Supplementary Information of "Quasi-equivalence of Width and Depth of Neural Networks"

Feng-Lei Fan[1], Rongjie Lai[2], Ge Wang[1]

Contact: fanf2@rpi.edu, lair@rpi.edu, wangg6@rpi.edu

[1] AI-based X-ray Imaging System (AXIS) Lab,
Rensselaer Polytechnic Institute, Troy 12180, NY, USA

[2] Department of Mathematics,
Rensselaer Polytechnic Institute, Troy 12180, NY, USA

October 2, 2020

Contents

I. Width as Related to Neural Tangent Kernel (NTK)

NTK sheds light on the power of a network when the width of this network goes to infinity. Previous results have suggested a link between the neural network and the Gaussian distribution when the network width increases infinitely. In [29, 23, 28, 32, 2], the Gaussian process is shown in the two-layer networks, the deep networks, and the convolutional networks. Weakly-trained networks [2] refer to the networks where only the top layer is trained after other layers are randomly initialized. Let $f(\boldsymbol{\theta}, x) \in \mathbb{R}$ denote the output of the network on input x where $\boldsymbol{\theta}$ denotes the parameters of the network, and $\boldsymbol{\theta}$ is an initialization over $\boldsymbol{\Theta}$. In the above context, training the top layer with the l_2 penalty reduces to the kernel regression:

$$ker^{(F)}(x, x') = \mathop{\mathbb{E}}_{\boldsymbol{\theta} \sim \boldsymbol{\Theta}} \langle f(\boldsymbol{\theta}, x), f(\boldsymbol{\theta}, x') \rangle, \tag{1}$$

where (F) denotes the functional space. On the other hand, an infinite wide network is considered as an over-parameterized network, with the motivation to explain the fact that why over-paramterized networks scale. There were extensive studies on this topic [11, 1, 7]. Notably, these studies indicate that the training renders relatively small changes when the network is sufficiently wide. Such a concentration behavior is justified by NTK in terms of the Gaussian process. The NTK is defined as the following:

$$ker^{(\partial)}(x, x') = \mathop{\mathbb{E}}_{\boldsymbol{\theta} \sim \boldsymbol{\Theta}} \langle \frac{\partial f(\boldsymbol{\theta}, x)}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{\theta}, x')}{\partial \boldsymbol{\theta}} \rangle \tag{2}$$

where (∂) denotes the gradient space. Along this line, K. Huang et al. [21] compared the NTK of the ResNet and the deep chain-like network, and found that the NTK of the chain-like network converges to a non-informative kernel as the depth goes to infinity, while the counterpart of ResNet does not. S. S. Du et al. [9] showed that the width provably matters in networks using the piecewise linear activation.

II. Transformation of an Arbitrary Network

Here, we provide the proof for **Theorem 1** and the experiment that shows the feasibility of the procedures in the proof of **Theorem 2**.

Theorem 1 (Equivalence of Univariate Regression Networks)**.** *Given any ReLU network $f : [-B,\ B] \to \mathbb{R}$ with one dimensional input and output variables. 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 $f(x) = \mathbf{H}_1(x) = \mathbf{H}_2(x), \forall x \in [-B,\ B]$.*

Proof. (**Theorem 1**) Since the function represented by a network $f : [-B,\ B] \to \mathbb{R}$ is piecewise linear, we express $f(x)$ as

$$f(x) = \begin{cases} w^{(0)}(x - x_0) + f(x_0), & x \in [x_0, x_1) \\ w^{(1)}(x - x_1) + f(x_1), & x \in [x_1, x_2) \\ w^{(2)}(x - x_2) + f(x_2), & x \in [x_2, x_3) \\ \quad\vdots & \\ w^{(n)}(x - x_n) + f(x_n), & x \in [x_n, x_{n+1}], \end{cases} \tag{3}$$

where $-B = x_0 < x_1 < x_2 < \cdots < x_n < x_{n+1} = B$, $w^{(i)} = \dfrac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$ to guarantee continuity, and the slopes of neighboring pieces are different; otherwise they can be fused together.

We first construct a wide ReLU network $\mathbf{H}_1(x)$ to represent f. This can be straightforward as follows:

$$\mathbf{H}_1(x) = f(x_0) + \sum_{i=0}^{n} (w^{(i)} - w^{(i-1)})\sigma(x - x_i), \tag{4}$$

where specially $w^{(-1)} = 0$. This network is of width $n + 1$. Now, we verify this claim. Given $x \in [x_j, x_{j+1})$, for some $j \geq 0$, we have the following:

$$\begin{aligned} & f(x_0) + \sum_{i=0}^{n} (w^{(i)} - w^{(i-1)})\sigma(x - x_i) \\ =& f(x_0) + \sum_{i=0}^{j} (w^{(i)} - w^{(i-1)})(x - x_i) \\ =& f(x_0) + w^{(j)}x + \sum_{i=0}^{j-1} w^{(i)}(x_{i+1} - x_i) - w^{(j)}x_j \\ =& f(x_0) + \sum_{i=0}^{j-1} (f(x_{i+1}) - f(x_i)) + w^{(j)}(x - x_j) \\ =& f(x_j) + w^{(j)}(x - x_j). \end{aligned} \tag{5}$$

Next, we construct an equivalent deep network dual to the above wide network. Note that (4) is the same as

$$f(x_0) + \sum_{i=0}^{n} \operatorname{sgn}(w^{(i)} - w^{(i-1)})\sigma\Big(|w^{(i)} - w^{(i-1)}|(x - x_i)\Big), \tag{6}$$

where $\operatorname{sgn}(x < 0) = -1$ and $\operatorname{sgn}(x > 0) = 1$. If we write $R_i(x) = \sigma(|w^{(i)} - w^{(i-1)}|(x - x_i)), i = 0, \cdots, n - 1$, then it is clear that the following recursive relation holds:

$$R_{i+1}(x) = \sigma\left(|w^{(i+1)} - w^{(i)}|\left(\frac{1}{|w^{(i)} - w^{(i-1)}|}R_i(x) - x_{i+1} + x_i\right)\right). \tag{7}$$

Thanks to the recurrent relation, each R_i can accurately represent a small monotonically increasing piece over $[x_i, x_{i+1}]$. We aggregate the outputs of those $n + 1$ pieces in the output neuron to get the desired deep ReLU network $\mathbf{H}_2(x)$:

$$\mathbf{H}_2(x) = f(x_0) + \sum_{i=0}^{n} \operatorname{sgn}(w^{(i)} - w^{(i-1)})R_i(x), \tag{8}$$

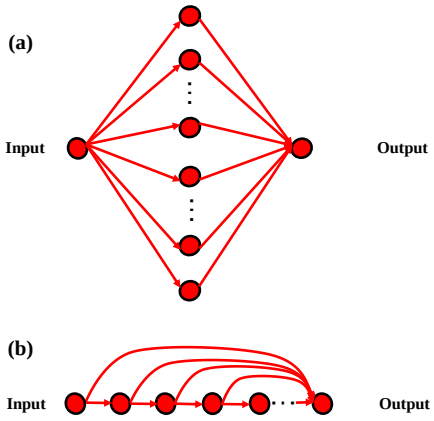

Figure 1: Equivalent wide (a) and deep (b) univariate ReLU networks.

which is the same as $\mathbf{H}_1(x)$. To construct $R_n(x)$, $n+1$ consecutive neurons are connected in series. Therefore, such a one-neuron-wide network has $n+2$ layers. We plot two types of neural networks $\mathbf{H}_1(x)$ and $\mathbf{H}_2(x)$ in **Figure 1**. □

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

$$\begin{aligned} \mathfrak{m}\Big(\{\mathbf{x} \mid h(\mathbf{x}) \neq \mathbf{H}_1(\mathbf{x})\}\Big) < \delta \\ \mathfrak{m}\Big(\{\mathbf{x} \mid h(\mathbf{x}) \neq \mathbf{H}_2(\mathbf{x})\}\Big) < \delta, \end{aligned} \tag{9}$$

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

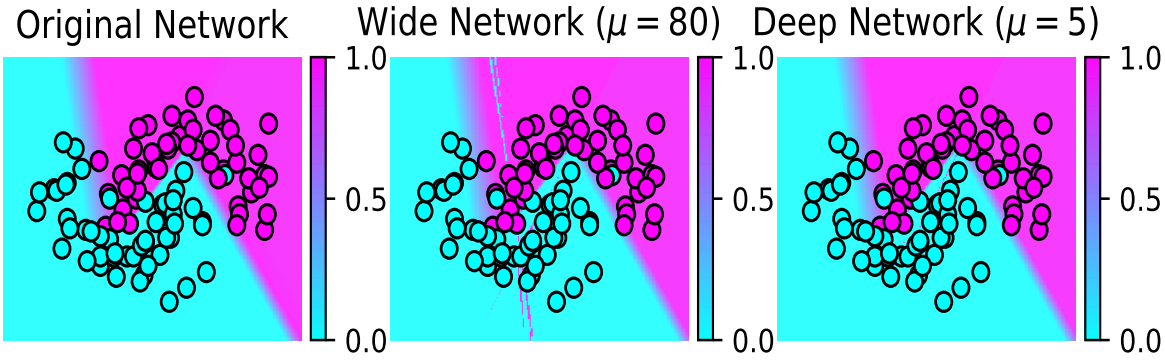

Figure 2: With the procedure in the proof for **Theorem 2**, an exemplary 2-6-2-1 network is transformed into a wide network ($\mu = 80$) and a deep network ($\mu = 5$) respectively.

Experiment: A 2-6-2-1 network was trained on the moon dataset. When the training process was finished, the accuracy of the network was 0.94. Then, we transformed the original network into a wide network ($\mu = 80$) and a deep network ($\mu = 5$) using the procedures described in the proof of **Theorem 2**. The patterns of outputs of the constructed wide network and deep network are respectively shown in **Figure 2**. It is seen that the constructed networks can approximate the original network well, and some expected artifacts can be squeezed by further increasing μ or using a post-processing network that can identify streaks and remove them.

III. Width-Depth equivalence by the De Morgan Law

We have the following formal statement for the quasi-equivalence in light of the De Morgan law. The proof of this theorem is constructive.

Theorem 3 (Quasi-equivalence in light of the De Morgan law)**.** *Given a disjoint rule system $\{A_i\}_{i=1}^n$, where each rule A_i is characterized by an indicator function $g_i(\mathbf{x})$ over a hypercube $\Gamma_i = [a_{1i}, b_{1i}] \times \cdots \times [a_{Di}, b_{Di}] \in [0,1]^D$:*

$$g_i(\mathbf{x}) = \begin{cases} 1, & \textbf{\textit{if}} \quad \mathbf{x} \in \Gamma_i \\ 0, & \textbf{\textit{if}} \quad \mathbf{x} \in \Gamma_i^c, \end{cases}$$

fulfilling the De Morgan law

$$A_1 \vee A_2 \cdots \vee A_n = \neg\Big((\neg A_1) \wedge (\neg A_2) \cdots \wedge (\neg A_n)\Big), \tag{10}$$

we can construct a wide ReLU network $\mathbf{H}_1(\mathbf{x})$ to represent the right hand side of the De Morgan law and a deep ReLU network $\mathbf{H}_2(\mathbf{x})$ of the De Morgan law to represent the left hand side, such that for any $\epsilon > 0$,

$$\mathfrak{m}\Big(\{\mathbf{x} \,|\mathbf{H}_1(\mathbf{x})\} \neq \mathbf{H}_2(\mathbf{x})\}\Big) < \epsilon, \tag{11}$$

where $\mathfrak{m}$ is a measurement in $[0,1]^D$.

We first show our construction for a single-variable binary classification problem as a warm-up, and then we provide a formal proof. Suppose that A_i is a rule defined as the following: IF $(x \in [a_i, b_i])$, THEN class is 1.

Without loss of generality, any two rules are disjoint, which means that any two intervals do not overlap with each other. As shown on the left side of **Figure 3**, the i^{th} network block (the green box) with three layers can represent the following function:

$$\begin{cases} 1 - \frac{a_1 - x}{\delta} & x \in [a_i - \delta, a_i] \\ 1 & x \in [a_i, b_i] \\ 1 - \frac{x - b_i}{\delta} & x \in [b_i, b_i + \delta] \end{cases}, \tag{12}$$

which gives a good approximation to A_i since δ can be made arbitrarily small. The output of the i^{th} block is $A_1 \vee A_2 \cdots \vee A_i$. As the network goes deeper, more blocks are stacked together to include more propositional rules over different intervals. In this process, these rules are integrated to represent $A_1 \vee A_2 \cdots \vee A_n$.

On the other hand, we can use a wide network to construct $\neg\Big((\neg A_1) \wedge (\neg A_2) \cdots \wedge (\neg A_n)\Big)$. As shown on the right side of **Figure 3**, the i^{th} block (the blue box) that represents $\neg A_i$ can be defined as

$$\begin{cases} 1 & x \in [-\infty, a_i - \delta] \\ 1 - \frac{x - a_i}{\delta} & x \in [a_i - \delta, a_i] \\ 0 & x \in [a_i, b_i] \\ 1 - \frac{b_i - x}{\delta} & x \in [b_i, b_i + \delta] \\ 1 & x \in [b_i + \delta, \infty] \end{cases}, \tag{13}$$

Concurrently, the n blocks can represent $\neg A_1, \cdots, \neg A_n$ respectively. When all $\neg A_i$ rules are prepared, with the output neuron we perform the logic intersection and negation. To this end, a summation is conducted of all the inputs before the thresholding operation. After thresholding, a negation is conducted. The threshold is set to $n-1$ for n propositional rules. Specifically, we have

$$\begin{aligned} &\neg\Big((\neg A_1) \wedge (\neg A_2) \cdots \wedge (\neg A_n)\Big) \\ &= 1 - ((\neg A_1) + (\neg A_2) + \cdots + (\neg A_n) - (n-1)). \end{aligned} \tag{14}$$

Proof. As illustrated in **Figures 4 and 5**, we provide the scheme to construct a wide ReLU network $\mathbf{H}_1(\mathbf{x})$ and a deep ReLU network $\mathbf{H}_2(\mathbf{x})$ respectively in light of the De Morgan law based on hypercubes in a high-dimensional space, which is an extension for the univariate results in our manuscript. To represent the rule

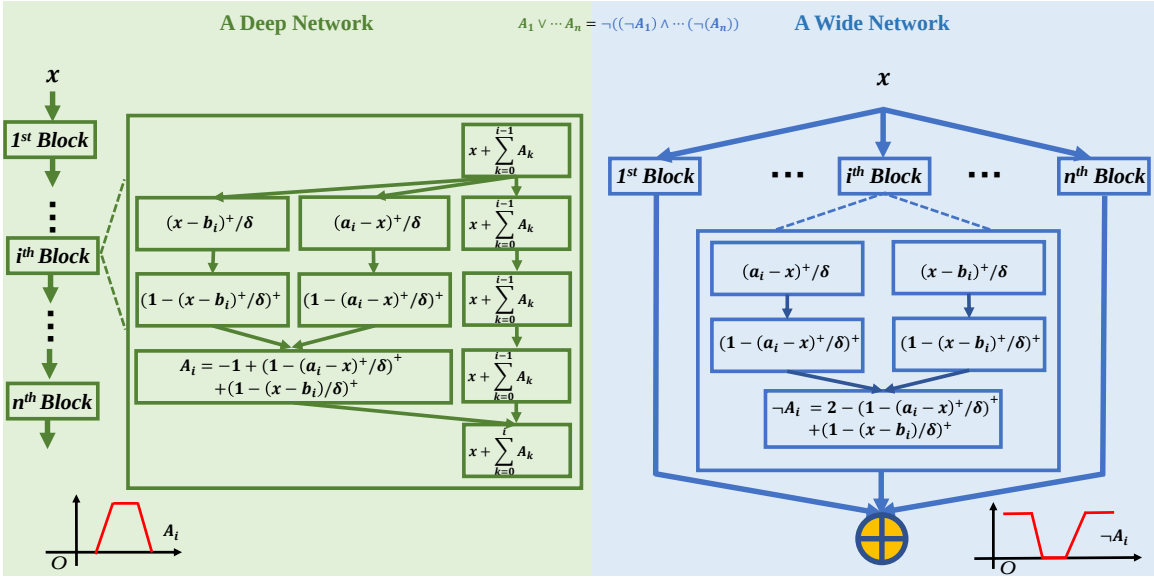

Figure 3: De Morgan equivalence. A deep network to implement $A_1 \vee A_2 \cdots \vee A_n$ using a trapezoid function and a wide version to implement $\neg\Big((\neg A_1) \wedge (\neg A_2) \cdots \wedge (\neg A_n)\Big)$ using the trap-like function. $(\cdot)^+$ denotes ReLU.

A_i that is equivalent to $g_i(\mathbf{x})$, a trapezoid function is built in the deep network, whereas a trap-like function is constructed to represent $\neg A_i$ that is equivalent to $1 - g_i(\mathbf{x})$. In our construction, for each hypercube the measures of $\{\mathbf{x} \in \Gamma_i | \mathbf{H}_1(\mathbf{x}) \neq g_i(\mathbf{x})\}$ and $\{\mathbf{x} \in \Gamma_i | \mathbf{H}_2(\mathbf{x}) \neq g_i(\mathbf{x})\}$ are no more than $vol(\Gamma_i)(1 - 2\delta)^n$, where $vol(\Gamma_i)$ is the volume of a hypercube Γ_i. Therefore, the total measure of errors for all the hypercubes is less than $\sum_i^n vol(\Gamma_i)(1 - 2\delta)^n = (1 - 2\delta)^n \sum_i^n vol(\Gamma_i) < (1 - 2\delta)^n$. As a result, let $\delta < (1 - (\epsilon/2)^{1/n})/2$, we have

$$\begin{aligned} \mathfrak{m}\Big(\{\mathbf{x} \,|\mathbf{H}_1(\mathbf{x})\} \neq \sum_{i=1}^{n} g_i(\mathbf{x})\}\Big) < \epsilon/2 \\ \mathfrak{m}\Big(\{\mathbf{x} \,|\mathbf{H}_2(\mathbf{x})\} \neq \sum_{i=1}^{n} g_i(\mathbf{x})\}\Big) < \epsilon/2, \end{aligned} \tag{15}$$

which lead to

$$\mathfrak{m}\Big(\{\mathbf{x} \,|\mathbf{H}_1(\mathbf{x})\} \neq \mathbf{H}_2(\mathbf{x})\}\Big) < \epsilon. \tag{16}$$

□

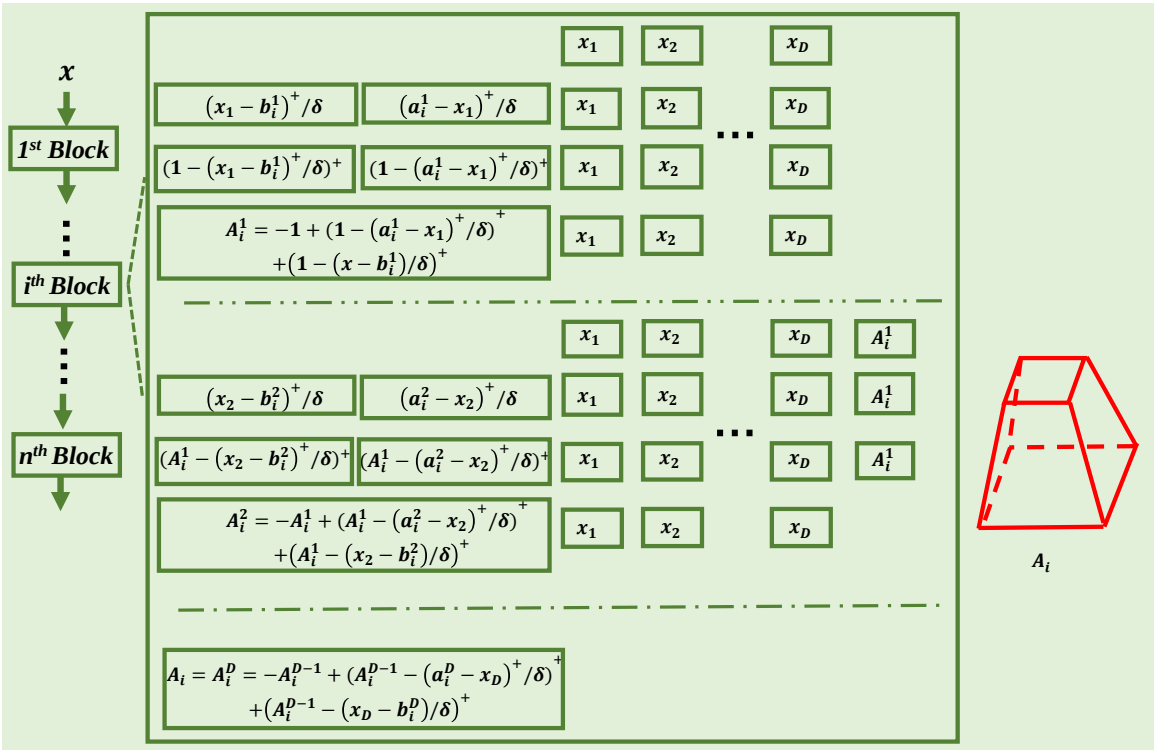

Figure 4: A deep network to realize the logic union of propositional rules in a high-dimensional space. Each propositional rule is associated with a hypercubic indicator function.

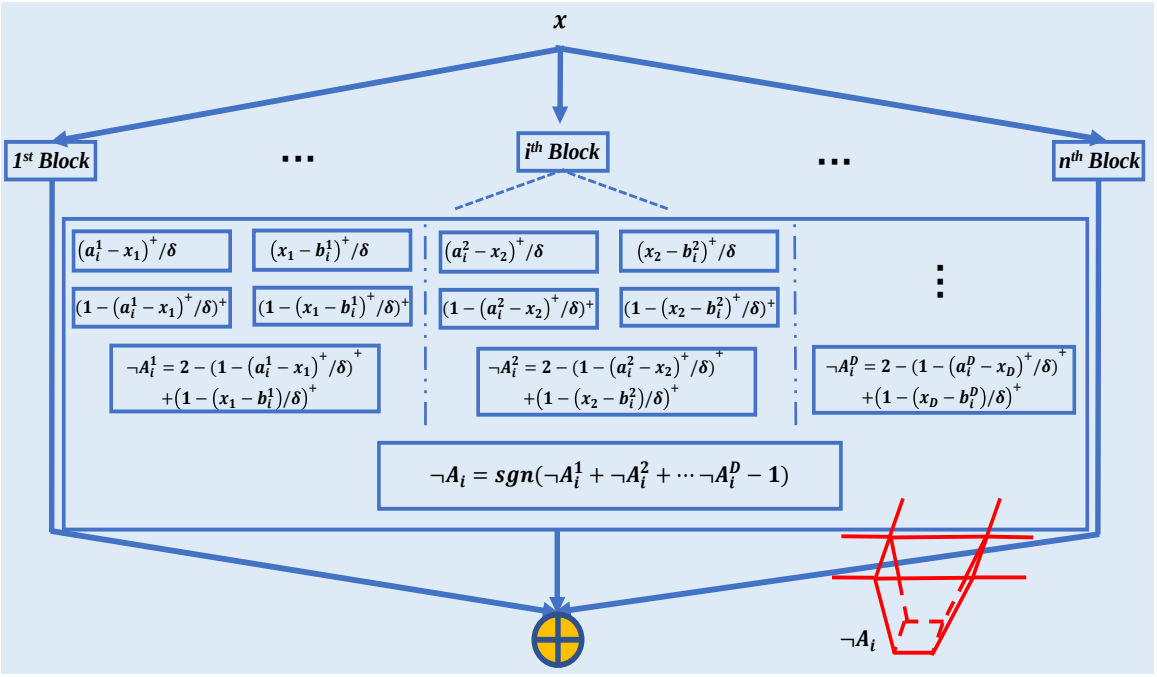

Figure 5: A wide network to realize the negation of the logic intersection of the negation of propositional rules in a high-dimensional space. The negation of each propositional rule is associated with a trap-like indicator function.

IV. Re-expressing the De Morgan Law

Let us re-express the De Morgan law. We partition the set $\{1, 2, \cdots, n\}$ into two groups of disjoint subsets $S_1, \cdots, S_P$ and $T_1, \cdots, T_Q$ satisfying $S_1 \cup \cdots \cup S_P \cup T_1 \cup \cdots \cup T_Q = \{1, 2, \cdots, n\}$, then we can re-express the De Morgan law as

$$\begin{aligned} & A_1 \vee A_2 \cdots \vee A_n \\ &= \sum_{p=1}^{P} \Big(\bigvee_{\alpha \in S_p} A_\alpha \Big) + \sum_{q=1}^{Q} \Big(\bigvee_{\beta \in T_q} A_\beta \Big) \\ &= \sum_{p=1}^{P} \neg \Big(\bigwedge_{\alpha \in S_p} (\neg A_\alpha) \Big) + \sum_{q=1}^{Q} \Big(\bigvee_{\beta \in T_q} A_\beta \Big) \\ &= \neg \Big((\neg A_1) \wedge (\neg A_2) \cdots \wedge (\neg A_n) \Big), \end{aligned} \tag{17}$$

which suggests the step-wise continuum of mutually equivalent networks, comprising not only a deep network and a wide network shown in **Figure 3** but also the transitional networks between them. In order to express $\sum_{p=1}^{P} \neg \Big(\bigwedge_{\alpha \in S_p} (\neg A_\alpha) \Big) + \sum_{q=1}^{Q} \Big(\bigvee_{\beta \in T_q} A_\beta \Big)$, a generic network is constructed with n blocks, each block representing either $\neg A_\alpha$ or A_β to form neural networks in various combinations of width and depth settings governed by the re-expressed De Morgan law.

To illustrate the mutual equivalency of the above scheme, the eight network architectures were constructed, including a deep network, a wide network and others between them as shown in **Figure 6**. In reference to **Figure 3**, each green box contains four layers with three neurons in each layer while each blue box contains three layers with two neurons per layer. The breast cancer Wisconsin dataset was selected as an example to perform our experiment, covering 30 attributes reflecting two pathological features categories: "Benign" and "Malignant". In our demonstration, we only used five attributes (radius, texture, perimeter, area, smoothness) to show the equivalence of networks. We randomly initialized all the parameters of each network configurations 20 times. The mean classification accuracy and the corresponding standard deviation of each network are tabulated in **Table 1**. The takeaway from **Table 1** is that the performance difference between any two networks is no more than 0.0102, which means that eight networks performed rather comparably. We did the t-test for experimental results from any pair of the networks, and the lowest p-value 0.0517 is from the results of networks II and V, which is not sufficient to reject that the networks II and V are equivalent. Such similarity suggests the existence of equivalent networks. In this regard, equivalence between a wide network and a deep network becomes a special case of a family of equivalent networks in the light of Boolean algebra. More experiment details are as follows.

Table 1: equivalence of different architectures.

Architecture	Mean	Standard Deviation
I	0.9264	0.0069
II	0.9293	0.0073
III	0.9300	0.0081
IV	0.9285	0.0079
V	0.9192	0.0117
VI	0.9280	0.0074
VII	0.9237	0.0103
VIII	0.9267	0.0101

Wisconsin Breast Cancer Dataset: This database is on UCI Machine Learning Repository, and can be also found in the SciPy library. In this dataset, ten real-valued features are calculated for each cell nucleus: a) radius (mean distance from center to points on the perimeter), b) texture (standard deviation of gray-scale values), c) perimeter, d) area, e) smoothness (local variation in radius lengths), f) compactness ($perimeter^2/area - 1.0$), g) concavity (severity of concave portions of the contour), h) concave points (number of concave portions of the contour), i) symmetry, j) fractal dimension ("coastline approximation" - 1). The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features.

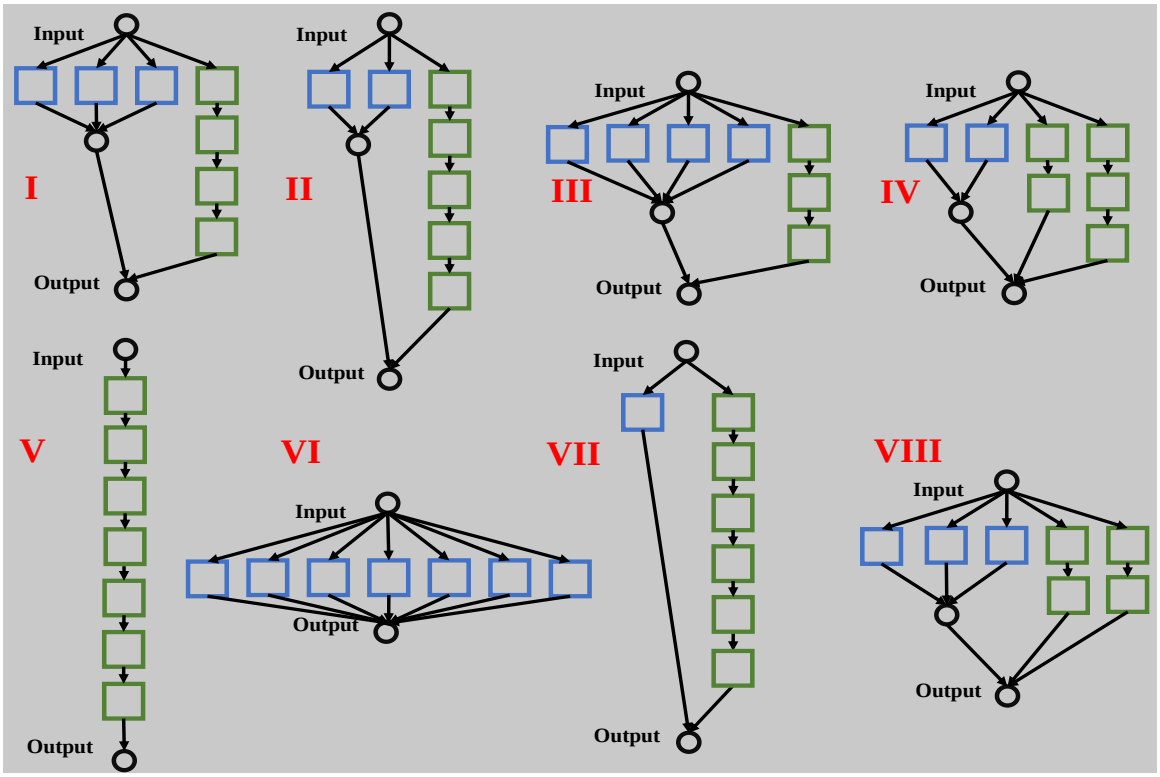

Figure 6: By re-expressing the De Morgan law, a family of equivalent network architectures comprising of a deep network, a wide network, and others between them are evaluated for breast cancer analysis in the example.

In total, 569 samples are included in this dataset, where 357 samples are benign while 212 samples are malignant.

Network Training: The eight network architectures in the abstracted view were constructed, including a deep network, a wide network and others between them, as shown in **Figure 6**. Each green box contains four layers with three neurons per layer while each blue box contains three layers with two neurons per layer. Taking the network IV be an example, **Figure** 7 shows its exact topological structure. From the breast cancer Wisconsin dataset, we only used five attributes (radius, texture, perimeter, area, smoothness) to show the equivalence of networks. The reason behind is that if we utilize all the attributes for classification, the problem is much easier, but it is tricky to discriminate if the similar performance metrics are due to the network equivalence or the saturation effect. The cross-entropy loss was utilized for a classification task, which is mathematically formulated as $-\sum_i (y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)$, where y_i is the ground truth label and $\hat{y}_i$ is the predicted digit. We randomly initialized all the parameters of each network configurations 20 times using the Gaussian distribution with mean of 0 and variance of 0.8. The network was optimized using Adam algorithm with a batch size of 569 for each iteration. The total number of iteration was set to 100,000 with the learning rate=0.0001 on an Intel CORE i7 CPU.

Evaluation: The classification accuracy of the eight networks are summarized in **Table 2**. According to the computation for the data, the mean performance difference between any two networks is no more than 0.0102, which suggests that the eight networks perform similarly. For rigor, we conducted the independent samples t-test, and the obtained p-values between results of any two networks are shown in **Table 3**. The null hypothesis is that data in two groups are from populations with equal means, without assuming that the populations also have equal variances. The lowest p-value 0.0517 is from the results of networks II and V, which is not sufficient to reject that the networks II and V behave differently. For the most of p-values, they are much larger than what suffices not to reject the null hypothesis. Such similarities suggest the existence of equivalent networks. In this regard, equivalence between a wide network and a deep network becomes a special case of a family of equivalent networks

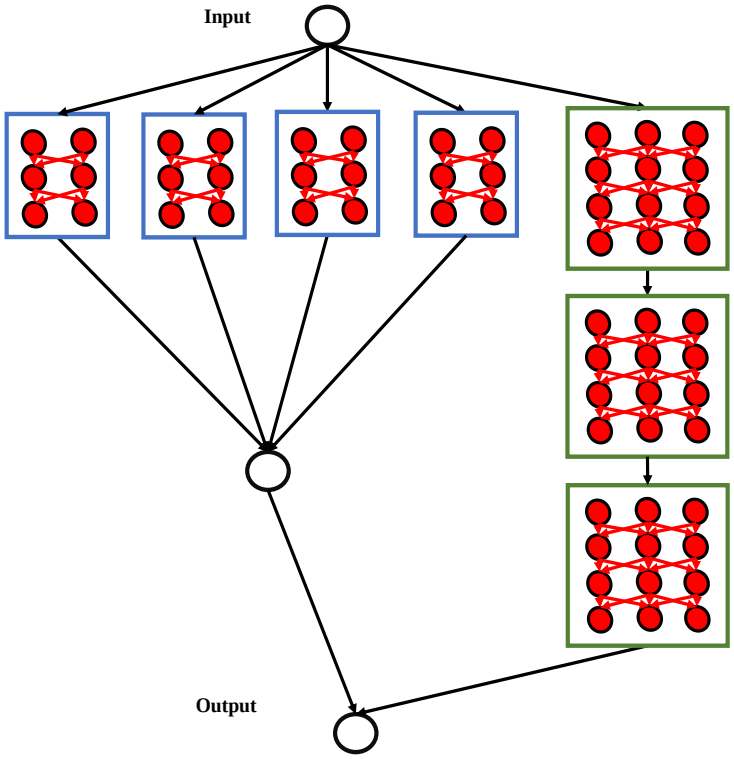

Figure 7: Structural details of the network IV. Each green box contains four layers with three neurons per layer while each blue box contains three layers with two neurons per layer.

in the light of Boolean algebra.

The code for this experiment was uploaded on our GitHub repository https://github.com/FengleiFan/Duality. The purpose of this experiment is to verify the network equivalence numerically.

Table 2: The classification accuracy of the eight networks in 20 random initializations.

	I	II	III	IV	V	VI	VII	VIII
1	0.920914	0.931459	0.926186	0.936731	0.919156	0.920914	0.931459	0.945518
2	0.920914	0.940246	0.927944	0.922671	0.920914	0.927944	0.920914	0.945518
3	0.938489	0.922671	0.920914	0.922671	0.922671	0.922671	0.912127	0.919156
4	0.920914	0.936731	0.920914	0.933216	0.920914	0.929701	0.920914	0.917399
5	0.926186	0.922671	0.936731	0.945518	0.917399	0.926186	0.931459	0.920914
6	0.919156	0.922671	0.929701	0.924429	0.922671	0.927944	0.926186	0.949033
7	0.926186	0.924429	0.926186	0.920914	0.938489	0.924429	0.922671	0.920914
8	0.942003	0.920914	0.920914	0.936731	0.922671	0.920914	0.903339	0.933216
9	0.922671	0.927944	0.929701	0.922671	0.927944	0.920914	0.906854	0.938489
10	0.920914	0.931459	0.929701	0.933216	0.931459	0.934974	0.924429	0.926186
11	0.922671	0.920914	0.940246	0.926186	0.922671	0.926186	0.927944	0.927944
12	0.929701	0.933216	0.922671	0.922671	0.908611	0.938489	0.940246	0.920914
13	0.920914	0.936731	0.927944	0.919156	0.945518	0.917399	0.927944	0.920914
14	0.922671	0.920914	0.934974	0.940246	0.933216	0.922671	0.936731	0.917399
15	0.929701	0.936731	0.919156	0.934974	0.920913	0.943761	0.919156	0.919156
16	0.920914	0.936731	0.936731	0.924429	0.919156	0.927944	0.933216	0.919156
17	0.927944	0.920914	0.922671	0.926186	0.926186	0.920914	0.906854	0.924429
18	0.922671	0.922671	0.931459	0.919156	0.929701	0.942003	0.922671	0.920914
19	0.938489	0.933216	0.922671	0.936731	0.910369	0.931459	0.936731	0.920914
20	0.933216	0.942003	0.922671	0.920914	0.922671	0.933216	0.922671	0.926186

Table 3: The p-values between experimental results of all the paired networks.

	I	II	III	IV	V	VI	VII	VIII
I		0.2048	0.5823	0.3734	0.3790	0.4633	0.3483	0.8984
II			0.4168	0.7448	0.0517	0.6003	0.0588	0.3679
III				0.6684	0.1676	0.8073	0.1692	0.7668
IV					0.1083	0.8568	0.1113	0.5440
V						0.1364	0.8847	0.3967
VI							0.1381	0.6405
VII								0.3607

V. Bounds for the Partially Separable Representation

The purpose of this section is to show that the partially separable function can be realized a quadratic network whose structure is bounded. Let us first introduce necessary preliminaries.

Quadratic Neuron [13, 12, 14]**:** The operation integrating n input variables to a quadratic/second-order neuron before being nonlinearly processed is expressed as

$$\begin{aligned} h(\mathbf{x}) &= (\sum_{i=1}^{n} w_{ir}x_i + b_r)(\sum_{i=1}^{n} w_{ig}x_i + b_g) + \sum_{i=1}^{n} w_{ib}x_i^2 + c \\ &= (\mathbf{w}_r\mathbf{x}^T + b_r)(\mathbf{w}_g\mathbf{x}^T + b_g) + \mathbf{w}_b(\mathbf{x}^2)^T + c, \end{aligned} \tag{18}$$

where $\mathbf{x}$ denotes the input vector, $\mathbf{w_r}, \mathbf{w_g}, \mathbf{w_b}$ are vectors of the same dimensionality as that of $\mathbf{x}$, and b_r, b_g, c are biases. Our quadratic function definition only utilizes $3n$ parameters, which is sparser than the general second-order representation which requires $\frac{n(n+1)}{2}$ parameters. Hereafter, we will use quadratic networks to refer networks using quadratic neurons.

Algebraic Structure: Any univariate polynomial of degree N can be perfectly computed by a quadratic ReLU network with the depth of $\log_2(N)$ and width of no more than N [15].

Partially Separable Representation: Approximating a multivariate function $f(x_1, x_2, \cdots, x_n)$ by a set of functions of fewer variables is a basic problem in approximation theory [26]. Despite that some function f is directly separable in the form of

$$f(x_1, x_2, \cdots, x_n) = \phi_1(x_1)\phi_2(x_2)\cdots\phi_n(x_n), \tag{19}$$

a more general formulation to express a multivariate function f is

$$f(x_1, x_2, \cdots, x_n) = \sum_{l=1}^{L}\prod_{i}^{n}\phi_{li}(x_i). \tag{20}$$

If L is permitted to be sufficiently large, then such a model is rather universal. For bivariate functions, an inspiring theorem has been proved [26]: Let $\{u_n\}_{n\in\mathcal{N}}$ and $\{v_n\}_{n\in\mathcal{N}}$ are orthornomal bases of $L^2(\mathbf{X})$ and $L^2(\mathbf{Y})$ respectively, then $\{u_m v_n\}_{(m,n)\in\mathcal{N}^2}$ is an orthornomal basis of $L^2(\mathbf{X}\times\mathbf{Y})$.

To approximate a general multivariate function, we relax the restrictive equality to the approximation in the L_1 sense and assume that, for every continuous n-variable function $f(x_1, \cdots, x_n)$ on $[0,1]^n$, given any positive number ϵ, there exists a group of ϕ_{li}, satisfying:

$$\int_{(x_1,\cdots,x_n)\in[0,1]^n} |f(x_1, \cdots, x_n) - \sum_{l=1}^{L}\prod_{i=1}^{n}\phi_{li}(x_i)| < \epsilon. \tag{21}$$

The key to demonstrate boundedness of the partially separable representation with a quadratic network is to use the Taylor's expansion, and estimate the remainder, leading to a realization of the partially separable representation. Based on such a realization, we obtain an upper bound of the needed width and depth for the partially separable representation. Let $\partial^{\boldsymbol{\alpha}} f(\mathbf{x}) = \frac{\partial^{\alpha_1}}{\partial x_1}\frac{\partial^{\alpha_2}}{\partial x_2}\cdots\frac{\partial^{\alpha_n}}{\partial x_n} f(\mathbf{x})$, $\boldsymbol{\alpha}! = \prod_{i=1}^{n}\alpha_i!$, and $\mathbf{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1}\cdots x_n^{\alpha_n}$, for the function class $\mathcal{F}^{n,k} = \{f \in C^{k+1}([0,\ 1]^n) \mid \|\partial^{\boldsymbol{\alpha}} f\|_\infty \le M, \quad \forall|\boldsymbol{\alpha}| \le k\}$, the following theorem holds.

Theorem 4. *For any $f \in \mathcal{F}^{n,k}$, if $\mathbf{x} \in [0,1]^n$, there exists a quadratic network $\mathcal{Q}(\mathbf{x})$ with the width no more than $k\binom{n+k}{n}$ and the depth no more than $\log_2(kn)$ to represent f in the partially separable representation, satisfying that*

$$\sup_{\mathbf{x}\in[0,1]^n} |f(\mathbf{x}) - \mathcal{Q}(\mathbf{x})| \le \frac{M}{(k+1)!}n^{k+1}. \tag{22}$$

Proof. Let $f \in \mathcal{F}^{n,k}$, if $\mathbf{x} \in [0,1]^n$, then based on classical multivariate calculus [34],

$$f(\mathbf{x}) = \sum_{|\boldsymbol{\alpha}|\le k}\frac{\partial^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}f(\mathbf{0})\mathbf{x}^{\boldsymbol{\alpha}} + R_{\mathbf{x},k}(\mathbf{x}), \tag{23}$$

where $R_{\mathbf{x},k}(\mathbf{x})$ is the remainder term. For any $\mathbf{x} \in [0,1]^n$, we have

$$R_{\mathbf{x},k}(\mathbf{x}) \leq \frac{M}{(k+1)!}(|x_1| + |x_2| + \cdots + |x_n|)^{k+1} \leq \frac{M}{(k+1)!} n^{k+1}. \tag{24}$$

It is observed from Eqs. 23 and 24 that, given $f \in \mathcal{F}^{n,k}$, the Taylor expansion forms a partially separable representation $\sum_{|\boldsymbol{\alpha}|\leq k} \frac{\partial^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} f(\mathbf{0})\mathbf{x}^{\boldsymbol{\alpha}}$ with $\binom{n+k}{n}$ products, and the error is no more than $\frac{M}{(k+1)!} n^{k+1}$. Along this line, we apply the quadratic network to express $\sum_{|\boldsymbol{\alpha}|\leq k} \frac{\partial^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} f(\mathbf{0})\boldsymbol{x}^{\boldsymbol{\alpha}}$. Any univariate polynomial of degree N can be computed by a quadratic network using the ReLU function with the depth of $\log_2(N)$ and the width of N. To approximate a general term $\frac{\partial^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} f(\mathbf{0})\boldsymbol{x}^{\boldsymbol{\alpha}} = \frac{\partial^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} f(\mathbf{0}) x_1^{\alpha_1} \cdots x_n^{\alpha_n}$, the depth of the quadratic network is no more than $\log_2(\max\{\alpha_1, \cdots, \alpha_n\}) \leq \log_2 k$, and the width is no more than $\sum_{i=1}^n \alpha_i = k$. In addition, the depth of a network that is used to express the product of n terms is $\log_2 n$. Therefore, the depth is no more than $\log_2 k + \log_2 n = \log_2 (kn)$. In contrast, the width is no more than $k\binom{n+k}{n}$, considering that there are $\binom{n+k}{n}$ terms in $\sum_{|\boldsymbol{\alpha}|\leq k} \frac{\partial^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} f(\boldsymbol{x})\boldsymbol{x}^{\boldsymbol{\alpha}}$. □

VI. Properties of the Partially Separable Representation

Let us compare the properties of the partially separable representation with those of the piecewise polynomial representation and the Kolmogorov–Arnold representation in terms of universality, finiteness, decomposability and smoothness. As a result, the partially separable representation is well justified.

1. Universality: The representation should have a sufficient ability to express the functions. A piecewise polynomial representation is capable of representing any continuous function in a local way. The Kolmogorov–Arnold representation theorem states that every multivariate continuous function can be represented as a combination of continuous univariate functions, which solves a generalized Hilbert thirteen problem. Therefore, the Kolmogorov–Arnold representation also possesses universality. As far as the partially separable representation is concerned, its universality in the L_1 distance is analyzed in the above Taylor expansion analysis. In summary, these three representation classes are all powerful enough to express continuous functions.

2. Finiteness: The multivariate Taylor expansion can approximate a general continuous function $f \in \mathcal{F}^{n,k}$ with a finite number of terms. Although the proof in the manuscript devises a specific strategy of obtaining a partially separable representation, the partially separable representation actually allows a variety of constructions in addition to the Taylor expansion. There is no fundamental reason that the partially separable representation will necessarily have exponentially many terms in a majority of practical tasks after reasonable assumptions are incorporated.

3. Decomposability: Decomposability is to decode the functionality of the model in terms of its building units, i.e., cells, layers, blocks, and so on. A plethora of engineering examples, such as software development and optical system design, have shown that modularized analysis is effective. Particularly, modularizing a neural network is advantageous for the model interpretability. For example, X. Chen et al. [8] developed InfoGAN to enhance decomposability of features learned by GAN [20], where InfoGAN maximizes the mutual information between the latent code and the observations, encouraging the use of noise to encode the semantic concept. Since the partially separable representation is more decomposable than its counterparts, the network constructed in reference to the partially separable representation is easily interpretable. For instance, such a network has simpler partial derivatives that can simplify interpretability analysis. [27].

4. Smoothness: A useful representation should be as smooth as possible such that the approximation can suppress high-frequency oscillations and be robust to noise. In the last eighties, Girosi and Poggio claimed that the use of the Kolmogorov-Arnold representation is doomed because the inner functions and the outer functions are highly non-smooth [17]. In the piecewise polynomial representation, the situation gets better since at least over each interval the representation is smooth. As far as the partially separable representation is concerned, because the expression is greatly relaxed (from the exact computation to approximate representation), it is feasible to make each element smooth as L goes large, and extract meaningful structures underlying big data.

Table 4: Comparison of the three functional representations, where "✓" means "yes", "X" means "no", and "–" means "mediocre".

Representation	Universality	Finiteness	Decomposability	Smoothness
Piecewise Polynomial	✓	X	X	–
Kolmogorov-Arnold	✓	✓	–	X
Partially Separable	✓	–	✓	✓

In summary, we compare the three representation schemes in terms of universality, finiteness, decomposability, and smoothness in **Table 4**. Clearly, the partially separable representation is suitable for machine learning tasks.

VII. Width-Depth Correlation Through Partially Separable Presentation

Now, let us analyze the complexity of the quadratic network in light of the aforementioned partially separate representation scheme as shown in **Figure 8**. Suppose that the polynomial P_{li} is of degree N_{li}, then the representation of each P_{li} can be done with a network of width $\sum_{l=1}^{L}\sum_{i=1}^{n} N_{li}$ and depth $\max_{l,i}\{\log_2(N_{li})\}$. Next, the multiplication demands an additional network of width Ln and depth $\log_2(n)$. Therefore, the overall quadratic network architecture will be of width $\max\{\sum_{l=1}^{L}\sum_{i=1}^{n} N_{li}, Ln\}$ and depth $\max_{l,i}\{\log_2(N_{li})\} + \log_2(n)$. Because the depth scales with a log function, which changes slowly when the dimensionality of the input is large. For simplicity, we take an approximation for depth $\max_{l,i}\{\log_2(N_{li})\}+\log_2(n) = \log_2(\max_{l,i}\{N_{li}\})+\log_2(n) \approx \alpha\log_2\sum_{l=1}^{L}\sum_{i=1}^{n} N_{li} + \log_2(n)$, where α is a positive constant. Let $\sum_{l=1}^{L}\sum_{i=1}^{n} N_{li} = N^{\Sigma}$, which describes the overall complexity of the function to be expressed, then the formulas to compute the width and depth are simplified as follows:

$$\begin{aligned} Width &= \max\{N^{\Sigma}, Ln\} \\ Depth &= \alpha log_2(N^{\Sigma}) + log_2(n). \end{aligned} \tag{25}$$

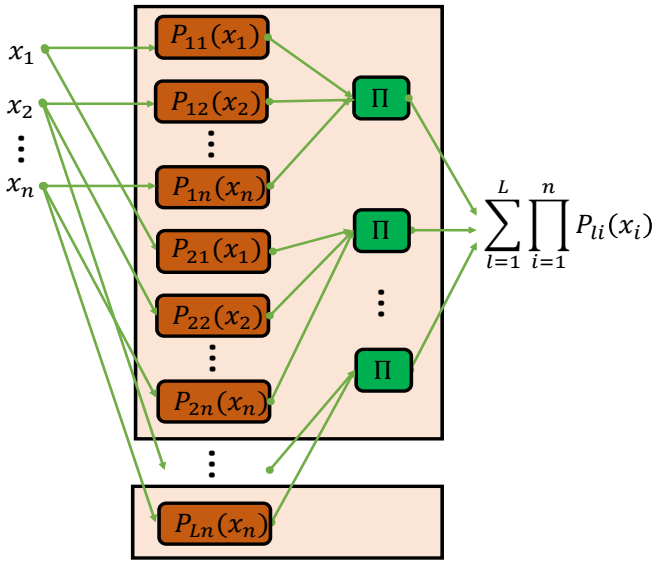

Figure 8: Use of a quadratic network to represent a partially separable representation. Suppose that the polynomial P_{li} is of degree N_{li}, the width and depth of the quadratic network to approximate P_{li} are N_{li} and $\log_2(N_{li})$ respectively.

One interesting point from (25) is that the lower bounds for depth and width to realize a partially separable representation are also suggested. As shown in **Figure 9**, we plot the width and depth as N^{Σ} changes. There are two highlights in **Figure 9**. The first is that the width is generally larger than the depth, which is different from the superficial impression on deep learning. The second is that, as the N^{Σ} goes up, the width/depth ratio is accordingly increased.

Table 5: Descriptions of different building blocks.

Modules	Degree	Operation	Width	Depth
P_{li}	N_{li}	Express ϕ_{li}	N_{li}	$\log_2(N_{li})$
Π	–	Express Φ_i	n	$\log_2(n)$

Remark: Through the complexity analysis, we realize that the width and depth of a network depend on the structure or complexity of the function to be approximated. In other words, they are controlled by the nature

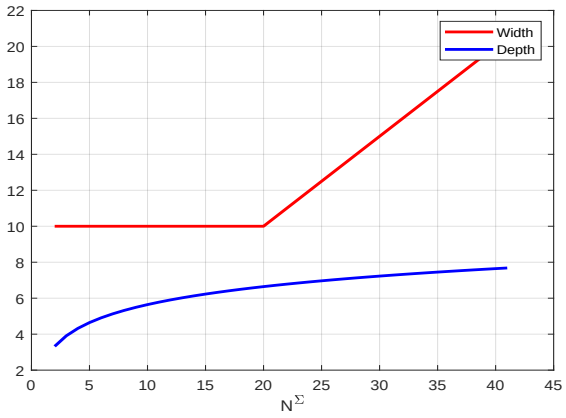

Figure 9: Width and depth versus N^{Σ} changes ($L = 4$, $n = 5$, and $\alpha = 1$ without loss of generality) assuming the partially separable representation.

of a specific task. As the task becomes complicated, the width and depth must increase accordingly, and the combination of the width and depth is not unique.

VIII. Effects of Width on Optimization, Generalization and VC dimension

We first illustrate the importance of width on optimization in the context of over-paramterization, kernel ridge regression, and NTK, and then report our findings that the existing generalization bounds and VC dimension results somehow suggest the width and depth equivalence for a given complexity of networks.

1. Optimization: The optimization mechanism is the key to understand the training process of neural networks [16]. From the view of optimization, the fact that randomly-initialized first-order methods can find well-generalizable minima on the training data is quite intriguing. Given the theme of this paper, we put our endeavor into the importance of width on the optimization of neural networks in hope to provide insights for practitioners. We divide the relevant literature into the three categories:

(1) Increase width for over-parameterization: Brutzkus et al. showed that a wide two-layer network using the hinge loss can generalize well on linearly separable data with stochastic gradient descent (SGD) [6]. Li and Liang showed that when data is normalized and fulfills a separability condition, a two-layer over-parameterized network can learn these data in a polynomial time [25]. Allen-Zhu et al. [1] showed that if the width of hidden layers is $\mathcal{O}\Big(n^{30}D^{30}\log^{30}(1/\epsilon)\Big)$, where n is the number of samples, D is the network depth, and ϵ is an expected error, then the gradient descent search converges with the error ϵ. This bound was further reduced to $n^4 2^D$ for the network using non-linear smooth activation functions such as soft-plus [10]. These paper established that a pre-specified small training error can be achieved by gradient descent when the network is very wide. The secret therein is that the weight matrix is very close to the initialization due to NTK [22].

(2) Kernel ridge regression: A neural network tends to give a Gaussian process when the width goes infinitely large. In this situation, only training the top layer of a network (weak training) will reduce to Kernel ridge regression. In a classical way, the weak training is to minimize the quadratic loss:

$$C(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda||\mathbf{w}||^2, \tag{26}$$

where $\mathbf{X} = [\mathbf{x}_1; ...; \mathbf{x}_n]$ is a collection of data $\mathbf{x}_i \in \mathbb{R}^{1\times d}, i = 1, ..., n$. Taking derivatives with respect to $\boldsymbol{w}$ and equating them to zero gives

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_d)^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{X}^T(\lambda\mathbf{I}_n + \mathbf{X}\mathbf{X}^T)^{-1}\mathbf{y}. \tag{27}$$

Given a new example x^*, the prediction is

$$y^* = x^*\mathbf{X}^T(\lambda\mathbf{I}_n + \mathbf{X}\mathbf{X}^T)^{-1}\mathbf{y}. \tag{28}$$

We replace the data samples with the high-dimensional representations of the neural network: $\mathbf{x} \to g_\theta(\mathbf{x}) \in \mathbb{R}^{1\times k}$ and let $\Phi_g(\mathbf{X}) = [g_\theta(\mathbf{x}_1); ...; g_\theta(\mathbf{x}_n)]$. By a similar derivation, we have

$$y^* = g_\theta(x^*)\Phi_g(\mathbf{X})^T(\lambda\mathbf{I}_n + \Phi_g(\mathbf{X})\Phi_g(\mathbf{X})^T)^{-1}\mathbf{y}. \tag{29}$$

When the width is very large, due to the Gaussian process, the weak training is reasonable, because $\Phi_g(\mathbf{X})$ converges to $\mathop{\mathbb{E}}_{\boldsymbol{\theta}\sim\boldsymbol{\Theta}}[\Phi_g(\mathbf{X})]$, and $\Phi_g(\mathbf{X})\Phi_g(\mathbf{X})^T$ converges to $\mathop{\mathbb{E}}_{\boldsymbol{\theta}\sim\boldsymbol{\Theta}}[\Phi_g(\mathbf{X})\Phi_g(\mathbf{X})^T]$, which can be approximated by the random initialization.

(3) Neural Tangent Kernel: In the above argument, we have justified the legitimacy of training the top layer by the large network width. How about training the entire network? Given the dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, we also consider training the network with the quadratic loss for regression: $l(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^n (f_{\boldsymbol{\theta}}(\mathbf{x}_i)) - y_i)^2$. Consider the gradient descent in an infinitesimally small learning rate, then

$$\frac{d\boldsymbol{\theta}(t)}{dt} = -\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}(t)) = -\sum_i^n (f_{\boldsymbol{\theta}(t)}(\mathbf{x}_i)) - y_i)\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}(t)}(\mathbf{x}_i). \tag{30}$$

Next, we can describe the dynamics of the model output $y(\boldsymbol{\theta})$ given the input $\mathbf{x}_j$ as follows:

$$\begin{aligned}\frac{df_{\boldsymbol{\theta}(t)}(\mathbf{x}_j)}{dt} &= \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}(t)}(\mathbf{x}_j)^T \frac{d\boldsymbol{\theta}}{dt} \\ &= -\sum_i^n (f_{\boldsymbol{\theta}(t)}(\mathbf{x}_i)) - y_i)\Big\langle \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}(t)}(\mathbf{x}_j), \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}(t)}(\mathbf{x}_i)\Big\rangle.\end{aligned} \tag{31}$$

Let us consider $\boldsymbol{u}(t) = (f_{\boldsymbol{\theta}(t)}(\mathbf{x}_i))_{i=1,\ldots,n}$ for all samples at time t and $\boldsymbol{y} = (y_i)_{i=1,\ldots,n}$ is the output, (31) can be written in a compact way:

$$\frac{d\boldsymbol{u}(t)}{dt} = -\mathbf{H}(t) \cdot (\boldsymbol{u}(t) - \boldsymbol{y}), \tag{32}$$

where the pq-entry of $\mathbf{H}(t)$ is

$$[\mathbf{H}(t)]_{pq} = \Big\langle \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}(t)}(\mathbf{x}_p), \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}(t)}(\mathbf{x}_q) \Big\rangle. \tag{33}$$

In the width limit, the Gaussian process shows that $\mathbf{H}(t)$ becomes a constant $\mathbf{H}^*$. Therefore,

$$\frac{d\boldsymbol{u}(t)}{dt} = -\mathbf{H}^* \cdot (\boldsymbol{u}(t) - \boldsymbol{y}), \tag{34}$$

which characterizes the trajectory of the training in the functional space instead of the parameter space.

Table 6: Representative bounds for chain-like neural networks, where $B_{l,2}$, $B_{l,F}$ and $B_{l,2\to1}$ to denote the upper bounds of the spectral norm $||\mathbf{W}_d||_2$, Frobenius norm $||\mathbf{W}_d||_F$, and $||\mathbf{W}_d||_{2,1}$ of the rank-r matrix in l^{th} layer. Generally, $||\mathbf{W}_d||_F$ and $||\mathbf{W}_d||_{2,1}$ is $\sqrt{r}$-times larger than $||\mathbf{W}_d||_2$, γ means the ramp loss function is $1/\gamma$-Lipschitz function, Γ is the lower bound for the product of the spectal norm of all the layers and the m is the size of data.

Reference	**Bound**	$\|\|\mathbf{W}_d\|\|_2 = 1$
[31]	$\mathcal{O}\Big(\frac{2^L \cdot \prod_{l=1}^{L} B_{l,F}}{\gamma\sqrt{m}}\Big)$	$\mathcal{O}\Big(\frac{2^L \cdot r^{L/2}}{\gamma\sqrt{m}}\Big)$
[5]	$\mathcal{O}\Big(\frac{\prod_{l=1}^{L} B_{l,F} \cdot \log(Lp)}{\gamma\sqrt{m}} \big(\sum_{l=1}^{L} \frac{B_{l,2\to1}^{2/3}}{B_{l,2}^{2/3}}\big)^{3/2}\Big)$	$\boldsymbol{\mathcal{O}\Big(\frac{\log(p)\sqrt{L^3 r}}{\gamma\sqrt{m}}\Big)}$
[30]	$\mathcal{O}\Big(\frac{\prod_{l=1}^{L} B_{l,F} \cdot \log(Lp)}{\gamma\sqrt{m}} \big(L^2 p \sum_{l=1}^{L} \frac{B_{l,F}^2}{B_{l,2}^2}\big)^{1/2}\Big)$	$\boldsymbol{\mathcal{O}\Big(\frac{\log(Lp)\sqrt{L^3 pr}}{\gamma\sqrt{m}}\Big)}$
[19]	$\mathcal{O}\Big(\frac{\prod_{l=1}^{L} B_{l,F}}{\gamma} \cdot \min\{\sqrt{\frac{\log\frac{1}{\Gamma}\prod_{l=1}^{L} B_{l,F}}{m^{1/2}}}, \sqrt{\frac{D}{m}}\}\Big)$	$\boldsymbol{\mathcal{O}\Big(\frac{\sqrt{Lpr}}{\gamma\sqrt{m}}\Big)}$

2. Generalization: Analysis of generalization bounds is a powerful tool to explain the excellent performance of neural networks. Traditional wisdom suggests that the increased model complexity will cause over-fitting to training data, which contradicts the fact that deep networks can easily fit random labels to the data and yet practically generalize well [35]. Recently, the generalization theory has gained increasingly more traction. In reference to Xingguo Li et al. [24], we have summarized the the state-of-the-art generalization bounds in **Table 6** and provided their complexities. In **Table 6**, we bold-face the bounds of interest which the width and depth dominate. As far as [31] is concerned, due to the exponential dependence on the depth, we will not focus on it. Instead, we argue that these bold-faced bounds somehow suggest the width and depth equivalence under a given complexity. Now, we use $B_1(L,p) = \log(p)\sqrt{L^3}$, $B_2(L,p) = \log(Lp)\sqrt{L^3 p}$, and $B_3(L,p) = \sqrt{Lp}$ to denote the bounds of interest. **Figure 10** shows the contours plots of B_1, B_2 and B_3. Along a contour, the width and depth changes for the fixed bound complexity. In other words, the depth and width of a neural network are mutually transformable without impacting the overall generalization ability theoretically.

3. VC Dimension: The equivalence of depth and width of a network can be appreciated in reference to VC dimension [33]. VC dimension is a measure on the capacity of a hypothesis space that can be learned by a neural network and other statistical learning models. The definition of VC dimension is based on the cardinality of the largest set of points that can be "shattered" by a model. Since VC dimension is originally geared towards binary classification, let us give the definition of VC dimension in the context of binary classification.

Given the hypothesis space $\mathcal{H}$ and the data collection $D = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$, every $h \in \mathcal{H}$ labels the instances in D, we denote the resultant labeling as

$$h_{|D} = \{(h(\mathbf{x}_1), h(\mathbf{x}_2), \cdot, h(\mathbf{x}_m))\}. \tag{35}$$

It is seen that, for m instances, the maximum number of possible labeling schemes is 2^m. If the hypothesis space $\mathcal{H}$ can realize all the possible labeling schemes on D, we say that D can be shattered by $\mathcal{H}$. VC dimension of $\mathcal{H}$ is the cardinality of the dataset comprising the maximum number of instances that can be shattered.

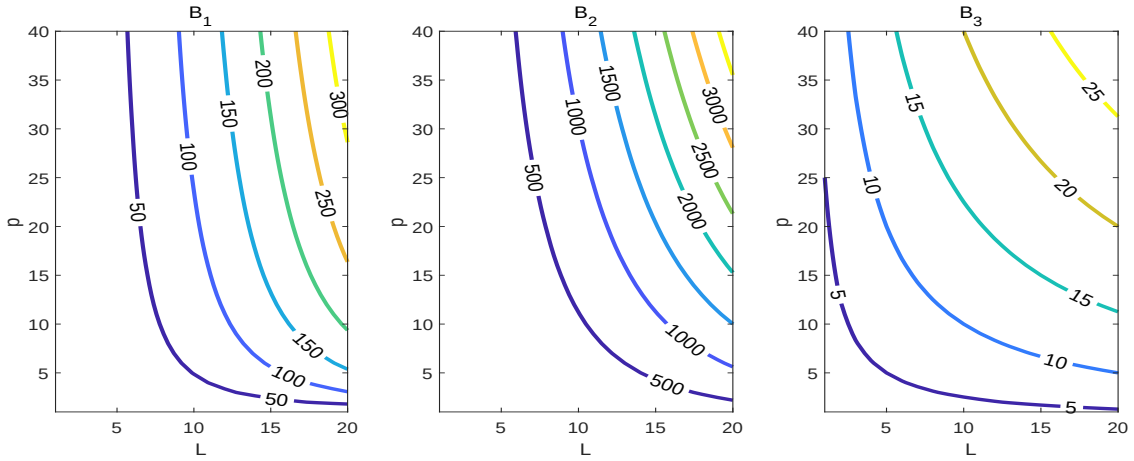

Figure 10: Contours of B_1, B_2 and B_3, where numbers on the curves of each sub-figure represent $\log(p)\sqrt{L^3}$, $\log(p)\sqrt{L^3}$ and $\sqrt{Lp}$ respectively. Along a contour, the width and depth changes to give the same bound.

The bound of VC dimension of a neural network has been extensively studied in the past decades. For example, let W be the number of weights and L be the number of layers, Goldberg and Jerrum [18] established the upper bound of VC dimension $\mathcal{O}(W^2)$ for a network with piecewise polynomial activation. Furthermore, Bartlett et al. [4] found that an upper VC dimension bound is $\mathcal{O}(WL^2 + WLlog(WL))$ and a lower bound is $\mathcal{O}(WL)$. Recently, Bartlett et al. [3] derived a nearly tight VC dimension bound for ReLU networks:

$$c \cdot WLlog(W/L) \leq VCdim(W, L) \leq C \cdot WLlog(W), \tag{36}$$

where c and C are constants.

Eq. (36) implies that the above upper and lower bounds for VC dimension of a neural network are determined by the number of parameters and the number of layers. As a semi-quantitative estimate, let all the layers have the same number of neurons, then we have

$$\begin{aligned} VCdim(Width, Depth) &\geq c \cdot Width \cdot Depth \cdot log(Width) \\ VCdim(Width, Depth) &\leq C \cdot Width \cdot Depth \cdot log(Width) \end{aligned} \tag{37}$$

Therefore, increasing either width or depth can boost the hypothesis space of a neural network. In other words, when it comes to promoting the expressive power of a network, increasing the width is essentially equivalent to increasing the depth in the sense of VC dimension, which also implies the equivalence of the width and depth of neural networks.

Reference

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*, pages 6155–6166, 2019.

[2] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, pages 8139–8148, 2019.

[3] P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20(63):1–7, 2019.

[4] P. L. Bartlett, V. Maiorov, and Ron Meir. Almost linear vc-dimension bounds for piecewise polynomial networks. *Neural Computation*, 10(8):2159–2173, 1998.

[5] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017.

[6] Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. Sgd learns over-parameterized networks that provably generalize on linearly separable data. *arXiv preprint arXiv:1710.10174*, 2017.

[7] Yuan Cao and Quanquan Gu. A generalization theory of gradient descent for learning over-parameterized deep relu networks. *arXiv preprint arXiv:1902.01384*, 2019.

[8] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

[9] Simon S Du and Wei Hu. Width provably matters in optimization for deep linear neural networks. *arXiv preprint arXiv:1901.08572*, 2019.

[10] Simon S Du, Jason D Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. *arXiv preprint arXiv:1811.03804*, 2018.

[11] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

[12] F. Fan, W. Cong, and G. Wang. Generalized backpropagation algorithm for training second-order neural networks. *International Journal for Numerical Methods in Biomedical Engineering.*, 34(5), 2018.

[13] F. Fan, W. Cong, and G. Wang. A new type of neurons for machine learning, international journal for numerical methods in biomedical engineering. *International Journal for Numerical Methods in Biomedical Engineering*, 34(2), 2018.

[14] Fenglei Fan and Ge Wang. Universal approximation with quadratic deep networks. *arXiv preprint arXiv:1808.00098*, 2018.

[15] Fenglei Fan, Jinjun Xiong, and Ge Wang. Universal approximation with quadratic deep networks. *Neural Networks*, 124:383–392, 2020.

[16] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, pages 797–842, 2015.

[17] Federico Girosi and Tomaso Poggio. Representation properties of networks: Kolmogorov's theorem is irrelevant. *Neural Computation*, 1(4):465–469, 1989.

[18] P. W. Goldberg and M. R. Jerrum. Bounding the vapnik-chervonenkis dimension of concept classes parameterized by real numbers. *Machine Learning*, 18(2):131–48, 1995.

[19] Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. *arXiv preprint arXiv:1712.06541*, 2017.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[21] Kaixuan Huang, Yuqing Wang, Molei Tao, and Tuo Zhao. Why do deep residual networks generalize better than deep feedforward networks?–a neural tangent kernel perspective. *arXiv preprint arXiv:2002.06262*, 2020.

[22] 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.

[23] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

[24] Xingguo Li, Junwei Lu, Zhaoran Wang, Jarvis Haupt, and Tuo Zhao. On tighter generalization bound for deep neural networks: Cnns, resnets, and beyond. *arXiv preprint arXiv:1806.05159*, 2018.

[25] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pages 8157–8166, 2018.

[26] W. A. Light and E. W. Cheney. Approximation theory in tensor product spaces. *Springer*, 2006.

[27] Zachary C Lipton. The mythos of model interpretability. *Queue*, 16(3):31–57, 2018.

[28] Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.

[29] Radford M Neal. Priors for infinite networks. In *Bayesian Learning for Neural Networks*, pages 29–53. Springer, 1996.

[30] Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*, 2017.

[31] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pages 1376–1401, 2015.

[32] Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Greg Yang, Jiri Hron, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. *arXiv preprint arXiv:1810.05148*, 2018.

[33] V. Vapnik, E. Levin, and Y. Lecun. Measuring the vc-dimension of a learning machine. *Neural computation*, 6(5):851–76, 1994.

[34] David Vernon Widder. *Advanced calculus*. Courier Corporation, 1989.

[35] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.