Journal of Machine Learning Research 24 (2023) 1-22 Submitted 5/21; Revised 3/23; Published 5/23

Quasi-Equivalence between Width and Depth of Neural Networks

Feng-Lei Fan FANF2@RPI.EDU
Department of Biomedical Engineering
School of Engineering
Biomedical Imaging Center
Center for Biotechnology and Interdisciplinary Studies
Rensselaer Polytechnic Institute
Troy, NY 12180, USA

Rongjie Lai LAIR@RPI.EDU
Department of Mathematics
Rensselaer Polytechnic Institute
Troy, NY 12180, USA

Ge Wang[*] WANGG6@RPI.EDU
Department of Biomedical Engineering
School of Engineering
Biomedical Imaging Center
Center for Biotechnology and Interdisciplinary Studies
Rensselaer Polytechnic Institute
Troy, NY 12180, USA

Editor: Joan Bruna

Abstract

While classic studies proved that wide networks allow universal approximation, recent research and successes of deep learning demonstrate the power of deep networks. Based on a symmetric consideration, we investigate if the design of artificial neural networks should have a directional preference, and what the mechanism of interaction is between the width and depth of a network. Inspired by the De Morgan law, we address this fundamental question by establishing a quasi-equivalence between the width and depth of ReLU networks. We formulate two transforms for mapping an arbitrary ReLU network to a wide ReLU network and a deep ReLU network respectively, so that the essentially same capability of the original network can be implemented. Based on our findings, a deep network has a wide equivalent, and vice versa, subject to an arbitrarily small error.

Keywords: Artificial neural networks, deep learning, wide learning, ReLU networks, quasi-equivalence.

1. Introduction

Over the past years, deep learning (Goodfellow et al., 2016) has become the mainstream approach of machine learning and achieved the state-of-the-art performance in many im-

∗. Corresponding author

portant tasks (Dahl et al., 2011; Kumar et al., 2016; Chen et al., 2017; Wang, 2016). One of the key reasons that accounts for the success of deep learning is the increased depth, which allows a hierarchical representation of features. A number of papers are dedicated to explaining why deep networks are better than shallow ones. Their idea is basically to find a special family of functions hard to be approximated by a shallow network but easy to be approximated by a deep network, or that a deep network can express complicated functions that a wide network falls short (Szymanski and McCane, 2014; Cohen et al., 2016; Mhaskar and Poggio, 2016; Eldan and Shamir, 2016; Montufar et al., 2014; Bianchini and Scarselli, 2014). For example, in Eldan and Shamir (2016) a special class of radial functions was constructed so that a one-hidden-layer network needs to use an exponential number of neurons to obtain a good approximation, but a two-hidden-layer network only requires a polynomial number of neurons for the same purpose. With the number of linear regions as the complexity measure, Montufar et al. (2014) showed that the number of linear regions grows exponentially with the depth of a network but only polynomially with the width of a network. In (Bianchini and Scarselli, 2014), a topological measure was utilized to characterize the complexity of functions. Then, it was shown that deep networks can represent more complex functions than what their shallow counterparts express, given the same amount of resources. Besides, width-bounded but depth-unbounded universal approximators were also developed (Lu et al., 2017; Lin and Jegelka, 2018; Fan et al., 2021) in analogy to the depth-bounded but width-unbounded universal approximators (Funahashi, 1989; Hornik et al., 1989).

Recently, the effects of width are discussed by more and more studies (Cheng et al., 2016; Chen and Liu, 2017; Zagoruyko and Komodakis, 2016). Since width and depth are the most basic topology measures of a neural network, exploring the roles of width and depth in neural networks is of strong interest and great importance. Currently, there exist both width-bounded and depth-bounded universal approximators. Since both width-bounded and depth-bounded networks can represent any function, they can represent each other as well, which suggests the width-depth equivalence of neural networks. Nevertheless, how a neural network learns a mapping is quite different from the way used in proving the universal approximation. Moreover, the core of the width-depth conversion is to employ a network to learn another network instead of a generic function. Therefore, the width-depth conversion based on universal approximation does not directly reveal the intrinsic relationship between the width and depth.

Specifically, we argue that the width-depth conversion via universal approximation lacks insight and is inefficient: 1) (Lack of insight) Universal approximation theoretically guarantees the representation capability of a model, but the way used in the universal approximation is different from how a model learns a mapping. For example, both decision trees and neural networks are universal approximators, but they are quite distinct. The way used in enabling the universal approximation is to divide a target function into many functions over tiny hypercubes. In practice, a network usually does not do so; *e.g.*, a ReLU network divides the space into polytopes, and a ReLU network is piecewise linear over polytopes. 2) (Inefficient) The published universal approximation analyses do not consider the character of a target function, and just inefficiently divide the function into many pieces over tiny hypercubes. However, in the width-depth transformation, the problem is how to express a wide (or deep) network using a deep (or wide) network. Thus, the properties of networks should be used to make a much more efficient transformation. In brief, utilizing universal approximation theorems to do the width-depth transformation do not accurately capture the model's essential characteristics in the width-depth conversion. Up to now, we still do not know how to non-trivially do transformation between a wide and a deep network, and what the mechanism of interaction is between the width and depth of a network.

To bridge this gap, inspired by the De Morgan law, here we demonstrate that the width and depth of neural networks are quasi-equivalent, which mainly leverages that a ReLU network is a piecewise linear function over polytopes. Specifically, we revisit the De Morgan law:

$$A_1 \vee A_2 \cdots \vee A_n = \neg\Big((\neg A_1) \wedge (\neg A_2) \cdots \wedge (\neg A_n)\Big), \tag{1}$$

where A_i is a propositional rule (*e.g.*, IF *input* $\in [a_i, b_i]$, THEN *input* belongs to some class), and such rules are disjoint. A network for classification can be linked to a rule-based system such as a collection of propositional rules. Straightforwardly, we can construct either a deep network to realize a union of propositional rules (left side) or a wide network that realizes the complement of the intersection of those rules after the complement (right side). As a result, the constructed deep and wide networks are equivalent to each other. Furthermore, we elaborate on the quasi-equivalence of ReLU networks by constructing two transforms mapping an arbitrary ReLU network to its quasi-equivalent wide network and quasi-equivalent deep network respectively, thereby verifying a general quasi-equivalence of the width and depth of ReLU networks. Our constructive scheme is largely based on the fact that a ReLU network partitions the space into polytopes (Chu et al., 2018). By partitioning each polytope into simplices, we have a simplicial complex in the space. Our strategy is to construct a modularized network to approximate an arbitrary linear function over a simplex as the essential building block. Then, we can aggregate these essential building blocks into either a wide or a deep network so as to establish the width-depth equivalence.

Our main contribution is the establishment of the width-depth quasi-equivalence of neural networks. We summarize our key results on ReLU networks in Table 1. Specifically, Table 1 lists the width and depth of the wide and deep networks constructed in our perspective, where the complexity measure M is the minimum number of simplices needed to partition the polytopes formed by a ReLU network (below we rigorously define the concept of partitioning). Given a proper complexity measure, the width of the constructed wide network is greater than the depth, while the depth of the constructed deep network is greater than the width.

Table 1: Networks obtained through the transformation, where D is the input dimension, and M is the complexity measure for a function class defined by ReLU networks.

	Network	Width	Depth
Transform Regression	Wide	$D^2(D+1)M$	$D+1$
Networks (**Theorem 10**)	Deep	$(D+1)D^2$	$(D+1)M$

To put our contributions in perspective, we would like to mention relevant studies. Jacot et al. (2018) proposed the theory of the neural tangent kernel (NTK), which provides a lens to understand a network when the width of a network goes to infinity. Kawaguchi et al. (2019) analyzed the effects of width and depth on the quality of local minima. It was shown that the quality of local minima improves toward the global minima as depth and width become larger. Levine et al. (2020) revealed the width-depth interplay in a self-attention network. To the best of our knowledge, our study is the first that reveals the width-depth quasi-equivalence of neural networks.

2. Quasi-Equivalence by De Morgan's Law

2.1 Preliminaries

For convenience, we use $\sigma(x) = \max\{0, x\}$ to denote the ReLU function. We mainly discuss ReLU networks in this study. Thus, all networks mentioned in the rest of this paper are ReLU networks, unless otherwise specified. At the same time, we focus on the fully-connected ReLU networks.

Definition 1 (Width and depth of a feedforward network) *Here, we follow the definition in (Arora et al., 2016). Given the number of hidden layers $k \in \mathbb{N}$, input and output dimensions $w_0, w_{k+1} \in \mathbb{N}$, a $\mathbb{R}^{w_0} \to \mathbb{R}^{w_{k+1}}$ feedforward network is given by specifying a sequence of k natural numbers $w_1, w_2, \ldots, w_k$ representing widths of the hidden layers, a set of k affine transformations $T_i : \mathbb{R}^{w_{i-1}} \to \mathbb{R}^{w_i}$ for $i = 1, \ldots, k$ and a linear transformation $T_{k+1} : \mathbb{R}^{w_k} \to \mathbb{R}^{w_{k+1}}$ corresponding to weights of the hidden layers. The function $f : \mathbb{R}^{w_0} \to \mathbb{R}^{w_{k+1}}$ computed or represented by this network is expressed as*

$$f = T_{k+1} \circ \sigma \circ T_k \circ \cdots \circ T_2 \circ \sigma \circ T_1, \tag{2}$$

where $\circ$ denotes function composition, and σ is an activation function. The depth of a ReLU DNN is defined as $k+1$. The width of a ReLU DNN is $\max\{w_1, \ldots, w_k\}$.

Definition 2 (Width and depth of a non-feedforward network) *Let $\mathbf{\Pi}$ be a network that is not feedforward, we delete a minimum number of links such that the resultant network $\mathbf{\Pi}'$ is a feedforward network without any isolated neuron. Then, we define the width and depth of $\mathbf{\Pi}$ as the width and depth of $\mathbf{\Pi}'$ respectively. If there are multiple ways to delete links with the same minimum number such that the resultant networks are feedforward without any isolated neuron, we take the maximum depth and width of these networks as the depth and width of $\mathbf{\Pi}$.*

Over the past several years, increasingly diversified network architectures, such as randomly wired networks (Xie et al., 2019), networks with stochastic structures (Deng et al., 2020), etc. are used as backbones for deep learning. Our definitions for width and depth are applicable to many unusual network configurations. Naturally, they are extensions of the conventional width and depth definitions and make sense for common networks. We have checked the networks constructed in our study. The width and depth of the networks can be uniquely determined by our definition because these architectures are simple modifications based on feedforward networks (Please see **Supplementary Information V** for details).

Definition 3 (Simplicial complex) *A D-simplex S is a D-dimensional convex hull provided by convex combinations of $D+1$ affinely independent vectors $\{\mathbf{v}_i\}_{i=0}^{D} \subset \mathbb{R}^D$. In other* *words,* $S = \left\{ \sum_{i=0}^{D} \xi_i \mathbf{v}_i \mid \xi_i \geq 0, \sum_{i=0}^{D} \xi_i = 1 \right\}$. *The convex hull of any subset of* $\{\mathbf{v}_i\}_{i=0}^{D}$ *is called a face of* S. *A simplicial complex* $\mathcal{S} = \bigcup_{\alpha} S_\alpha$ *is composed of a set of simplices* $\{S_\alpha\}$ *satisfying: 1) every face of a simplex from* $\mathcal{S}$ *is also in* $\mathcal{S}$*; 2) the non-empty intersection of any two simplices* $S_1, S_2 \in \mathcal{S}$ *is a face of both* S_1 *and* S_2.

Proposition 1 *Suppose that* $f(\mathbf{x})$ *is a function represented by a ReLU network, then* f *is a piecewise linear function that splits the space into polytopes, where each polytope is convex, and* f *is a linear function over each polytope.*

Proof Inspired by the idea in (Chu et al., 2018), the proof here is for all ReLU networks, including networks using shortcuts. Let a vector $\mathbf{C} = \{c_1, ..., c_N\}$ denote the firing states of all neurons in the network, where N is the total number of neurons, and $c_i \in \{0, 1\}$. $c_i = 0$ means that the i-th neuron is not fired and vice versa. The firing state of every neuron is determined by the input $\mathbf{x}$, and we denote the set of instances that share the same collective neuron firing state $\mathbf{C_h}$ as the polytope P_h: $P_h = \{\mathbf{x} \mid \mathbf{C}(\mathbf{x}) = \mathbf{C_h}\}$. For $\mathbf{x} \in P_h$, the output of the neuron i is a linear function, denoted as $n^{(i)}(\mathbf{x})$. Then, the firing state of the neuron i is controlled by a linear inequality ($n^{(i)}(\mathbf{x}) > 0$ or $n^{(i)}(\mathbf{x}) \leq 0$). In total, $\mathbf{C}(\mathbf{x}) = \mathbf{C_h}$ is equivalent to a set of N linear inequality constraints, indicating that P_h is a convex polytope. ■

Definition 4 *A partition of a polytope* P *into simplices fulfills that the union of all simplices equals* P*, and the intersection of any two simplices is a common face or empty.*

Definition 5 *We define the complexity of the function represented by a ReLU network as the minimum number of simplices,* M*, that is needed to partition each and every polytope to support the function of the ReLU network.*

Here we elaborate on why M is a good measure. Previously, because a deep network with piecewise linear activation is a piecewise linear function, the number of linear regions (polytopes) was intensively studied to measure the complexity of a neural network. For example, Montufar et al. (2014) and Serra et al. (2018) estimated the upper and lower bounds of the number of linear regions with respect to the number of neurons at each layer. Xiong et al. (2020) computed the bounds for convolutional neural networks. Park et al. (2021) proposed neural activation coding to maximize the number of linear regions to improve the model performance. Despite these results, there exists a problem with the number of linear regions as a complexity measure. It may happen that simple and complex networks realize the same number of regions for a given task. As shown in Figure 1, two networks divide the space into two regions to separate concentric rings. However, one network uses three neurons to define a triangle domain, while the other has six neurons to form a hexagon. According to the number of linear regions, the two networks have the same complexity but the six-neuron network is more complex than the other. To address this problem, the complexity of a linear region should be taken into account as well. We argue

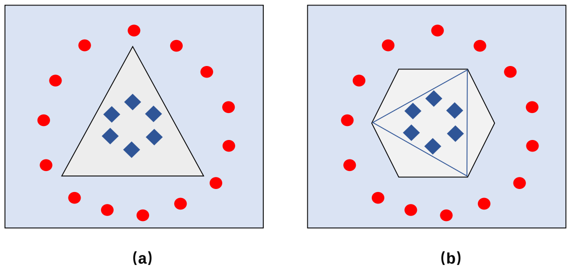

Figure 1: Two networks for classification of concentric rings of red circles and blue diamonds. (a) A one-hidden-layer network with 3 neurons to classify concentric rings whose polytope contains 1 simplex; and (b) a one-hidden-layer network with 6 neurons to classify concentric rings whose polytope comprises 4 simplices. Clearly, the number of simplices is a more meaningful complexity measure of ReLU networks than the number of polytopes.

that how many simplices a linear region comprises indicates how complex a linear region is. Therefore, we propose to use the number of simplices as a legitimate complexity measure. In Figure 1, counting the number of simplices, the complexity of two networks are 1 and 4 respectively, which is a better characterization.

Let us estimate the lower bound of M. To this end, we need to take advantage of the lower bound of the number of polytopes. Empirical bounds of the number of polytopes (N_p) for a feedforward ReLU network were estimated in (Montufar et al., 2014; Serra et al., 2018; Serra and Ramalingam, 2020), where one result in (Montufar et al., 2014) states that let $n_i, i = 1, \cdots, L$, be the number of neurons in the i-th layer, and D be the dimension of the input space, N_p is lower bounded by $\left(\prod_{i=1}^{L-1}[n_i/D]^D\right)\cdot\sum_{j=0}^{D}\binom{n_L}{j}$. Thus, the minimum number of simplices that fill these polytopes is naturally also lower bounded by

$$M \geq \left(\prod_{i=1}^{L-1}[n_i/D]^D\right)\cdot\sum_{j=0}^{D}\binom{n_L}{j}. \tag{3}$$

As the original network becomes more complicated in terms of the minimum number of simplices (M) to partition the input space, the width of the constructed wide network and the depth of the constructed deep network will increase accordingly.

Definition 6 *We measure the depth and width of a network in terms of complexity in M, where M is the minimum number of simplices to partition the polytopes represented by the original network. More precisely, given M, we call the constructed network wide if its width is C_1M and its depth is C_2, where C_1 and C_2 are constants independent of M. Similarly, we call the constructed network deep if its depth is C_3M and its width is C_4, where C_3 and C_4 are also constants independent of M.*

It is underscored that we use two different concepts: the complexity of the function class represented by networks and the structural complexity of a network. The former measures the complexity of the function, while the latter measures the topological structure of a network. In our transformation scheme, the structures of constructs/networks are determined by the complexity of a function of interest.

Definition 7 *We call that a wide network* $\mathbf{N}_1 : \Omega \to \mathbb{R}$ *is equivalent to a deep network* $\mathbf{N}_2 : \Omega \to \mathbb{R}$, *if* $\mathbf{N}_1(\mathbf{x}) = \mathbf{N}_2(\mathbf{x}), \forall x \in \Omega$. *We call that a wide network* $\mathbf{N}_1$ *is quasi-equivalent to a deep network* $\mathbf{N}_2$, *if there is a pre-specified small constant* $\delta > 0$, $\mathfrak{m}(\{\mathbf{x} \in \Omega \mid \mathbf{N}_1(\mathbf{x}) \neq \mathbf{N}_2(\mathbf{x})\} < \delta$, *where* $\mathfrak{m}$ *is a Lebesgue measurement defined on* Ω.

Definition 8 (Fan-shaped functions/domains) *Given vectors* $\{\mathbf{v}_i\}_{i=0}^{D} \subset \mathbb{R}^D$ *that are affinely independent, we define the fan-shaped domain with the vertex* $\mathbf{v}_0$ *in* $\mathbb{R}^D$ *as*

$$\{\sum_{i=1}^{D} \xi_i(\mathbf{v}_i - \mathbf{v}_0) \mid \xi_i \geq 0\}. \tag{4}$$

The fan-shaped function is a linear function over the fan-shaped domain and zero outside that domain.

2.2 Motivating Example

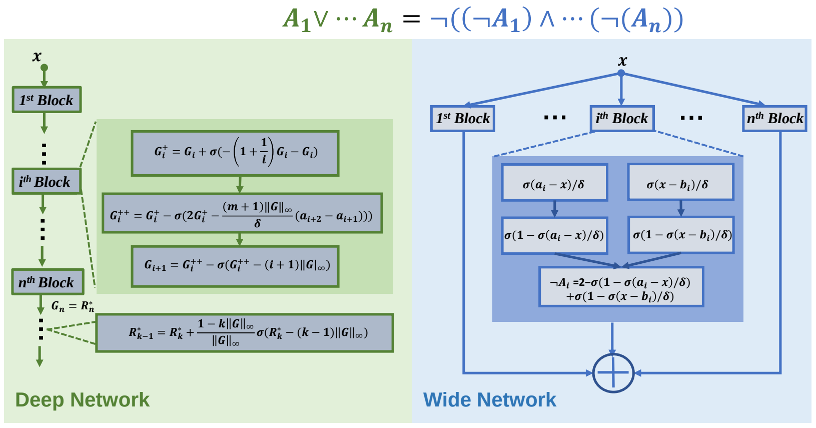
$A_1 \vee \cdots A_n = \neg((\neg A_1) \wedge \cdots (\neg(A_n))$
x
1st Block
ith Block
nth Block
$G_i^{+} = G_i + \sigma(-\left(1+\frac{1}{i}\right)G_i - G_i)$
$G_i^{++} = G_i^{+} - \sigma(2G_i^{+} - \frac{(m+1)\|G\|_\infty}{\delta}(a_{i+2} - a_{i+1})))$
$G_{i+1} = G_i^{++} - \sigma(G_i^{++} - (i+1)\|G|_\infty)$
$G_n = R_n^*$
$R_{k-1}^* = R_k^* + \frac{1-k\|G\|_\infty}{\|G\|_\infty}\sigma(R_k^* - (k-1)\|G\|_\infty)$
Deep Network
$\sigma(a_i - x)/\delta$
$\sigma(x - b_i)/\delta$
$\sigma(1 - \sigma(a_i - x)/\delta)$
$\sigma(1 - \sigma(x - b_i)/\delta)$
$\neg A_i = 2 - \sigma(1 - \sigma(a_i - x)/\delta) + \sigma(1 - \sigma(x - b_i)/\delta)$
Wide Network

Figure 2: The width and depth equivalence in light of the De Morgan law. In this equivalence construction, a deep network implements $A_1 \vee A_2 \cdots \vee A_n$ using a trapezoid function, while a wide network implements $\neg\Big((\neg A_1) \wedge (\neg A_2) \cdots \wedge (\neg A_n)\Big)$ using the trap-like function.

An important school of neural network interpretability research is to extract interpretable rules from a network for classification using decompositional or pedagogical methods (Fan and Wang, 2020; Adadi and Berrada, 2018; Thrun, 1995; Setiono and Liu, 1995; Saad and II, 2007; Thrun, 1995). Pedagogical methods decode a set of rules that imitate the input-output relationship of a network, and these rules do not necessarily correspond to the parameters of the network. One common type of rules are propositional in the IF-THEN format, where the preconditions are provided as a set of hypercubes in the input space:

IF *input* $\in [a_i, b_i]^m$, THEN *input* belongs to some class.

Since there is a connection between the rule-based inference and the network-based inference, we consider a neural network in terms of propositional rules. Furthermore, we know that the De Morgan law holds true for disjoint propositional rules. Mathematically, the De Morgan law is formulated as

$$A_1 \vee A_2 \cdots \vee A_n = \neg\Big((\neg A_1) \wedge (\neg A_2) \cdots \wedge (\neg A_n)\Big), \tag{5}$$

where A_i is a rule, and $\neg A_i$ is its negation. The De Morgan law gives a duality in the sense of binary logic, which means that for any propositional rule system described by $A_1 \vee A_2 \cdots \vee A_n$, there exists an equivalent dual propositional rule system $\neg\Big((\neg A_1) \wedge (\neg A_2) \cdots \wedge (\neg A_n)\Big)$.

Regarding each rule as an indicator function over a hypercube:

$$g_i(\mathbf{x}) = \begin{cases} 1, & \textbf{if} \quad \mathbf{x} \in \text{a} \quad \text{hypercube} \\ 0, & \textbf{if} \quad \mathbf{x} \notin \text{a} \quad \text{hypercube} \end{cases}$$

in Figure 2, we construct a deep network that realizes a logic union of propositional rules (the left hand side of Eq. (5)) and a wide network that realizes the negation of the logic intersection of those rules after negation (the right hand side of Eq. (5)). As a result, the constructed deep and wide networks are equivalent by the De Morgan law.

The above motivating example inspires us to consider the width-depth equivalence in a broader sense. First, a ReLU network is a piecewise linear function over polytopes. To generate rules, such a piecewise linear function should be divided into simplices instead of hypercubes. As shown in Figure 3, we only need two rules if we build rules over simplices, which is much more efficient than building rules over hypercubes. Second, the network can generate continuous values instead of discrete values. Thus, representing a linear function rather than an indicator function is demanded. Based on these two considerations, we generalize an indicator function over a hypercube to a linear function over a simplex S_i in a bounded domain:

$$g_i(\mathbf{x}) = \begin{cases} \mathbf{w}\mathbf{x} + c, & \textbf{if} \quad \mathbf{x} \in S_i \\ 0, & \textbf{if} \quad \mathbf{x} \in S_i^c. \end{cases}$$

2.3 Quasi-Equivalence between Width and Depth of Networks

This section describes the main contribution of our paper. We formulate the transformation from an arbitrary ReLU network to a wide network and a deep network respectively. We use a network-based building block to represent a linear function over a simplex. Integrating such building blocks can represent any piecewise linear function over polytopes, thereby elaborating a general equivalence between the width and depth of networks. In particular, a ReLU network can be converted into wide and deep ReLU networks respectively (**Theorem 10**). The transformation of a univariate network is rather different from that of a multivariate network. Whereas the equivalence is precise between the wide and deep networks in the univariate case, the multivariate wide and deep networks are made approximately equivalent, up to an arbitrarily small error. Furthermore, in the multivariate case the width

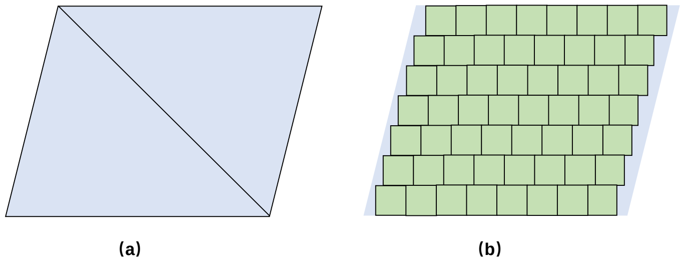

Figure 3: Given a polytope, one can partition it into simplices or hypercubes. Because a polytope is naturally partitioned by simplices, the number of needed simplices is much smaller than that of tiny hypercubes needed for universal approximation. Thus, simplices are much more efficient than hypercubes.

of the wide network is not the same as the depth of the deep network. This is one reason why we term such an equivalence as a quasi-equivalence.

The sketch of transforming a ReLU network to a quasi-equivalent form is that we first construct either a wide modular network or a deep modular network to represent the corresponding function over each and every simplex, and then we aggregate the results into deep or wide networks in series or parallel respectively, to represent the original network well.

Theorem 9 (Equivalence of Univariate ReLU Networks) *Given any ReLU network $f : [-B,\ B] \to \mathbb{R}$, there is a wide ReLU network $\mathbf{H}_1 : [-B,\ B] \to \mathbb{R}$ and a deep ReLU network $\mathbf{H}_2 : [-B,\ B] \to \mathbb{R}$, such that $f(x) = \mathbf{H}_1(x) = \mathbf{H}_2(x), \forall x \in [-B,\ B]$.*

Our main result is formally summarized as the following quasi-equivalence theorem for the multivariate case.

Theorem 10 (Quasi-Equivalence of Multivariate ReLU Networks) *Suppose that the representation of an arbitrary ReLU network is $h : [-B,\ B]^D \to \mathbb{R}$, and M is the minimum number of simplices to partition the polytopes supporting h, for any $\delta > 0$, there exist a wide ReLU network $\mathbf{H}_1$ of width $\mathcal{O}\left[D^2(D+1)M\right]$ and depth $D+1$ and a deep ReLU network $\mathbf{H}_2$ of width $(D+1)D^2$ and depth $\mathcal{O}\left[(D+1)M\right]$, satisfying that*

$$\begin{aligned} \mathfrak{m}\Big(\mathbf{x} \mid h(\mathbf{x}) \neq \mathbf{H}_1(\mathbf{x})\}\Big) &< \delta \\ \mathfrak{m}\Big(\mathbf{x} \mid h(\mathbf{x}) \neq \mathbf{H}_2(\mathbf{x})\}\Big) &< \delta, \end{aligned} \tag{6}$$

where $\mathfrak{m}(\cdot)$ is the standard measure in $[-B,\ B]^D$.

We defer the proof of **Theorem 9** to **Supplementary Information I**, and split the proof of **Theorem 10** into the two-dimensional case (more intuitive) in Appendix and the general case in **Supplementary Information I** for better readability.

The key idea to represent a linear function over a simplex is to construct high-dimensional fan-shaped functions that are supported in fan-shaped domains, and use these constructs

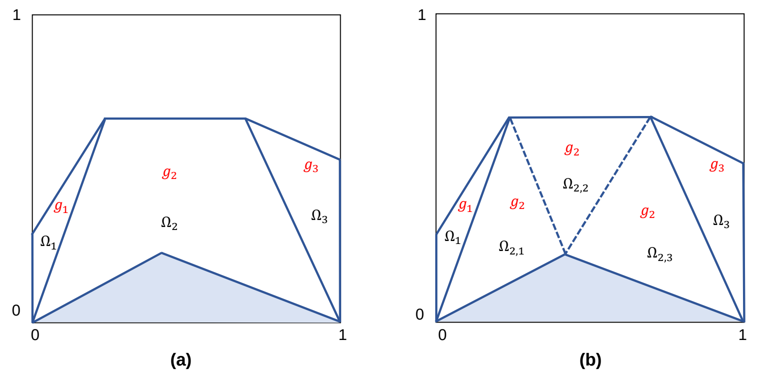

Figure 4: Representing a function that is piecewise linear over a non-convex region and zero otherwise. (a) The representations in (Wang and Sun, 2005; He et al., 2018) are handicapped in representing a function over polytopes that make a non-convex region. It can be seen that due to the unboundedness and continuity, $f = \max\{g_1, g_2, g_3\}$ is inaccurate over the shaded area (The function value over the shaded area should be zero, but f generates a linear function over the shaded area); (b) our representation is accurate over the shaded area because it directly represents each linear function over its simplex.

to eliminate non-zero functional values outside the simplex of interest. This is a new and local way to represent a piecewise linear function over polytopes. In contrast, there are two global ways to represent piecewise linear functions (Wang and Sun, 2005; He et al., 2018). In (Wang and Sun, 2005), for every piecewise linear function $f : \mathbb{R}^n \to \mathbb{R}$, there exists a finite set of linear functions $g_1, \cdots, g_m$ and subsets $T_1, \cdots, T_P \subseteq \{1, 2, \cdots, m\}$ such that $f = \sum_{p=1}^{P} s_p \max_{i\in T_p}\{g_i\}$, where $s_p \in \{-1, +1\}, p = 1, \cdots, P$. In (He et al., 2018), the representation is $f = \max_{1\leq p\leq P} \min_{i\in T_p}\{g_i\}$. Nevertheless, due to the unboundedness and continuity, the global representation of a piecewise linear function is handicapped over polytopes that make a non-convex region. Let us use Figure 4 to illustrate our point: the target function over $\Omega_1, \Omega_2, \Omega_3$ is piecewise linear, where the relations of g_1, g_2, g_3 are summarized in Table 2, and the function value over the shaded area is zero.

Table 2: Regions and relations of linear functions.

Region	Relation
Ω_1	$g_1 \geq g_2 \geq g_3$
Ω_2	$g_2 \geq g_1, g_2 \geq g_3$
Ω_3	$g_3 \geq g_2 \geq g_1$

- Representation in (Wang and Sun, 2005; He et al., 2018): $f = \max\{g_1, g_2, g_3\}$
- Ours: $f = (g_1)_{\{\mathbf{x}\in\Omega_1\}} + (g_2)_{\{\mathbf{x}\in\Omega_{2,1}\}} + (g_2)_{\{\mathbf{x}\in\Omega_{2,2}\}} + (g_2)_{\{\mathbf{x}\in\Omega_{2,3}\}} + (g_3)_{\{\mathbf{x}\in\Omega_3\}}$.

where $(g_1)_{\{\mathbf{x}\in\Omega_1\}}$ means restriction of g_1 on Ω_1 and the rest notations follow the same rule. It can be seen that due to the unboundedness and continuity, $f = \max\{g_1, g_2, g_3\}$ is inaccurate over the shaded area (The function value over the shaded area should be zero, but f generates a linear function over the shaded area). In contrast, our representation is accurate over the shaded area because it is local. We highlight the construction of fan-shaped functions that provides a new way for the representation of a high-dimensional piecewise linear function. Particularly, the use of fan-shaped functions allows a resultant neural network to express a manifold effectively and efficiently, since a manifold usually occupies a non-convex region.

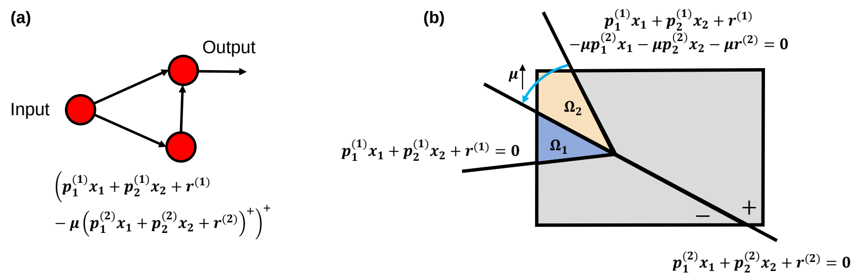
(a)
Output
Input
$\left(p_1^{(1)}x_1 + p_2^{(1)}x_2 + r^{(1)} - \mu\left(p_1^{(2)}x_1 + p_2^{(2)}x_2 + r^{(2)}\right)^+\right)^+$
(b)
$p_1^{(1)}x_1 + p_2^{(1)}x_2 + r^{(1)} - \mu p_1^{(2)}x_1 - \mu p_2^{(2)}x_2 - \mu r^{(2)} = 0$
μ
Ω_2
Ω_1
$p_1^{(1)}x_1 + p_2^{(1)}x_2 + r^{(1)} = 0$
$-$
$+$
$p_1^{(2)}x_1 + p_2^{(2)}x_2 + r^{(2)} = 0$

Figure 5: Fan-shaped functions constructed by a modularized network to eliminate non-zero functional values outside a simplex of interest. (a) The network module $F(x_1, x_2)$ is defined in terms of $h_1(\mathbf{x}) = p_1^{(1)}x_1 + p_2^{(1)}x_2 + r^{(1)}$ and $h_2(\mathbf{x}) = p_1^{(2)}x_1 + p_2^{(2)}x_2 + r^{(2)}$. (b) When $\mu \to \infty$, the line $h_1(\mathbf{x}) - \mu h_2(\mathbf{x})$ gets closer to the line $h_2(\mathbf{x})$. Thus, $F(x_1, x_2)$ gradually becomes a fan-shaped function, which is a linear function h_1 in the region Ω_1 and 0 outside Ω_1. Then, $F(x_1, x_2)$ will be used to cut the space to obtain a nearly perfect simplex.

Since such a fan-shaped function is a basic building block for our construction of wide and deep equivalent networks, let us explain it in a two-dimensional case for easy visualization. An essential building block expressed by a network in Figure 5(a) is based on the following function:

$$F(\mathbf{x}) = \sigma \circ (h_1(\mathbf{x}) - \mu\sigma \circ h_2(\mathbf{x})), \tag{7}$$

where $h_1(\mathbf{x}) = p_1^{(1)}x_1 + p_2^{(1)}x_2 + r^{(1)}$, and $h_2(\mathbf{x}) = p_1^{(2)}x_1 + p_2^{(2)}x_2 + r^{(2)}$ are provided by two linearly independent vectors $\{(p_1^{(1)}, p_2^{(1)}), (p_1^{(2)}, p_2^{(2)})\}$, and μ is a positive controlling factor. Eq. (7) is a ReLU network of depth=2 and width=2 according to our width-depth definition. As illustrated in Figure 5(b), the support of $F(\mathbf{x})$ contains three boundaries and four polytopes (two of which only allow zero value of F). For convenience, given a linear function $\ell(\mathbf{x}) = c_1x_1 + c_2x_2 + c_3$, we define $\ell^- = \{\mathbf{x} \in \mathbb{R}^2 \mid \ell(\mathbf{x}) < 0\}$ and $\ell^+ = \{\mathbf{x} \in \mathbb{R}^2 \mid \ell(\mathbf{x}) \geq 0\}$. Thus, we can write $\Omega_1 = h_1^+ \cap h_2^-$ and $\Omega_2 = (h_1 - \mu h_2)^- \cap h_2^+$. There are three properties of $F(\mathbf{x})$. First, the common line shared by Ω_1 and Ω_2 is $h_2(\mathbf{x}) = 0$. Second, the size of Ω_2 is adjustable by controlling μ. Note that $h_1(\mathbf{x}) - \mu h_2(\mathbf{x}) = 0$ will be arbitrarily close to $h_2(\mathbf{x}) = 0$ as $\mu \to \infty$, which makes Ω_2 negligible. In the limiting case, the support of $F(\mathbf{x})$ converges to the fan-shaped domain Ω_1. Because $h_1(\mathbf{x}) - \mu h_2(\mathbf{x}) = 0$ is almost parallel to $h_2(\mathbf{x}) = 0$ when μ is large enough, we approximate the area of Ω_2 as the product of the length of $h_2(\mathbf{x}) = 0$ within $[-B, B]^2$ and the distance between two lines, which yields $|\Omega_2| \leq 2\sqrt{2}B/\mu$. Third, the function F over the fan-shaped area Ω_1 is h_1. Linearly aggregating a number of these functions can represent an arbitrary linear function over Ω_1 (see our further elaboration in Figures 6 and 7 below).

What we approximate is a piecewise linear function defined by a ReLU network, not a general continuous function considered in the context of traditional universal approximation. The key is that for a piecewise linear function over polytopes, we do not need to keep dividing polytopes at increasingly finer resolution. We partition each polytope into simplices. The error measure we take is a standard measure in $[-B, B]^D$. The error arises because we cannot perfectly express a function over a fan-shaped function. Figure 5(b) shows what $F(\mathbf{x})$ looks like. Ideally, Ω_2 should not exist so that we can use it to cut the space into a perfect simplex. However, Ω_2 is needed for our construction. Nevertheless, we only need to increase μ to make its measure (area) arbitrarily small, without using more neurons, since the boundary of Ω_2 is adjustable with μ. Figure 6 illustrates how we use two nearly perfect fan-shaped functions to obtain a linear function over a simplex. Although the resultant simplex is inexact, the area of erroneous regions can be controlled to be insignificant. Please read the proof of our main theorem for details.

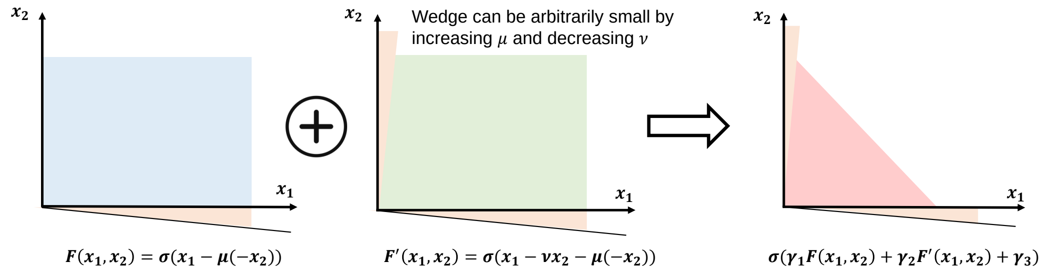
x_2
x_1
$F(x_1, x_2) = \sigma(x_1 - \mu(-x_2))$
x_2
Wedge can be arbitrarily small by increasing μ and decreasing ν
x_1
$F'(x_1, x_2) = \sigma(x_1 - \nu x_2 - \mu(-x_2))$
x_2
x_1
$\sigma(\gamma_1 F(x_1, x_2) + \gamma_2 F'(x_1, x_2) + \gamma_3)$

Figure 6: Combination of two fan-shaped functions leads to a linear function over a simplex. The area of erroneous regions can be arbitrarily small by adjusting the parameters of the neurons (increasing μ and decreasing ν). $\gamma_1, \gamma_2, \gamma_3$ are coefficients of linearly combining $F(x_1, x_2)$ and $F'(x_1, x_2)$.

Remark 1. As a ReLU network of interest partitions the space into more and more polytopes, the number of needed simplices will increase. Because the lower bound of M in Eq. (3) is far larger than D or $(D+1)D^2$, the width of $\mathbf{H}_1(\mathbf{x})$ and the depth of $\mathbf{H}_2(\mathbf{x})$ will dominate. Furthermore, the width of $\mathbf{H}_1(\mathbf{x})$ is greater than its depth by an order of magnitude in terms of M, and the depth of $\mathbf{H}_2(\mathbf{x})$ is greater than its width similarly. Therefore, $\mathbf{H}_1(\mathbf{x})$ is a wide network, and $\mathbf{H}_2(\mathbf{x})$ is a deep network.

Remark 2. Inspired by the network structure used to define the proposed fan-shaped function, we find that intra-layer links can enhance the representation capability of a shallow network, without increasing its width. In the current deep learning research, the dominating network architectures often use shortcuts. However, few studies, if not none, considered the depth separation theory in the shortcut paradigm. Our work reveals that the insertion of intra-layer links can reduce width; *i.e.*, without the need to go as wide as before, a shallow network can express a complicated function as well as a deep network could. Thus, the depth separation theory can be extended to the shortcut paradigm. Along this direction, we have estimated the bound for intra-linked networks; please see Fan et al. (2023) for details.

Remark 3. The network conversion through our proposed transformation is superior to universal approximation in the following two aspects.

First, despite that both our conversion scheme and the universal approximation suffer from the exponential dependency on the ambient space dimension, our scheme is still much more efficient than universal approximation in the sense of finiteness; *i.e.*, **the width and depth of our constructed networks are fixed given a prescribed error, while the width and depth of universal approximation need to increase to decrease the error.** Table 3 summarizes recent key advances in the universal approximation theory of neural networks. To our best knowledge, in all these schemes the network structure will increase as the prescribed error drops. This is because the basic idea in universal approximation to achieve a lower error is to partition the space at finer resolution (hypercubes, hyper-trapezoid, etc.), which demands more neurons and increases width or depth accordingly. However, the goal of our construction is not to accomplish universal approximation but to use a wide or deep ReLU network to approximate an arbitrary ReLU network. Therefore, we can leverage the functional structure of ReLU networks to increase both accuracy and efficiency. Very recently, Shen et al. (2021a) developed a feed-forward neural network for the universal approximation with a fixed number of neurons. However, the employed activation functions ($\sigma_1(x) = |x - [\frac{x+1}{2}]|$, where $[\cdot]$ is a floor function, and $\sigma_2 = \frac{x}{|x|+1}$) are quite unusual, while our work assumes the ReLU function, which is practically dominant.

Second, our transformation between wide and deep ReLU networks takes the intrinsic functional structure into full account, which is a novel approach. Through this lens, a wide network can be converted to an equivalent deep network and vice versa, not only because their output functions can approximate each other sufficiently well but also because they partition the space in the same way. In contrast, the universal approximation scheme only mimics the final output function and not the internal space partition.

Table 3: Relationship between the network structure and the error with recent universal approximation methods, where W and L are the network width and depth respectively, $\epsilon > 0$ is an error, and $w_f(\cdot)$ is the modulus of continuity of f.

Reference	Target Function	Relation
Shen et al. (2021b)	$f \in C([0,1]^D)$	$\epsilon \sim \mathcal{O}\left(2w_f(2\sqrt{D})2^{-W} + w_f(2\sqrt{D}2^{-W})\right)$
Lu et al. (2021)	$f \in C^s([0,1]^D)$	$\epsilon \sim \mathcal{O}\left(\|f\|_{C^s([0,1]^d)} W^{-2s/D} L^{-2s/D}\right)$
Shen et al. (2019)	$f \in C([0,1]^D)$	$\epsilon \sim \mathcal{O}\left(19\sqrt{D} w_f(W^{-2/D} L^{-2/D})\right)$
Yarotsky (2017)	$f \in \mathcal{W}^{n,\infty}([0,1]^D)$	$L \sim \mathcal{O}\left(c(d,n)\mathrm{In}(1/\epsilon)\right)$

3. Discussions

Opportunities through Width-Depth Equivalence. Popular deep learning theories, such as neural tangent kernel (NTK) and neural network Gaussian process (NNGP), were developed assuming an infinite width. Essentially, these theories characterize what will happen when a network becomes wide. However, in the era of deep learning, we are more concerned with how depth affects the behavior of a neural network. Characterizing depth-induced network behaviors is important in deciphering the mechanism of deep learning and shedding light on the interpretability of deep networks. Therefore, depth-oriented theories would be highly desirable. Our work suggests that the width-oriented theories might also hold in a major sense when the width is finite but the depth is infinite, since width and depth are equivalent.

Width-Depth Correlation. Every continuous n-variable function f on $[0,1]^n$ can be in the L_1 sense represented by partially separable multivariate functions (Light and Cheney, 2006):

$$\int_{(x_1,\cdots,x_n)\in[0,1]^n} |f(x_1,\cdots,x_n) - \sum_{l=1}^{L}\prod_{i=1}^{n}\phi_{li}(x_i)| < \epsilon, \tag{8}$$

where ϵ is an arbitrarily small positive number, ϕ_{li} is a continuous function, and L is the number of products. In **Supplementary Information IV**, we justify the suitability of this partially separable representation, and compare it with other representations.

Furthermore, we can correlate the width and depth of a network to the structure of a function to be approximated. In a nutshell, each continuous function ϕ_{li} can be approximated by a polynomial of some degree, which can be appropriately represented by quadratic neurons. As a consequence, via a quadratic representation scheme, the width and depth of a network structure must reflect the structure of $\sum_{l=1}^{L}\prod_{i=1}^{n}\phi_{li}(x_i)$. In other words, they are controlled by the nature of a specific task. As the task becomes demanding, the width and/or depth must be increased accordingly, and the combination of the width and depth is not unique. For more details, please see **Supplementary Information IV**.

4. Conclusion

Inspired by the De Morgan law and through a systematic analysis, we have established the quasi-equivalence between the depth and width of ReLU neural networks. We have formulated two transforms for mapping an arbitrary regression ReLU network to a wide ReLU network and a deep ReLU network respectively. This quasi-equivalence represents a step forward in understanding the relationship between the width and depth of networks. More efforts are needed to refine this quasi-equivalence relationship and find its real-world applications.

Acknowledgments

F. L. Fan is supported by the Rensselaer-IBM AI Research Collaboration Program (`http://airc.rpi.edu`), part of the IBM AI Horizons Network (`http://ibm.biz/AIHorizons`), R. Lai is partially supported by an NSF Career Award DMS–1752934 and NSF DMS-2134168. G. Wang is partially supported by the NIH grants R01CA237267, R01HL151561, R01EB031102, and R01EB032716.

Appendix. Proof of Theorem 10 (2D)

Here, we show the correctness of Theorem 10 in the 2D case. Regarding the transformation of an arbitrary multivariate network, the situation is more complicated than in the case of univariate networks. Nevertheless, we can still establish the δ-equivalence, which is a slightly relaxed result.

The sketch of proof: A ReLU network is a piecewise linear function over polytopes, which can be decomposed into a summation of linear functions over a simplex. **Lemma** 11 below shows that a network module $\mathbf{N}(\mathbf{x})$ can represent an arbitrary linear function over a simplex. Then, in **Theorem** 10, to transform an arbitrary ReLU network into a wide and a deep network, we horizontally aggregate network modules $\mathbf{N}(\mathbf{x})$ to have a wide network, and we use shortcuts to sequentially establish a deep network with $\mathbf{N}(\mathbf{x})$.

A D-simplex S is a D-dimensional convex hull formed by the convex combination of $D+1$ affinely independent vectors $\{\mathbf{v}_i\}_{i=0}^D \subset \mathbb{R}^D$. In other words, $S = \left\{\sum_{i=0}^{D} \xi_i \mathbf{v}_i \mid \xi_i \geq 0, \sum_{i=0}^{D} \xi_i = 1\right\}$. In the 2D case, if we write $V = (\mathbf{v}_1 - \mathbf{v}_0, \mathbf{v}_2 - \mathbf{v}_0)$, then V is invertible, and $S = \{\mathbf{v}_0 + V\mathbf{x} \mid \mathbf{x} \in \Delta\}$, where $\Delta = \left\{\mathbf{x} \in \mathbb{R}^2 \mid \mathbf{x} \geq 0, \mathbf{1}^\top \mathbf{x} \leq 1\right\}$ is a template simplex in $\mathbb{R}^2$. It is clear that the following one-to-one affine mapping between S and Δ exists, which is

$$T : S \rightarrow \Delta, \mathbf{p} \mapsto T(\mathbf{p}) = V^{-1}(\mathbf{p} - \mathbf{v}_0). \tag{9}$$

Therefore, we only need to make the construction in the special case where $S = \Delta$ to simplify our analysis. The coordinate transform in Eq. (9) can conveniently map the construction from Δ to S.

Given a linear function $\ell(\mathbf{x}) = c_1 x_1 + c_2 x_2 + c_3$, we write $\ell^- = \{\mathbf{x} \in \mathbb{R}^2 \mid \ell(\mathbf{x}) < 0\}$ and $\ell^+ = \{\mathbf{x} \in \mathbb{R}^2 \mid \ell(\mathbf{x}) \geq 0\}$. Δ is enclosed by three lines provided by $\ell_1(\mathbf{x}) = x_1$, $\ell_2(\mathbf{x}) = x_2$, and $\ell_3(\mathbf{x}) = -x_1 - x_2 + 1$. We write three vertices of Δ as $\mathbf{v}_0 = (0,0), \mathbf{v}_1 = (1,0), \mathbf{v}_2 = (0,1)$. Then, $f : [-B,\ B]^2 \rightarrow \mathbb{R}$ supported on Δ is expressed as follows:

$$f(\mathbf{x}) = \begin{cases} \mathbf{a}^\top \mathbf{x} + b, & \textbf{if} \quad \mathbf{x} \in \Delta \\ 0, & \textbf{if} \quad \mathbf{x} \in \Delta^c \end{cases}, \tag{10}$$

where $\mathbf{a} = (f(\mathbf{v}_1) - f(\mathbf{v}_0), f(\mathbf{v}_2) - f(\mathbf{v}_0)), b = f(\mathbf{v}_0)$.

Lemma 11 *Suppose that the representation of an arbitrary ReLU network is $f : [-B,\ B]^D \rightarrow \mathbb{R}$ expressed as Eq. (10), for any $\delta > 0$, there exists a ReLU network $\mathbf{N}$ of width $(D+1)D^2$ and depth $D+1$, satisfying that*

$$\mathfrak{m}\Big(\mathbf{x} \mid f(\mathbf{x}) \neq \mathbf{N}(\mathbf{x})\}\Big) < \delta. \tag{11}$$

Proof (**D=2**) Our goal is to approximate the given piecewise linear function f over Δ. Therefore, we need to cancel f outside its domain. We first index the polytopes separated by three lines $\ell_1(\mathbf{x}) = 0, \ell_2(\mathbf{x}) = 0$, and $\ell_3(\mathbf{x}) = 0$ as $\mathcal{A}^{(\chi_1,\chi_2,\chi_3)} = \ell_1^{\chi_1} \cap \ell_2^{\chi_2} \cap \ell_3^{\chi_3}, \chi_i \in \{+,-\}, i = 1,2,3$. It is clear that $\Delta = \mathcal{A}^{(+,+,+)}$. Additionally, we use $\vee$ to exclude a

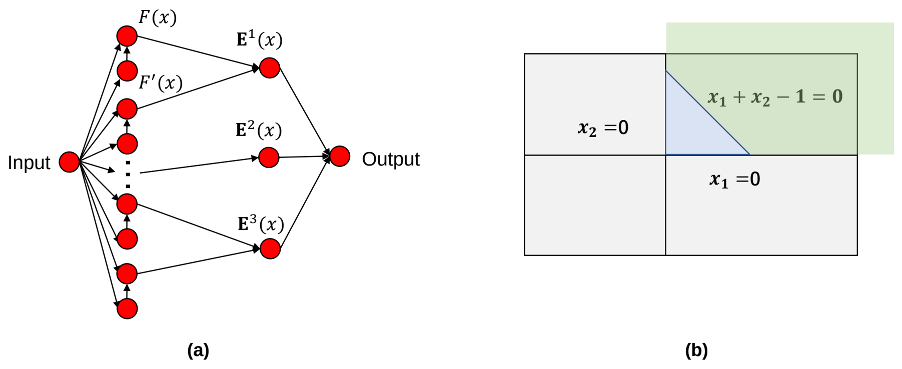

Figure 7: Quasi-equivalence analysis in the 2D case. (a) The structure of the deep network to represent f over Δ. (b) The polytopes outside Δ have three fan-shaped domains, on which f can be canceled by three linearly independent functions over Δ.

component. For instance, $\mathcal{A}^{(\chi_1,\vee,\chi_3)} = \ell_1^{\chi_1} \cap \ell_3^{\chi_3}$. It can be easily verified that $\mathcal{A}^{(\chi_1,\vee,\chi_3)} = \mathcal{A}^{(\chi_1,+,\chi_3)} \cup \mathcal{A}^{(\chi_1,-,\chi_3)}$.

Allowing more layers in a network provides a way to represent f. Let $F(x_1,x_2) = \sigma \circ (x_1 - \mu\sigma \circ (-x_2))$ and $F^{'}(x_1,x_2) = \sigma \circ (x_1 - \nu x_2 - \mu\sigma \circ (-x_2))$, both of which are approximately enclosed by boundaries $x_1 = 0$ and $x_2 = 0$. Therefore, the fan-shaped regions of $F(x_1,x_2)$ and $F^{'}(x_1,x_2)$ almost overlap as ν is small. To obtain the third boundary $\ell_3(\mathbf{x}) = 0$ for building the simplex Δ, we stack one more layer with only one neuron to separate the fan-shaped region of $F(x_1,x_2)$ with the boundary $-x_1 - x_2 + 1 = 0$ as follows:

$$\mathbf{E}^1(\mathbf{x}) = (\gamma_1^* F(\mathbf{x}) + \gamma_2^* F'(\mathbf{x}) + \gamma_3^*)^+, \tag{12}$$

where $(\gamma_1^*, \gamma_2^*, \gamma_3^*)$ are roots of the following system of equations:

$$\begin{cases} \gamma_1 + \gamma_2 = -1 \\ -\nu\gamma_2 = -1 \\ \gamma_3 = 1. \end{cases} \tag{13}$$

Thus, $\mathbf{E}^1(x_1,x_2)$ will represent the function $-x_1 - x_2 + 1$ over Δ and zero in the rest area. The depth and width of $\mathbf{E}^1(x_1,x_2)$ are 3 and 4, respectively. Similarly, due to the employment of the two fan-shaped functions, the area of the erroneous regions is estimated as

$$2\sqrt{2}B\left(\nu + 2/\mu\right). \tag{14}$$

To acquire f over Δ, similarly, we need three linear independent functions as linear independent bases. We modify ℓ_3 slightly to get $\ell_3^{'} = \ell_3 - \tau' x_1$ and $\ell_3^{''} = \ell_3 - \tau'' x_2$. Repeating the procedure described in (1), for $\ell_3^{'}$ we construct the network $\mathbf{E}^2(x_1,x_2)$ that is $\ell_3 - \tau' x_1$ over $\ell_1^+ \cap \ell_2^+ \cap (\ell_3^{'})^+$, while for $\ell_3^{''}$ we construct the network $\mathbf{E}^3(x_1,x_2)$ that is $\ell_3 - \tau' x_1$ over $\ell_1^+ \cap \ell_2^+ \cap (\ell_3^{''})^+$. We set positive numbers τ' and τ'' small enough to have two triangular domains $\ell_1^+ \cap \ell_2^+ \cap (\ell_3')^+$ and $\ell_1^+ \cap \ell_2^+ \cap (\ell_3'')^+$ almost identical with Δ. Moreover, let τ' and τ'' satisfy

$$\begin{bmatrix} -1 & -1-\tau' & -1 \\ -1 & -1 & -1-\tau'' \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \rho_1^* \\ \rho_2^* \\ \rho_3^* \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ b \end{bmatrix}, \tag{15}$$

where $\rho_1^*, \rho_2^*, \rho_3^*$ are solutions. As a consequence, we obtain the deep network (illustrated in Figure 7:

$$\mathbf{N}(\mathbf{x}) = \rho_1^* \mathbf{E}^1(\mathbf{x}) + \rho_2^* \mathbf{E}^2(\mathbf{x}) + \rho_3^* \mathbf{E}^3(\mathbf{x}) \tag{16}$$

that produces f on Δ. The depth and width of the network are 3 and 12. Similarly, the area of the erroneous regions is bounded by

$$2\sqrt{2}B\left(3\nu + \tau' + \tau'' + 6/\mu\right). \tag{17}$$

Therefore, for any $\delta > 0$, if we choose

$$0 < \nu, \tau', \tau'', 1/\mu < \frac{\delta}{2\sqrt{2}B(3+2+6)} = \frac{\delta}{22\sqrt{2}B}, \tag{18}$$

then the constructed network $\mathbf{N}$ will satisfy

$$\mathfrak{m}\left(\{\mathbf{x} \in \mathbb{R}^2 | f(\mathbf{x}) \neq \mathbf{N}(\mathbf{x})\}\right) < \delta. \tag{19}$$

■

Proof (**Theorem 10**, $\mathbf{D} = \mathbf{2}$) The network h is piecewise linear and splits the space into polytopes. It is feasible to employ a number of simplices to represent the polytopes defined by h (Ehrenborg, 2007). Given that M is the minimum number of required simplices, we have

$$h(\mathbf{x}) = \sum_{m=1}^{M} f^{(m)}(\mathbf{x}), \tag{20}$$

where

$$f^{(m)}(\mathbf{x}) = \begin{cases} a_1^{(m)} x_1 + a_2^{(m)} x_2 + b^{(m)}, & \textbf{if} \quad \mathbf{x} \in S^{(m)} \\ 0, & \textbf{if} \quad \mathbf{x} \notin S^{(m)} \end{cases}, \tag{21}$$

and $S^{(m)}$ is the m-th simplex. For the construction of a wide network, we use network modules to represent $f^{(m)}(\mathbf{x})$ and then horizontally aggregate them into a wide network. In contrast, for construction of a deep network, we sequentially express $f^{(m)}(\mathbf{x})$ in terms of a network module without linking them to the input.

Representing f with a wide ReLU network: Lemma 11 suggests that a wide network module $\mathbf{N}$ can generically represent a piece-wise linear function over a template simplex. To represent $f^{(m)}$ over $S^{(m)}$, we need to use Eq. (9) to transform the function from the barycentric coordinate system to the Euclidean coordinate system. Let three vertices of $S^{(m)}$ be $\{\mathbf{v}_0^{(m)}, \mathbf{v}_1^{(m)}, \mathbf{v}_2^{(m)}\}$ and $V^{(m)} = (\mathbf{v}_1^{(m)} - \mathbf{v}_0^{(m)}, \mathbf{v}_2^{(m)} - \mathbf{v}_0^{(m)})$, we have

$$\mathbf{N}_1^{(m)}(\mathbf{x}) = \mathbf{N}((V^{(m)})^{-1}(\mathbf{x} - \mathbf{v}_0^{(m)})), \tag{22}$$

satisfying

$$\mathfrak{m}\left(\{\mathbf{x} \in \mathbb{R}^2 \mid f^{(m)}(\mathbf{x}) \neq \mathbf{N}_1^{(m)}(\mathbf{x})\}\right) < \delta. \tag{23}$$

By aggregating the network $\mathbf{N}_1^{(m)}(\mathbf{x})$ horizontally, we have the following wide network:

$$\mathbf{H}_1(\mathbf{x}) = \sum_{m=1}^{M} \mathbf{N}_1^{(m)}(\mathbf{x}). \tag{24}$$

Therefore, the constructed wide network $\mathbf{H}_1(\mathbf{x})$ is of width $\mathcal{O}(12M)$ and depth 3. It is clear that the width $\mathcal{O}(12M)$ of the wide network $\mathbf{H}_1(\mathbf{x})$ dominates, as the number of needed simplices becomes sufficiently large.

Representing f with a deep ReLU network: For a deep construction, the fundamental difficulty is how to sequentially express each $f^{(m)}$; *i.e.*, the input of each block can only come from the earlier block instead of the input, like what we did in one-dimensional case (Figure 8(a)). Let us derive via induction how to sequentially represent each $f^{(m)}$. We still adopt the idea of modularized networks, but now each network module has two outputs. **Lemma 11** suggests that a network module $\mathbf{N}$ can generically represent a function over a template simplex.

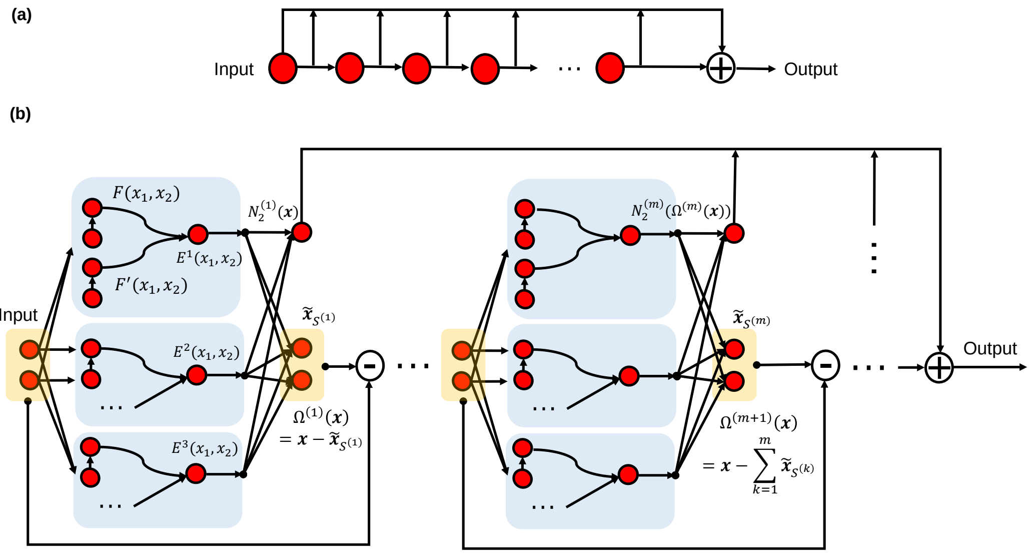
(a)
Input
Output
(b)
$F(x_1, x_2)$
$N_2^{(1)}(\boldsymbol{x})$
$E^1(x_1, x_2)$
$F'(x_1, x_2)$
Input
$\widetilde{\boldsymbol{x}}_{S^{(1)}}$
$E^2(x_1, x_2)$
$\Omega^{(1)}(\boldsymbol{x}) = \boldsymbol{x} - \widetilde{\boldsymbol{x}}_{S^{(1)}}$
$E^3(x_1, x_2)$
$N_2^{(m)}(\Omega^{(m)}(\boldsymbol{x}))$
$\widetilde{\boldsymbol{x}}_{S^{(m)}}$
$\Omega^{(m+1)}(\boldsymbol{x}) = \boldsymbol{x} - \sum_{k=1}^{m} \widetilde{\boldsymbol{x}}_{S^{(k)}}$
Output

Figure 8: Illustration of deep networks in (a) 1D and (b) 2D cases, respectively.

Step 1. Assume that the two outputs of the first block are $\mathbf{N}_2^{(1)}$ and $\Omega^{(1)}$. To represent $f^{(1)}$ over $S^{(1)}$, similarly we need to transform the function from the barycentric coordinate system to the Euclidean coordinate system. Let three vertices of $S^{(1)}$ be $\{\mathbf{v}_0^{(1)}, \mathbf{v}_1^{(1)}, \mathbf{v}_2^{(1)}\}$ and $V^{(1)} = (\mathbf{v}_1^{(1)} - \mathbf{v}_0^{(1)}, \mathbf{v}_2^{(1)} - \mathbf{v}_0^{(1)})$, we have

$$\mathbf{N}_2^{(1)} = \mathbf{N}((V^{(1)})^{-1}(\mathbf{x} - \mathbf{v}_0^{(1)})), \tag{25}$$

which is one output of the first block.

Next, we derive the other output $\Omega^{(1)}(\mathbf{x})$. Note that we are not allowed to use the input directly. Encouraged by the inversion idea in the univariate case, we invert the function domain into the input domain to get a function that is approximately only supported over $S^{(1)}$:

$$\tilde{\mathbf{x}}_{S^{(1)}} = \begin{cases} \mathbf{x}, & \textbf{if} \quad \mathbf{x} \in S^{(1)} \\ 0, & \textbf{if} \quad \mathbf{x} \notin S^{(1)} \end{cases}. \tag{26}$$

To do so, recall that we have $\mathbf{E}^1, \mathbf{E}^2, \mathbf{E}^3$ in constructing $\mathbf{N}_2$, we will have $(\xi_1^*, \xi_2^*, \xi_3^*)$ such that

$$\tilde{\mathbf{x}}_{S^{(1)}} = \xi_1^*\mathbf{E}^1((V^{(1)})^{-1}(\mathbf{x}-\mathbf{v}_0^{(1)}))+\xi_2^*\mathbf{E}^2((V^{(1)})^{-1}(\mathbf{x}-\mathbf{v}_0^{(1)}))+\xi_3^*\mathbf{E}^3((V^{(1)})^{-1}(\mathbf{x}-\mathbf{v}_0^{(1)})) \tag{27}$$

As shown in Figure 8(b), use the residual connection, we compute $\Omega^{(1)}(\mathbf{x}) = \mathbf{x} - \tilde{\mathbf{x}}_{S^{(1)}}$, which is zero over S^1 and $\mathbf{x}$ for other regions. $\Omega^{(1)}(\mathbf{x})$ will be used to feed the next block to construct $f^{(2)}$.

Step 2. Suppose that the two output functions of the m-th block are $\mathbf{N}_2^m(\mathbf{x})$ and $\Omega^{(m)}(\mathbf{x}) = \mathbf{x} - \sum_{i=1}^m \tilde{\mathbf{x}}_{S^{(i)}}$. $\Omega^m(\mathbf{x})$ is fed into the $(m+1)$-th block as the input. Because $S^{(m+1)}$ is outside the domain of $S^{(1)} \cup \cdots \cup S^{(m)}$, with the same technique used in **Step 1**, we construct

$$\mathbf{N}_2^{(m+1)}(\Omega^{(m)}(\mathbf{x})) = \mathbf{N}_2((V^{(m+1)})^{-1}(\Omega^{(m)}(\mathbf{x}) - \mathbf{v}_0^{(1)})), \tag{28}$$

where $\{\mathbf{v}_0^{(m+1)}, \mathbf{v}_1^{(m+1)}, \mathbf{v}_2^{(m+1)}\}$ are three vertices of $S^{(m+1)}$, and $V^{(m+1)} = (\mathbf{v}_1^{(m+1)} - \mathbf{v}_0^{(m+1)}, \mathbf{v}_2^{(m+1)} - \mathbf{v}_0^{(m+1)})$ to express $f^{(m+1)}$ over $S^{(m+1)}$. $\Omega^{(m)}(\mathbf{x})$ is functionally equivalent to $\mathbf{x}$ for two reasons. First, all simplices do not overlap. $\Omega^{(m+1)}(\mathbf{x})$ is $\mathbf{x}$ outside the domain of $S^{(1)} \cup \cdots \cup S^{(m)}$, and hence zero value over $S^{(1)} \cup \cdots \cup S^{(m)}$. Second, w.l.o.g., we assume that $(0,0)$ is outside of all simplices or lies on the boundary of a certain simplex. As such, we have $\mathbf{N}_2^{(m+1)}((0,0)) = 0, \forall m$, and $\mathbf{N}_2^{(m+1)}(\Omega^{(m)}(\mathbf{x}))$ will not erroneously produce a non-zero constant over $S^{(1)} \cup \cdots \cup S^{(m)}$. Also, we obtain a function $\tilde{\mathbf{x}}_{S^{(m+1)}}$ inside the $(m+1)$-th block. Applying the residual operation, we have

$$\Omega^{(m+1)}(\mathbf{x}) = \Omega^{(m)}(\mathbf{x}) - \tilde{\mathbf{x}}_{S^{(m+1)}} = \mathbf{x} - \sum_{i=1}^{m+1} \tilde{\mathbf{x}}_{S^{(i)}}. \tag{29}$$

Lastly, similar to the one-dimensional deep network, we aggregate $\mathbf{N}_2^{(m)}(\Omega^{(m-1)}(\mathbf{x}))$ via shortcut connection to obtain the following deep network:

$$\mathbf{H}_2(\mathbf{x}) = \sum_{m=1}^{M} \mathbf{N}_2^{(m)}(\Omega^{(m-1)}(\mathbf{x})). \tag{30}$$

Therefore, the constructed deep network $\mathbf{H}_2(\mathbf{x})$ is of depth $\mathcal{O}(3M)$ and width 12. Please note that in the above equation, the summation is implemented with shortcuts. It is clear that the depth of $\mathbf{H}_2(\mathbf{x})$ dominates over the width.

■

References

A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 6:52138–60, 2018.

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*, 2016.

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

CL Philip Chen and Zhulin Liu. Broad learning system: An effective and efficient incremental learning system without the need for deep architecture. *IEEE Transactions on Neural Networks and Learning Systems*, 29(1):10–24, 2017.

Hu Chen, Yi Zhang, Mannudeep K Kalra, Feng Lin, Yang Chen, Peixi Liao, Jiliu Zhou, and Ge Wang. Low-dose CT with a residual encoder-decoder convolutional neural network. *IEEE Transactions on Medical Imaging*, 36(12):2524–2535, 2017.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10, 2016.

Lingyang Chu, Xia Hu, Juhua Hu, Lanjun Wang, and Jian Pei. Exact and consistent interpretation for piecewise linear neural networks: A closed form solution. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1244–1253, 2018.

Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. In *Conference on Learning Theory*, pages 698–728. PMLR, 2016.

George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, 2011.

Zhijie Deng, Yinpeng Dong, Shifeng Zhang, and Jun Zhu. Understanding and exploring the network with stochastic architectures. *Advances in Neural Information Processing Systems*, 33, 2020.

Richard Ehrenborg. The perles-shephard identity for non-convex polytopes. Technical report, Technical Report, University of Kentucky, 2007.

Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Conference on Learning Theory*, pages 907–940. PMLR, 2016.

F. Fan and G. Wang. Fuzzy logic interpretation of quadratic networks. *Neurocomputing*, 374:10–21, 2020.

Feng-Lei Fan, Dayang Wang, Hengtao Guo, Qikui Zhu, Pingkun Yan, Ge Wang, and Hengyong Yu. On a sparse shortcut topology of artificial neural networks. *IEEE Transactions on Artificial Intelligence*, 3(4):595–608, 2021.

Feng-Lei Fan, Ze-Yu Li, Huan Xiong, and Tieyong Zeng. Rethink depth separation with intra-layer links. *arXiv preprint arXiv:2305.07037*, 2023.

Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183–192, 1989.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press, Cambridge, MA, US, 2016.

Juncai He, Lin Li, Jinchao Xu, and Chunyue Zheng. Relu deep neural networks and linear finite elements. *arXiv preprint arXiv:1807.03973*, 2018.

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

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, pages 8571–8580, 2018.

Kenji Kawaguchi, Jiaoyang Huang, and Leslie Pack Kaelbling. Effect of depth and width on local minima in deep learning. *Neural Computation*, 31(7):1462–1498, 2019.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387. PMLR, 2016.

Yoav Levine, Noam Wies, Or Sharir, Hofit Bata, and Amnon Shashua. Limits to depth efficiencies of self-attention. *arXiv preprint arXiv:2006.12467*, 2020.

William A Light and Elliot W Cheney. *Approximation theory in tensor product spaces*, volume 1169. Springer, Berlin, Germany, 2006.

Hongzhou Lin and Stefanie Jegelka. Resnet with one-neuron hidden layers is a universal approximator. *Advances in Neural Information Processing Systems*, 31:6169–6178, 2018.

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

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: a view from the width. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6232–6240, 2017.

Hrushikesh N Mhaskar and Tomaso Poggio. Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, 14(06):829–848, 2016.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2924–2932, 2014.

Yookoon Park, Sangho Lee, Gunhee Kim, and David Blei. Unsupervised representation learning via neural activation coding. In *International Conference on Machine Learning*, pages 8391–8400. PMLR, 2021.

E. W. Saad and D. C. Wunsch II. Neural network explanation using inversion. *Neural Networks*, 20(1):78–93, 2007.

Thiago Serra and Srikumar Ramalingam. Empirical bounds on linear regions of deep rectifier networks. In *AAAI*, pages 5628–5635, 2020.

Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. Bounding and counting linear regions of deep neural networks. In *International Conference on Machine Learning*, pages 4558–4566. PMLR, 2018.

Rudy Setiono and Huan Liu. Understanding neural networks via rule extraction. In *IJCAI*, volume 1, pages 480–485, 1995.

Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network approximation characterized by number of neurons. *arXiv preprint arXiv:1906.05497*, 2019.

Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network approximation: Achieving arbitrary accuracy with fixed number of neurons. *arXiv preprint arXiv:2107.02397*, 2021a.

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

Lech Szymanski and Brendan McCane. Deep networks are effective encoders of periodicity. *IEEE Transactions on Neural Networks and Learning Systems*, 25(10):1816–1827, 2014.

Sebastian Thrun. Extracting rules from artificial neural networks with distributed representations. pages 505–512. Morgan Kaufmann Publishers, Burlington, MA, USA, 1995.

Ge Wang. A perspective on deep imaging. *IEEE Access*, 4:8914–8924, 2016.

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

Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1284–1293, 2019.

Huan Xiong, Lei Huang, Mengyang Yu, Li Liu, Fan Zhu, and Ling Shao. On the number of linear regions of convolutional neural networks. In *International Conference on Machine Learning*, pages 10514–10523. PMLR, 2020.

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

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.