i,\ell} - \sum_{\ell=0}^{k} d_{i,\ell}. \end{aligned}$$

There exists a continuous piecewise linear function ϕ_1 with $2n$ breakpoints such that

$$\phi_1(x) = a_{im} \quad \text{for any } x \in [im, im+m-1] \text{ and } i = 0, 1, \cdots, n-1.$$

Then, we have

$$\phi_1(j) = a_{im} \quad \text{for } j = 0, 1, \cdots, nm-1,$$

where (i, k) is the unique index pair satisfying $j = im + k$ with $i \in \{0, 1, \cdots, n-1\}$ and $k \in \{0, 1, \cdots, m-1\}$. By Lemma D.1, ϕ_1 can be realized by a one-hidden-layer ReLU network of width $2n + 1$.

By Lemma D.3, there exist $\phi_2, \phi_3 \in \mathcal{NN}_s\big\{58(s+7)^2(n+1)\big\}$ such that

$$\phi_2(j) = \sum_{\ell=0}^{k} c_{i,\ell} \quad \text{and} \quad \phi_3(j) = \sum_{\ell=0}^{k} d_{i,\ell} \quad \text{for } j = 0, 1, \cdots, nm-1,$$

where (i, k) is the unique index pair satisfying $j = im + k$ with $i \in \{0, 1, \cdots, n-1\}$ and $k \in \{0, 1, \cdots, m-1\}$.

Hence, by indexing $j \in \{0, 1, \cdots, nm-1\}$ as $j = im+k$ for $i = \{0, 1, \cdots, n-1\}$ and $k \in \{0, 1, \cdots, m-1\}$, we have

$$a_j = a_{im} + \sum_{\ell=0}^{k} c_{i,\ell} - \sum_{\ell=0}^{k} d_{i,\ell} = \phi_1(j) + \phi_2(j) - \phi_3(j).$$

By defining

$$\widetilde{\phi}(x) \coloneqq \Big(\phi_1(x) + \phi_2(x) + \phi_3(x)\Big)\varepsilon \quad \text{for any } x \in \mathbb{R},$$

we have $\widetilde{\phi}(j) = a_j\varepsilon$ for $j = 0, 1, \cdots, nm-1$ and $\widetilde{\phi}$ can be realized by the height-s NestNet in Figure 19.

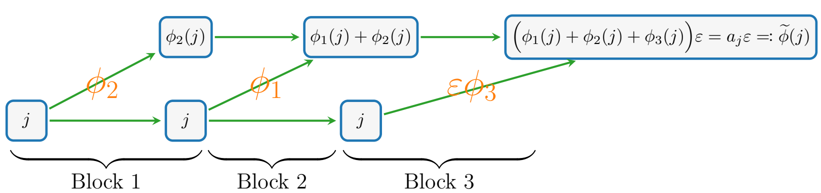


Figure 19: An illustration of the NestNet realizing $\widetilde{\phi}$ for $j = 0, 1, \cdots, J-1$.

In Figure 19, Block 1 or 3 has at most

$$3\Big(58(s+7)^2(n+1)\Big) = 174(s+7)^2(n+1)$$

parameters; Block 2 is of width $(2n+1) + 2 = 2n+3$ and depth 1, and hence has at most

$$(2+1)(2n+3) + \big((2n+3)+1\big)2 = 10n + 17$$

parameters. Then, $\widetilde{\phi}$ can be realized by a height-s ReLU NestNet with at most

$$2\big(174(s+7)^2(n+1)\big) + 10n + 17 = 349(s+7)^2(n+1)$$

parameters. Note that $\widetilde{\phi}$ may not be bounded. Thus, we define

$$\psi(x) \coloneqq \min\big\{\sigma(x),\, M\big\} \quad \text{for any } x \in \mathbb{R},$$

where

$$M = \max\{y_j : j = 0, 1, \cdots, nm - 1\}.$$

Then, the desired function ϕ can be define via $\phi \coloneqq \psi \circ \widetilde{\phi}$. Clearly,

$$0 \le \phi(x) \le M = \max\{y_j : j = 0, 1, \cdots, J - 1\} \quad \text{for any } x \in \mathbb{R}.$$

It follows from $0 \le a_j \varepsilon = \lfloor y_j / \varepsilon \rfloor \varepsilon \le y_j \le M$ for $j = 0, 1, \cdots, J - 1$ that

$$\phi(j) = \psi \circ \widetilde{\phi}(j) = \psi\big(a_j \varepsilon\big) = \min\big\{\sigma(a_j \varepsilon),\, M\big\} = a_j \varepsilon,$$

implying

$$\big|\phi(j) - y_j\big| = \big|a_j \varepsilon - y_j\big| = \Big|\lfloor y_j/\varepsilon \rfloor \varepsilon - y_j\Big| = \Big|\lfloor y_j/\varepsilon \rfloor - y_j/\varepsilon\Big|\varepsilon \le \varepsilon.$$

It remains to show that ϕ can be realized by a height-s ReLU NestNet with the desired size. Clearly, ψ can be realized by the network in Figure 20, which is of width 4 and depth 2.

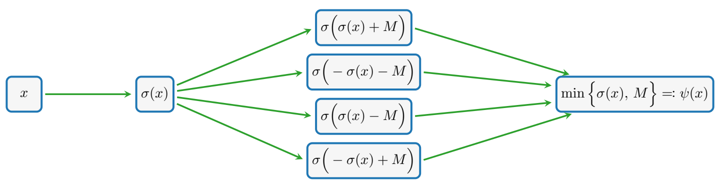


Figure 20: An illustration of the network realizing ψ based on the fact $\min\{a, b\} = \frac{1}{2}\big(\sigma(a+b) - \sigma(-a-b) - \sigma(a-b) - \sigma(-a+b)\big)$.

Therefore, ϕ can be realized by a height-s ReLU NestNet with at most

$$349(s+7)^2(n+1) + (4+1)4(2+1) \le 350(s+7)^2(n+1)$$

parameters. Hence, we finish the proof of Proposition B.2.

D.3 Proof of Lemma D.2 for Proposition B.2

We will use the mathematical induction to prove Lemma D.2. To this end, we introduce two lemmas for the base case and the induction step.

Lemma D.4. *Given any $n \in \mathbb{N}^+$, there exists a function ϕ realized by a ReLU network with $128n + 294$ parameters such that: For any $\theta_1, \theta_2, \cdots, \theta_n \in \{0, 1\}$, we have*

$$\phi\big(k + \mathrm{bin}\, 0.\theta_1\theta_2 \cdots \theta_n\big) = \sum_{\ell=1}^{k} \theta_\ell \quad \text{for } k = 0, 1, \cdots, n. \tag{15}$$

Lemma D.5. *Given any $n, r, \widehat{n} \in \mathbb{N}^+$, if $g \in \mathcal{NN}_r\{\widehat{n}\}$ satisfying*

$$g\big(p + \mathrm{bin}\, 0.\xi_1\xi_2 \cdots \xi_{n^r}\big) = \sum_{j=1}^{p} \xi_j \quad \text{for any } \xi_1, \xi_2, \cdots, \xi_{n^r} \in \{0, 1\} \text{ and } p = 0, 1, \cdots, n^r, \tag{16}$$

then there exists $\phi \in \mathcal{NN}_{r+1}\big\{\widehat{n} + 114(r+7)(n+1)\big\}$ such that: For any $\theta_1, \theta_2, \cdots, \theta_{n^{r+1}} \in \{0, 1\}$, we have

$$\phi\big(k + \mathrm{bin}\, 0.\theta_1\theta_2 \cdots \theta_{n^{r+1}}\big) = \sum_{\ell=1}^{k} \theta_\ell \quad \text{for } k = 0, 1, \cdots, n^{r+1}.$$

The proofs of Lemmas D.4 and D.5 can be found in Sections D.3.1 and D.3.2, respectively. We remark that the function ϕ in Lemma D.5 is independent of $\theta_1, \theta_2, \cdots, \theta_{nm}$. The proof of Lemma D.2 mainly relies on Lemma D.4 and repeated applications of Lemma D.5. The details can be found below.

Proof of Lemma D.2. We will use the mathematical induction to prove Lemma D.2. First, let us consider the base case $s = 1$. By Lemma D.4, there exists a function realized by a ReLU network with $128n + 294$ parameters such that: For any $\theta_1, \theta_2, \cdots, \theta_n \in \{0, 1\}$, we have

$$\phi\Big(k + \text{bin}\, 0.\theta_1\theta_2\cdots\theta_n\Big) = \sum_{\ell=1}^{k} \theta_\ell \quad \text{for } k = 0, 1, \cdots, n.$$

That means Equation (14) holds for $s = 1$. Moreover, ϕ can also be regarded as a height-1 ReLU NestNet with $128n + 294 \le 57(s+7)^2(n+1)$ parameters for $s = 1$, which means Lemma D.2 is proved for the case $s = 1$.

Next, assume Lemma D.2 holds for $s = r \in \mathbb{N}^+$. We need to show that it is also true for $s = r + 1$ by applying Lemma D.5. By the induction hypothesis, there exists

$$g \in \mathcal{NN}_r\Big\{57(r+7)^2(n+1)\Big\}$$

such that: For any $\xi_1, \xi_2, \cdots, \xi_{n^r} \in \{0, 1\}$, we have

$$g\Big(k + \text{bin}\, 0.\xi_1\xi_2\cdots\xi_{n^r}\Big) = \sum_{\ell=1}^{k} \theta_\ell \quad \text{for } k = 0, 1, \cdots, n^r.$$

It follows from $m = n^r$ that

$$g\Big(p + \text{bin}\, 0.\xi_1\xi_2\cdots\xi_m\Big) = \sum_{j=1}^{p} \xi_j \quad \text{for any } \xi_1, \xi_2, \cdots, \xi_m \in \{0, 1\} \text{ and } p = 0, 1, \cdots, m,$$

which means g satisfies Equation (16). Then, by Lemma D.5 with $m = n^r$ and $\widehat{n} = 57(r+7)^2(n+1)$ therein, there exists

$$\phi \in \mathcal{NN}_{r+1}\Big\{\widehat{n} + 114(r+7)(n+1)\Big\}$$

such that: For any $\theta_1, \theta_2, \cdots, \theta_{nm} \in \{0, 1\}$, we have

$$\phi\Big(k + \text{bin}\, 0.\theta_1\theta_2\cdots\theta_{nm}\Big) = \sum_{\ell=1}^{k} \theta_\ell \quad \text{for } k = 0, 1, \cdots, nm.$$

It follows from $m = n^r$ that, for any $\theta_1, \theta_2, \cdots, \theta_{n^{r+1}} \in \{0, 1\}$, we have

$$\phi\Big(k + \text{bin}\, 0.\theta_1\theta_2\cdots\theta_{n^{r+1}}\Big) = \sum_{\ell=1}^{k} \theta_\ell \quad \text{for } k = 0, 1, \cdots, n^{r+1},$$

which means Equation (14) holds for $s = r + 1$. Moreover, we have

$$\begin{aligned} \widehat{n} + 114(r+7)(n+1) &= 57(r+7)^2(n+1) + 114(r+7)(n+1) \\ &= 57(n+1)\Big((r+7)^2 + 2(r+7)\Big) \\ &\le 57(n+1)\big((r+7)+1\big)^2 = 57\big((r+1)+7\big)^2(n+1). \end{aligned}$$

This implies that

$$\phi \in \mathcal{NN}_{r+1}\Big\{\widehat{n} + 114(r+7)(n+1)\Big\} \subseteq \mathcal{NN}_{r+1}\Big\{57\big((r+1)+7\big)^2(n+1)\Big\}.$$

Thus, we prove Lemma D.2 for the case $s = r + 1$, which means we finish the induction step. Hence, by the principle of induction, we complete the proof of Lemma D.2. □

D.3.1 Proof of Lemma D.4 for Lemma D.2

To simplify the proof of Lemma D.4, we introduce the following lemma.

Lemma D.6. *Given any $n \in \mathbb{N}^+$, there exists a function ϕ realized by a ReLU network of width 7 and depth $2n + 1$ such that: For any $\theta_1, \theta_2, \cdots, \theta_n \in \{0, 1\}$, we have*

$$\phi\Big(\text{bin}\, 0.\theta_1\theta_2\cdots\theta_n,\ k\Big) = \sum_{\ell=1}^{k} \theta_\ell \quad \text{for } k = 0, 1, \cdots, n.$$

Lemma D.6 is the Lemma 3.5 of [35]. The detailed proof can be found therein. With Lemma D.6 in hand, we are ready to prove Lemma D.4.

Proof of Lemma D.4. By Lemma D.6, there exists a function ϕ_0 realized by a ReLU network of width 7 and depth $2n+1$ such that: For any $\theta_1, \theta_2, \cdots, \theta_n \in \{0,1\}$, we have

$$\phi_0\big(\text{bin}\,0.\theta_1\theta_2\cdots\theta_n,\, k\big) = \sum_{\ell=1}^{k} \theta_\ell \quad \text{for } k = 1, 2, \cdots, n.$$

The equation above is not true for $k = 0$. We will construct ϕ_2 such that

$$\phi_2(\text{bin}\,0.\theta_1\theta_2\cdots\theta_n,\, k) = \sum_{\ell=1}^{k} \theta_\ell \quad \text{for } k = 0, 1, \cdots, n.$$

To this end, we first set

$$M = \max\big\{|\phi_0(x,\, y)| : x \in [0,1],\, y \in [0,n]\big\}$$

and define

$$\phi_1(x,y) \coloneqq \min\big\{M + \phi_0(x,y),\, 2My\big\} \quad \text{for any } x \in [0,1] \text{ and } y \in [0,n].$$

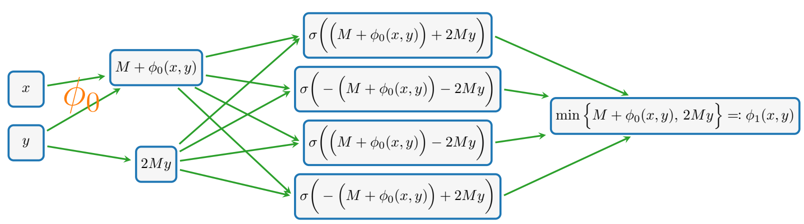


Figure 21: An illustration of the network realizing ϕ_1 for any $x \in [0,1]$ and $y \in [0,n]$ based on the fact $\min\{a,b\} = \frac{1}{2}\big(\sigma(a+b) - \sigma(-a-b) - \sigma(a-b) - \sigma(-a+b)\big)$.

As we can see from Figure 21, ϕ_1 can be realized by a ReLU network of width $\max\{7,4\} = 7$ and depth $(2n+1)+2 = 2n+3$. Moreover, we have

$$\begin{aligned}\phi_1(\text{bin}\,0.\theta_1\theta_2\cdots\theta_n,\, k) &= \min\big\{M + \phi_0(\text{bin}\,0.\theta_1\theta_2\cdots\theta_n,\, k),\, 2Mk\big\}\\ &= \begin{cases} M + \sum_{\ell=1}^{k}\theta_\ell & \text{for } k = 1,2,\cdots,n \\ 0 & \text{for } k = 0.\end{cases}\end{aligned}$$

Define

$$\phi_2(x,y) \coloneqq \sigma\big(\phi_1(x,y) - M\big) \quad \text{for any } x \in [0,1] \text{ and } y \in [0,\infty).$$

Then, ϕ_2 can be realized by a ReLU network of width 7 and depth $(2n+3)+1 = 2n+4$. Moreover, we have

$$\begin{aligned}\phi_2(\text{bin}\,0.\theta_1\theta_2\cdots\theta_n,\, k) &= \sigma\big(\phi_1(\text{bin}\,0.\theta_1\theta_2\cdots\theta_n,\, k) - M\big)\\ &= \begin{cases} \sigma\big(\sum_{\ell=1}^{k}\theta_\ell\big) = \sum_{\ell=1}^{k}\theta_\ell & \text{for } k = 1,2,\cdots,n \\ \sigma(-M) = 0 & \text{for } k = 0.\end{cases}\end{aligned}$$

That is,

$$\phi_2(\text{bin}\,0.\theta_1\theta_2\cdots\theta_n,\, k) = \sum_{\ell=1}^{k} \theta_\ell \quad \text{for } k = 0, 1, \cdots, n.$$

Next, we will construct $\boldsymbol{\Psi}$ to extract k and $\text{bin}\,0.\theta_1\theta_2\cdots\theta_n$ from $k + \text{bin}\,0.\theta_1\theta_2\cdots\theta_n$. It is easy to construct a continuous piecewise linear function $\psi : \mathbb{R} \to \mathbb{R}$ with $2n$ breakpoints satisfying

$$\psi(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{\ell=0}^{n-1} [\ell,\, \ell+1-\delta] \text{ with } \delta = 2^{-n}.$$

By Lemma D.1 with $p = 2n$ therein, ψ can be realized by a one-hidden-layer ReLU network of width $2n+1$. By defining

$$\mathbf{\Psi}(x) \coloneqq \begin{bmatrix} x - \psi(x) \\ \psi(x) \end{bmatrix} = \begin{bmatrix} \sigma(x) - \psi(x) \\ \psi(x) \end{bmatrix} \quad \text{for any } x \in [0,\infty).$$

Then, $\mathbf{\Psi}$ can be realized by a one-hidden-layer ReLU network of width $1 + 2(2n+1) = 4n+3$. That means, the network realizing $\mathbf{\Psi}$ has at most

$$(1+1)(4n+3) + \big((4n+3)+1\big)2 = 16n + 14$$

parameters. Moreover, for any $\theta_1, \theta_2, \cdots, \theta_n \in \{0,1\}$ and $k = 0, 1, \cdots, n$, we have

$$\psi(k + \text{bin}\, 0.\theta_1\theta_2\cdots\theta_n) = \lfloor k + \text{bin}\, 0.\theta_1\theta_2\cdots\theta_n \rfloor = k,$$

implying

$$\begin{aligned} \mathbf{\Psi}(k + \text{bin}\, 0.\theta_1\theta_2\cdots\theta_n) &= \begin{bmatrix} k + \text{bin}\, 0.\theta_1\theta_2\cdots\theta_n - \psi(k + \text{bin}\, 0.\theta_1\theta_2\cdots\theta_n) \\ \psi(k + \text{bin}\, 0.\theta_1\theta_2\cdots\theta_n) \end{bmatrix} \\ &= \begin{bmatrix} \text{bin}\, 0.\theta_1\theta_2\cdots\theta_n \\ k \end{bmatrix}. \end{aligned}$$

Finally, the desired function ϕ can be defined via $\phi \coloneqq \phi_2 \circ \mathbf{\Psi}$. Clearly, the network realizing ϕ_2 is of width 7 and depth $2n+4$, and hence has at most

$$(7+1)7\big((2n+4)+1\big) = 56(2n+5)$$

parameters, implying ϕ can be realized by a ReLU network with at most

$$56(2n+5) + (16n+14) = 128n + 294$$

parameters. Moreover, for any $\theta_1, \theta_2, \cdots, \theta_n \in \{0,1\}$ and $k = 0, 1, \cdots, n$, we have

$$\begin{aligned} \phi(k + \text{bin}\, 0.\theta_1\theta_2\cdots\theta_n) &= \phi_2 \circ \mathbf{\Psi}(k + \text{bin}\, 0.\theta_1\theta_2\cdots\theta_n) \\ &= \phi_2(\text{bin}\, 0.\theta_1\theta_2\cdots\theta_n,\, k) = \sum_{\ell=1}^{k} \theta_\ell. \end{aligned}$$

Thus, we finish the proof of Lemma D.4. □

D.3.2 Proof of Lemma D.5 for Lemma D.2

The key idea of proving Lemma D.5 is to construct a network with n blocks, each of which extracts the sum of n^r bits via g. Then the whole network can extract the sum of n^{r+1} bits as we expect.

To simplify our notation, we set $m = n^r$. Given any nm binary bits $\theta_\ell \in \{0,1\}$ for $\ell = 1, 2, \cdots, nm$, we divide these nm bits into n classes according to their indices, where the i-th class is composed of m bits $\theta_{im+1}, \cdots, \theta_{im+m}$ for $i = 0, 1, \cdots, n-1$. We will show how to extract the m bits of the i-th class, stored in $\text{bin}\, 0.\theta_{im+1}\cdots\theta_{im+m}$.

First, let us show how to construct a network to extract k and $\text{bin}\, 0.\theta_1\theta_2\cdots\theta_{nm}$ from $k + 0.\theta_1\theta_2\cdots\theta_{nm}$. By setting $\widetilde{n} = 2n$ and Proposition B.1 with $J = 2^{\widetilde{n}^r}$ therein, there exists

$$\widetilde{g} \in \mathcal{NN}_r\big\{36(r+7)\widetilde{n}\big\} = \mathcal{NN}_r\big\{36(r+7)(2n)\big\} = \mathcal{NN}_r\big\{72(r+7)n\big\}$$

such that

$$\widetilde{g}(x) = \lfloor x \rfloor \quad \text{for any } x \in \bigcup_{\ell=0}^{J-1} [\ell, \ell+1-\delta].$$

Observe that

$$J - 1 = 2^{\widetilde{n}^r} = 2^{(2n)^r} - 1 \ge 2^{2(n^r)} - 1 = 2^{2m} - 1 = 4^m - 1 \ge m^2 \ge nm.$$

It follows from $\text{bin}\, 0.\theta_1\theta_2\cdots\theta_{nm} \le 1 - 2^{-nm} = 1 - \delta$ that

$$k + \text{bin}\, 0.\theta_1\theta_2\cdots\theta_{nm} \in \bigcup_{\ell=0}^{nm} [\ell, \ell+1-\delta] \subseteq \bigcup_{\ell=0}^{J-1} [\ell, \ell+1-\delta]$$

for $k = 0, 1, \cdots, nm$. Thus, we have

$$\widetilde{g}(k + \mathrm{bin}\, 0.\theta_1\theta_2\cdots\theta_{nm}) = k \quad \text{for } k = 0, 1, \cdots, nm. \tag{17}$$

It is easy to verify that

$$2^m \cdot \mathrm{bin}\, 0.\theta_{im+1}\cdots\theta_{nm} \in \bigcup_{\ell=0}^{2^m-1} [\ell, \ell + 1 - \delta] \quad \text{for } i = 0, 1, \cdots, n-1.$$

Since $2^m - 1 = 2^{n^r} - 1 \le 2^{(2n)^r} - 1 = J - 1$, we have

$$\widetilde{g}(2^m \cdot \mathrm{bin}\, 0.\theta_{im+1}\cdots\theta_{nm}) = \lfloor 2^m \cdot \mathrm{bin}\, 0.\theta_{im+1}\cdots\theta_{nm} \rfloor \quad \text{for } i = 0, 1, \cdots, n-1.$$

Therefore, for $i = 0, 1, \cdots, n-1$, we have

$$\mathrm{bin}\, 0.\theta_{im+1}\cdots\theta_{im+m} = \frac{\lfloor 2^m \cdot \mathrm{bin}\, 0.\theta_{im+1}\cdots\theta_{nm} \rfloor}{2^m} = \frac{\widetilde{g}(2^m \cdot \mathrm{bin}\, 0.\theta_{im+1}\cdots\theta_{nm})}{2^m}$$

and

$$\begin{aligned} \mathrm{bin}\, 0.\theta_{(i+1)m+1}\cdots\theta_{nm} &= 2^m\Big(\mathrm{bin}\, 0.\theta_{im+1}\cdots\theta_{nm} - \mathrm{bin}\, 0.\theta_{im+1}\cdots\theta_{im+m}\Big) \\ &= 2^m\Big(\mathrm{bin}\, 0.\theta_{im+1}\cdots\theta_{nm} - \frac{\widetilde{g}(2^m \cdot \mathrm{bin}\, 0.\theta_{im+1}\cdots\theta_{nm})}{2^m}\Big). \end{aligned}$$

By defining

$$\phi_1(x) \coloneqq \frac{\widetilde{g}(2^m x)}{2^m} \quad \text{and} \quad \phi_2(x) \coloneqq 2^m\Big(x - \frac{\widetilde{g}(2^m x)}{2^m}\Big) = \Big(\sigma(x) - \frac{\widetilde{g}(2^m x)}{2^m}\Big) \quad \text{for } x \ge 0,$$

we have

$$\mathrm{bin}\, 0.\theta_{im+1}\cdots\theta_{im+m} = \phi_1(\mathrm{bin}\, 0.\theta_{im+1}\cdots\theta_{nm}) \tag{18}$$

and

$$\mathrm{bin}\, 0.\theta_{(i+1)m+1}\cdots\theta_{nm} = \phi_2(\mathrm{bin}\, 0.\theta_{im+1}\cdots\theta_{nm}) \tag{19}$$

for any $i \in \{0, 1, \cdots, n-1\}$. Moreover, ϕ_1 can be realized by a one-hidden-layer $\widetilde{g}$-activated network of width 1; ϕ_2 can be realized by a one-hidden-layer $(\sigma, \widetilde{g})$-activated network of width 2.

Define

$$\phi_{3,i}(x) \coloneqq \min\{\sigma(x - im),\, m\} \quad \text{for any } x \in \mathbb{R} \text{ and } i = 0, 1, \cdots, n-1.$$

For any $k \in \{1, 2, \cdots, nm\}$, there exist $k_1 \in \{0, 1, \cdots, n-1\}$ and $k_2 \in \{1, 2, \cdots, m\}$ such that $k = k_1 m + k_2$. Then we have

$$\phi_{3,i}(k) = \min\{\sigma(k - im),\, m\} = \begin{cases} m & \text{if } i \le k_1 - 1 \\ k_2 & \text{if } i = k_1 \\ 0 & \text{if } i \ge k_1 + 1. \end{cases} \tag{20}$$

Observe that

$$\begin{aligned} \big\{1, 2, \cdots, k\big\} &= \big\{1, 2, \cdots, k_1 m + k_2\big\} \\ &= \Big(\bigcup_{i=1}^{k_1-1} \big\{im + j : j = 1, 2, \cdots, m\big\}\Big) \bigcup \big\{k_1 m + j : j = 1, 2, \cdots, k_2\big\}. \end{aligned}$$

It follows that

$$\begin{aligned} \sum_{\ell=1}^{k} \theta_\ell = \sum_{\ell=1}^{k_1 m + k_2} \theta_\ell &= \sum_{i=0}^{k_1-1}\Bigg(\sum_{j=1}^{m} \theta_{im+j}\Bigg) + \sum_{j=1}^{k_2} \theta_{k_1 m + j} + 0 \\ &= \sum_{i=0}^{k_1-1}\Bigg(\sum_{j=1}^{m} \theta_{im+j}\Bigg) + \sum_{i=k_1}^{k_1}\Bigg(\sum_{j=1}^{k_2} \theta_{im+j}\Bigg) + \sum_{i=k_1+1}^{n-1}\Bigg(\sum_{j=1}^{0} \theta_{im+j}\Bigg) \\ &= \sum_{i=0}^{k_1-1}\Bigg(\sum_{j=1}^{\phi_{3,i}(k)} \theta_{im+j}\Bigg) + \sum_{i=k_1}^{k_1}\Bigg(\sum_{j=1}^{\phi_{3,i}(k)} \theta_{im+j}\Bigg) + \sum_{i=k_1+1}^{n-1}\Bigg(\sum_{j=1}^{\phi_{3,i}(k)} \theta_{im+j}\Bigg) \\ &= \sum_{i=0}^{n-1}\Bigg(\sum_{j=1}^{\phi_{3,i}(k)} \theta_{im+j}\Bigg) \end{aligned} \tag{21}$$

for $k \in \{1, 2, \cdots, nm\}$, where the second to last equality comes from Equation (20). It is easy to verify that Equation (21) also holds for $k = 0$, i.e.,

$$\sum_{\ell=1}^{0} \theta_\ell = 0 = \sum_{i=0}^{n-1} \left(\sum_{j=1}^{0} \theta_{im+j} \right) = \sum_{i=0}^{n-1} \left(\sum_{j=1}^{\phi_{3,i}(0)} \theta_{im+j} \right).$$

Therefore, we have

$$\sum_{\ell=1}^{k} \theta_\ell = \sum_{i=0}^{n-1} \left(\sum_{j=1}^{\phi_{3,i}(k)} \theta_{im+j} \right) \quad \text{for any } k \in \{0, 1, \cdots, nm\}. \tag{22}$$

Fix $i \in \{0, 1, \cdots, n-1\}$. By setting $p = \phi_{3,i}(k) \in \{0, 1, \cdots, m\}$ and $\xi_j = \theta_{im+j}$ for $j = 1, 2, \cdots, m$ in Equation (16), we have

$$g\Big(\phi_{3,i}(k) + \text{bin}\, 0.\theta_{im+1}\theta_{im+2}\cdots\theta_{im+m}\Big) = \sum_{j=1}^{\phi_{3,i}(k)} \theta_{im+j}. \tag{23}$$

With Equations (17), (18), (19), (22), and (23) in hand, we are ready to construct the desired function ϕ, which can be realized by the NestNet in Figure 22. Clearly, we have

$$\phi(k + \text{bin}\, 0.\theta_1\cdots\theta_{nm}) = \sum_{\ell=1}^{k} \theta_\ell \quad \text{for } k = 0, 1, \cdots, nm.$$

Note that $nm = n \cdot n^r = n^{r+1}$. Then we have

$$\phi(k + \text{bin}\, 0.\theta_1\cdots\theta_{n^{r+1}}) = \sum_{\ell=1}^{k} \theta_\ell \quad \text{for } k = 0, 1, \cdots, n^{r+1}.$$

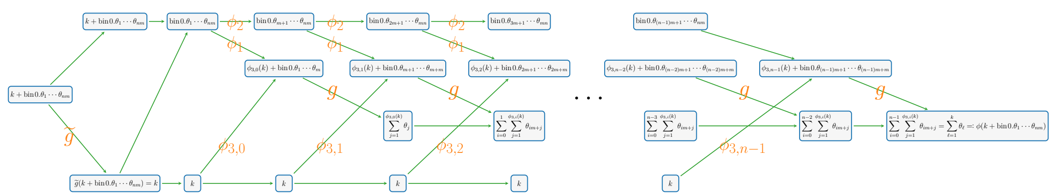


Figure 22: An illustration of the NestNet realizing ϕ based on Equations (17), (18), (19), (22), and (23). Here, g and $\widetilde{g}$ are regarded as activation functions.

It remains to estimate the number of parameters in the NestNet realizing ϕ. Recall that ϕ_1 can be realized by a one-hidden-layer $\widetilde{g}$-activated network of width 1 and ϕ_2 can be realized by a one-hidden-layer $(\sigma, \widetilde{g})$-activated network of width 2.

Observe that

$$\min\{a, b\} = \tfrac{1}{2}\Big(\sigma(a+b) - \sigma(-a-b) - \sigma(a-b) - \sigma(-a+b)\Big) \quad \text{for any } a, b \in \mathbb{R}.$$

As we can see from Figure 23, $\phi_{3,i}$ can be realized by a σ-activated network of width 4 and depth 2 for each $i \in \{0, 1, \cdots, n-1\}$.

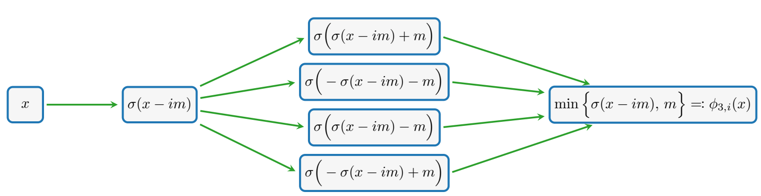


Figure 23: An illustration of $\phi_{3,i}$ for each $i \in \{0, 1, \cdots, n-1\}$.

Thus, the network in Figure 22 can be regarded as a $(\sigma, g, \widetilde{g})$-activated network of width $2 + 1 + 1 + 1 + 4 + 1 = 10$ and depth $2 + (2+1)n = 3n + 2$. Recall that $g \in \mathcal{NN}_r\{\widehat{n}\}$ and $\widetilde{g} \in \mathcal{NN}_r\{72(r+7)n\}$. This implies that ϕ can be realized by a height-$(r+1)$ NestNet with at most

$$\underbrace{(10+1)10\Big((3n+2)+1\Big)}_{\text{outer network}} + \underbrace{\widehat{n}}_{g} + \underbrace{72(r+7)n}_{\widetilde{g}} \le \widehat{n} + 114(r+7)(n+1)$$

parameters, which means we finish the proof of Lemma D.5.