Characterizing ResNet's Universal Approximation Capability

Chenghao Liu [1] **Enming Liang** [1] **Minghua Chen** [1]

Abstract

Since its debut in 2016, ResNet has become arguably the most favorable architecture in deep neural network (DNN) design. It effectively addresses the gradient vanishing/exploding issue in DNN training, allowing engineers to fully unleash DNN's potential in tackling challenging problems in various domains. Despite its practical success, an essential theoretical question remains largely open: how well/best can ResNet approximate functions? In this paper, we answer this question for several important function classes, including polynomials and smooth functions. In particular, we show that ResNet with constant width can approximate Lipschitz continuous function with a Lipschitz constant μ using $\mathcal{O}(c(d)(\varepsilon/\mu)^{-d/2})$ tunable weights, where $c(d)$ is a constant depending on the input dimension d and $\epsilon > 0$ is the target approximation error. Further, we extend such a result to Lebesgue-integrable functions with the upper bound characterized by the modulus of continuity. These results indicate a factor of d reduction in the number of tunable weights compared with the classical results for ReLU networks. Our results are also order-optimal in ε, thus achieving optimal approximation rate, as they match a generalized lower bound derived in this paper. This work adds to the theoretical justifications for ResNet's stellar practical performance.

1. Introduction

One trend in deep learning in the past decade is the use of larger and deeper neural networks to process higher-dimensional data. However, for deep neural networks, training is notoriously difficult, due to gradient vanishing (Glorot & Bengio, 2010). In 2016, the emergence of ResNet (He et al., 2016) addresses the issue of gradient vanishing or exploding encountered during neural network training and has achieved outstanding performance in applications.

The practical success of ResNet naturally leads to an essential theoretical question: how well can ResNet approximate functions? Along this line, a milestone result in (Lin & Jegelka, 2018) shows the universal approximation capability of ResNet (even with one neuron per layer): it can approximate any Lebesgue-integrable function arbitrarily well as the number of tunable weights goes to infinity. This result gives theoretical justification to using ResNet to approximate general functions and spurs a number of follow-up studies. For example, the authors in (Oono & Suzuki, 2019) explore the approximation capabilities of ResNet-type convolutional neural networks, establishing that they can approximate smooth functions with a comparable quantity of tunable weights as their feed-forward ReLU network counterparts. However, the question of whether this represents the optimal use of ResNet's resources remains unanswered. We delve deeper into this and other related research in Section 1.1. Despite of these exciting results, it remains largely open today to fully characterize the universal approximation capability of ResNet. That is, *how many tunable weights are needed for ResNet with optimized structures to approximate a function up to an error* ε?

In this paper, we seek answers to the above question, by developing upper-/lower- bounds on tunable weights for ResNet with constant width to approximate popular classes of functions. We summarize our **contributions** as follows:

▷ In Sec. 3, we explicitly establish the relationship between ResNet and feedforward networks (FNNs) (see Proposition 1). We show that ResNet can be viewed as an FNN and thus derive lower bounds on the number of tunable parameters for ResNet to approximate various classes of functions.

▷ In Sec. 4, we show that ResNet with constant width, by leveraging specific tunable weights, can approximate functions in various function classes. These include monomials with degree p, requiring the number of weights $\mathcal{O}(p \log p/\varepsilon)$, polynomials of degree p, needing $\mathcal{O}(p\#\text{terms} \log p/\varepsilon)$[1], and smooth functions differentiable

[1] School of Data Science, City University of Hong Kong. Correspondence to: Minghua Chen <minghua.chen@cityu.edu.hk>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1] The notation '#' is an abbreviation of number and '#terms' refers to the number of terms in the polynomial.

Table 1. A summary of existing and our results on approximation rate of ResNet structure. In the table, the upper bound refers to the number of parameters needed using the network architecture to approximate any function in the target function space to ε.

Paper	Function Space *	Network Architecture	Upper Bound	Optimality
(Lin & Jegelka, 2018)	Lipschitz Continuous Functions over $[0,1]^d$	ResNet with one neuron per layer	$\mathcal{O}_d(\varepsilon^{-d})$	Suboptimal‡
(Oono & Suzuki, 2019)	Lipschitz Continuous Functions over $[0,1]^d$	ResNet CNN with $\mathcal{O}(1)$ channels	$\mathcal{O}_d(\varepsilon^{-d})$	Suboptimal‡
Ours(Theorem 7)	Lipschitz Continuous Functions over $[0,1]^d$	ResNet with constant($=4$) width	$\mathcal{O}_d(\varepsilon^{-\frac{d}{2}})$ †	Optimal

$*$ Note that (Lin & Jegelka, 2018) and our Theorem 7 consider approximating Lebesgue-integrable functions and (Oono & Suzuki, 2019) consider Hölder class. All of them include Lipschitz functions that are used for comparison.
‡ The bound is optimal in terms of the entropy limitation. One could refer to Theorem 3 (Yarotsky, 2017) or (Yarotsky & Zhevnerchuk, 2020). However, the upper bound is not optimal in terms of Vapnik-Chervonenkis (VC) dimension (Shen et al., 2022b; Siegel, 2023).
† Our rate derived is explicit for all input-related parameters such as d.

up to degree r, needing $\mathcal{O}_{d,r}(\varepsilon^{-d/r}\log 1/\varepsilon)$[2]. In addition, we show that ResNet with one neuron per hidden layer can generate any continuous piece-wise linear function. Last but not least, we derive a tight upper bound of the number of tunable parameters of ResNet for approximating Lebesgue-integrable functions over $[0,1]^d$ which is $\mathcal{O}_d\left(\omega_f^{-1}(\varepsilon)^{-d/2}\right)$ where $\omega_f(t) := \sup\{|f(x)-f(y)| : \|x-y\|_2 \leq t\}$ is the modulus of continuity and $\omega_f^{-1}(r) := \sup\{t : \omega_f(t) \leq r\}$. Moreover, if f is Lipschitz continuous with Lipschitz constant μ, the upper bound becomes $\mathcal{O}(c(d)(\varepsilon/\mu)^{-d/2})$. Besides, Our bounds are explicit for all related parameters including the input dimension, the target function space, and the desired accuracy.

$\triangleright$ We highlight our results in Theorem 7, which achieves the optimal upper bound of the number of parameters of ResNet for approximating Lebesgue-integrable functions even by ResNet with constant width. This is a non-trivial extension of the work (Lin & Jegelka, 2018), which shows that for Lebesgue-integrable function f over $[0,1]^d$, there exists a ResNet R with one neuron per layer and not more than $\mathcal{O}(\omega_f^{-1}(\varepsilon)^{-d})$ parameters such that $\|f-R\| \leq \varepsilon$. Our results further establish that ResNet with constant width can approximate any Lebesgue-integrable function over $[0,1]^d$ to an error ε with an ε-order optimal upper bound $\mathcal{O}(\omega_f^{-1}(\varepsilon)^{-d/2})$ of the number of parameters needed. We summarize the comparison in Table 1.

These findings add to the theoretical justifications for ResNet's outstanding practical performance, and shed light on analysis for further research on NN design optimization.

1.1. Related Work

In recent years, the expressive capabilities of various neural network architectures have garnered increased attention, spurred by their remarkable and noteworthy successes. In this subsection, we will discuss previous research on the topic through the lens of approximation theory.

Universality. The universality, i.e., universal approximation property, of a function family implies that this family is dense in the space of continuous functions, meaning it can approximate any continuous function to an arbitrary precision. In the earlier years, (Cybenko, 1989; Hornik, 1991; Leshno et al., 1993; Pinkus, 1999) made a groundbreaking argument by demonstrating that shallow neural networks equipped with suitable activation functions such as sigmoid, non-polynomial functions, possess universal approximation properties. In recent years, the universality of narrow deep networks has also attracted considerable attention. (Hanin & Sellke, 2017) determined that a deep ReLU neural network must have a minimum width of $d+1$ to ensure universality, where d is the input dimension. (Kidger & Lyons, 2020; Park et al., 2020; Cai, 2022) later showed that deep narrow networks with reasonable activation functions can achieve universality, providing the minimum width of neural networks for achieving the universal approximation property. Over the past decades, a variety of network architectures have been developed to cater to diverse tasks and objectives, extending beyond feedforward ReLU networks. The universal approximation theorem has been shown for multiple network architectures, including standard ReLU convolutional neural networks (CNNs) (Zhou, 2018; 2020), 2D ReLU CNNs with classical structures (He et al., 2022), continuous-time recurrent neural network (RNN) (Li et al., 2020; 2022b), ResNet (Lin & Jegelka, 2018; Li et al., 2022a), and Transformers (Yun et al., 2019).

[2] The notation $a(\varepsilon) = \mathcal{O}(g(\varepsilon))$ means $a(\varepsilon) \leq Cg(\varepsilon)$ for sufficiently small ε where C is a constant independent of ε. Importantly, throughout this paper, we employ the notation $\mathcal{O}_d(\cdot)$ to underscore the hidden constant C depending on d.

Approximation Capabilities. There has been substantial progress in enhancing our theoretical understanding of neural networks. Some studies have focused on comparing the expressive power of both shallow and deep neural networks, examining their respective capabilities (e.g., (Arora et al., 2016; Eldan & Shamir, 2016; Liang & Srikant, 2016; Telgarsky, 2016; Yarotsky, 2017; Poggio et al., 2017)). Some others have quantified the number of linear regions within deep neural networks, casting light on their complexity (e.g., (Montufar et al., 2014; Serra et al., 2018; Arora et al., 2016)). Besides, constructive methods have been utilized to probe the approximation capabilities across different function classes. Notably, researchers have delved into the optimal approximation of continuous functions (e.g., (Shen et al., 2022b; Yarotsky, 2018)), the optimal approximation of smooth functions (e.g., (Yarotsky, 2017; Lu et al., 2021; Montanelli & Du, 2019)), and the approximation of analytic functions (e.g., (Wang et al., 2018; Schwab & Zech, 2021)). Recently, there is some interesting work on the special network architecture like parameters sharing (Zhang et al., 2023) and nested network (Zhang et al., 2022). These diverse investigations collectively deepen our understanding of the potential and constraints of neural networks in approximating various functions.

Perspectives on the Curse of Dimensionality. The 'curse of dimensionality' coined by (Bellman, 1957) refers to a phenomenon that a model class will suffer an exponential increase in its complexity as the input dimension increases. Importantly, the curse of dimensionality, not limited to FNNs, is also a challenge for almost all classes of function approximators due to the entropy limitation (Kolmogorov & Tikhomirov, 1959). More specifically, any continuous function approximator (refer to Ssec. 2.2 for detailed explanations) will suffer the curse of dimension in the Hölder space $C^r([0,1]^d)$ (DeVore et al., 1989) because the metric entropy in $C^r([0,1]^d)$ with respect to the uniform topology is $\Theta(\varepsilon^{-d/r})$. The property is applied to ReLU neural networks in Thm. 3 (Yarotsky, 2017).

In an attempt to mitigate the curse of dimensionality, initial strategies involved the consideration of smaller function classes whose metric entropy is expected to reduce such as analytical functions (Wang et al., 2018), bandlimited functions (Montanelli et al., 2019), Korobove space (Montanelli & Du, 2019), a space derived by Kolmogorov Superposition Theorem(KST) (Lai & Shen, 2021; He, 2023). More recently, researchers have shifted their focus toward the structure of neural networks, suggesting a potential solution to circumvent the curse of dimensionality. Simultaneously, a more recent trend aims to serve neural networks as discontinuous function approximators, thereby examining neural networks with novel activation functions (e.g. (Shen et al., 2020; Jiao et al., 2023; Shen et al., 2021; 2022a)). However, the failure of these model classes in practice is due to the discontinuity of the function approximators, wherein even minor perturbations in the training data can lead to chaotic changes in the input-output relationship. Consequently, to circumvent the curse of dimensionality, it is imperative to make appropriate choices within the unstable model class and the restricted objective function space.

ResNet Architecture. Since ResNet's first appearance, it has made a big success in practice and has become a core component of popular structures like Transformer (Vaswani et al., 2017), inspiring much research for theoretical understanding. (Hardt & Ma, 2016) show that ResNet can represent any classifier on any finite sample perfectly, i.e., it can represent any discrete function. Later, (Lin & Jegelka, 2018) extended this discrete setting to continuous, which shows that ResNet with one neuron per layer can approximate any Lebesgue-integrable function. The authors in (Oono & Suzuki, 2019) derive approximation and estimation error rates for ResNet-type CNN using $\mathcal{O}(1)$ channels. They show a block-sparse FNN can be realized by a ResNet-type CNN and then study the approximation capabilities of block-sparse FNNs. Our paper differs in this regard. We show that ResNet can be implemented by ReLU FNNs (Prop. 1), and we utilize this relationship to set the lower bounds on the approximation capability of ResNets. We establish the upper bounds by directly constructing ResNet to approximate different function classes. Since CNNs and FNNs are different structures, the results in (Oono & Suzuki, 2019) can not be translated to ours. Notably, while they show ResNet-type CNNs with not more than $\mathcal{O}(\varepsilon^{-d})$ parameters can achieve approximation for Lipschitz functions to an error ε, our Theorem 7 shows $\mathcal{O}(\varepsilon^{-d/2})$ parameters are adequate using ResNet with constant width. We summarize the comparison of the results in Table 1. Recently, (He et al., 2022) study the approximation properties of CNNs and show the universal approximation property of shallow ResNet-type CNN with a large number of channels.

2. Mathematical Modeling of ResNet and Proof Ideas

2.1. Mathematical modeling of ResNet

ResNet's original proposal (He et al., 2016) includes complex structures such as convolutional layers. In this paper, we focus on ResNet-type FNNs as it is a fundamental structure and enough to achieve a strong approximation capability and unless otherwise specified, we will refer to ResNet as ResNet-type FNNs. ResNet consists of residual blocks and identity shortcut connections. Specifically, with the basic notations of addition $(f+g)(x) = f(x) + g(x)$ and composition $f \circ g(x) = f(g(x))$ of mappings f, g, a ResNet is a function $R(x)$ from $\mathbb{R}^d$ to $\mathbb{R}$ given by

$$R(x) = \mathcal{L} \circ (\mathcal{T}^{[L]} + Id) \circ \cdots (\mathcal{T}^{[1]} + Id) \circ \mathcal{A}_Q(x), \quad (1)$$

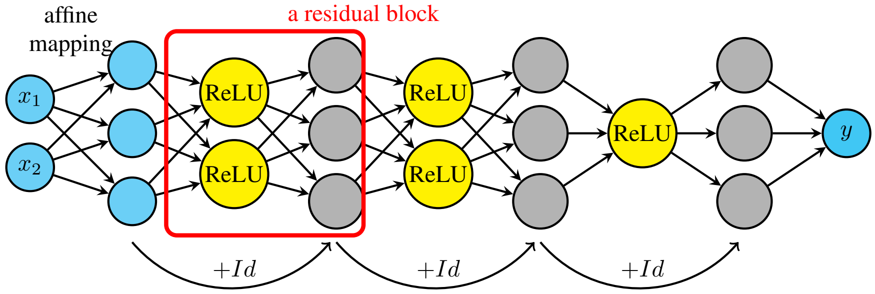


Figure 1. An example of a ResNet from $\mathbb{R}^2$ to $\mathbb{R}$ belonging to $\mathcal{RN}\,(Q=3, N=2, L=3)$. Every residual block is composed of an activation layer followed by a linear layer. The activation layer neurons are colored yellow, while the linear layer neurons are grey. The maximum number of neurons in each activation layer is 2, each linear layer has 3 neurons, and it has 3 blocks.

where $\mathcal{A}_Q : \mathbb{R}^d \to \mathbb{R}^Q$ and $\mathcal{L} : \mathbb{R}^Q \to \mathbb{R}$ are affine transformations, $Id : z \mapsto z$ is the identity mapping, and $\mathcal{T}^{[i]} (i = 0, 1, ..., L)$ are basic residual blocks. Each block $\mathcal{T}^{[i]} : \mathbb{R}^Q \to \mathbb{R}^Q$ further consists of two layers:

- an activation layer $\mathbb{R}^Q \to \mathbb{R}^{n_i} : z \mapsto \sigma(W_i z + b_i)$ with n_i neurons, and
- a linear layer $\mathbb{R}^{n_i} \to \mathbb{R}^Q : \sigma(W_i z + b_i) \mapsto V_i \sigma(W_i z + b_i)$ with Q neurons,

where parameters $W_i \in \mathbb{R}^{n_i \times Q}, V_i \in \mathbb{R}^{Q \times n_i}, b_i \in \mathbb{R}^{n_i}$ $(i = 1, 2, \cdots, L)$, and $\sigma(\cdot)$ is the generalized ReLU activation function for vector output, i.e., $\sigma(x_1, x_2, \cdots, x_n) = (\max 0, x_1, \cdots, \max 0, x_n)$ for $x_1, x_2, \cdots, x_n \in \mathbb{R}$. To this end, we write $\mathcal{T}^{[i]}(z) = V_i \sigma(W_i z + b_i)$.

We use L to denote ResNet's depth defined as the number of residual blocks[3], and its width N is defined as the maximum number of **activation layer neurons**, i.e., $\max\{n_1, n_2, \cdots, n_L\}$. For conciseness of notation, we denote by $\mathcal{RN}\,(Q, N, L)$ the set of ResNet functions from $\mathbb{R}^d$ to $\mathbb{R}$, with width N, depth L, and Q neurons in each linear layer. To keep our study interesting, we always assume that $Q \geq d$ where d is the input dimension. Otherwise, the universality of the ResNet does not hold[4]. For our later discussion, we define a ResNet as **ResNet with bottleneck blocks (b-ResNet)** if its width is an absolute constant, i.e., it belongs to $\mathcal{RN}\,(Q, N = C, L)$ where C is an absolute constant (independent of input dimension d). We have an example of a ResNet structure in Fig. 1.

2.2. Problem statement

In this paper, we care about the number of parameters of a ResNet needed for approximating a given function, which characterizes ResNet's approximation capabilities. Specifically, we give the following problem statement.

Given $f^\star : [0,1]^d \to \mathbb{R}$ belong to some function space $\mathcal{F}$ and fix $d \in \mathbb{N}_+$ and the ResNet model $\mathcal{H} = \mathcal{RN}\,(Q, N, L)$, we are interested in the following two questions specifically.

$\triangleright$ Lower bound. What is the minimum number of weights of ResNet required to approximate any $f^\star \in \mathcal{F}$ to an error ε?

$\triangleright$ Upper bound. What is the number of weights of a ResNet architecture sufficient to approximate any $f^\star \in \mathcal{F}$ to an error ε? Or does there exists Q, N, L such that $\inf_{\tilde{f} \in \mathcal{H}} \left\| \tilde{f} - f^\star \right\| \leq \varepsilon$ holds with relatively smaller number of tunable parameters?

Here tunable parameters refer to non-zero parameters in a ResNet. Note the answer to the two questions usually depends on the desired error ε, the input dimension d, and the function space $\mathcal{F}$. Usually we will consider continuous function space $C([0,1]^d)$ under uniform norm $\|f\|_\infty = \max_{x \in [0,1]^d} |f(x)|$ and L^p-integrable function space $L^p([0,1]^d)$ under norm $\|f\|_{L^p([0,1]^d)} = \int_{[0,1]^d} |f(x)| \,\mathrm{d}x$ where we always assume $p \in [1, \infty)$ without any specification. Our main results show that b-ResNet, i.e. ResNet in $\mathcal{RN}\,(Q, N, L)$ with $Q = d + c_1, N = c_2$ $(c_1, c_2$ are absolute constants) and proper L, can achieve powerful approximation capability in various function classes. It can approximate specific functions with fewer tunable functions than that of FNNs. Besides, b-ResNet can achieve optimal approximation for Lebesgue-integrable functions.

Remarks on continuous/discontinuous approximators. While the difference between continuous and discontinuous

[3] Note that we define the depth as the number of residual blocks because the activation and linear layers are always coupled together within each block. It is important to note that this definition differs from the one commonly used in applications, where the depth of a ResNet refers to the number of layers excluding the shortcuts. However, this difference does not affect the presentation of our results.

[4] If $Q < d$, the ResNet with one neuron per activation layer belongs to the set of narrow networks with widths smaller than or equal to d (Prop. 1). Therefore, the ResNet may lose the universality since narrow-width (smaller than $d+1$) ReLU networks cannot approximate all $\mathbb{R}^d$ continuous functions (Hanin & Sellke, 2017).

will play an important role in the lower bounds, we provide explanations about it here to make readers understand it readily. In approximation theory, we aim to approximate all functions in a space $\mathcal{F}$ using a model class as an approximator (e.g., neural networks). We achieve this by choosing different parameters for different functions, meaning the parameters $\theta \in \Theta$ can be seen as a mapping of the target functions, i.e., $\theta = h(f)$ where $h : \mathcal{F} \to \Theta$. If this mapping h is continuous, we refer to the approximator as a continuous approximator. Conversely, "discontinuous deep networks" are characterized by a discontinuous mapping h. In this paper, the term "arbitrary/unconstrained deep networks" refers to discontinuous approximators, and "construction in a continuous (discontinuous) phase" means the constructed neural networks are continuous (discontinuous) approximators. Significantly, the bound of approximation power for continuous approximators is limited by metric entropy, while for discontinuous approximators, it is limited by Vapnik-Chervonenkis (VC) dimension (Goldberg & Jerrum, 1993).

2.3. Proof Ideas and Novelty

Lower Bound. We establish that ResNet can be conceptualized as a sparse FNN (Prop. 1), implying that the lower bound of ResNet must exceed that of FNN. We then derive the generalized lower bounds of ResNet on the approximation of various function classes from the lower bounds of FNNs.

Upper Bounds for Approximation of Polynomials and Smooth Functions. Drawing inspiration from the work of (Yarotsky, 2017), where DNNs are constructed to approximate the function x^2 and xy, we construct ResNet to approximate these fundamental functions. Unlike the work (Yarotsky, 2017) which computes the product function using $xy = ((x+y)^2 - x^2 - y^2)/2$, we select $xy = ((x+y)/2)^2 - ((x-y)/2)^2$ as utilized in some previous work, e.g., (Suh et al., 2022). The selection is more efficient for construction and leads to fewer parameters. Next, we note that any polynomial can be expressed as a composition of the product function xy, and polynomials can approximate smooth functions due to the local Taylor expansion property. By constructing ResNet to approximate these functions, we can derive upper bounds on the approximation of polynomials and smooth functions. These constructions are in a continuous phase, which leads to the result Proposition 3 and Theorem 4 and 5. Our results show that b-ResNet is enough to approximate these smooth functions and has fewer tunable weights than that of FNN.

Table 2. High-level steps of constructing b-ResNet to achieve optimal approximation where the framework of constructive methods is from (Shen et al., 2019; 2022b). Details are in Appendix F.

Step 1: Space Partitions $[0,1]^d\backslash\Omega = \cup_{\boldsymbol{\beta}\in\{0,1,\cdots,\lfloor L^{2/d}\rfloor-1\}^d} Q_{\boldsymbol{\beta}}$. • $Q_{\boldsymbol{\beta}}$: small hypercubes with side length $\mathcal{O}(L^{-2/d})$ • $x_{\boldsymbol{\beta}}$: a representative for a cube $Q_{\boldsymbol{\beta}}$ • Ω: a small enough region • f: the target (Lipschitz) function from $[0,1]^d$ to $\mathbb{R}$
Step 2: Constructing a b-ResNet Φ such that $\Phi(x) = \boldsymbol{\beta}$. • b-ResNet Φ: depth $\mathcal{O}(L)$
Step 3: Constructing a b-ResNet ϕ such that $\phi(\boldsymbol{\beta}) \approx f(x_{\boldsymbol{\beta}})$. • b-ResNet ϕ: depth $\mathcal{O}(L)$
Step 4: Error Estimation. • Constructed b-ResNet $R(x) = \phi \circ \Phi(x) \approx f(x_{\boldsymbol{\beta}}) \approx f(x)$ • $\lvert f(x_{\boldsymbol{\beta}}) - f(x)\rvert = \mathcal{O}(L^{-2/d})$, $\lvert R(x) - f(x_{\boldsymbol{\beta}})\rvert = \mathcal{O}(L^{-2/d})$

Optimal Approximation for Lebesgue-Integrable Functions. Building upon the constructive techniques from (Shen et al., 2019; 2022b) on ReLU FNNs, we show how b-ResNet can achieve optimal approximation for Lebesgue-integrable functions (refer to Table 2 for high-level steps). The basic idea is to construct a ResNet with depth $\mathcal{O}(L)$ to generate a step function with much more steps to achieve a higher approximation rate. In our construction, we utilize linear layers for value storage and activation layers for intermediate computation. In each block, we designate a constant number of neurons within in the activation layer for the computation in relation to one of the corresponding neurons in the linear layer, with the outcomes refreshed via identity mappings in the next block. In distinct blocks, activation layer neurons perform computations associated with different neurons from the linear layer. This procedure is replicated for each residual block, which is then sequentially combined to form a b-ResNet which will exhibit a large expressive power with fewer parameters than that of FNNs. More details can be found in Appendix F. It is important to note that the FNN results in (Shen et al., 2022b) cannot be directly extended to ResNet.

Our novelty lies in the construction and analysis of b-ResNet, as well as in the approximation power analysis of ResNet in general. Our proof utilizes a new construction of ResNet blocks for function approximation. Our work is the first to show ResNet even with constant width, can achieve optimal approximation for Lebesgue-integrable functions. Furthermore, the role of identity mappings is greatly leveraged in the construction. We uncover the extensive expressive power of b-ResNet, providing theoretical guarantees and insights into the successful performance of ResNet.

3. Lower Bounds on ResNet's Function Approximation Capability

In this section, we build the explicit relation between ResNet and FNNs to establish the lower bound on the complexity of ResNet to approximate polynomials and other function classes such as smooth function class. The lower bounds help us to analytically show our upper bounds are optimal in terms of ε in Sec. 4. The proof in this section is postponed to Appendix B.

First, we propose a key argument that a ResNet can be regarded as a special sparse ReLU network.

Proposition 1. *For any ResNet $R(x) \in \mathcal{RN}(Q, N, L)$ of the input $x \in [0,1]^d$ with W number of tunable parameters, there exists an equivalent ReLU FNN $\Phi(x) : [0,1]^d \to \mathbb{R}$ with width $N+Q$, depth $2L$, $W + 2QL$ number of tunable parameters, such that $R(x) = \Phi(x)$, $\forall x \in [0,1]^d$.*

This proposition implies that if a ResNet can approximate a function up to an error ε, then an FNN with a larger size can also approximate the same function up to ε. Thus one can bound ResNet's universal approximation capability by studying that of a larger-size FNN (might have an increase in tunable parameters by a factor of d). That said, the lower bound of the complexity of ResNet must be larger than or equal to that of FNN **in terms of** ε when approximating the same function.

Thus, building on the above discussion, lower bounds for FNNs in the existing literature can be applied to ResNets in terms of ε. We discuss in the following. Regarding the smooth space $C^r([0,1]^d)$, (Yarotsky, 2017) established lower bounds of $\Theta_r(\epsilon^{-d/r})$[5] for continuous ReLU network approximators and $\Theta_{r,d}(\epsilon^{-d/2r})$ for unconstrained deep ReLU networks. Later, (Yarotsky, 2018) demonstrated optimal error approximation rates of $\mathcal{O}_d(\omega_f(W^{-1/d}))$ for continuous ReLU network approximators, and $\mathcal{O}_d(\omega_f(W^{-2/d}))$ for unconstrained deep ReLU networks, where W represents the number of tunable weights. Suppose f is Lipschitz continuous, the optimal upper bound for the ReLU FNN is $\mathcal{O}_d(\varepsilon^{-d})$ for continuous weight selection, and $\mathcal{O}_d(\varepsilon^{-d/2})$ for unconstrained deep networks.[6] We will return to these discussions in the next section. In the end of this section, we will focus on the lower bound for polynomial function space (a smaller space compared to continuous and smooth functions). However, the aforementioned lower bound is not applicable and it is not available in the existing literature. Below we give the lower bound of the complexity of ResNet on the approximation of polynomials. The result will be used to show the upper bound in Thm. 4 is ε-order optimal in the next section.

We begin with some notations. Let $x \in \mathbb{R}^d$ and $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_d)$ where $\alpha_i \in \mathbb{N}$. Define $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d}$ and this is called a monomial. The degree of the monomial is $|\alpha| := \alpha_1 + \alpha_2 + \cdots + \alpha_d$. Then a multivariate polynomial is a sum of several monomials and its degree is the highest degree among these monomials. We then give the theorem as below.

Theorem 2. *Let $x = [x_1, x_2, \ldots, x_d] \in [0,1]^d$ and $\mathcal{P}(d,p) (p \geq d)$ be the set of d dimension polynomial functions with degree p. If a ReLU FNN $\Psi(x) : [0,1]^d \to \mathbb{R}^d$ with width N, depth L and $T = NL$ neurons can approximate any $f \in \mathcal{P}(d,p)$ to an error ε, i.e.*

$$|\Psi(x) - f(x)| < \varepsilon, \forall x \in [0,1]^d,$$

then we have $T \geq \Theta_d(\log 1/\varepsilon)$. Note this lower bound can also be applied to ResNet.

When we compare Theorem 2 with prior findings, we observe that the lower bound for the polynomial function space, in terms of ε, is notably smaller than the lower bounds $\Theta_r(\epsilon^{-d/r})$ established for smooth functions $C^r([0,1]^d)$. This is because polynomial functions constitute a substantially smaller subset of the smooth function space. Consequently, they exhibit a reduced complexity in approximation scenarios, reflecting the inherent simplicity of their structural characteristics.

4. Upper Bounds on ResNet's Function Approximation Capability

In this section, we characterize ResNet's approximation capability by establishing its upper bounds for important function classes. Moreover, our results show that b-ResNet can approximate polynomials and smooth functions with fewer tunable parameters than those of FNNs. The organization of this section is in the following. Subsection 4.1 presents the upper bounds of the complexity of ResNet on approximating monomials. Subsequently in subsection 4.2, we extend the results to polynomials, and smooth functions in Sobolev space following the work of (Yarotsky, 2017). In Subsection 4.3, the properties of ResNet on approximating continuous piecewise linear functions are discussed. Last but not least, Subsection 4.4 shows that b-ResNet can achieve the optimal approximation for Lebesgue-integrable functions.

4.1. Approximating Monomials

Our first key result shows that b-ResNet $\mathcal{RN}(Q = d + \mathcal{O}(1), N = C, L)$ can approximate any monomials where L depends on d and the desired error ε, and C is an absolute constant independent of d. Importantly, this result explains why b-ResNet can approximate polynomials and smooth functions with fewer tunable parameters. The proof in this section can be found in Appendix C.

Proposition 3. *Let $x = [x_1, x_2, \cdots, x_d] \in [-M, M]^d$ and $\alpha \in \mathbb{N}^d$, where $M \geq 1$ and x^α be any given monomial with*

[5] Here $f(\varepsilon) = \Theta_d(g(\varepsilon))$ means $f(\varepsilon) \sim Cg(\varepsilon)$ for sufficient small ε where C is a constant which does not depend on ε but can depend on d. The subscript d on Θ emphasizes that the constant C may depend on d.

[6] As pointed out in Sec. 2.2, the bound of approximation power for continuous approximators is limited by metric entropy, while for discontinuous approximators, it is limited by VC dimension.

degree p, i.e., $|\alpha| = p$*. Then there exists a b-ResNet*

$$R \in \mathcal{RN}\left(d+3, 4, \mathcal{O}(p\log(p/\varepsilon) + p^2 \log M)\right)$$

such that

$$\|R - x^\alpha\|_{C([-M,M]^d)} < \varepsilon,$$

while having $\mathcal{O}\left(p\log\left(p/\varepsilon\right) + p^2 \log M\right)$ *tunable weights.*

Note that the number of total weights of R is $\mathcal{O}(dp\log(p/\varepsilon))$ when $M = 1$. However, our constructive proof shows that each residual block only contains a constant number of non-zero weights. Therefore, it suffices to adjust $\mathcal{O}(p\log(p/\varepsilon))$ weights. This analysis also holds when $M > 1$. Furthermore, it is important to note that the upper bound in the above theorem is independent of d. In fact, the dimension d is incorporated into p because any monomial with degree p can always be interpreted as a product function of dimension p. For instance, $x_1^2 x_2 x_3^2 = \pi(x_1, x_1, x_2, x_3, x_3)$ where $\pi(x_1, \cdots, x_d) = x_1 x_2 \cdots x_d$ is the product function.

Next, we highlight several key observations and discussions from the analysis and results in the following.

ResNet vs FNNs. We show that ResNet is capable of approximating any monomial with degree p on $[0,1]^d$ with $\mathcal{O}\left(p\log(p/\varepsilon)\right)$ number of tunable weights, a reduction by a factor d as compared to that of ReLU FNNs. According to (DeVore et al., 2021), a ReLU network with width $\mathcal{O}(d)$ and depth $\mathcal{O}\left(p\log(p/\varepsilon)\right)$ can approximate any monomial with degree p, resulting in a total weight count of $\mathcal{O}\left(d^2 p\log(p/\varepsilon)\right)$. According to their construction, there are $\mathcal{O}\left(dp\log(p/\varepsilon)\right)$ tunable weights. Thus, our result has a reduction by a factor d. As a closed remark, while we compare the upper bounds, it is an open and interesting direction to obtain the lower bound of the size of neural networks with respect to input dimension d.

Root of reduction. Note that each identity mapping can be realized by $2d$ ReLU units ($x = [x]_+ - [-x]_+$) but it only may lead to an "additive" reduction of d tunable weights compared with an FNN. In our study, however, this additive reduction of d tunable weights, as seen in our b-ResNet model, does translate into a multiplicative reduction by a factor of d. We establish this result by constructively proving that a b-ResNet with a constant number of tunable weights per residual block can approximate functions with the same accuracy as a ReLU FNN requiring $\mathcal{O}(d)$ tunable weights in each layer. The high-level ideas are in the following. First, one can construct an FNN with width $d + \mathcal{O}(1)$ that can approximate a function, and in each layer, there are d neurons for storing the value of the input. Hence if we have an identity mapping, then we can move the d neurons in each layer and the role of identity mapping is to forward the input value (or input-related value). Then it will have a factor d reduction.

Deep vs Shallow. The author in (Shapira, 2023) provides a lower bound on the complexity of shallow FNNs to approximate any non-normalized monomial over $[-M, M]^d$ which scales exponentially with d (refer to Thm. 3 (Shapira, 2023)). By proposition 1, this lower bound also applies to shallow ResNet. Conversely, Theorem 3 gives a mild upper bound for deep ResNet which scales polynomially with d. This underscores the benefits of deep networks.

4.2. Approximating Polynomials and Smooth Functions

Polynomials are the summation of monomials and a smooth function can be approximated by a polynomial as per the local Taylor expansion. In this subsection, we display the upper bounds on the approximation of polynomials (Thm. 4) and smooth functions (Thm. 5). The proof of the two theorems can be found in Appendix C and D.

Theorem 4. *Let* $x = [x_1, x_2, \cdots, x_d] \in [0,1]^d$*. For a multivariate polynomial* $P(x)$ *with degree p, i.e.,* $P(x) = \sum_{\alpha\in E} c_\alpha x^\alpha$ *where* $E = \{\alpha \in \mathbb{N}^d : |\alpha| \le p\}$*, there exists a ResNet*

$$R \in \mathcal{RN}\left(d+4, 4, \mathcal{O}\left(p|E|\log\left(p/\varepsilon\right)\right)\right)$$

such that

$$|R(x) - P(x)| < \varepsilon \cdot \sum_{|\alpha|\in E} |c_\alpha|, \quad \forall x \in [0,1]^d.$$

Additionally, the ResNet has $\mathcal{O}(p|E|\log(p/\varepsilon))$ *tunable weights.*

Note $|E| \le \binom{p+d}{p}$ which exponentially increase in d when p or d is very large. Nonetheless, we can see this upper bound is optimal in terms of ε according to Thm. 2.

The Sobolev space $W^{r,\infty}([0,1]^d)$ is the set of functions belonging to $C^{r-1}([0,1]^d)$ whose $(r-1)$-th order derivatives are Lipschitz continuous. Further definitions can be found in Appendix D. We then give the upper bounds of ResNet's complexity in the following theorem.

Theorem 5. *Fix* $r, d \in \mathbb{N}_+$*. There is a ResNet* $R(x) \in \mathcal{RN}\left(d+4, N=4, L\right)$ *that can approximate any function from the unit ball of* $W^{r,\infty}([0,1]^d)$ *to* ε *where* $L = \mathcal{O}_{d,r}(\varepsilon^{-d/r}\log 1/\varepsilon)$*.*

The number of tunable parameters is still less than that of FNN by a factor d based on the polynomial approximation methods. However, the hidden constant $c(d,r)$ in $\mathcal{O}_{d,r}(\varepsilon^{-d/r}\log 1/\varepsilon)$ is very large in d where an estimation is given as $(\frac{2^{\frac{d+1}{r}}d}{r})^d < c(d,r) < (\frac{2^{\frac{d+1}{r}}d}{r})^d d^{r+2}(d+r)r$. As we discussed before, (Yarotsky, 2017) established the lower bound $\Theta_r(\varepsilon^{-d/r})$ of the tunable weights for continuous ReLU network approximators on the approximation of the Sobolev space $W^{r,\infty}([0,1]^d)$. Based on Prop. 1, the lower bound can also apply to ResNet. Thus, we can see the upper bound in Theorem 5 is nearly tight up to a $\log$ factor for ResNet. Besides, it is noteworthy that recent results (Yang & Zhou, 2024) show that shallow ReLU neural networks can also reach this upper bound when $r < (d+3)/2$.

4.3. Representing Continuous Piecewise Linear Functions

Piecewise linear interpolation holds a significant position in approximation theory, as it is a basic method of approximating functions. Therefore, studying the expressive power of neural networks for piecewise linear functions becomes particularly important. In this section, we show that ResNet even with one neuron per activation layer can generate any continuous piecewise linear function (CPwL), as shown in the theorem below where the proof is in Appendix E.

Theorem 6. *For any CPwL function $f : \mathbb{R}^d \to \mathbb{R}$, there exists a ResNet $R(x) \in \mathcal{RN}(d+1, 1, L)$ with $L = \mathcal{O}(Md)$ that can exactly represent f where M is an f-dependent number.*

Note that M is implicit which is an f-dependent number. It depends on the property of the input CPwL function f including the number of pieces and linear components. More details about the representation of CPwL functions can be found in (Tarela & Martinez, 1999; Wang & Sun, 2005). Moreover, a recent work (Chen et al., 2022) derived a dimension-independent bound for ReLU networks if the number of pieces and linear components of the target CPwL function is known. Their constructions are also possible for ResNet. It should be importantly noted that Theorem 6 provides another approach to demonstrating the universal property as the CPwL functions are dense in continuous function space under uniform norms. While (Lin & Jegelka, 2018) shows that ResNet with one neuron per activation layer can approximate any step function, the construction of Theorem 6 is much easier.

4.4. Optimal Approximation for Lebesgue-Integrable Functions

In this subsection, we show that even b-ResNet can achieve optimal approximation for L^p-integrable functions. It is well-known that $C([0,1]^d)$ space is dense in $L^p([0,1]^d)$ where $p \in [1, \infty)$ under L^p norm. Thus, we just need to consider the approximation for $C([0,1]^d)$ as shown in the following theorem. The proof can be found in Appendix F.

Theorem 7. *Let $d \in \mathbb{N}^+$, $d \geq 5$ and $p \in [1, \infty)$. For any given continuous function $f \in C([0,1]^d)$, there exists a ResNet $R \in \mathcal{RN}(d+1, 4, 24L+9d+4)$ such that*

$$\|f - R\|_{L^p([0,1]^d)} \leq 7\sqrt{d}\omega_f(L^{-2/d}).$$

As a direct corollary, the number of parameters needed is $\mathcal{O}_d\left(\omega_f^{-1}(\varepsilon)^{-d/2}\right)$ for a ResNet to approximate the function f to an error ε.

To discuss about the theorem, we have the following concluding remarks.

Optimality. As discussed in Sec. 3, the lower bound of the size of deep ReLU networks for approximating continuous functions over $[0,1]^d$ under uniform norm is $\mathcal{O}(\varepsilon^{-d/2})$. Moreover, this lower bound also holds for the approximation under L^p norm which comes from a recent result (Siegel, 2023). Hence, it follows from Proposition 1 that this lower bound can also apply to ResNet which shows our upper bound in Theorem 7 is ε-order optimal.

Extension to Uniform Approximation. The optimal bound in Theorem 7 is extendable to the case under the uniform norm, as facilitated by Lemma 3.4 (Lu et al., 2021). This extension, however, necessitates an increased width of the ResNet architecture. As such, it is non-trivial to demonstrate that bottleneck ResNets (b-ResNets) can maintain this optimal rate when approximating continuous functions under the uniform norm.

Extension to Entired Domain $\mathbb{R}^d$. Note that for any function f belonging to $L^p(\mathbb{R}^d)$, and given an arbitrary error margin $\epsilon > 0$, there exists a compact set $\mathcal{H}$ upon which h is defined and outside of which it is zero (i.e., $h(x) = 0$ for $x \in \mathbb{R}^d \setminus \mathcal{H}$) (Walter, 1987). If we take $\mathcal{H}$ to be contained within the hypercube $[-H, H]^d$, we can define a new function $g(x) = h\left(\frac{x+H}{2H}\right)$ mapped onto the unit cube $[0,1]^d$. Function g is continuous by construction and adheres to the assumptions of Theorem 7.

Non-trivial Bound Extension. In this part, we clarify that our results are non-trivial and can not be derived from the work (Yarotsky, 2018). It is important to note that his construction allows ReLU FNNs with a width of $2d + 10$ to attain the approximation rate in our Theorem 7. However, within Yarotsky's framework, d neurons per layer can be adapted to form an identity mapping. If one were to attempt to extend these bounds to ResNets in a straightforward manner, the conclusion would be that a ResNet with width $d + \mathcal{O}(1)$ could achieve the mentioned rate. Our Theorem 7, on the other hand, demonstrates that a ResNet with a constant width—completely independent of d—is capable of achieving the same rate of approximation. This finding is interesting as it implies a reduction in the number of parameters by a factor of d when compared to FNNs, underscoring the non-trivial nature of our extension.

Trade-off between Depth and Width. As a closing remark, this paper concentrates on b-ResNet, that is, ResNet with a constant width. Besides, for the set of ResNets denoted as $\mathcal{RN}(Q, N, L)$, it is an interesting future direction to explore the trade-off between Q, N and L.

Table 3. A summary of upper bounds of the size of FNN and ResNet on the approximation of two representative types of functions: polynomials and Lipschitz functions. Note the relevant papers primarily focus on approximating functions such as Hölder functions, Sobolev functions, continuous functions, and Lebesgue-integrable functions. However, they all encompass Lipschitz functions, so we use Lipschitz functions for our comparison.

Functions Network	Polynomial of Degree p with $\|E\|$ Terms	Lipschitz Functions with Lip. Constant 1
Shallow ReLU FNNs	$\mathcal{O}\left(\|E\|p^{3/2}\varepsilon^{-1}\log(1/\varepsilon)\right)$,[1]	$\mathcal{O}\left(c_1(d)^{\dagger}\varepsilon^{-d}\right)$,[3]
Deep ReLU FNNs	$\mathcal{O}\left(\|E\|pd\log(p/\varepsilon)\right)$,[2]	$\mathcal{O}\left(c_2(d)^{\dagger}\varepsilon^{-d/2}\right)$,[4,5]
ResNet with Constant Width	$\mathcal{O}\left(\|E\|p\log(p/\varepsilon)\right)$	$\mathcal{O}\left(c_3(d)^{\dagger}\varepsilon^{-d/2}\right)$

†: $c_1(d)$ implicitly depends on d, $c_2(d) > (3^d d^2)^d$, and $c_3(d) = (7\sqrt{d})^{d/2}$.
The related references are: [1] (Blanchard & Bennouna, 2021),[2] (DeVore et al., 2021),[3] (Yang & Zhou, 2024),[4] (Yarotsky & Zhevnerchuk, 2020) and [5] (Yarotsky, 2018).

Table 4. Comparison of MSE loss.

NN structure	$d = 100$	$d = 200$	$d = 300$
$\mathcal{NN}(d+1, d/10)$	0.0139	0.0216	0.0472
$\mathcal{RN}(d+1, 10, d/10)$	0.0102	0.0131	0.0225
$\mathcal{RN}(d+1, 20, d/10)$	0.0093	0.0127	0.0228
$\mathcal{RN}(d+1, 40, d/10)$	0.0091	0.0131	0.0230

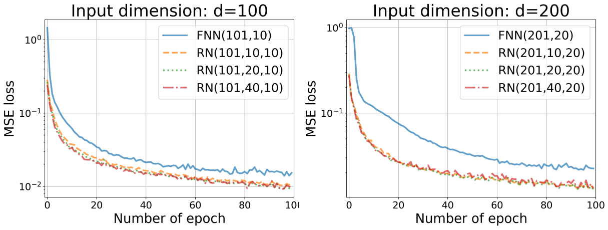


Figure 2. Comparison of testing MSE loss for FNN and b-RestNet to approximate high dimensional functions defined in Equation (2). More experiment results can be found in Appendix G.

5. Experiments

In this section, we provide function approximation results to numerically validate the theoretical results presented in Sec. 4. To emphasize the approximation error, we involve a sufficiently complex target function for the experiment. Specifically, we utilize the following set of functions (where a_i, b_i are parameters) to test the universal approximation capability of b-ResNet.

$$f(x) = \sum_{i=1}^{m} [a_i \prod_{j \in S_i^1} x_j + b_i \sin(\prod_{k \in S_i^2} x_k)]. \tag{2}$$

The parameter settings in Equation (2) are included in Appendix G. We then compare b-ResNet with fully connected NN for approximating the function in Equation (2) with different dimensions. The results are shown in Figure 2 and Table 4.

From the experiments, it is evident that under the same dimensionality, the b-ResNet has a reduced number of parameters compared to a classical FNN. However, it achieves a lower MSE. This, in some sense, demonstrates the remarkable function approximation capability of the b-ResNet. Moreover, we take into account that the function approximation task is of high dimensionality, which in turn reflects the capacity of ResNets to learn high-dimensional functions.

In conclusion, the experiment results demonstrate (i) the exceptional approximation capability of b-ResNet for learning complex functions, (ii) efficient structure design which highly reduces training parameters, and (iii) strong scalability for approximating high-dimension functions.

6. Conclusion

In this study, we provide a substantial theoretical contribution to the understanding of ResNet's capabilities in function approximation. We show that ResNet with constant width possesses the remarkable ability to approximate various important functions including polynomials, smooth functions, and piecewise linear functions. Importantly, we show that even ResNet with constant width can achieve optimal approximation for Lebesgue-integrable functions which are frequently encountered in practical applications. Moreover, we obtain some improvement compared with FNN. To make more explicit comparisons between rates of approximation by ResNet and FNN, we have Table 3 showing a summary. While ResNet has significant optimization advantages over FNNs during training, our results indicate that ResNets can achieve strong approximation capabilities with even fewer parameters than FNNs. These findings add to the theoretical justifications for ResNet's stellar practical performance.

Acknowledgements

This work is supported in part by a General Research Fund from Research Grants Council, Hong Kong (Project No. 11200223), an InnoHK initiative, The Government of the HKSAR, Laboratory for AI-Powered Financial Technologies, and a Shenzhen-Hong Kong-Macau Science & Technology Project (Category C, Project No. SGDX20220530111203026). The authors would also like to thank the anonymous reviewers for their helpful comments.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*, 2016.

Bellman, R. Dynamic programming (new jersey: Princeton university press). 1957.

Blanchard, M. and Bennouna, M. A. Shallow and deep networks are near-optimal approximators of korobov functions. In *International Conference on Learning Representations*, 2021.

Cai, Y. Achieve the minimum width of neural networks for universal approximation. In *The Eleventh International Conference on Learning Representations*, 2022.

Chen, K.-L., Garudadri, H., and Rao, B. D. Improved bounds on neural complexity for representing piecewise linear functions. *Advances in Neural Information Processing Systems*, 35:7167–7180, 2022.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

DeVore, R., Hanin, B., and Petrova, G. Neural network approximation. *Acta Numerica*, 30:327–444, 2021.

DeVore, R. A., Howard, R., and Micchelli, C. Optimal nonlinear approximation. *Manuscripta mathematica*, 63: 469–478, 1989.

Eldan, R. and Shamir, O. The power of depth for feedforward neural networks. In *Conference on learning theory*, pp. 907–940. PMLR, 2016.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Goldberg, P. and Jerrum, M. Bounding the vapnik-chervonenkis dimension of concept classes parameterized by real numbers. In *Proceedings of the sixth annual conference on Computational learning theory*, pp. 361–369, 1993.

Hanin, B. and Sellke, M. Approximating continuous functions by relu nets of minimal width. *arXiv preprint arXiv:1710.11278*, 2017.

Hardt, M. and Ma, T. Identity matters in deep learning. *arXiv preprint arXiv:1611.04231*, 2016.

He, J. On the optimal expressive power of relu dnns and its application in approximation with kolmogorov superposition theorem. *arXiv preprint arXiv:2308.05509*, 2023.

He, J., Li, L., and Xu, J. Approximation properties of deep relu cnns. *Research in the Mathematical Sciences*, 9(3): 38, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Jiao, Y., Lai, Y., Lu, X., Wang, F., Yang, J. Z., and Yang, Y. Deep neural networks with relu-sine-exponential activations break curse of dimensionality in approximation on hölder class. *SIAM Journal on Mathematical Analysis*, 55(4):3635–3649, 2023.

Kidger, P. and Lyons, T. Universal approximation with deep narrow networks. In *Conference on learning theory*, pp. 2306–2327. PMLR, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kolmogorov, A. N. and Tikhomirov, V. M. ε-entropy and ε-capacity of sets in function spaces. *Uspekhi Matematicheskikh Nauk*, 14(2):3–86, 1959.

Lai, M.-J. and Shen, Z. The kolmogorov superposition theorem can break the curse of dimensionality when approximating high dimensional functions. *arXiv preprint arXiv:2112.09963*, 2021.

Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

Li, Q., Lin, T., and Shen, Z. Deep learning via dynamical systems: An approximation perspective. *Journal of the European Mathematical Society*, 2022a.

Li, Z., Han, J., Li, Q., et al. On the curse of memory in recurrent neural networks: Approximation and optimization analysis. *arXiv preprint arXiv:2009.07799*, 2020.

Li, Z., Han, J., Weinan, E., and Li, Q. Approximation and optimization theory for linear continuous-time recurrent neural networks. *J. Mach. Learn. Res.*, 23:42–1, 2022b.

Liang, S. and Srikant, R. Why deep neural networks for function approximation? *arXiv preprint arXiv:1610.04161*, 2016.

Lin, H. and Jegelka, S. Resnet with one-neuron hidden layers is a universal approximator. *Advances in neural information processing systems*, 31, 2018.

Lu, J., Shen, Z., Yang, H., and Zhang, S. Deep network approximation for smooth functions. *SIAM Journal on Mathematical Analysis*, 53(5):5465–5506, 2021.

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

Montanelli, H., Yang, H., and Du, Q. Deep relu networks overcome the curse of dimensionality for bandlimited functions. *arXiv preprint arXiv:1903.00735*, 2019.

Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. On the number of linear regions of deep neural networks. *Advances in neural information processing systems*, 27, 2014.

Oono, K. and Suzuki, T. Approximation and non-parametric estimation of resnet-type convolutional neural networks. In *International conference on machine learning*, pp. 4922–4931. PMLR, 2019.

Park, S., Yun, C., Lee, J., and Shin, J. Minimum width for universal approximation. In *International Conference on Learning Representations*, 2020.

Pinkus, A. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195, 1999.

Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., and Liao, Q. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5):503–519, 2017.

Schwab, C. and Zech, J. Deep learning in high dimension: Neural network approximation of analytic functions in $L^2(\mathbb{R}^d, \gamma_d)$. *arXiv preprint arXiv:2111.07080*, 2021.

Serra, T., Tjandraatmadja, C., and Ramalingam, S. Bounding and counting linear regions of deep neural networks. In *International Conference on Machine Learning*, pp. 4558–4566. PMLR, 2018.

Shapira, I. Expressivity of shallow and deep neural networks for polynomial approximation. *arXiv preprint arXiv:2303.03544*, 2023.

Shen, Z., Yang, H., and Zhang, S. Deep network approximation characterized by number of neurons. *arXiv preprint arXiv:1906.05497*, 2019.

Shen, Z., Yang, H., and Zhang, S. Deep network approximation with discrepancy being reciprocal of width to power of depth. *arXiv preprint arXiv:2006.12231*, 2020.

Shen, Z., Yang, H., and Zhang, S. Neural network approximation: Three hidden layers are enough. *Neural Networks*, 141:160–173, 2021.

Shen, Z., Yang, H., and Zhang, S. Deep network approximation: Achieving arbitrary accuracy with fixed number of neurons. *The Journal of Machine Learning Research*, 23(1):12653–12712, 2022a.

Shen, Z., Yang, H., and Zhang, S. Optimal approximation rate of relu networks in terms of width and depth. *Journal de Mathématiques Pures et Appliquées*, 157:101–135, 2022b.

Siegel, J. W. Optimal approximation rates for deep relu neural networks on sobolev and besov spaces. *Journal of Machine Learning Research*, 24(357):1–52, 2023.

Suh, N., Zhou, T.-Y., and Huo, X. Approximation and nonparametric estimation of functions over high-dimensional spheres via deep relu networks. In *The Eleventh International Conference on Learning Representations*, 2022.

Tarela, J. and Martinez, M. Region configurations for realizability of lattice piecewise-linear models. *Mathematical and Computer Modelling*, 30(11-12):17–27, 1999.

Telgarsky, M. Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101*, 2015.

Telgarsky, M. Benefits of depth in neural networks. In *Conference on learning theory*, pp. 1517–1539. PMLR, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Walter, R. Real and complex analysis. 1987.

Wang, Q. et al. Exponential convergence of the deep neural network approximation for analytic functions. *arXiv preprint arXiv:1807.00297*, 2018.

Wang, S. and Sun, X. Generalization of hinging hyperplanes. *IEEE Transactions on Information Theory*, 51(12):4425–4431, 2005.

Yang, Y. and Zhou, D.-X. Optimal rates of approximation by shallow relu k neural networks and applications to nonparametric regression. *Constructive Approximation*, pp. 1–32, 2024.

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

Yarotsky, D. Optimal approximation of continuous functions by very deep relu networks. In *Conference on learning theory*, pp. 639–649. PMLR, 2018.

Yarotsky, D. and Zhevnerchuk, A. The phase diagram of approximation rates for deep neural networks. *Advances in neural information processing systems*, 33:13005–13015, 2020.

Yun, C., Bhojanapalli, S., Rawat, A. S., Reddi, S. J., and Kumar, S. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*, 2019.

Zhang, S., Shen, Z., and Yang, H. Neural network architecture beyond width and depth. *Advances in Neural Information Processing Systems*, 35:5669–5681, 2022.

Zhang, S., Lu, J., and Zhao, H. On enhancing expressive power via compositions of single fixed-size relu network. *arXiv preprint arXiv:2301.12353*, 2023.

Zhou, D.-X. Deep distributed convolutional neural networks: Universality. *Analysis and Applications*, 16(06):895–919, 2018.

Zhou, D.-X. Universality of deep convolutional neural networks. *Applied and computational harmonic analysis*, 48 (2):787–794, 2020.

A. Preliminaries

In this section, we introduce some basic notations for use in subsequent proofs.

A.1. Feedforward neural networks (FNNs)

As is known to all, FNN is a function $\Phi : \mathbb{R}^d :\rightarrow \mathbb{R}$ which is formed as the alternating compositions of ReLU function σ, and affine transformations $\mathcal{A}^{[i]}(y) = U_i y + v_i$ with $U_i \in \mathbb{R}^{d_i \times d_{i-1}}, v_i \in \mathbb{R}^{d_i}, d_0 = d$ for $i = 1, 2, \cdots, L$. Specifically,

$$\Phi(x) = \mathcal{L} \circ \sigma \circ \mathcal{A}^{[L]} \circ \sigma \circ \mathcal{A}^{[L-1]} \circ \cdots \circ \sigma \circ \mathcal{A}^{[1]}(x)$$

where $\mathcal{L}$ is a final affine transformation. Here L denotes the number of layers of the FNN, and the width of the FNN is conventionally defined by $\max\{d_1, d_2, \cdots, d_L\} := K$. The ReLU activation function is defined by:

$$\sigma(x) := \operatorname{ReLU}(x) = \max(x, 0) = (x)_+, x \in \mathbb{R}$$

and for $x \in \mathbb{R}^d$, $\sigma(x) := (\sigma(x_1), \cdots, \sigma(x_d))$. Typically, it is presumed that the number of neurons in each layer of an FNN is the same, which is equal to the width K, as any neuron deficits in a layer can be dealt with by adding $K - d_j$ neurons whose biases are zero in layer j. The weights between these extra neurons are consequently assigned to zero.

A.2. ResNet

Let $d, Q \in \mathbb{N}^+$. ResNet $R(x) : \mathbb{R}^d \rightarrow \mathbb{R}$ is a combination of an initial affine layer, multiple basic residual blocks with identity mapping, and a final affine output layer:

$$R(x) = \mathcal{L} \circ (\mathcal{T}^{[L]} + Id) \circ (\mathcal{T}^{[L-1]} + Id) \circ \cdots \circ (\mathcal{T}^{[1]} + Id) \circ \mathcal{A}_Q(x), \tag{3}$$

where $\mathcal{A}_Q : \mathbb{R}^d \rightarrow \mathbb{R}^Q$ and $\mathcal{L} : \mathbb{R}^Q \rightarrow \mathbb{R}$ are affine transformations. Besides, $\mathcal{T}^{[i]}(i = 0, 1, ..., L)$ are basic residual blocks, i.e., $\mathcal{T}^{[i]}(z) = V_i \sigma(W_i z + b_i)$ where $W_i \in \mathbb{R}^{n_i \times Q}, V_i \in \mathbb{R}^{Q \times n_i}, b_i \in \mathbb{R}^{n_i}$.

Concretely, we denote the output of the i-th block by $z^{[i]}$. Then the outputs of each block can be formulated as follows:

$$\begin{aligned}
&z^{[0]} = \mathcal{A}_Q(x) = W_0 x + b_0, \\
&\mathcal{T}^{[i]}(z) = V_i \sigma(W_i z + b_i), \\
&z^{[i]} = \mathcal{T}^{[i]}\left(z^{[i-1]}\right) + z^{[i-1]}, \quad i = 1, 2, \cdots, L, \\
&R(x) = \mathcal{L}\left(z^{[L]}\right) = B z^{[L]},
\end{aligned} \tag{4}$$

where $W_0 \in \mathbb{R}^{Q \times d}, b_0 \in \mathbb{R}^Q, W_i \in \mathbb{R}^{n_i \times Q}, V_i \in \mathbb{R}^{Q \times n_i}, b_i \in \mathbb{R}^{n_i}, B \in \mathbb{R}^{1 \times Q}$ and $x \in \mathbb{R}^d$.

The ResNet's depth, denoted by L, is defined as the number of residual blocks. The ResNet's width is the maximum number of neurons in the activation layer, that is $\max\{n_1, n_2, ..., n_L\}$. The subscript Q of $\mathcal{A}_Q$ refers to the number of neurons in the linear layer. We denote by $\mathcal{RN}(Q, N, L)$ the set of ResNet functions width N, depth L and Q neurons in each linear layer.

Additionally, we define

$$\begin{aligned}
&\zeta^{[i]} = \mathcal{T}_1^{[i]}(z^{[i-1]}) = \sigma(W_i z + b_i), \\
&\gamma^{[i]} = \mathcal{T}^{[i]}\left(z^{[i-1]}\right) = \mathcal{T}_2^{[i]}\left(\zeta^{[i]}\right) = V_i \zeta^{[i]} = V_i \mathcal{T}_1^{[i]}\left(z^{[i-1]}\right) \quad \text{and} \\
&z^{[i]} = z^{[i-1]} + \gamma^{[i]}
\end{aligned} \tag{5}$$

for $i = 1, 2, \cdots, L$. See Fig. 3 for an illustration.

A.3. Notations

We summarize the notations we will use in this paper in the following.

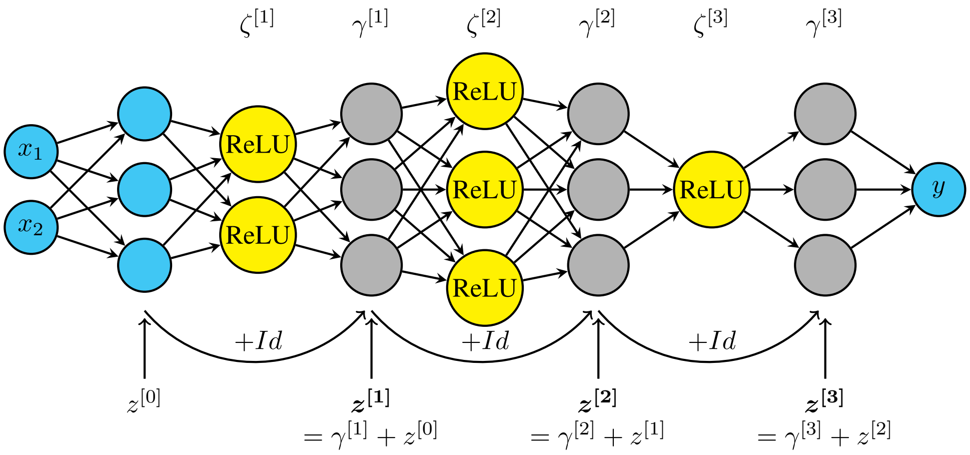


Figure 3. An illustration figure for the ResNet notation. The yellow neurons are in the activation layer and the grey neurons are in the linear layer.

- Let column vectors $a_i \in \mathbb{R}^{m_i}$, where $i = 1, 2, \cdots, n$ and $m_i \in \mathbb{N}_+ := \{1, 2, \cdots\}$. To represent these vectors concisely, we use $(a_1, a_2, \cdots, a_n)$ to denote $\left[a_1^T, a_2^T, \cdots, a_n^T\right]^T \in \mathbb{R}^{m_1+m_2+\cdots+m_n}$. Here, a^T denotes the transpose of a. Let $a \in \mathbb{R}^m$. If we write a vector as $(\mathbb{R}^l, a) \in \mathbb{R}^{m+l}$, this implies that the value of the vector in the position represented by '$\mathbb{R}^l$' does not matter. If $l = 1$, we always use '-' to substitute '$\mathbb{R}$', i.e., $(-, a) \in \mathbb{R}^{m+1}$ implies the value of the position represented by '-' does not matter. For a vector v in $\mathbb{R}^m$, v_i is the i-th entry of v for $i = 1, 2, \cdots, m$.

- Denote by $\mu(\mathcal{T})$ the Lebesgue measure of a measurable set $\mathcal{T}$.

- Let $\mathbb{1}_S$ be the characteristic function on a set S, i.e., $\mathbb{1}_S = 1$ on set S and 0 otherwise.

- For two sets A, B, $A \backslash B := \{x : x \in A, x \notin B\}$.

- For any $\xi \in \mathbb{R}$, let $\lfloor \xi \rfloor := \max\{i : i \leq \xi, i \in \mathbb{Z}\}$ and $\lceil \xi \rceil := \min\{i : i \geq \xi, i \in \mathbb{Z}\}$.

B. Proof of Proposition 1 and Theorem 2

In this appendix, we provide the proofs of conclusions in Sec. 3.

B.1. Proof of Proposition 1

For a ResNet in $\mathcal{RN}(Q, N, L)$ from $[0,1]^d$ to $\mathbb{R}$ defined by the formula 3, 4 and 5, we now construct a special network with depth L and width $N + Q$ having the same output. We first suppose the input of the network is $y^{[0]} = \mathcal{A}_Q(x) = z^{[0]}$ and denote the output of the i-th layer by $y^{[i]}$. What's more, we assume in each layer the bottom Q neurons of each layer are ReLU-free, i.e., the activation function of them is identity mapping $\sigma(x) = x$. The activation of the rest of neurons are ReLU. Then by assigning some weights to the first layer, we can have $y^{[1]} = (\zeta^{[1]}, z^{[0]})$. In the next layer, we can easily compute $z^{[1]} = V_1\zeta^{[1]} + z^{[0]}$. Then we have $y^{[2]} = (\mathbb{R}^N, z^{[1]})$. Now assume $y^{[2i]} = (\mathbb{R}^N, z^{[i]})$. Then in the first N neurons of the next layer, we compute $\mathcal{T}_1^{[i+1]}(z^{[i]}) = \mathrm{ReLU}(W_{i+1}z^{[i]} + b^i)$. In the bottom Q ReLU-free neurons, we copy $z^{[i]}$. Then $y^{[i+1]} = \left(\zeta^{[i+1]}, z^{[i]}\right)$. Then in the next layer, we can compute

$$z^{[i+1]} = V_{i+1}\zeta^{[i+1]} + z^{[i]}$$

i.e., $y^{[2(i+1)]} = (\mathbb{R}^N, z^{[i+1]})$. By induction, we have found a special $2L$ deep network with top N ReLU neurons and bottom Q ReLU-free neurons having the same output as the ResNet. This process can be seen in Figure 4.

Next, we construct a real ReLU network Φ that has the same size and output as the special Network. Because the domain $[0,1]^d$ is compact, there exists $C_i \in \mathbb{R}^Q$ such that $z^{[i]} + C_i > 0$ for all $i = 0, 1, 2, \cdots, L$. Now, we suppose the first layer of Φ is $u^{[0]} = \mathrm{ReLU}(\mathcal{A}_Q(x) + C_0) = \mathrm{ReLU}(z^{[0]} + C_0)$. Denote the j-th layer of Φ is $u^{[j-1]}$. In each subsequent layer of

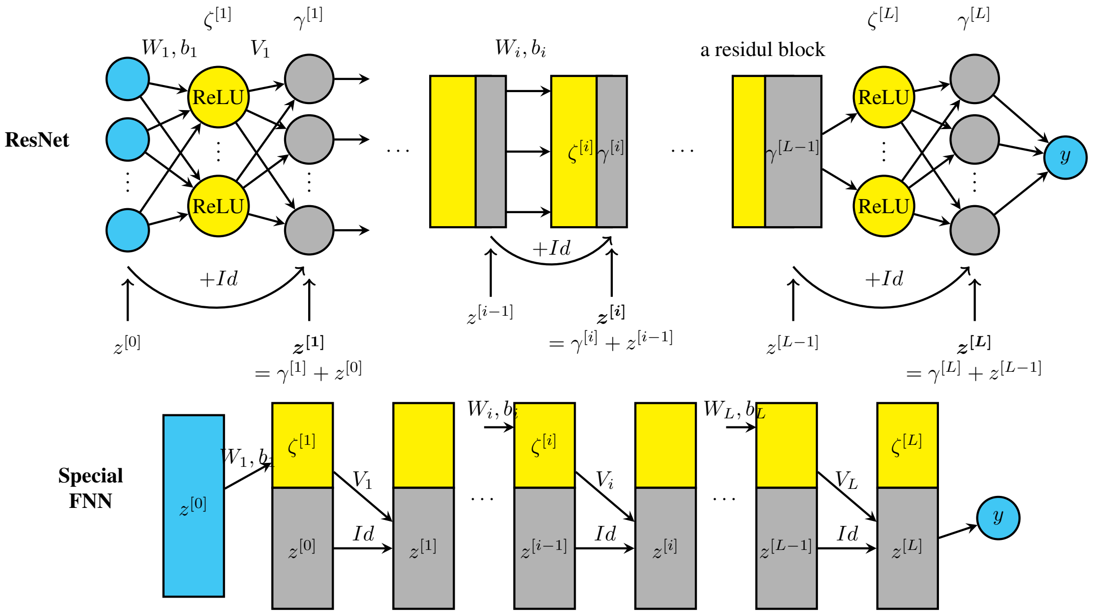


Figure 4. ResNet (top) can be generated by a special FNN (bottom). All grey neurons are ReLU-free and all yellow neurons are with ReLU activation. Moreover, for ResNet, the yellow neurons are in the activation layer and the grey neurons are in the linear layer.

Φ, the width is $N+Q$. We denote $u^{[i]} = (u^{[i]}_{(N)}, u^{[i]}_{(Q)})$ where $u^{[i]}_{(N)}$ is the value of top N neurons and $u^{[i]}_{(Q)}$ is the value of bottom Q neurons in the layer $i+1$. Then note $\mathcal{T}_1^{[1]}(z^{[0]}) = \mathrm{ReLU}(W_1 z^{[0]} + b_1)$. We can compute

$$u^{[1]}_{(N)} = \mathrm{ReLU}\left(W_1(u^{[0]} - C_0) + b_1\right) = \mathrm{ReLU}(W_1 z^{[0]} + b_1) = \zeta^{[1]}$$

and $u^{[1]}_{(Q)} = u^{[0]} = \mathrm{ReLU}(z^{[0]} + C_0)$. Note $z^{[i]} = \mathrm{ReLU}(z^{[i]} - C_i) + C_i$. We can compute

$$u^{[2]}_{(Q)} = \mathrm{ReLU}\left(V_1 u^{[1]}_{(N)} + u^{[1]}_{(Q)} - C_0 + C_1\right) = \mathrm{ReLU}\left(V_1 \zeta^{[1]} + z^{[0]} + C_1\right) = \mathrm{ReLU}(z^{[1]} + C_1).$$

Now, we suppose $u^{[2j]} = (\mathbb{R}^N, \mathrm{ReLU}(z^{[j]} + C_j))$. Then we can compute

$$u^{[2j+1]}_{(N)} = \mathrm{ReLU}\left(W_{j+1}(u^{[2j]} - C_j) + b_{j+1}\right) = \mathrm{ReLU}(W_{j+1} z^{[j]} + b_{j+1}) = \mathcal{T}_1^{[j+1]}(z^{[j]}) = \zeta^{[j+1]}$$

and $u^{[j+1]}_{(Q)} = \mathrm{ReLU}(u^{[j]}_{(Q)}) = \mathrm{ReLU}(z^{[0]} + C_0)$. Then in the next layer,

$$\begin{aligned} u^{[2j+2]}_{(Q)} &= \mathrm{ReLU}\left(V_{j+1} u^{[2j+1]}_{(N)} + u^{[2j+1]}_{(Q)} - C_j + C_{j+1}\right) \\ &= \mathrm{ReLU}\left(V_{j+1} \mathcal{T}_1^{[j+1]}(z^{[j]}) + z^{[j]} + C_{j+1}\right) \\ &= \mathrm{ReLU}(z^{[j+1]} + C_{j+1}). \end{aligned}$$

By induction, we can output $z^{[L]}$ in the last layer, i.e., $u^{[2L]} = (\mathbb{R}^N, \mathrm{ReLU}(z^{[L]} + C_L))$. Then output $z^{[L]}$ by some affine transformation $\mathcal{A}(u^{[2L]}) = \mathrm{ReLU}(z^{[L]} + C_L) - C_L = z^{[L]}$.

From the construction of the ReLU network, we can see the FNN has $W + 2kL$ non-zero training weights.

B.2. Proof of Theorem 2

Because the product function $f(x) = x_1 x_2 \cdots x_d$ belongs to $\mathcal{P}(d,p)$, it suffices to show the lower bound of the complexity of an FNN on the approximation of f is $\Theta_d(\log 1/\varepsilon)$. Let $x \in [0,1]$. Define $\widetilde{\Psi}(x) = \Psi(x, x, \cdots, x)$ and $\widetilde{f}(x) =$

$f(x, x, \cdots, x) = x^d$. Then by the assumption, we have

$$\left|\widetilde{\Psi}(x) - \widetilde{f}(x)\right| < \varepsilon, \quad x \in [\frac{1}{2}, 1].$$

In the interval $[1/2, 1]$, $\widetilde{f}$ is strictly convex because

$$\widetilde{f}''(x) = d(d-1)x^d \geq d(d-1)(\frac{1}{2})^d := c_1 > 0.$$

By lemma 2.1 in (Telgarsky, 2015), $\widetilde{\Psi}$ is a CPwL function over $[0, 1]$ with at most $(2N)^L$ linear pieces, i.e. $[1/2, 1]$ is partitioned into at most $(2N)^L$ intervals for which $\widetilde{\Psi}$ is linear. Now, we divide $[1/2, 1]$ into $(2N)^L$ intervals. Thus, there exists an interval $[a, b] \subset [1/2, 1]$ with $b - a \geq \frac{1}{2(2N)^L}$ over which $\widetilde{\Psi}$ is linear. Then define

$$G(x) = \widetilde{f}(x) - \widetilde{\Psi}, x \in [a, b].$$

Then $|G(x)| < \varepsilon$ and $G''(x) \geq c_1 > 0$ for any $x \in [a, b]$ due to the linearity of $\widetilde{f}$. Then we consider $x \in [a, b]$ and local taylor expansion at $(a + b)/2$:

$$G(x) = G(\frac{a+b}{2}) + G'(\frac{a+b}{2})(x - \frac{a+b}{2}) + \frac{G''}{2}(\xi)(x - \frac{a+b}{2})^2 \quad \text{where} \xi \in [a, b].$$

Then let $x = a$ and $x = b$, we have

$$\begin{aligned} G(a) &= G(\frac{a+b}{2}) - G'(\frac{a+b}{2})(\frac{b-a}{2}) + \frac{G''}{2}(\xi)(\frac{b-a}{2})^2, \quad \xi \in [a, \frac{a+b}{2}] \quad \text{and} \\ G(b) &= G(\frac{a+b}{2}) + G'(\frac{a+b}{2})(\frac{b-a}{2}) + \frac{G''}{2}(\eta)(\frac{b-a}{2})^2, \quad \eta \in [\frac{a+b}{2}, b]. \end{aligned}$$

It follows that

$$\max\{G(a), G(b)\} \geq G(\frac{a+b}{2}) + \frac{c_1}{2}(\frac{b-a}{2})^2.$$

Thus, by noting $b - a \geq \frac{1}{2(2N)^L}$ we have

$$2\varepsilon > \max\{G(a), G(b)\} - G(\frac{a+b}{2}) \geq \frac{c_1}{2}(\frac{b-a}{2})^2 \geq \frac{c_1}{2}\left(\frac{1}{4(2N)^L}\right)^2.$$

Then

$$(2N)^{2L} \geq \frac{c}{\varepsilon}$$

where c is a constant depending on d. It follows from the number of neurons $T = NL$ that

$$\begin{aligned} &\log \frac{2T}{L} \geq \frac{1}{2L} \log \frac{c}{\varepsilon} \\ \Longleftrightarrow\ &4T \geq \frac{u}{\log u} \log \frac{c}{\varepsilon} \geq \log \frac{c}{\varepsilon} \quad \text{where} \quad u = \frac{2T}{L}. \end{aligned}$$

Therefore, the number of neurons must be at least the order $\log 1/\varepsilon$.

C. Proof of Proposition 3 and Theorem 4

Our ideas in this appendix are from (DeVore et al., 2021). It should be noted that while (DeVore et al., 2021) inspired our approach to polynomial approximation, there are big differences in the construction details. Significantly, our main contribution is the successful demonstration of ResNet's construction proof.

The discussion in Subsection C.1 commences with a consideration of the fundamental functions, ranging from x^2 to xy, which we use to construct our approximation using ResNet. Subsequently, in Subsection C.2, we begin by establishing Proposition 3 for the case $[0, 1]^d$. This is then extended to the case $[-M, M]^d$ for $M > 1$ in Subsection C.3.

C.1. prelininaries

We recall that the so-called hat function h is defined by

$$h(x) = 2(x)_+ - 4\left(x - \frac{1}{2}\right)_+ + 2(x-1)_+. \tag{6}$$

Let $h_m(x)$ be the m-fold composition of the function h, i.e. $h_m = \underbrace{h \circ h \circ \cdots \circ h}_{m \text{ times}}$ which is the so-called sawtooth function. Then

$$x^2 = x - \sum_{m \geq 1} 4^{-m} h_m(x), \quad x \in [0,1].$$

Next, we define

$$S(x) := x^2 \quad \text{and} \quad S_n(x) := x - \sum_{m=1}^{n} 4^{-m} h_m(x), \quad n \geq 1, \quad x \in [0,1].$$

We then have

$$|S(x) - S_n(x)| \leq \sum_{i=n+1}^{\infty} 4^{-i} \leq \frac{1}{3} \cdot 4^{-n}, \quad x \in [0,1]. \tag{7}$$

$S_n(x)$ is a piecewise linear interpolation of S on $[0,1]$, using $2^n + 1$ uniformly distributed breakpoints, as indicated in (Yarotsky, 2017) (see Proposition 1). Using equation 7, we can generate S_n and approximate x^2.

Proposition 8. *There exists a ResNet $R(x) \in \mathcal{RN}(2,4,L)$ with $L = \mathcal{O}(\log 1/\varepsilon)$ such that*

$$\left|R(x) - x^2\right| < \varepsilon, x \in [0,1]$$

while having $\mathcal{O}(\log 1/\varepsilon)$ neurons. Especially, $R(x) \in \mathcal{RN}(4,2,n)$ generate S_n exactly.

Proof. It suffices to construct a ResNet required to represent S_n. Then we let the right-hand side of (7) equal to ε, i.e., $\frac{1}{3}4^{-n} = \varepsilon$. We then have $n = \mathcal{O}(\log 1/\varepsilon)$. Next, we construct a ResNet $R(x) \in \mathcal{RN}(4,2,n)$ generating $S_n(x)$ exactly.

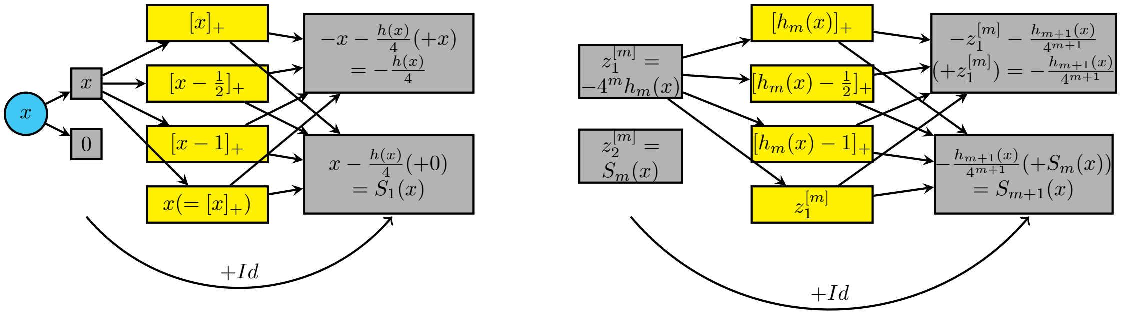

x
x
0
$[x]_+$
$[x-\frac{1}{2}]_+$
$[x-1]_+$
$x(=[x]_+)$
$-x - \frac{h(x)}{4}(+x) = -\frac{h(x)}{4}$
$x - \frac{h(x)}{4}(+0) = S_1(x)$
$+Id$
$z_1^{[m]} = -4^m h_m(x)$
$z_2^{[m]} = S_m(x)$
$[h_m(x)]_+$
$[h_m(x)-\frac{1}{2}]_+$
$[h_m(x)-1]_+$
$z_1^{[m]}$
$-z_1^{[m]} - \frac{h_{m+1}(x)}{4^{m+1}} (+z_1^{[m]}) = -\frac{h_{m+1}(x)}{4^{m+1}}$
$-\frac{h_{m+1}(x)}{4^{m+1}}(+S_m(x)) = S_{m+1}(x)$
$+Id$

Figure 5. Illustration for the constructive first block (left) and the m-th block (right). Grey represents the linear layer and yellow represents the activation layer.

Let $z^{[0]} = \mathcal{A}_2(x) = (x, 0)$. Then by equation (6), we can use three ReLU units to store $(x)_+, (x-1/2)_+$ and $(x-1)_+$ and hence compute $h(x)$. At the same layer, use one ReLU unit to copy x. Then in the first residual block, we can compute

$$\gamma^{[1]} = \mathcal{T}^{[1]}(z^{[0]}) = (-x - h(x)/4, x - h(x)/4)$$

so that

$$z^{[1]} = \gamma^{[1]} + z^{[0]} = (-x - h(x)/4, x - h(x)/4) + (x, 0) = (-h(x)/4, S_1(x)).$$

See Figure 5 (left) for illustration.

Now we assume the output of the m-th block is $z^{[m]} = (-4^m h_m(x), S_m(x))$. Also by (6) and assigning appropriate weights, we can use three units to compute $h(h_m(x)) = h_{m+1}(x)$ and use one unit to copy $-4^m h_m(x)$ by $a = -(-a)_+$. Thus, we can output

$$\gamma^{[m+1]} = \mathcal{T}^{[m+1]}(z^{[m]}) = (-4^{m+1}h_{m+1}(x) - 4^m h_m(x), -4^{m+1}h_{m+1}(x))$$

in the next block so that

$$z^{[m+1]} = (-4^{m+1}h_{m+1}(x) - 4^m h_m(x), -4^{m+1}h_{m+1}(x)) + z^{[m]} = (-4^{m+1}h_{m+1}(x), S_{m+1}(x)).$$

See Figure 5 (right) for illustration. By induction, we complete our proof. Concretely, $z^{[n]} = (-4^n h_n(x), S_n(x))$ and $R(x) = \mathcal{L}(z^{[n]}) = S_n(x)$ by letting $\mathcal{L}(x_1, x_2) = x_2$. □

Let $x, y \in [0,1]$. We can approximate the product function xy by using the equality $xy = (\frac{|x+y|}{2})^2 - (\frac{|x-y|}{2})^2$. Define

$$\pi_n(x,y) = S_n(\frac{|x+y|}{2}) - S_n(\frac{|x-y|}{2}). \tag{8}$$

It then follows from equation (7),

$$|\pi_n(x,y) - xy| \leq 4^{-n}, \quad \forall x, y \in [0,1]. \tag{9}$$

For the later rigorous derivation, we also need to prove the following lemma:

Lemma 9.

$$\pi_n(x,y) \in [0,1], \quad \forall x, y \in [0,1]. \tag{10}$$

Proof. According to the definition of S_n, we have

$$x^2 \leq S_n(x) \leq x, \quad \text{for} \quad x \in [0,1].$$

Then

$$\begin{aligned}
\pi_n(x,y) &= S_n(\frac{x+y}{2}) - S_n(\frac{|x-y|}{2}) \\
&\leq \frac{x+y}{2} - \frac{(x-y)^2}{4} \\
&= \frac{1}{4}\left(x(2-x) + y(2-y) + 2xy\right) \\
&\leq \frac{1}{4}(1+1+2) = 1.
\end{aligned}$$

Next, we show $\pi_n(x,y) \geq 0$. We start from

$$\begin{aligned}
\pi_n(x,y) &= S_n(\frac{x+y}{2}) - S_n(\frac{|x-y|}{2}) \\
&= \frac{x+y}{2} - \sum_{i=1}^{n} 4^{-i} h_i(\frac{x+y}{2}) - \frac{|x-y|}{2} + \sum_{i=1}^{n} 4^{-i} h_i(\frac{|x-y|}{2}) \\
&= \min\{x,y\} + \sum_{i=1}^{n} 4^{-i}\left(h_i(\frac{|x-y|}{2}) - h_i(\frac{x+y}{2})\right).
\end{aligned} \tag{11}$$

Now we introduce the function

$$\zeta(x) := 2\min\{|x-s| : s \in \mathbb{Z}\}, \quad x \in \mathbb{R}.$$

Then for $x \in [0,1]$ we have

$$h(x) = \zeta(x) \quad \text{and} \quad h_m(x) = \zeta(2^{m-1}x), m \geq 2.$$

Since ζ is subadditive, i.e. $\zeta\left(t+t'\right) \leq \zeta(t)+\zeta\left(t'\right)$, we have

$$\begin{aligned} h_i(\frac{x+y}{2}) &= h_i(\frac{|x-y|}{2}+\min\{x,y\}) \\ &= \zeta(2^{i-1}(\frac{|x-y|}{2}+\min\{x,y\})) \\ &\leq \zeta(2^{i-1}\frac{|x-y|}{2})+\zeta(2^{i-1}\min\{x,y\}) \\ &= h_i(\frac{|x-y|}{2})+h_i(\min\{x,y\}). \end{aligned} \tag{12}$$

Namely,

$$h_i(\frac{|x-y|}{2})-h_i(\frac{x+y}{2}) \geq -h_i(\min\{x,y\}).$$

From (11), we then have

$$\pi_n(x,y) \geq \min\{x,y\}-\sum_{i=1}^{n} 4^{-i}\left(\min\{x,y\}\right) = S_n(\min\{x,y\}) \geq \min\{x,y\}^2 \geq 0.$$

□

For $x,y \in [-M,M]$, we can approximate xy by the following remark.

Remark 10. *If $x,y \in [-M,M]$, we can approximate xy by using the equality $xy = M^2\left[(\frac{|x+y|}{2M})^2-(\frac{|x-y|}{2M})^2\right]$. Because the domain of S_n is $[0,1]$, we define*

$$\widehat{\pi}_n(x,y) = M^2\pi_n(\frac{x}{M},\frac{y}{M}) = M^2\left[S_n(\frac{|x+y|}{2M})-S_n(\frac{|x-y|}{2M})\right]. \tag{13}$$

We have

$$|\widehat{\pi}_n(x,y)-xy| \leq M^2 4^{-n}, \forall x,y \in [-M,M]. \tag{14}$$

Now we show ResNet can approximate the product function xy.

Proposition 11. *Let $x,y \in [-M,M]$. There exists a ResNet*

$$R \in \mathcal{RN}\left(3,(4,L)\right)$$

from $[-M,M]$ to $\mathbb{R}$ with $L = \mathcal{O}(\log M/\varepsilon)$ such that

$$|R(x,y)-xy| < \varepsilon, x,y \in [-M,M]$$

while having $\mathcal{O}(\log M/\varepsilon)$ neurons and tunable weights. Especially, the ResNet $R(x,y)$ with width 4 and depth $2n$ can generate $\pi_n(x,y)$ exactly.

Proof. It suffices to construct a ResNet required to output $\pi_n(x,y)$. Let the right-hand side of (14) equal ε. We can get $n = \mathcal{O}(\log M/\varepsilon)$. Now, we construct a ResNet with width 4 and depth $2n$ to represent $\pi_n(x,y)$.

Let $\mathcal{A}_3(x) = (\frac{x+y}{2M},\frac{x-y}{2M},0)$. Next, we can use the first block to output $z^{[1]} = (\frac{|x+y|}{2M},\frac{|x-y|}{2M},0)$ by the simple observation $|a| = (a)_+ + (-a)_+$. See Figure 6 for illustration.

Then, it follows from the proof of 8 that we can use the first n layers and 4 units in each layer to output $z^{[n+1]} = (-,\frac{|x-y|}{2M},S_n(\frac{|x+y|}{2M}))$ while keeping the value of the second neuron in each linear layer unchanged. Next, by the same operation, we use the next n blocks to output

$$z^{[2n+1]} = (-,-,S_n(\frac{x+y}{2})-S_n(\frac{|x-y|}{2})) = (-,-,\pi_n(x,y)/M^2).$$

Thus, $R(x,y) = \mathcal{L}(-,-,\pi_n(x,y)/M^2) = \pi_n(x,y)$ by letting $\mathcal{L}(x_1,x_2,x_3) = M^2 x_3$. See Figure 7 for illustration.

□

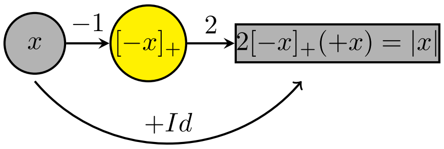


Figure 6. Illustration for computing $|x|$ by a residual block.

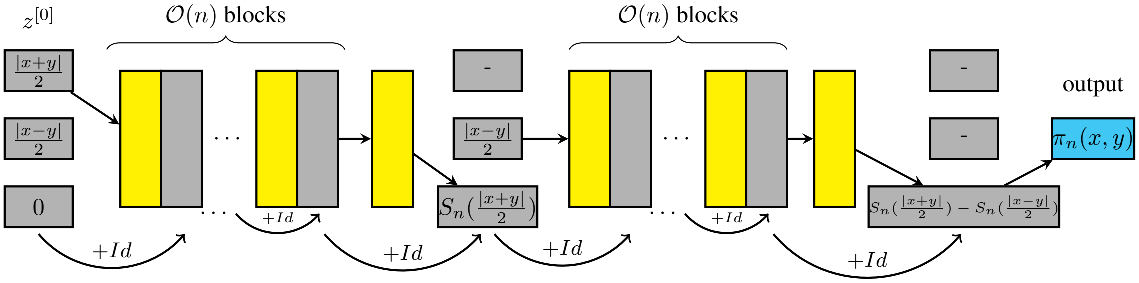


Figure 7. Illustration for generating $\pi_n(x, y)$ by the constructive ResNet. Grey represents the linear layer and yellow represents the activation layer.

Moreover, we can approximate the multiple product function $x_1x_2\cdots x_d$ where $x_1, x_2, \cdots, x_d \in [0, 1]$. We can well-define by (10) that

$$\pi_n^m\left(x_1, x_2, \cdots, x_m\right) = \pi_n\left(\pi_n^{m-1}\left(x_1, x_2, \cdots, x_{m-1}\right), x_m\right), m = 3, 4, \cdots \quad \text{for} \quad x_1, \cdots, x_d \in [0, 1] \tag{15}$$

and $\pi_n^2(x_1, x_2) = \pi_n(x_1, x_2)$. Then we have

Proposition 12.

$$|\pi_n^m\left(x_1, x_2, \cdots, x_m\right) - x_1x_2\cdots x_m| \le em4^{-n}, \quad x_1, x_2, \cdots, x_m \in [0, 1] \tag{16}$$

as long as $n \ge 1 + \log_2 m$.

Proof. First, It follows from the definition of S_n and $S(x) = x^2$ that

$$S(x) - S_n(x) = -\sum_{m=n+1}^{\infty} 4^{-m}h_m(x).$$

Note h_m has the Lipschitz norm 2^m. We have

$$\begin{aligned} \|S' - S_n'\|_{L_\infty([0,1])} &\le |\sum_{m=n+1}^{\infty} 4^{-m}| \cdot \|h_m'(x)\|_{L_\infty([0,1])} \\ &\le |\sum_{m=n+1}^{\infty} 4^{-m}2^m| \\ &\le 2^{-n}, \quad n \ge 1. \end{aligned}$$

$S'(x) = 2x$ so we have

$$S_n'(x) = 2x + \delta \quad \text{where} \quad \delta \le 2^{-n}. \tag{17}$$

Define $\pi(x, y) := xy$. We have

$$\begin{aligned} \partial_1\pi_n\left(x, y\right) &= S_n'\left(\frac{x+y}{2}\right) - \frac{1}{2}\cdot\mathbb{1}\{x \ge y\}S_n'\left(\frac{x-y}{2}\right) + \frac{1}{2}\cdot\mathbb{1}\{x < y\}S_n'\left(\frac{y-x}{2}\right) \\ &= \frac{1}{2}\left(x + y + \delta\right) - \frac{1}{2}\cdot\mathbb{1}\{x \ge y\}\left(x - y + \delta_1\right) + \frac{1}{2}\cdot\mathbb{1}\{x < y\}\left(y - x + \delta_2\right) \\ &= y + \delta - \mathbb{1}\{x \ge y\}\delta_1 + \mathbb{1}\{x < y\}\delta_2 \\ &= \partial_1\pi\left(x, y\right) + \delta - \mathbb{1}\{x \ge y\}\delta_1 + \mathbb{1}\{x < y\}\delta_2, \end{aligned} \tag{18}$$

where $\delta, \delta_1, \delta_2 \leq 2^{-n}$. Moreover, it is the same when considering about $\partial_2 \pi_n(x,y)$. Thus, we have

$$\|\partial_i \pi - \partial_i \pi_n\|_{L_\infty([0,1]^2)} \leq 2^{-n+1}, \quad \text{where } \partial_i := \partial_{x_i}, i = 1, 2. \tag{19}$$

Define $\pi^m(x_1, x_2, \cdots, x_m) = x_1 x_2 \cdots x_m$. Now we assume

$$\|\pi^j - \pi_n^j\|_{C([0,1]^j)} \leq c_j 4^{-n} \tag{20}$$

holds for all $j \leq m-1$. Note, the inequality literally holds for $j = 2$ by letting $c_2 = 1$. Then we have

$$\begin{aligned}
\|\pi^m - \pi_n^m\|_{C([0,1]^m)} &\leq \|\pi\left(x_m, \pi^{m-1}\right) - \pi_n\left(x_m, \pi^{m-1}\right)\|_{C([0,1]^m)} + \|\pi_n\left(x_m, \pi^{m-1}\right) - \pi_n\left(x_m, \pi_n^{m-1}\right)\|_{C([0,1]^m)} \\
&\leq 4^{-n} + \|\partial_2 \pi_n\|_{L_\infty([0,1]^2)} \left\|\pi^{m-1} - \pi_n^{m-1}\right\|_{C([0,1]^{m-1})} \\
&\leq 4^{-n} + \left(1 + 2^{-n+1} c_{m-1}\right) 4^{-n} \\
&= (1 + \alpha_n c_{m-1}) \cdot 4^{-n}.
\end{aligned}$$

By induction, we have proved

$$\|\pi^m - \pi_n^m\|_{C([0,1]^m)} \leq c_m 4^{-n}$$

where c_m satisfies the recurrence formula $c_m = 1 + \alpha_n c_{m-1}, m \geq 3$, with initial value $C_2 = 1$. The solution is

$$c_m = \sum_{j=0}^{m-2} \alpha_n^j \leq (m-1)\alpha_n^{m-2}.$$

If $m \leq 2^{n-1}$, we have

$$c_m \leq (m-1)\left(1 + \frac{1}{2^{n-1}}\right)^{m-2} < m\left(1 + \frac{1}{m}\right)^m < em.$$

Then we complete the proof and get

$$\|\pi^m - \pi_n^m\|_{C([0,1]^m)} \leq em4^{-n}$$

for $m = 3, 4, 5, \cdots$, as long as $n \geq 1 + \log_2 m$. □

C.2. Proof of Proposition 3 over $[0,1]^d$

Now, we are ready to prove Proposition 3 over $[0,1]^d$. For the completeness, we show the following theorem.

Proposition 13 (Proposition 3)**.** *Let $x = (x_1, x_2, \cdots, x_d) \in [0,1]^d, \alpha \in \mathbb{N}^d$ and x^α be any given monomial with degree p. Then*

(1) there is a ResNet $R_1 \in \mathcal{RN}\left(d+1, 4, \mathcal{O}(d\log(d/\varepsilon))\right)$ such that

$$\|R_1 - x_1 x_2 \cdots x_d\|_{C([0,1]^d)} < \varepsilon$$

while having $\mathcal{O}(d\log(d/\varepsilon))$ tunable weights. Moreover, there is a ResNet belonging to $R_1 \in \mathcal{RN}\left(d+1, 4, \mathcal{O}(nd)\right)$ can generate $\pi_n^d(x_1, x_2, \cdots, x_3)$ exactly.

(2) there is a ResNet $R_2 \in \mathcal{RN}\left(d+3, 4, \mathcal{O}(p\log(p/\varepsilon))\right)$ such that

$$\|R_2 - x^\alpha\|_{C([0,1]^d)} < \varepsilon$$

while having $\mathcal{O}\left(p\log\left(p/\varepsilon\right)\right)$ tunable weights.

Proof. We prove (1) first. Let the right-hand side of inequality 16 equal to ε and note $m = d$ under the condition of (1). Then, n is the order $\log d/\varepsilon$. Now we construct a ResNet required with width 4 and depth $\mathcal{O}(nd)$ generating $\pi_n^d\left(x_1, x_2, \cdots, x_d\right)$.

Let $\mathcal{A}_{d+1} = (x_1, x_2, \cdots, x_d, 0)$. It follows from the proof of proposition 11 that we can assign some weights for the first $2n$ blocks to output

$$z^{[2n]} = (-, -, x_3, x_4, \cdots, x_d, \pi_n(x_1, x_2))$$

while only changing the value of the first, second, and last neurons in each activation layer. In the next block, we set the value of the first and second neurons to zero by using identity mapping. In the next block, we then can output

$$z^{[2n+1]} = (0, 0, x_3, x_4, \cdots, x_d, \pi_n(x_1, x_2)).$$

The zero-value neuron in the activation layer is ready to store the results in the next phase. Then in the next $2n$ blocks we can compute $\pi_n^3(x_1, x_2, x_3)$. Concretely, by the proof of proposition 11, we can have

$$z^{[4n+1]} = (0, \pi_n^3(x_1, x_2, x_3), -, x_4, \cdots, x_d, -).$$

By repeatedly doing the operation above, we can use about $(2n+1)(d-1)$ blocks totally with width 4 to approximate $\pi_n^d(x_1, x_2, \cdots, x_d)$. Moreover, there are about $4d+4$ weights in each building block. However, from the operation above, we can see only a constant number of weights are non-zero. Therefore, this network has at most cdn tunable weights where c is an absolute constant.

For (2), if $p \leq d$, the case can be the same with (1). Let's assume $p > d$. We just need to note $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d}$ can be approximated by

$$\pi_n^d(\pi_n^{\alpha_1}(x_1, \cdots, x_1), \cdots, \pi_n^{\alpha_d}(x_d, \cdots, x_d)) = \pi_n^p(\underbrace{x_1, \cdots, x_1}_{\alpha_1 \text{ times}}, \underbrace{x_2, \cdots, x_2}_{\alpha_2 \text{ times}}, \cdots, \underbrace{\alpha_d, \cdots, \alpha_d}_{\alpha_d \text{ times}}). \tag{21}$$

Thus, we must store the value of $x_1, \cdots, x_d$ in each building block. However, in the proof of (1), if we complete the output of $\pi_n(x_1, x_2)$, we will lose the value of x_1 and x_2 in the neurons. That's why we need $d+3$ neurons in the linear layer in this case. The two more neurons in the linear layer can help us preserve $x_1, \cdots, x_d$ in each linear layer. Here we briefly show a constructive ResNet generating 21 exactly.

Let $\mathcal{A}_{d+3} = (x, x_1, x_1, 0)$. By the proof of proposition 11, we can compute $\pi_n(x_1, x_1)$ using $2n$ blocks and get $z^{[2n]} = (x, -, -, \pi_n(x_1, x_1))$ in the $2n$-th block. In the next block, we compute $z^{[2n+1]} = (x, 0, x_1, \pi_n(x_1, x_1))$. Then in the next $2n$ block, we compute $\pi_n^3(x_1, x_1, x_1) = \pi_n^2(x_1, \pi_n(x_1, x_1))$ and hence get $z^{[4n+1]} = (x, \pi_n^3(x_1, x_1, x_1), -, -)$. Then by doing it repeatedly, we can use $\mathcal{O}(pn)$ blocks to generate (21). Similarly by the inequality (16) where $m = p$, we can get the depth is $\mathcal{O}(p \log \frac{p}{\varepsilon})$ if the desired accuracy is ε. Moreover, note there is only a constant number of weights being non-zero in each block. Thus, the total number of the non-zero weights (tunable weights) is $\mathcal{O}(p \log \frac{p}{\varepsilon})$. We can see an illustration in Figure 8.

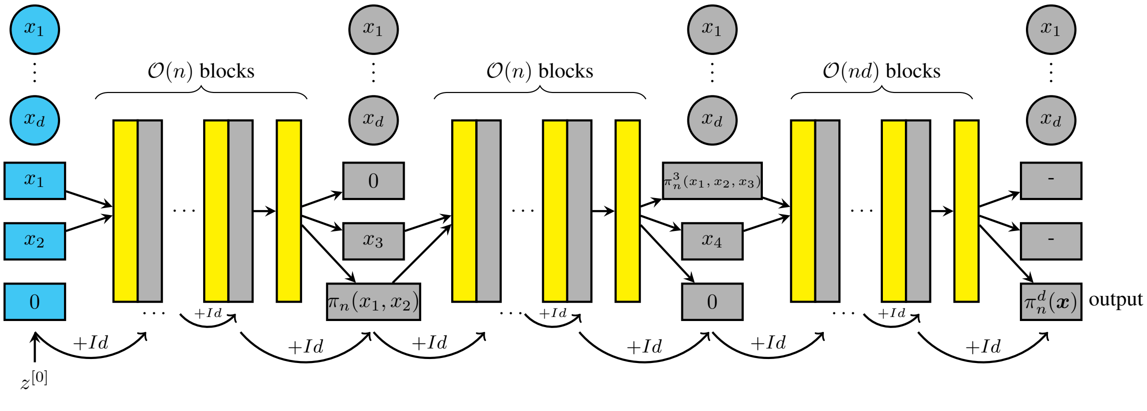


Figure 8. Illustration for generating $\pi_n^d(x_1, \cdots, x_d)$ by the constructive ResNet. The topmost d neurons are used for storing values. Grey represents the linear layer and yellow represents the activation layer.

□

Following this theorem, we supplement some discussions about **why are monomials important.** Monomials are the essential constituents of polynomials, which serve an integral role in both theory and applications. Additionally, monomials possess a straightforward mathematical structure, which aids in analyzing and comparing the approximation capabilities of neural networks. Numerous intriguing studies have been conducted in this field. Both deep (Yarotsky, 2017) and shallow

(Blanchard & Bennouna, 2021) ReLU networks can efficiently approximate monomials over $[0,1]^d$ with $poly(d)$ neurons. However, for monomials over $[0,M]^d$ ($M>1$), shallow ResLU networks will at least cost $exp(d)$ neurons approximating it to ε (Shapira, 2023). Further, the cost of shallow networks will be reduced to $poly(d)$ if the monomial is normalized over $[0,M]^d$ (multiplied by some normalization constant M^{-p}). More comprehensive discussion can be found in (Shapira, 2023).

C.3. Proof of Proposition 3

In this subsection, we extend Thm, 13 to $[-M,M]^d$ where $M>1$. First, we extend the lemma 9 to the $[-1,1]$.

Lemma 14. *Let $x, y \in [-1,1]$. Then $\pi_n(x,y) \in [-1,1]$*

Proof. Since $x, y \in [-1,1]$, it suffices to show

$$|\pi_n(x,y)| = \left| S_n(\frac{|x+y|}{2}) - S_n(\frac{|x-y|}{2}) \right| \leq 1. \tag{22}$$

First, we show that $S_n(x)$ is monotone incresing over $[0,1]$. For any $0 \leq x \leq y \leq 1$,

$$S_n(x) - S_n(y) = x - y + \sum_{i=1}^{n} 4^{-i} \left(h_i(y) - h_i(x) \right).$$

Then by

$$h_i(x) = h_i(y + x - y) = \zeta(2^{i-1}(y + x - y)) \leq \zeta(2^{i-1}y) + \zeta(2^{i-1}(x-y))) = h_i(y) + h_i(x-y)$$

we have

$$S_n(x) - S_n(y) = x - y + \sum_{i=1}^{n} 4^{-i} \left(h_i(y) - h_i(x) \right) \geq x - y - \sum_{i=1}^{n} 4^{-i} h_i(x-y) \geq (x-y)^2 \geq 0.$$

Note $|x+y| \geq |x-y|$ is equivalent to $xy \geq 0$. So we only care about the following case to show 22:

- $x, y \leq 0$.

$$\begin{aligned} \pi_n(x,y) &= S_n(\frac{|x+y|}{2}) - S_n(\frac{|x-y|}{2}) \\ &\leq \frac{|x|+|y|}{2} - \frac{(x-y)^2}{4} \\ &= \frac{1}{4}\left(|x|(2-|x|) + |y|(2-|y|) + 2xy \right) \\ &\leq \frac{1}{4}(1+1+2) = 1. \end{aligned}$$

- $xy \leq 0$.

$$\begin{aligned} \pi_n(x,y) &= S_n(\frac{|x-y|}{2}) - S_n(\frac{|x+y|}{2}) \\ &\leq \frac{|x|+|y|}{2} - \frac{(x+y)^2}{4} \\ &= \frac{1}{4}\left(|x|(2-|x|) + |y|(2-|y|) - 2xy \right) \\ &\leq \frac{1}{4}(1+1+2) = 1. \end{aligned}$$

□

Thus, π_n^m can be well-defined over $[-1,1]$ (equation 15). Next, we show that Proposition 12 holds for $x_1, x_2, \cdots, x_d \in [-1,1]$, i.e.,

Proposition 15.

$$|\pi_n^m(x_1, x_2, \cdots, x_m) - x_1 x_2 \cdots x_m| \le em4^{-n}, \quad x_1, x_2, \cdots, x_m \in [-1,1] \tag{23}$$

as long as $n \ge 1 + \log_2 m$.

Proof. For $x, y \in [-1,1]$, the only change of the proof is equation 18. We note

$$S_n'(\frac{|x+y|}{2}) = \mathbb{1}\{x+y \ge 0\}\frac{1}{2}(x+y+\varepsilon_1) - \mathbb{1}\{x+y<0\}\frac{1}{2}(-x-y+\varepsilon_2) = \frac{1}{2}(x+y+\varepsilon)$$

where $\varepsilon_1, \varepsilon_2, \varepsilon \le 2^{-n}$. Then we can still get

$$\|\partial_i \pi - \partial_i \pi_n\|_{L_\infty([-1,1]^2)} \le 2^{-n+1}, \quad \text{where } \partial_i := \partial_{x_i}, i = 1, 2. \tag{24}$$

Then we can show the result following the proof of C.1. □

Then Thm. 3 can be easily showed by the following remark.

Remark 16 (Proposition 3)**.** *Let $x_1, x_2, \cdots, x_m \in [-M, M]$. To approximate $x_1 x_2 \cdots x_m$, we consider a function defined by $\widehat{\pi}_n^m(x_1, \ldots, x_m) := M^m \pi_n^m(|x_1|/M, \ldots, |x_m|/M)$ with the approximation accuracy*

$$|x_1 x_2 \cdots x_m - \widehat{\pi}_n^m(x_1, \cdots, x_m)| \le emM^m \cdot 4^{-n}, \quad \forall x_1, x_2, \cdots, x_m \in [-M, M] \tag{25}$$

as long as $n \ge 1 + \log_2 m$. Moreover $\widehat{\pi}_n^m$ can be generated by a ResNet $R(x) \in \mathcal{RN}(m+1, 4, \mathcal{O}(mn))$ while having at most $\mathcal{O}(mn)$ tunable weights.

Proof. The remark is the direct corollary from the proof of Proposition 13. By letting the right hand side of Equation 25 equal to ε where $m = p$ and p is the degree of the monomial, we complete the proof of Proposition 3. □

C.4. Proof of Theorem 4

Proof. By Proposition 3, for each $x^\alpha : \alpha \in E$, we can use a ResNet with width 4, depth $\mathcal{O}(p \log p/\varepsilon)$ and $d+3$ neurons in each linear layer to output $R_\alpha(x)$ such that

$$|R_\alpha(x) - x^\alpha| < \varepsilon, \quad x \in [0,1]^d.$$

Thus,

$$|\sum_{\alpha \in E} c_\alpha R_\alpha(x) - \sum_{\alpha \in E} c_\alpha x^\alpha| < \varepsilon \cdot \sum_{|\alpha| \in E} |c_\alpha|, \quad x \in [0,1]^d.$$

Then Let $\mathcal{A}_{d+4} = (x, x_1, x_1, 0, 0)$. To generate each $R_\alpha(x)$, we need 4 more computational units in each linear layer and depth $\mathcal{O}(p \log p/\varepsilon)$ while having $\mathcal{O}(p \log p/\varepsilon)$ non-zero weights. Then store the value of $R_\alpha(x)$ in the last neuron in each linear layer. Then we can output $(x, -, -, -, \sum_{\alpha \in E} c_\alpha R_\alpha(x))$ finally with depth $\mathcal{O}(p|E|\log(p/\varepsilon))$ while having $\mathcal{O}(p|E|\log(p/\varepsilon))$ non-zero weights totally. □

D. Proof of Theorem 5

In this section, we prove Theorem 5. Before that, we will give the definition supplement about Sobolev space in subsection 4.2.

D.1. Definition supplement of Sobolev spaces

For $\alpha = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}^d$ and $x = (x_1, \ldots, x_d) \in [0,1]^d$, define

$$D^\alpha f = \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}}$$

where $|\alpha| = \alpha_1 + \cdots + \alpha_d$. Let $r \in \mathbb{N}_+$. The Sobolev space $W^{r,\infty}([0,1]^d)$ is the set of functions belonging to $C^{r-1}([0,1]^d)$ whose $(r-1)$-th order derivatives are Lipschitz continuous with the norm

$$\|f\|_{W_\infty^r} := \max_{\alpha:|\alpha| \le r} \operatorname{esssup}_{x \in [0,1]^d} |D^\alpha f(x)| < \infty.$$

We denote by $U^r([0,1]^d)$ the unit ball of $W^{r,\infty}([0,1]^d)$, i.e. $U^r([0,1]^d) = \{f \in W^{r,\infty}([0,1]^d) : \|f\|_{W_\infty^r} \le 1\}$. Note

$$\operatorname{ess\,sup} f = \inf\{a \in \mathbb{R} : \mu(\{x : f(x) > a\}) = 0\}$$

where μ is Lebesgue measure.

D.2. Proof of Theorem 5

We follow the proof of theorem 1 in (Yarotsky, 2017). In our proof, we skip some details and focus on the constructions of ResNet. The details can be found in theorem 1 of (Yarotsky, 2017). Now let $f \in U^r([0,1]^d)$ and $\boldsymbol{\alpha} \in \mathbb{N}^d$.

Let N be a positive integer to be determined and $\mathbf{m} = (m_1, \ldots, m_d) \in \{0, 1, \ldots, N\}^d$. The function $\phi_{\mathbf{m}}$ is defined as the product

$$\phi_{\mathbf{m}}(\mathbf{x}) = \prod_{k=1}^{d} \psi\left(3N\left(x_k - \frac{m_k}{N}\right)\right)$$

where

$$\psi(x) = \begin{cases} 1, & |x| < 1, \\ 0, & 2 < |x|, \\ 2 - |x|, & 1 \le |x| \le 2. \end{cases}$$

Let

$$f_1(\mathbf{x}) = \sum_{\mathbf{m} \in \{0,\ldots,N\}^d} \sum_{\boldsymbol{\alpha}:|\boldsymbol{\alpha}|<r} a_{\mathbf{m},\boldsymbol{\alpha}} \phi_{\mathbf{m}}(\mathbf{x}) \left(\mathbf{x} - \frac{\mathbf{m}}{N}\right)^{\boldsymbol{\alpha}}$$

where $a_{\mathbf{m},\boldsymbol{\alpha}}$ are some specific coefficients when considering the locally Taylor expansion of f. Then by choosing

$$N = \left\lceil \left(\frac{r!}{2^d d^r} \frac{\varepsilon}{2}\right)^{-1/r} \right\rceil \tag{26}$$

where $\lceil \cdot \rceil$ is the celling function, we have

$$\|f - f_1\|_\infty \le \frac{\varepsilon}{2}.$$

Now, we consider to approximate $\phi_{\mathbf{m}}(\mathbf{x}) \left(\mathbf{x} - \frac{\mathbf{m}}{N}\right)^{\boldsymbol{\alpha}}$ by ResNet.

The following lemma follows directly from remark 16 that

Lemma 17. *Let $\mathbf{x} \in [0,1]^d$ and $g_1(x_1), \cdots, g_2(x_d) \in [-1,1]$. Then*

$$\left|\pi_n^d\left(g_1(x_1), g_2(x_2), \cdots, g_d(x_d)\right) - g_1(x_1)g_2(x_2)\cdots g_d(x_d)\right| \le ed4^{-n}, \quad n \ge 1 + \log_2 d$$

for all $x_1, x_2, \cdots, x_d \in [-1,1]$.

Moreover, we need the following lemma for the construction.

Lemma 18. *There is a ResNet $R(x) \in \mathcal{RN}\,(Q = 1, N = 3, L = 4)$ such that $R(x) = \psi(x)$ for any $x \in \mathbb{R}$.*

Proof of lemma 18. (Lin & Jegelka, 2018) has shown that the following operations are realizable by a single basic residual block of ResNet with one neuron: (a) Shifting by a constant: $R^+ = R + c$ for any $c \in \mathbb{R}$. (b) Min or Max with a constant: $R^+ = \min\{R, c\}$ or $R^+ = \max\{R, c\}$ for any $c \in \mathbb{R}$. (c) Min or Max with a linear transformation: $R^+ = \min\{R, \alpha R + \beta\}$ (or max) for any $\alpha, \beta \in \mathbb{R}$.

For the ResNet, $z^{[0]} = x$. In the first layer, we use one computational unit to compute $\mathrm{ReLU}(x+2)$ and two units to compute $x = (x)_+ - (-x)_+$. Then we output $z^{[1]} = \mathrm{ReLU}((x+2) - x + x = \mathrm{ReLU}(x+2)$ in the first layer. In the remaining layers, we only need one neuron per layer. we can output $z^{[2]} = z^{[1]} - 2(z^{[1]} - 2)$, $z^{[3]} = \min\{z^{[2]}, 1\}$ and $z^{[4]} = \max\{z^{[3]}, 0\} = \psi(x)$. □

Then now we define

$$R_{\mathbf{m},\boldsymbol{\alpha}}(x) = \pi_n^{2d}\left(\psi(3Nx_1 - 3m_1), \cdots, \psi(3Nx_d - 3m_d), \pi_n^{\alpha_1}(x_1 - \frac{m_1}{N}), \cdots, \pi_n^{\alpha_d}(x_d - \frac{m_d}{N})\right)$$

$$= \pi_n^{d+|\boldsymbol{\alpha}|}\left(\psi(3Nx_1 - 3m_1), \cdots, \psi(3Nx_d - 3m_d), \underbrace{x_1 - \frac{m_1}{N}, \cdots, x_1 - \frac{m_1}{N}}_{\alpha_1 \text{times}}, \cdots, \underbrace{x_d - \frac{m_d}{N}, \cdots, x_d - \frac{m_d}{N}}_{\alpha_d \text{times}}\right).$$

By lemma 17 and note

$$|\psi(3Nx_i - 3m_i)| \le 1 \quad \text{and} \quad |x_i - \frac{m_i}{N}| \le 1 \quad \text{for} \quad i = 1, 2, \cdots, d$$

, we have

$$|R_{\mathbf{m},\boldsymbol{\alpha}}(\mathbf{x}) - \phi_{\mathbf{m}}(\mathbf{x})\left(\mathbf{x} - \frac{\mathbf{m}}{N}\right)^{\boldsymbol{\alpha}}| < e(r+d)4^{-n} = \varepsilon_0, \quad \mathbf{x} \in [0,1]^d, \tag{27}$$

by letting $n = \mathcal{O}(\log \frac{e(r+d)}{\varepsilon_0})$. Then with similar proof of Proposition 3 and lemma 18, we have a ResNet with $Q = d + 3$ to generate $R_{\mathbf{m},\boldsymbol{\alpha}}(\mathbf{x})$ such (27) satisfies while having $\mathcal{O}(d(r+d)\log e(r+d)/\varepsilon_0)$ weights. It follows from the proof of Theorem 4, we have a ResNet with $Q = d + 4$ can generate

$$\widetilde{f}(\mathbf{x}) = \sum_{\mathbf{m} \in \{0,\ldots,N\}^d} \sum_{\boldsymbol{\alpha}: |\boldsymbol{\alpha}| < r} a_{\mathbf{m},\boldsymbol{\alpha}} R_{\mathbf{m},\boldsymbol{\alpha}}(\mathbf{x})$$

while having

$$\mathcal{O}\left((N+1)^d d^r d(d+r)\log\frac{e(r+d)}{\varepsilon_0}\right) \tag{28}$$

weights. From the proof of theorem 1 in (Yarotsky, 2017) we then have

$$|\widetilde{f}(\mathbf{x}) - f_1(\mathbf{x})| \le 2^d d^r \varepsilon_0.$$

Let $\varepsilon_0 = \varepsilon/(2^{d+1}d^r)$. We have $\|\widetilde{f} - f_1\|_\infty \le \varepsilon/2$. Thus,

$$\|f - \widetilde{f}\|_\infty \le \|f - f_1\|_\infty + \|f_1 - \widetilde{f}\|_\infty \le \varepsilon.$$

Now, substitute $\varepsilon_0 = \varepsilon/(2^{d+1}d^r)$ and N with equation 26 into (28), the upper bound on the total weights of the ResNet are

$$\mathcal{O}\left(\left(\frac{r!}{2^d d^r}\frac{\varepsilon}{2}\right)^{-d/r} d^{r+1}(d+r)\log\frac{e(r+d)2^d d^r}{\varepsilon}\right) = \mathcal{O}_{d,r}\left(\varepsilon^{-\frac{d}{r}}\log\frac{1}{\varepsilon}\right).$$

By the well-known Stirling's approximation

$$\sqrt{2\pi r}\left(\frac{r}{e}\right)^r e^{\frac{1}{12r+1}} < r! < \sqrt{2\pi r}\left(\frac{r}{e}\right)^r e^{\frac{1}{12r}}, \tag{29}$$

the hidden constant $c(d,r)$ in the $\mathcal{O}_{d,r}$ notation can be bounded by

$$(\frac{2^{\frac{d+1}{r}}d}{r})^d < c(d,r) < (\frac{2^{\frac{d+1}{r}}d}{r})^d d^{r+2}(d+r)r \tag{30}$$

where C is an absolute constant.

E. Proof of Theorem 6

In this appendix, we give the proof of Theorem 6. The proof of Theorem 6 is mainly based on the following lemma.

Lemma 19. *Fix the integer $d \geq 1$ and let $f : \mathbb{R}^d \to \mathbb{R}$ be a CPwL function. Then there exist affine functions $p_\alpha, q_\beta : \mathbb{R}^d \to \mathbb{R}$ such that f can be written as the difference of positive convex functions:*

$$f = p - q, \quad \textit{where} \quad p := \max_{1\leq \alpha \leq A} p_\alpha, \quad q := \max_{1\leq \beta\leq B} q_\beta$$

where A, B are some positive numbers.

Proof. For any CPwL function $f : \mathbb{R}^d \to \mathbb{R}$, by theorem 1 in (Wang & Sun, 2005), there exists a finite set of affine linear functions $\ell_1, \ldots, \ell_k$ and a finite integer M such that

$$f = \sum_{j=1}^{M} \sigma_j \left(\max_{i\in S_j} \ell_i \right)$$

where $S_j \subseteq \{1, \ldots, k\}$, $|S_j| \leq d + 1$ and $\sigma_i \in \{+1, -1\}$ for all $i = 1, 2, \cdots, M$. We write

$$f = \sum_{j=1}^{M} \sigma_j \left(\max_{i\in S_j} \ell_i \right) = \sum_{j:\sigma_j=1} \max_{i\in S_j} \ell_i - \sum_{j:\sigma_j=-1} \max_{i\in S_j} \ell_i = p - q.$$

The last equation holds by the fact that the sum of convex functions is convex and the sum of CPwL functions is also CPwL. We can easily see A, B is bounded by Md. However, M is an implicit number that may depend on the property of the CPwL function (e.g., the number of pieces, and the number of linear components). More details can be found in the proof details in (Tarela & Martinez, 1999; Wang & Sun, 2005). □

Now we are ready for the proof of Theorem 6.

Proof of Theorem 6. The proof is based on the observation

$$\max\{x, y\} = y + \mathrm{ReLU}(x - y)$$

Now we construct a single-neuron per hidden layer ResNet with $k = d + 1$ to output f exactly. We use the same notation as lemma 19. Let $z^{[0]} = \mathcal{A}_{d+1}(x) = (x, p_1(x))$. In the next block, we compute $\zeta^{[1]} = \mathcal{T}_1^{[1]}(z^{[0]}) = \mathrm{ReLU}(p_2(x) - p_1(x))$ in the activation layer and $\gamma^{[1]}\mathcal{T}_2^{[1]}(z^{[0]}) = (0, \mathrm{ReLU}(p_2(x) - p_1(x)))$ in the linear layer by choosing the appropriate weights. Then we can output

$$z^{[1]} = \gamma^{[1]} + z^{[0]} = (0, \mathrm{ReLU}(p_2(x) - p_1(x)) + z^{[0]} = (x, \max\{p_1, p_2\}).$$

By repeatedly doing this, we can output $z^{[A]} = (x, \max_{1\leq\alpha\leq A}\{p_\alpha\})$ in the A-th block. In the next two block, we output $z^{[A+2]} = (x, p(x) - q_1(x))$. Here we use two single-neuron blocks to compute $q_1(x) = (q_1)_+ - (-q_1)_+$. Then by the same operation, we can get the result $z^{[A+B+2]} = (x, p - q)$ in the $(A + B + 2)$-th block. Then the ResNet outputs $f = p - q$ exactly.

□

F. Proof of Theorem 7

We first introduce some terms we will use. For a univariate continuous piecewise linear function $f(x)$, x_0 is called a breakpoint of f if $\lim_{x\to x_0^+} f'(x) \neq \lim_{x\to x_0^-} f'(x)$. We will abbreviate 'continuous piecewise linear' as 'CPwL'. For a finite sample set $A = \{(x_i, y_i) : 1 \leq i \leq m\}$ and let x_i be increasing for i, we have a CPwL function f such that: i) $f(x_i) = y_i$ and ii) f is linear in each interval $(-\infty, x_1], [x_1, x_2], \cdots, [x_{m-1}, x_m], [x_m, \infty]$. In this case, we say f is a CPwL function defined by the sample set A. Note that the set of the breakpoints of f is a subset of $\{x_1, \cdots, x_m\}$.

Next, we introduce some basic lemma we will use for the construction of ResNet later.

Definition 20. *A function $g : \mathbb{R}^d \to \mathbb{R}^v$ is a max-min string of length $L \geq 1$ on d input variables and v output variables if there exist affine functions $\ell_1, \dots, \ell_L : \mathbb{R}^d \to \mathbb{R}^v$ such that*

$$g = \tau_{L-1}\left(\ell_L, \tau_{L-2}\left(\ell_{L-1}, \dots, \tau_2\left(\ell_3, \tau_1\left(\ell_2, \ell_1\right)\right) \cdots\right)\right),$$

where each τ_i is either a coordinate-wise max or a min.

Lemma 21. *Suppose $g : \mathbb{R}^d \to \mathbb{R}$ is a max-min string of length L, i.e.*

$$g = \tau_{L-1}\left(\ell_L, \tau_{L-2}\left(\ell_{L-1}, \dots, \tau_2\left(\ell_3, \tau_1\left(\ell_1, \ell_2\right)\right) \cdots\right)\right)$$

Then there is a ResNet $R \in \mathcal{RN}(d+1, 1, L)$ such that $R(x) = (x, g(x))$.

Proof. The proof is based on the observation

$$\max\{x, y\} = y + \mathrm{ReLU}(x - y) \quad \text{and} \quad \min\{x, y\} = y - \mathrm{ReLU}(y - x)$$

We may consider the case $v = 1$. Now we construct a single-neuron per hidden layer ResNet with $k = d + 1$ to output f exactly. Let $z^{[0]} = \mathcal{A}_{d+1}(x) = (x, \ell_1(x))$. Without loss of generality, we assume $\tau_1 = \max$. In the next block, we compute $\zeta^{[1]} = \mathcal{T}_1^{[1]}(z^{[0]}) = \mathrm{ReLU}(\ell_2(x) - \ell_1(x))$ in the activation layer and $\gamma^{[1]}\mathcal{T}_2^{[1]}(z^{[0]}) = (0, \mathrm{ReLU}(p_2(x) - p_1(x)))$ in the linear layer by choosing the appropriate weights. Then we can output

$$z^{[1]} = \gamma^{[1]} + z^{[0]} = (0, \mathrm{ReLU}(\ell_2(x) - \ell_1(x)) + z^{[0]} = (x, \max\{p_1, p_2\}) = (x, \tau_1(\ell_1, \ell_2)).$$

If $\tau_1 = \min$, we have a similar process. Now we can assume the output of the $k-1$-th block is

$$z^{[k-1]} = \left(x, \tau_{k-1}\left(\ell_k, \tau_{k-2}\left(\ell_{k-1}, \cdots, \tau_1\left(\ell_2, \ell_1\right)\right)\right)\right) := (x, g_{k-1}).$$

Without loss of generality, we assume $\tau_k = \min$. Then we can compute in the next residual block that

$$\zeta^{[k]} = \mathrm{ReLU}(g_{k-1} - \ell_{k+1})$$

and

$$\gamma^{[k]} = \left(0, -\mathrm{ReLU}(g_{k-1} - \ell_{k+1}(x))\right).$$

Thus, the output of the next block is

$$z^{[k]} = z^{[k-1]} + \gamma^{[k]} = (x, g_{k-1}) + (0, -\mathrm{ReLU}(g_{k-1} - \ell_{k+1})) = \left(x, \tau_k(\ell_{k+1}, \tau_{k-1}\left(\ell_k, \cdots, \tau_1\left(\ell_2, \ell_1\right)\right)\right)\right).$$

By induction, we can output $z^{[L]} = (x, g)$ in the L-th block. Thus, we finish the proof. □

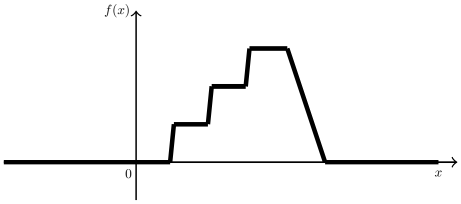
There is a special CPwL function which can be represented by max-min string which is shown in the following lemma.

Lemma 22. *Let L be an positive integer and ε be a sufficient small number. Assume a CPwL function is defined by the sample set*

$$\{(k, k), (k + 1 - \varepsilon, k) : k = 0, 1, \cdots, L - 1\} \bigcup \{(L, L)\} \bigcup \{(L + 1, 0)\} \bigcup \{(L + 2, 0)\}$$

with $2L$ breakpoints. Then there exists a ResNet $R \in \mathcal{RN}(d + 1, 1, 2L + 1)$ such that $R(x) = (x, f(x))$.

Proof. It is easy to show the CPwL function on the assumption can be exactly represented by a max-min string of length L by induction. We have an illustration for the kind of step function $f(x)$ as shown in the following when $L = 4$. Thus, the result follows from Lemma 21 immediately.



□

Lemma 23. *Let $S \subseteq \mathbb{R}^d$ be a finite set. Then any function $f : S \to \mathbb{R}$ can be computed exactly by a ResNet $R(x) \in \mathcal{RN}(d+1, 1, 2|S|)$.*

Proof. As shown in Proposition 4 (Hanin & Sellke, 2017), f can be exactly generated by a min-max string of length $2|S|$. Then with Lemma 21, we directly have the result. □

F.1. Proof of Theorem 7

We first given the definition of the small region Ω. Given $K \in N^+$and $\delta \in \left(0, \frac{1}{K}\right)$, define a trifling region $\Omega\left([0,1]^d, K, \delta\right)$ of $[0,1]^d$ as

$$\Omega\left([0,1]^d, K, \delta\right) := \bigcup_{i=1}^{d} \left\{ \mathbf{x} = (x_1, x_2, \cdots, x_d) \in [0,1]^d : x_i \in \cup_{k=1}^{K-1} \left(\frac{k}{K} - \delta, \frac{k}{K} \right) \right\}.$$

In particular, $\Omega\left([0,1]^d, K, \delta\right) = \varnothing$ if $K = 1$.

Thus, given the following theorem, we can prove Theorem 7. The proof of Theorem 24 can be found in Sec. F.2.

Theorem 24. *Let $d \in \mathbb{N}^+$ and $d \geq 5$. Given $f \in C\left([0,1]^d\right)$, for any $L \in \mathbb{N}^+$, there exists a ResNet*

$$R : \mathbb{R}^d \to \mathbb{R} \in \mathcal{RN}(d+1, 4, 24L + 9d + 4)$$

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

$$|f(\mathbf{x}) - R(\mathbf{x})| \leq 6\sqrt{d}\omega_f\left(L^{-2/d}\right), \quad \textit{for any } \mathbf{x} \in [0,1]^d \backslash \Omega\left([0,1]^d, K, \delta\right),$$

where $K = \lfloor L^{2/d} \rfloor$ and δ is an arbitrary number in $\left(0, \frac{1}{3K}\right]$.

Proof of Theorem 7. Without loss of generality, we assume f is not a constant function as it is a trivial case. Hence $\omega_f(r) > 0$ for any $r > 0$. Moreover, it is obvious that $|f(\mathbf{x}) - f(\mathbf{0}| < \omega_f(\sqrt{d}))$ hence $|f(\mathbf{x})| \leq |f(\mathbf{0})| + \omega_f(\sqrt{d})$ for $\mathbf{x} \in [0,1]^d$.

By Theorem 24, there exists a ResNet $R \in \mathcal{RN}(d+1, 4, 24L + 9d + 4)$ such that $\|R\|_{L^\infty(\mathbb{R}^d)} \leq |f(\mathbf{0})| + \omega_f(\sqrt{d})$ and

$$|f(\mathbf{x}) - R(\mathbf{x})| \leq 6\sqrt{d}\omega_f\left(L^{-2/d}\right), \quad \text{for any } \mathbf{x} \in [0,1]^d \backslash \Omega\left([0,1]^d, K, \delta\right),$$

Now, we set $K = \lfloor L^{2/d} \rfloor$ and choose a small $\delta \in \left(0, \frac{1}{3K}\right]$ such that for $x \in [0,1]^d$

$$\mu(\Omega) \cdot |f(\mathbf{x}) - R(\mathbf{x})|^p \leq Kd\delta\left(2|f(\mathbf{0})| + 2\omega_f(\sqrt{d})\right)^p = \lfloor L^{2/d} \rfloor d\delta\left(2|f(\mathbf{0})| + 2\omega_f(\sqrt{d})\right)^p \leq \left(\omega_f\left(L^{-2/d}\right)\right)^p.$$

It follows that

$$\begin{aligned}
\|f - R\|^p_{L^p([0,1]^d)} &= \int_{\Omega([0,1]^d, K, \delta)} |f(\mathbf{x}) - R(\mathbf{x})|^p \, \mathrm{d}\mathbf{x} + \int_{[0,1]^d \backslash \Omega([0,1]^d, K, \delta)} |f(\mathbf{x}) - R(\mathbf{x})|^p \, \mathrm{d}\mathbf{x} \\
&\leq Kd\delta\left(2|f(\mathbf{0})| + 2\omega_f(\sqrt{d})\right)^p + \left(6\sqrt{d}\omega_f\left(L^{-2/d}\right)\right)^p \\
&\leq \left(\omega_f\left(L^{-2/d}\right)\right)^p + \left(6\sqrt{d}\omega_f\left(L^{-2/d}\right)\right)^p \\
&\leq \left(7\sqrt{d}\omega_f\left(L^{-2/d}\right)\right)^p.
\end{aligned}$$

Hence, $\|f - R\|_{L^p([0,1]^d)} \leq 7\sqrt{d}\omega_f\left(L^{-2/d}\right)$.

□

F.2. Proof of Theorem 24

Given the following two propositions, we prove Theorem 24. The main steps are from (Shen et al., 2019; 2022b) and we mainly focus on the construction of ResNet. The proof of Proposition 25 can be found in Sec. F.3 and the Proof of Proposition 26 can be found in Sec. F.4.

Proposition 25. *For any $L, d \in \mathbb{N}^+$ and $\delta \in \left(0, \frac{1}{3K}\right]$ with $K = \lfloor L^{2/d} \rfloor$, there exists a ResNet $\widehat{R} : [0,1] \to \mathbb{R}^2, \widehat{R}(x) = (x, R(x))$ in $\mathcal{RN}(2, 1, 4L^{\frac{1}{d}} + 4)$ such that*

$$R(x) = k, \quad \textit{if } x \in \left[\frac{k}{K}, \frac{k+1}{K} - \delta \cdot \mathbb{1}_{\{k \le K-2\}}\right]$$

for $k = 0, 1, \cdots, K-1$.

Proposition 26. *Given any $\varepsilon > 0$ and arbitrary $L, J \in \mathbb{N}^+$ with $J \le L^2$ and samples $y_j \ge 0$ for $j = 0, 1, \cdots, J-1$ with*

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

then there exists a ResNet $R \in \mathcal{RN}(6, 4, 13L + 7)$ such that

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

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

Proof of Theorem 24. Without loss of generality, we assume f is not a constant. By the definition of $\omega_f(\cdot)$, we have $|f(\mathbf{x}) - f(\mathbf{0})| \le \omega_f(\sqrt{d})$ for any $\mathbf{x} \in [0,1]^d$. We can define $\widetilde{f} = f - f(\mathbf{0}) + \omega_f(\sqrt{d})$, then $0 \le \widetilde{f}(\mathbf{x}) \le 2\omega_f(\sqrt{d})$ for any $\mathbf{x} \in [0,1]^d$. Now set $K = \lfloor L^{2/d} \rfloor$, and δ to an arbitrary number in $\left(0, \frac{1}{3K}\right]$.

The details of the proof can be divided into the following four steps.

Step 1: Divide $[0,1]^d$ into $\{Q_{\boldsymbol{\beta}}\}_{\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d}$ and $\Omega\left([0,1]^d, K, \delta\right)$. Define $\mathbf{x}_{\boldsymbol{\beta}}$ is the vertex of $Q_{\boldsymbol{\beta}}$ with minimum $\|\cdot\|_1$ norm, i.e., $\mathbf{x}_{\boldsymbol{\beta}} := \boldsymbol{\beta}/K$ and

$$Q_{\boldsymbol{\beta}} := \left\{\mathbf{x} = (x_1, \cdots, x_d) \in [0,1]^d : x_i \in \left[\frac{\beta_i}{K}, \frac{\beta_i + 1}{K} - \delta \cdot 1_{\{\beta_i \le K-2\}}\right], i = 1, \cdots, d\right\}$$

for each d-dimensional index $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_d) \in \{0, 1, \cdots, K-1\}^d$. Clearly,

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

Step 2: Construct Φ mapping $\mathbf{x} \in Q_{\boldsymbol{\beta}}$ to $\boldsymbol{\beta}$. By Proposition 25, there exists a ResNet $\widehat{R}_1 = (x, R_1(x)) \in \mathcal{RN}(2, 1, 4L^{1/d} + 3)$ such that

$$R_1(x) = k, \quad \text{if } x \in \left[\frac{k}{K}, \frac{k+1}{K} - \delta \cdot \mathbb{1}_{\{k \le K-2\}}\right] \text{ for } k = 0, 1, \cdots, K-1.$$

It follows that $R_1(x_i) = \beta_i$ if $\mathbf{x} = (x_1, x_2, \cdots, x_d) \in Q_{\boldsymbol{\beta}}$ for each $\boldsymbol{\beta} = (\beta_1, \beta_2, \cdots, \beta_d)$. Let

$$\begin{aligned}
&\mathcal{L}_0(x_1, \cdots, x_d) = (x_1, x_2, \cdots, x_d, 0), \\
&\varphi_i(x_1, x_2, \cdots, x_d, 0) = (x_1, x_2, \cdots, x_d, R_1(x_i)), \quad \text{for } i = 1, 2, \cdots, d, \\
&\mathcal{L}_i(x_1, \cdots, x_d, y) = (x_1, \cdots, x_{i-1}, y, x_{i+1}, \cdots, x_d, 0), \quad \text{for } i = 1, 2, \cdots, d,
\end{aligned}$$

Then define $\Phi(x_1, \cdots, x_d) = \mathcal{L}_d \circ \varphi_d \circ \cdots \circ \mathcal{L}_1 \circ \varphi_1 \circ \mathcal{L}_0(x_1, \cdots, x_d)$. We have Φ is a ResNet in $\mathcal{RN}(d+1, 1, 4dL^{1/d} + 4d) \subset \mathcal{RN}(d+1, 1, 4L + 8d - 4)$ and

$$\Phi(\mathbf{x}) := \left(R_1(x_1), R_1(x_2), \cdots, R_1(x_d), 0\right), \quad \text{for any } \mathbf{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d$$

i.e., $\Phi(\mathbf{x}) = (\boldsymbol{\beta}, 0)$ if $\mathbf{x} \in Q_{\boldsymbol{\beta}}$ for $\boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d$. The computation process is illustrated by Fig. 9. Note that $4dL^{1/d} + 4d \le 4L + 8d - 4$ comes form the inequality $na^{1/n} \le a + n - 1$ for any non-negative real number a and positive integer n.

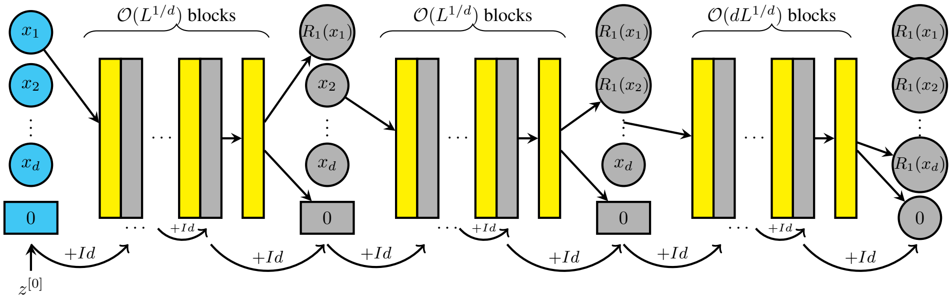

$\mathcal{O}(L^{1/d})$ blocks
$\mathcal{O}(L^{1/d})$ blocks
$\mathcal{O}(dL^{1/d})$ blocks
x_1
x_2
x_d
0
$R_1(x_1)$
$R_1(x_2)$
$R_1(x_d)$
$+Id$
$z^{[0]}$

Figure 9. Illustration of the computation of $\Phi(\mathbf{x})$ by ResNet. The last neuron in each layer is used for the intermediate computation and will be cleared to zero by a linear mapping in each stage.

Step 3: Construct a ResNet ϕ mapping $\boldsymbol{\beta}$ approximately to $\widetilde{f}\,(\mathbf{x}_{\boldsymbol{\beta}})$. The construction of the sub-network generating ϕ essentially relies on Proposition 26. We follow the work of (Shen et al., 2019; 2022b) in this step. To meet the requirements of applying Proposition 26, one can define two auxiliary sets $\mathcal{A}_1$ and $\mathcal{A}_2$ as

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

and

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

Clearly, $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \{1\} = \left\{\frac{j}{2K^d} : j = 0, 1, \cdots, 2K^d\right\}$ and $\mathcal{A}_1 \cap \mathcal{A}_2 = \varnothing$.

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 $\mathcal{A}_1$. Define

$$\psi_1(\mathbf{x}) := \frac{x_d}{2K^d} + \sum_{i=1}^{d-1} \frac{x_i}{K^i}, \qquad \text{for any } \mathbf{x} = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d.$$

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

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

Step 3.2: Construct g to satisfy $g \circ \psi_1(\boldsymbol{\beta}) = \widetilde{f}\,(\mathbf{x}_{\boldsymbol{\beta}})$ and to meet the requirements of Proposition 26. According to (Shen et al., 2019), we have a CPwL function $g : [0, 1] \to \mathbb{R}$ with a set of breakpoints $\left\{\frac{j}{2K^d} : j = 0, 1, \cdots, 2K^d\right\} = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \{1\}$ and the values of g at these breakpoints satisfy the following properties:

- The values of g at the breakpoints in $\mathcal{A}_1$ are set as

$$g\left(\psi_1(\boldsymbol{\beta})\right) = \widetilde{f}\,(\mathbf{x}_{\boldsymbol{\beta}}), \qquad \text{for any } \boldsymbol{\beta} \in \{0, 1, \cdots, K-1\}^d;$$

- At the breakpoint 1 , let $g(1) = \widetilde{f}(\mathbf{1})$, where $\mathbf{1} = (1, 1, \cdots, 1) \in \mathbb{R}^d$;
- 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 26. To achieve this, we can let g be linear on each interval $\left[\frac{i}{K^{d-1}} - \frac{K+1}{2K^d}, \frac{i}{K^{d-1}}\right]$ for $i = 1, 2, \cdots, K^{d-1}$.

Then g satisfies the condition of Proposition 26:

$$\left|g\left(\frac{j}{2K^d}\right)-g\left(\frac{j-1}{2K^d}\right)\right|\le\max\left\{\omega_f\left(\frac{1}{K}\right),\omega_f(\sqrt{d})/K\right\}\le\omega_f\left(\frac{\sqrt{d}}{K}\right),\qquad\text{for } j=1,2,\cdots,2K^d,$$

and

$$0\le g\left(\frac{j}{2K^d}\right)\le 2\omega_f(\sqrt{d}),\qquad\text{for } j=0,1,\cdots,2K^d$$

Step 3.3: Construct ψ_2 approximating g well on $\mathcal{A}_1\cup\mathcal{A}_2\cup\{1\}$. Since $2K^d=2\left(\lfloor L^{2/d}\rfloor\right)^d\le 2\,L^2\le\widetilde{L}^2$, where $\widetilde{L}=\frac{3}{2}L$, by Proposition 26 (set $y_j=g\left(\frac{j}{2K^2}\right)$ and $\varepsilon=\omega_f\left(\frac{\sqrt{d}}{K}\right)>0$ therein), there exists a ResNet $\widetilde{\psi}_2\in\mathcal{RN}(6,4,13\widetilde{L}+7)$ such that

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

and

$$0\le\widetilde{\psi}_2(x)\le\max\left\{g\left(\frac{j}{2K^d}\right):j=0,1,\cdots,2K^d-1\right\}\le 2\omega_f(\sqrt{d}),\qquad\text{for any } x\in\mathbb{R}.\tag{31}$$

By defining $\psi_2(x):=\widetilde{\psi}_2\left(2K^dx\right)$ for any $x\in\mathbb{R}$, we have $\psi_2\in\mathcal{RN}(6,4,20L+7)$, $0\le\psi_2(x)=\widetilde{\psi}_2\left(2K^dx\right)\le 2\omega_f(\sqrt{d})$, for any $x\in\mathbb{R}$, and

$$\left|\psi_2\left(\frac{j}{2K^d}\right)-g\left(\frac{j}{2K^d}\right)\right|=\left|\widetilde{\psi}_2(j)-g\left(\frac{j}{2K^d}\right)\right|\le\omega_f\left(\frac{\sqrt{d}}{K}\right),\qquad\text{for } j=0,1,\cdots,2K^d-1.$$

We then define the desired function ϕ as $\phi:=\psi_2\circ\psi_1$. Note that $\psi_1:\mathbb{R}^d\to\mathbb{R}$ is a linear function and $\psi_2\in\mathcal{RN}(6,4,20L+7)$. Thus, $\phi\in\mathcal{RN}(6,4,20L+7)$. Then we have

$$\left|\phi(\boldsymbol{\beta})-\widetilde{f}\left(\mathbf{x}_{\boldsymbol{\beta}}\right)\right|=\left|\psi_2\left(\psi_1(\boldsymbol{\beta})\right)-g\left(\psi_1(\boldsymbol{\beta})\right)\right|\le\omega_f\left(\frac{\sqrt{d}}{K}\right),\tag{32}$$

for any $\boldsymbol{\beta}\in\{0,1,\cdots,K-1\}^d$. Equation (31) and $\phi=\psi_2\circ\psi_1$ implies

$$0\le\phi(\mathbf{x})\le 2\omega_f(\sqrt{d}),\tag{33}$$

for any $\mathbf{x}\in\mathbb{R}^d$.

Step 4: Construct the final network to implement the desired function R. Define $R:=\phi\circ\Phi+f(\mathbf{0})-\omega_f(\sqrt{d})$ and the specific ResNet structure is illustrated by Fig. 10. Since $\Phi\in\mathcal{RN}(d+1,4L+8d-4)$ and $\phi\in\mathcal{RN}(6,4,20L+7)$, $R=\phi\circ\Phi+f(\mathbf{0})-\omega_f(\sqrt{d})$ is in

$$\mathcal{RN}\left(d+1,4,24L+9d+4\right)\ \text{if } d\ge 5$$

as shown in Fig. 10.

Now let us estimate the approximation error. Note that $f=\widetilde{f}+f(\mathbf{0})-\omega_f(\sqrt{d})$. By Equation (32), for any $\mathbf{x}\in Q_{\boldsymbol{\beta}}$ and $\boldsymbol{\beta}\in\{0,1,\cdots,K-1\}^d$, we have

$$\begin{aligned}|f(\mathbf{x})-R(\mathbf{x})|&=\left|\widetilde{f}(\mathbf{x})-\phi\left(\Phi(\mathbf{x})\right)\right|=\left|\widetilde{f}(\mathbf{x})-\phi(\boldsymbol{\beta})\right|\\&\le\left|\widetilde{f}(\mathbf{x})-\widetilde{f}\left(\mathbf{x}_{\boldsymbol{\beta}}\right)\right|+\left|\widetilde{f}\left(\mathbf{x}_{\boldsymbol{\beta}}\right)-\phi(\boldsymbol{\beta})\right|\\&\le\omega_f\left(\frac{\sqrt{d}}{K}\right)+\omega_f\left(\frac{\sqrt{d}}{K}\right)\le 2\omega_f\left(2\sqrt{d}L^{-2/d}\right),\end{aligned}$$

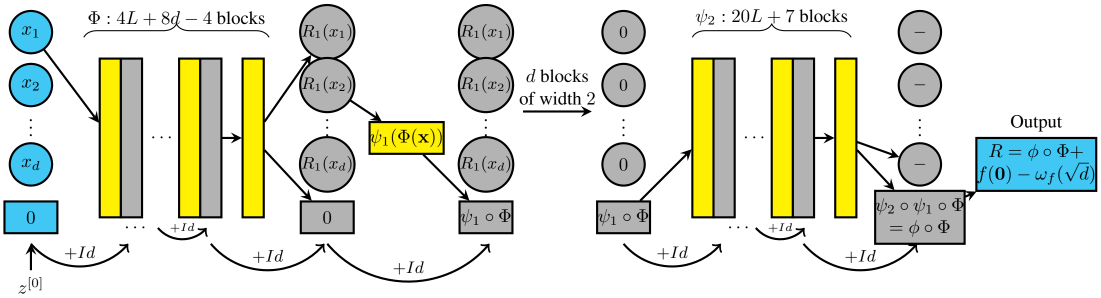


Figure 10. Illustration of implementing ResNet $R = \phi \circ \Phi + f(\mathbf{0} - \omega_f(\sqrt{d}))$.

where the last inequality comes from the fact $K = \lfloor L^{2/d} \rfloor \geq \frac{L^{2/d}}{2}$ for any $L \in \mathbb{N}^+$. Recall that $\omega_f(nr) \leq n\omega_f(r)$ for any $n \in \mathbb{N}^+$and $r \in [0, \infty)$. Therefore, for any $\mathbf{x} \in \cup_{\boldsymbol{\beta} \in \{0,1,\cdots,K-1\}^d} Q_{\boldsymbol{\beta}} = [0,1]^d \backslash \Omega\left([0,1]^d, K, \delta\right)$, we have

$$|f(\mathbf{x}) - R(\mathbf{x})| \leq 2\omega_f\left(2\sqrt{d}L^{-2/d}\right) \leq 2\lceil 2\sqrt{d}\rceil \omega_f\left(L^{-2/d}\right) \leq 6\sqrt{d}\omega_f\left(L^{-2/d}\right).$$

Last, by Equation (33) and $R = \phi \circ \Phi + f(\mathbf{0}) - \omega_f(\sqrt{d})$, it follows that $\|R\|_{L^\infty(\mathbb{R}^d)} \leq |f(\mathbf{0})| + \omega_f(\sqrt{d})$. Thus, we finish the proof.

□

F.3. Proof of Propostion 25

proof of Prop. 25. Without loss of generality, assume $K = \lfloor L^{2/d} \rfloor = \widetilde{L}^2$ where $\widetilde{L} = L^{1/d}$. We first consider the sample set

$$\{(1, \widetilde{L} - 1), (2, 0)\} \cup \left\{\left(\frac{m}{\widetilde{L}}, m\right) : m = 0, 1, \cdots, \widetilde{L} - 1\right\} \cup \left\{\left(\frac{m+1}{\widetilde{L}} - \delta, m\right) : m = 0, 1, \cdots, \widetilde{L} - 2\right\}.$$

Let $R_1(x)$ be a CPwL function by the sample set. It follows from Lemma 22 that there exists a ResNet $\widehat{R}_1 \in \mathcal{RN}(2, 1, 2\widetilde{L} + 1)$ such that $\widehat{R}_1(x) = (x, R_1(x))$ where $R_1(x)$ satisfy

- $R_1\left(\frac{\widetilde{L}-1}{\widetilde{L}}\right) = R_1(1) = \widetilde{L} - 1$ and $R_1\left(\frac{m}{\widetilde{L}}\right) = R_1\left(\frac{m+1}{\widetilde{L}} - \delta\right) = m$ for $m = 0, 1, \cdots, \widetilde{L} - 2$, and
- R_1 is linear on $\left[\frac{\widetilde{L}-1}{\widetilde{L}}, 1\right]$ and each interval $\left[\frac{m}{\widetilde{L}}, \frac{m+1}{\widetilde{L}} - \delta\right]$ for $m = 0, 1, \cdots, \widetilde{L} - 2$.

Thus, we have

$$R_1(x) = \ell, \quad \text{for } x \in \left[\frac{m}{\widetilde{L}}, \frac{m+1}{\widetilde{L}} - \delta \cdot \mathbb{1}_{\{\ell \leq \widetilde{L}-2\}}\right].$$

Next for the sample set

$$\left\{\left(\frac{1}{\widetilde{L}}, \widetilde{L} - 1\right), (2, 0)\right\} \cup \left\{\left(\frac{\ell}{\widetilde{L}^2}, \ell\right) : \ell = 0, 1, \cdots, \widetilde{L} - 1\right\} \cup \left\{\left(\frac{\ell+1}{\widetilde{L}^2} - \delta, \ell\right) : \ell = 0, 1, \cdots, \widetilde{L} - 2\right\}.$$

with size $2\widetilde{L} + 1$, similarly by Lemma 22, there exists a ResNet $\widehat{R}_2 \in \mathcal{RN}(2, 1, 2\widetilde{L} + 1)$ such that $\widehat{R}_2 = (x, R_2(x))$ and

- $R_2\left(\frac{\widetilde{L}-1}{\widetilde{L}^2}\right) = R_2\left(\frac{1}{\widetilde{L}}\right) = \widetilde{L} - 1$ and $R_2\left(\frac{\ell}{\widetilde{L}^2}\right) = R_2\left(\frac{\ell+1}{\widetilde{L}^2} - \delta\right) = \ell$ for $\ell = 0, 1, \cdots, \widetilde{L} - 2$;
- R_2 is linear on $\left[\frac{\widetilde{L}-1}{\widetilde{L}^2}, \frac{1}{\widetilde{L}}\right]$ and each interval $\left[\frac{\ell}{\widetilde{L}^2}, \frac{\ell+1}{\widetilde{L}^2} - \delta\right]$ for $\ell = 0, 1, \cdots, \widetilde{L} - 2$.

It follows that, for $m, \ell = 0, 1, \cdots, \widetilde{L}-1$,

$$R_2\left(x - \frac{m}{\widetilde{L}}\right) = \ell, \quad \text{for } x \in \left[\frac{m\widetilde{L}+\ell}{\widetilde{L}^2}, \frac{m\widetilde{L}+\ell+1}{\widetilde{L}^2} - \delta \cdot \mathbb{1}_{\{\ell \le \widetilde{L}-2\}}\right].$$

Since $K = \widetilde{L}^2$, each $k \in \{0, 1, \cdots, K-1\}$ can be unique represented by $k = m\widetilde{L} + \ell$ for $m, \ell = 0, 1, \cdots, \widetilde{L}-1$. For any $x \in \left[\frac{k}{K}, \frac{k}{K} - \delta \cdot \mathbb{1}_{\{k \le K-2\}}\right]$ for $k \in \{0, 1, \cdots, K-1\}$, $\widehat{R}_1(x) = (x, R(x)) = (x, m)$. Next, we define an affine mapping R_0 such that $R_0(x, m) = \left(m, x - \frac{m}{\widetilde{L}}\right)$. Finally, let $\widehat{R}_2(x) = (x, R_2(x))$. Then

$$\widehat{R}_2\left(m, x - \frac{m}{\widetilde{L}}\right) = \left(m, R_2\left(x - \frac{m}{\widetilde{L}}\right)\right) = (m, \ell).$$

With a final affine layer $\mathcal{L}(\ell, m) = m\widetilde{L} + \ell = k$, we can output the desired value.

Thus, the desired ResNet $R := \mathcal{L} \circ \widehat{R}_2 \circ R_0 \circ \widehat{R}_1$ satisfy

$$R(x) = k, \quad \text{if } x \in \left[\frac{k}{K}, \frac{k}{K} - \delta \cdot \mathbb{1}_{\{k \le K-2\}}\right] \text{ for } k \in \{0, 1, \cdots, K-1\}.$$

Note that an affine function can be computed by a residul block. Hence $R \in \mathcal{RN}(2, 1, 4\widetilde{L} + 4)$.

□

F.4. Proof of Proposition 26

For $\theta \in [0, 1)$, suppose its base-q representation is $\theta = \sum_{\ell=1}^{\infty} \theta_\ell q^{-\ell}$ with $\theta_\ell \in \{0, 1, \cdots, q-1\}$. Then we use the notation $0.\theta_1\theta_2\cdots\theta_L$ to denote the L-term base-q representation of θ, i.e., $0.\theta_1\theta_2\cdots\theta_L := \sum_{\ell=1}^{L} \theta_\ell q^{-\ell}$. We first show some fundamental lemmas which are the keys to the proof of Proposition 26.

Lemma 27. *For any $L \in \mathbb{N}^+$, there exists a ResNet R in $\mathcal{RN}(5, 4, 7L)$ such that, for any $\theta_1, \theta_2, \cdots, \theta_L \in \{0, 1, 2\}$, we have*

$$R(0.\theta_1\theta_2\cdots\theta_L, \ell) = \sum_{j=1}^{\ell} \theta_j, \quad \textit{for } \ell = 1, 2, \cdots, L$$

Proof. Given $\theta_1, \theta_2, \cdots, \theta_L \in \{0, 1, 2\}$, define

$$\xi_j := 0.\theta_j\theta_{j+1}\cdots\theta_L = \sum_{i=j}^{L} \frac{\theta_i}{3^{i-j+1}}, \quad \text{for } j = 1, 2, \cdots, L$$

and $\mathcal{T}_3(x) : [0, 3) \to \mathbb{R}$ as

$$\mathcal{T}_3(x) = k, \quad \text{if } x \in [k, k+1-\varepsilon] \text{ for } k = 0, 1, 2,$$

where ε is a parameter to be determined later. Then we have

$$\theta_j = \lfloor 3\xi_j \rfloor \text{ for } j = 1, 2, \cdots, L,$$

and

$$\xi_{j+1} = 3\xi_j - \theta_j \text{ for } j = 1, 2, \cdots, L-1.$$

Moreover, $\lfloor\cdot\rfloor$ can be approximated by the ReLU network $\mathcal{T}_3(\cdot)$. Note that if $\varepsilon < 3^{-L}$, then $\lfloor 3x \rfloor = \mathcal{T}_3(3x)$ for any $x = 0.\theta_1\theta_2\cdots\theta_l$ where $l \le L$. Thus, $\theta_j = \mathcal{T}_3(3\xi_j)$ for $j = 1, 2, \cdots, L$. Now let

$$\mathcal{T}(x) := \begin{cases} 1, x \ge 0, \\ 0, x < 0. \end{cases}$$

If $x \in \mathbb{Z}$, $\mathcal{T}(x) = \mathcal{T}_0(x)$ where $\mathcal{T}_0(x)$ is defined by

$$\mathcal{T}_0(x) := \begin{cases} 1, x \geq 0, \\ x+1, 0 \geq x \geq -1, \\ -1, x < -1. \end{cases}$$

Now we have

$$\sum_{j=1}^{\ell} \theta_j = \sum_{j=1}^{L} \theta_j \mathcal{T}(\ell - j) = \sum_{j=1}^{L} \theta_j \mathcal{T}_0(\ell - j) := \sum_{j=1}^{L} z_{\ell,j}$$

for $\ell = 1, 2, \cdots, L$ where $z_{\ell,j} = \theta_j \mathcal{T}_0(\ell - j) \geq 0$.

Here the multiplication of $x \in \{0,1,2\}$ and $y \in \{0,1\}$ can be done by $xy = \sigma(x+y-1) - \sigma(x-y-1)$.

Now, we construct a ResNet R such that $R\left(0.\theta_1\theta_2 \cdots \theta_L, \ell\right) = \sum_{j=1}^{\ell} \theta_j$ for $\ell = 1, 2, \cdots, L$. Note that $\mathcal{T}_3$ is a linear interpolation at the sample set

$$\{(k,k) : k = 0,1,2\} \cup \{(k+1-\varepsilon, k) : k = 0,1,2\}.$$

Then by Lemma 22, $\mathcal{T}_3$ can be generated by a ResNet $\widehat{H} \in \mathcal{RN}(2,1,6)$, i.e., $\widehat{H}(x) = (x, \mathcal{T}_3(x))$. Similarly, $\mathcal{T}_0$ can be generated by a ResNet $\widehat{P} \in \mathcal{RN}(2,1,2)$ where $\widehat{P}(x) = (x, \mathcal{T}_0(x))$. Note that if $\mathcal{A}(x)$ is CPwL and satisfies the condition of Lemma 22, so is $\mathcal{A}(ax+b)$ for $a, b \in \mathbb{R}$.

Then we have the following ResNet to output the desired value. Let input be (ξ_1, ℓ)

$$z^{[0]} = (\xi_1, \ell, 0, 0, 0).$$

According to the above discussion, after 6 residual blocks, we can compute

$$z^{[6]} = (\xi_1, \ell, \theta_1, \mathcal{T}_0(\ell - 1) := y_1, 0).$$

In the next residual block, we can compute

$$\zeta^{[7]} = (\theta_1, y_1, \sigma(\theta_1 + y_1 - 1), \sigma(\theta_1 - y_1 - 1))$$

and

$$\gamma^{[7]} = (-3\theta_1, 0, -\theta_1, -y_1, z_{\ell,1}).$$

Hence,

$$z^{[7]} = \gamma^{[7]} + z^{[6]} = (\xi_2, \ell, 0, 0, z_{\ell,1})$$

This computation process is shown in Fig. 11. Now, for some $m \geq 1$ we may assume the output of $7(m-1)$-th block is

$$z^{[7(m-1)]} = (\xi_m, \ell, 0, 0, \sum_{j=1}^{m-1} z_{\ell,j}).$$

As the same process, using 6 blocks, we can compute

$$z^{[7(m-1)+6]} = (\xi_m, \ell, \theta_m, \mathcal{T}_0(\ell - m) := y_m, 0).$$

Similarly, in the next block, by choosing appropriate weights, we can compute

$$\zeta^{[7(m-1)+7]} = (\theta_m, y_m, \sigma(\theta_m + y_m - 1), \sigma(\theta_m - y_m - 1))$$

and

$$\gamma^{[7]} = (-3\theta_m, 0, -\theta_m, -y_m, \sum_{j=1}^{m} z_{\ell,j}).$$

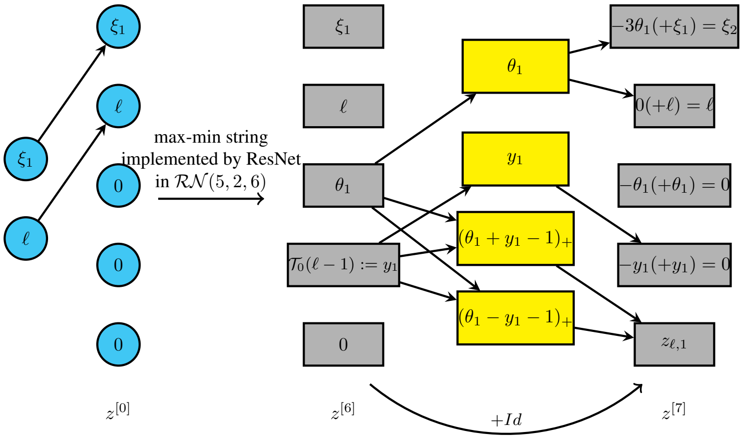


Figure 11. Illustration for the computation of $z_{\ell,1}$. The grey shape is in ReLU-free free and the yellow one is equipped with ReLU activation function.

Hence we have

$$z^{[7m]} = \gamma^{[7m]} + z^{[7m-1]} = (\xi_{m+1}, \ell, 0, 0, \sum_{j=1}^{m} z_{\ell,j})$$

By induction, there is a ResNet $R : \mathbb{R}^2 \to \mathbb{R}^5$ in $\mathcal{RN}(5,4,7L)$ such that

$$R(\xi_1, \ell) = (-, \ell, -, -, \sum_{j=1}^{L} z_{\ell,L} = \sum_{j=1}^{\ell} \theta_j).$$

Hence by choosing appropriate linear mapping $\mathcal{L}$ from $\mathbb{R}^5$ to $\mathbb{R}$, we have $\mathcal{L} \circ R(\xi_1 = 0.\theta_1\theta_2 \cdots \theta_L, \ell) = \sum_{j=1}^{\ell} \theta_j$.

□

Lemma 28. *For any $L \in \mathbb{N}^+$, any $\theta_{k,\ell} \in \{0,1,2\}$ for $k, \ell = 0, 1, \cdots, L-1$, there exists a function $R : \mathbb{R}^2 \to \mathbb{R}$ and $R \in \mathcal{RN}(5,4,9L+2)$ such that*

$$R(k,\ell) = \sum_{j=0}^{\ell} \theta_{k,j}, \quad \textit{for } k, \ell = 0, 1, \cdots, L-1.$$

Proof. Denote $y_k := 0.\theta_{k,0}\theta_{k,1}\cdots\theta_{k,L-1}$, for $k = 0, 1, \cdots, L-1$. Then for the sample set $\{(k, y_k) : k = 0, 1, \cdots, L\}$, whose size is $L+1$, it follows from Lemma 23 that there exists a ResNet $R_1 \in \mathcal{RN}(2,1,2L+2) \subset \mathcal{RN}(5,4,2L+1)$ such that

$$R_1(k) = y_k, \quad \text{for } k = 0, 1, \cdots, L-1.$$

By Lemma 27, there exists $R_2 \in \mathcal{RN}(5,4,7L)$ such that, for any $\xi_1, \xi_2, \cdots, \xi_L \in \{0,1,2\}$, we have

$$R_2\left(0.\xi_1\xi_2\cdots\xi_L, \ell\right) = \sum_{j=1}^{\ell} \xi_j, \quad \text{for } \ell = 1, 2, \cdots, L.$$

It follows that, for any $\xi_0, \xi_1, \cdots, \xi_{L-1} \in \{0,1,2\}$, we have

$$R_2\left(0.\xi_0\xi_1\cdots\xi_{L-1}, \ell+1\right) = \sum_{j=0}^{\ell} \xi_j, \quad \text{for } \ell = 0, 1, \cdots, L-1.$$

Thus, for $k, \ell = 0, 1, \cdots, L-1$, we have a ResNet R such that

$$R(k,\ell) = R_2\left(R_1(k), \ell+1\right) = R_2\left(y_k, \ell+1\right) = R_2\left(0.\theta_{k,0}\theta_{k,1}\cdots\theta_{k,L-1}, \ell+1\right) = \sum_{j=0}^{\ell} \theta_{k,j}.$$

Moreover, it is clear that R is a ResNet in $\mathcal{RN}(5, 4, 9L+2)$. □

Lemma 29. *For any $\varepsilon > 0, L \in \mathbb{N}^+$, and a sample set $\{y_{k,\ell} \geq 0\}_{k.\ell}$ with*

$$|y_{k,\ell} - y_{k,\ell-1}| \leq \varepsilon, \quad \textit{for } k, \ell = 0, 1, \cdots, L-1,$$

there exists a ReLU netework $\phi \in \mathcal{RN}(6, 4, 11L+5)$ such that

(i) $|\phi(k,\ell) - y_{k,\ell}| \leq \varepsilon$, for $k, \ell = 0, 1, \cdots, L-1$, and

(ii) $0 \leq \phi(x_1, x_2) \leq \max_{k,\ell=0,1,\cdots,L-1}\{y_{k,\ell}\}$, for any $x_1, x_2 \in \mathbb{R}$.

Proof. First we define

$$a_{k,\ell} := \lfloor y_{k,\ell}/\varepsilon \rfloor, \quad \text{for } k, \ell = 0, 1, \cdots, L-1.$$

We will construct a ResNet $R : \mathbb{R}^2 \to \mathbb{R}$ such that $R(k,\ell) = a_{k,\ell}\varepsilon$ for $k, \ell = 0, 1, \cdots, L-1$.

We denote $b_{k,0} := 0$ and $b_{k,\ell} := a_{k,\ell} - a_{k,\ell-1}$ for $k, \ell = 0, 1, \cdots, L-1$. It follows from $|y_{k,\ell} - y_{k,\ell-1}| \leq \varepsilon$ for all k and ℓ that $b_{k,\ell} \in \{-1, 0, 1\}$. Thus, we have

$$a_{k,\ell} = a_{k,0} + \sum_{j=1}^{\ell}(a_{k,j} - a_{k,j-1}) = a_{k,0} + \sum_{j=1}^{\ell} b_{k,j} = a_{k,0} + \sum_{j=0}^{\ell} b_{k,j}$$

for $k, \ell = 0, 1, \cdots, L-1$. For the sample set $\{(k, a_{k,0}) : k = 0, 1, \cdots, L-1\} \cup \{(L, 0)\}$, whose size is $L+1$, it follows from Lemma 23 that there exists a ResNet $\widehat{R}_1 \in \mathcal{RN}(2, 1, 2L+2)$ such that

$$\widehat{R}_1(k) = (k, R_1(k) = a_{k,0}), \quad \text{for } k = 0, 1, \cdots, L-1.$$

By Lemma 28, there exists a ResNet $R_2 \in \mathcal{RN}\left(5, 4, 9L+2\right)$ such that

$$R_2(k,\ell) = \sum_{j=0}^{\ell} b_{k,j} = \sum_{j=0}^{\ell}(b_{k,j}+1) - \ell.$$

Here note that $b_{k,j} \in \{0, 1, 2\}$ which will satisfy the condition of Lemma 28.

Thus, we can compute $a_{k,0}$ first by $\widehat{R}_1$. Then use one neuron in each linear layer to preserve the value of $a_{k,0}$ and compute $\sum_{j=0}^{\ell} b_{k,j}$ by R_2. For the new ResNet denoted by R_3, which is the composition of R_1 and R_2, we show how to construct it below. We let input be (k,ℓ) and

$$z^{[0]} = (k, \ell, 0, 0, 0, 0).$$

We use $\widehat{R}_1$ to compute

$$z^{[2L+2]} = (k, \ell, 0, 0, 0, a_{k,0}).$$

Then view $z^{[2L+2]}$ as the input of R_2. We can compute by Lemma 28

$$z^{[11L+4]} = (-, -, -, -, R_2(k,\ell), a_{k,0}).$$

By choosing appropriate affine mapping $\mathcal{L}(x) = \varepsilon(x_5 + x_6)$, we have $R(k,\ell) = \varepsilon a_{k,\ell}$.

Moreover, let $R_0(x) = \min\{\sigma(x), y_{\max}\}$. Then for $x, y \in \mathbb{R}$, $R(x,y) := R_0 \circ R_3(x,y) \leq y_{\max}$ and $R(k,\ell) = \min\{\varepsilon a_{k,\ell}, y_{\max}\} = a_{k,\ell}$ for $k, \ell = 0, 1, \cdots, L$.

Note $\min\{a,b\} = a - (a-b)_+$. Then R is a ResNet in $\mathcal{RN}\left(6, 4, 11L+5\right)$.

□

Now, we are ready to prove Prop. 26.

Proof of Prop. 26. Without loss of generality, assume $J = L^2$ since we can set $y_{J-1} = y_J = y_{J+1} = \cdots = y_{L^2-1}$ if $J < L^2$. For the sample set

$$\{(kL, k) : k = 0, 1, \cdots, L\} \cup \{(kL + L - 1, k) : k = 0, 1, \cdots, L - 1\}$$

with size is $2L + 1$, there is a ResNet $\widehat{R}_1 \in \mathcal{RN}(2, 1, 2L + 1)$ by Lemma 22 such that

- $\widehat{R_1}(x) = (x, R_1(x))$,
- $R_1(L^2) = L$ and $R_1(kL) = R_1(kL + L - 1) = k$ for $k = 0, 1, \cdots, L - 1$, and
- R_1 is a CPwL function defined by the sample set

It follows that

$$R_1(j) = k \quad \text{and} \quad j - L \cdot R_1(j) = \ell, \quad \text{where } j = kL + \ell,$$

for $k, \ell \in \{0, 1, \cdots, L-1\}$. Since any number j in $\{0, 1, \ldots, J-1\}$ can be uniquely decomposed as the form $j = kL+\ell$ for $k, \ell = 0, 1, \cdots, L - 1$, we denote $y_j = y_{kL+\ell}$ as $y_{k,\ell}$. Then by Lemma 29, there exists a ResNet $R_2 \in \mathcal{RN}(6, 4, 11L + 5)$ such that

$$|R_2(k, \ell) - y_{k,\ell}| \leq \varepsilon, \quad \text{for } k = 0, 1, \cdots, L - 1 \text{ and } \ell = 0, 1, \cdots, L - 1,$$

and

$$0 \leq R_2\left(x_1, x_2\right) \leq y_{\max}, \quad \text{for any } x_1, x_2 \in \mathbb{R}.$$

So for any $j = kL + \ell \in \{0, 1, \cdots, J - 1\}$, we have an affine mapping $\mathcal{L}$ such that

$$\mathcal{L} \circ \widehat{R}_1(j) = \mathcal{L}(k, j) = (k, \ell).$$

Let $R = R_2 \circ \mathcal{L} \circ R_1$. Moreover, note that an affine mapping can be computed by a residual block. Thus, we have R is a ReLU network in $\mathcal{RN}(6, 4, 13L + 7)$ and it satisfy

$$|R(j) - y_j| \leq \varepsilon.$$

Then we have finished the proof.

□

G. Experiments

In this appendix, we specify the experiment setting in Section 5. First, we assume that an appropriate algorithm can effectively manage the optimization error (e.g., Adam optimizer (Kingma & Ba, 2014)). We then choose a sufficiently complex target function to ensure that the approximation error is the dominant factor. To reduce the randomness and errors in our experiments, we conduct function approximation experiments across a set of functions. Specifically, we utilize the following set of functions to test the universal approximation capability of b-ResNet.

$$f(x) = \sum_{i=1}^{m} [a_i \prod_{j \in S_i^1} x_j + b_i \sin(\prod_{k \in S_i^2} x_k)], \tag{34}$$

where $x \in [0, 1]^d$, S_i^1 and S_j^1 are the index sets randomly sampled from $\{1, ..., d\}$ with replacement, a_i and b_i are coefficients, and m is the total number of terms.

Specifically, for each case of $d = 100, 200, 300$, we randomly selected 30 functions from the set for function approximation experiments. After removing some outliers, we took the average testing loss. For each $d \in \{100, 200, 300\}$, we set the parameters as $m = d/10$, card$(S_i^1) \leq \sqrt{d}$, card$(S_i^2) \leq \sqrt{d}$, $a_i \in [0, 1]$, and $b_i \in [0, 0.1]$, where the index sets (e.g., S_i^1) and coefficients (e.g., a_i) are randomly sampled.

We then compare b-ResNet with fully-connected (FC) NN for approximating each sampled function, with network structure as $\mathcal{RN}\,(d+1, n, d/10)$ for $n \in \{10, 20, 40\}$, and $\mathcal{NN}\,(d+1, d/10)$, respectively. Next for each approximated function, we conduct uniform sampling with $1000 \cdot d$ samples and use 90% for training and 10% for testing, and then take the average loss. We optimize the network parameters using Adam (Kingma & Ba, 2014) with a learning rate of 10^{-3} and present the test performance over iteration. The results include the mean square error (MSE) and the maximum absolute error (MAX) on testing samples.

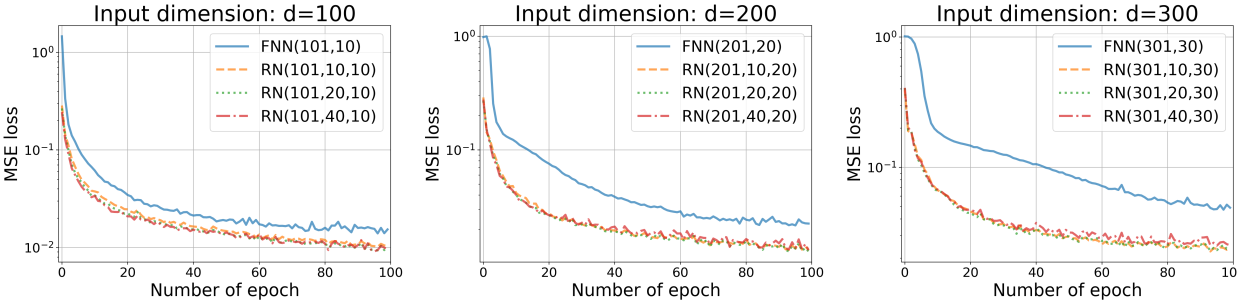


Figure 12. Comparison of MSE loss on testing samples when training with MSE loss.

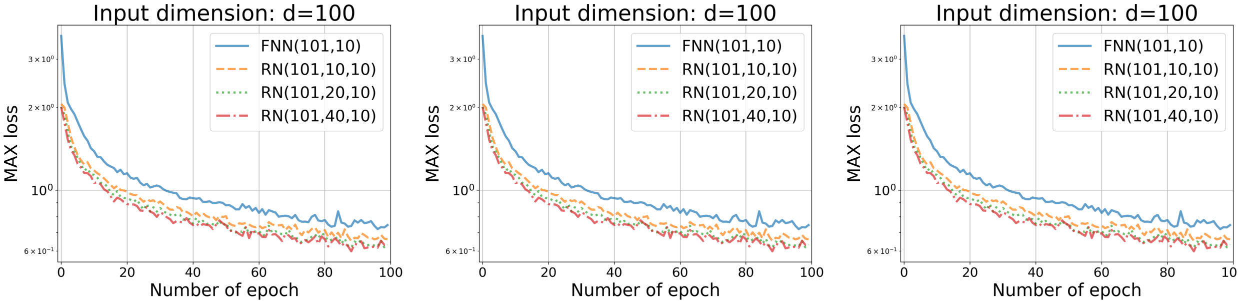


Figure 13. Comparison of MAX loss on testing samples when training with MSE loss.