Model Complexity of Deep Learning: A Survey

Xia Hu[1], Lingyang Chu[2], Jian Pei[1], Weiqing Liu[3], and Jiang Bian[3]

[1]School of Computing Science, Simon Fraser University, Burnaby, Canada, huxiah@sfu.ca, jpei@cs.sfu.ca
[2]Department of Computing and Software, McMaster University, Hamilton, Canada, chul9@mcmaster.ca
[3]Microsoft Research, Beijing, China, {Weiqing.liu, Jiang.bian}@microsoft.com

March 10, 2021

Abstract

Model complexity is a fundamental problem in deep learning. In this paper we conduct a systematic overview of the latest studies on model complexity in deep learning. Model complexity of deep learning can be categorized into expressive capacity and effective model complexity. We review the existing studies on those two categories along four important factors, including model framework, model size, optimization process and data complexity. We also discuss the applications of deep learning model complexity including understanding model generalization capability, model optimization, and model selection and design. We conclude by proposing several interesting future directions.

1 Introduction

Mainly due to its superior performance, deep learning is disruptive in many applications, such as computer vision [40], natural language processing [55] and computational finance [91]. At the same time, however, a series of fundamental questions about deep learning models remain, such as why deep learning can achieve substantially better expressive power comparing to classical machine learning models, how to understand and quantify the generalization capability of deep models, and how to understand and improve the optimization process.

Model complexity of deep learning is a core problem and is related to many those fundamental questions.

Model complexity of deep learning is concerned about, for a certain deep learning architecture, how complicated problems the deep learning model can express [15, 44, 70, 89]. Understanding the complexity of a deep model is a key to precisely understanding the capability and limitation of the model. Exploring model complexity is necessary not only for understanding a deep model itself, but also for investigating many other related fundamental questions. For example, from the statistical learning theory point of view, the expressive power of a model is used to bound the generalization error [69]. Some recent studies propose norm-based model complexity [60] and sensitivity-based model complexity [76, 81] to explore the generalization capability of deep models. Moreover, detecting changes of model complexity in a training process can provide insights into understanding and improving the performance of model optimization and regularization [44, 74, 89].

The investigation of machine learning model complexity dates back to decades ago. A series of early studies in 1990s discuss the complexity of classical machine learning models [16, 20, 21, 98]. One representative model is decision tree [19], whose complexity is always measured by tree depth [20] and number of leaf nodes [16]. Another frequent subject in model complexity analysis is logistic regression, which is the foundation of a large number of parameterized models. The model complexity of logistic regression is investigated from the perspectives of Vapnik-Chervonenicks theory [26, 96], Rademacher complexity [46], Fisher Information matrix [21], and the razor of model [6]. Here, the razor of model is a theoretical index of the complexity of a parametric family of models comparing to the true distribution. However, deep learning models are dramatically different from those classical machine learning models discussed decades ago [70]. The complexity analysis of classical machine learning models cannot be directly applied or straightforwardly extended to deep models.

Recently, model complexity in deep learning has attracted more and more attention [13, 60, 70, 78, 81, 89]. However, to the best of our knowledge, there is no existing survey on model complexity in deep learning. The lack of survey on this emerging and important subject motivates us to conduct this survey of the latest studies. In this article, we use the terms "deep learning model" and "deep neural network" interchangeably.

There are prolific studies on complexity of classical machine learning models decades ago, which are summarized in excellent surveys [20, 21, 61, 93]. In this section, we very briefly review the complexity of several typical models, including decision tree, logistic regression and bayesian network models. We also discuss the differences between the model complexity of deep neural networks and that of those models.

There are relatively well accepted standard measurements for complexity of decision trees. The complexity of a decision tree can be represented by the number of leaf nodes [16, 61] or the depth of the tree [20, 98]. Since a decision tree is constructed by recursively splitting the input space and proposing a local simple model for each resulting region [71], the number of leaf nodes in principle uses the number of resulting regions to represent the complexity of the tree. The depth of a decision tree as the complexity measure quantifies in the worst case the number of queries needed to make a classification [20]. A series of studies [16, 61] investigate tree optimization based on the accuracy-complexity tradeoff. Furthermore, Buhrman and De Wolf [20] associate the complexity of decision trees with the complexity of functions represented by certificate complexity [4], sensitivity complexity [28] and approximating polynomials [80], and use these complexity measures of functions to bound the complexity of decision trees.

Logistic regression is the foundation of a large number of parameterized models [21, 46]. In the early 1990s, the degree of effective freedom is proposed as a complexity measure for linear models and penalized linear models, which is represented by Vapnik-Chervoneniks (VC) dimension [26]. Besides, Rademacher complexity and Gaussian complexity [8, 12] are also used to measure model complexity of logistic regression models [46]. Comparing to VC dimension, Rademacher complexity takes data distribution into consideration and therefore reflects finer-grained model complexity. Later, Bulso *et al.* [21] and Balasubramanian [6] suggest that model complexity of logistic regression models is related to the number of distinguishable distributions that can be represented by the models. Bulso *et al.* [21] define a complexity measure of logistic regression models based on the determinant of the Fisher Information matrix.

Spiegelhater *et al.* [93] systematically investigate the model complexity of Bayesian hierarchical models. A unique challenge in measuring the complexity of Bayesian hierarchical models is that the number of model parameters is not clearly defined. Spiegelhater *et al.* [93] define a model complexity measure by the number of effective parameters. Using the information theoretic argument, they show that this complexity measure can be estimated by the difference between the posterior mean of the deviance and the deviance at the posterior estimates of the parameters of interest, and is approximately the trace of the product of Fisher's information [33] and the posterior covariance matrix. In addition, they suggest that adding such a complexity measure to the posterior mean deviance gives a deviance information criterion for comparing models.

Complexity measures are often model specific. The definition of model complexity largely depends on model structures. Different model frameworks often call for different definitions of complexity measures according to their structural characteristics. The complexity measures of different model frameworks usually

cannot be directly compared.

Deep learning models are structurally different from traditional machine learning models and have dramatically more parameters. Deep learning models are always substantially more complex than traditional models. Therefore, the previous methods of model complexity of traditional machine learning models cannot be directly applied to deep learning models to obtain valid complexity measures. For example, measuring the complexity of a decision tree by the depth of the tree [20, 98] and the number of leaf nodes [16, 61] is obviously not applicable to deep learning models. Measuring model complexity by the number of trainable parameters [46] has a very limited effect on deep learning models since deep learning models are often over-parameterized.

The rest of this survey is organized as follows. In Section 2, we introduce deep learning model complexity and the two categories, expressive capacity and effective model complexity. In Section 3 we review the existing studies on the expressive capacity of deep learning models. In Section 4 we survey the existing studies on the effective complexity of deep learning models. In Section 5 we discuss the applications of deep learning model complexity. In Section 6 we conclude this survey and discuss some future directions.

2 Deep Learning Model Complexity

In this section, we first divide deep model complexity into two categories, expressive capacity and effective model complexity. Then, we discuss a series of important factors of deep learning model complexity, and group the representative existing studies accordingly.

2.1 What is Deep Learning Model Complexity?

The term "model complexity" may refer to two different meanings in deep learning. First, model complexity may refer to capacity of deep models in expressing or approximating complicated distribution functions [13]. Second, it may describe how complicated the distribution functions are with some parameterized deep models [89]. These two meanings are captured by the notions of model expressive capacity and model effective complexity, respectively.

Expressive capacity, also known as representation capacity, expressive power, and complexity capacity [60, 86], captures the capacity of deep learning models in approximating complex problems. Informally, the expressive capacity describes the upper bound of the complexity of any model in a family of models.

This notion is consistent with the description of hypothesis space complexity [69, 96]. The hypothesis space is a family of hypotheses, such as the family of all neural networks with a fixed model structure. Considering the hypothesis

space represented by a fixed model structure, the model expressive capacity is also the hypothesis space complexity. In statistical learning theory, the complexity of an infinite hypothesis space is represented by its expressive power, that is, the richness of the family of hypothesises [69]. A notion to capture hypothesis space complexity is Rademacher complexity [12], which measures the degree to which a hypothesis space can fit random noise. Another notion is VC dimension [26], which reflects the size of the largest set that can be shattered by the hypothesis space. Exploring expressive capacity helps to obtain the guarantee of learnability of deep models and derive generalization bounds [69].

Effective model complexity, also known as practical complexity, practical expressivity, and usable capacity [37, 81], reflects the complexity of the functions represented by deep models with specific parameterization [89]. The effective model complexity is for a model with parameters fixed. The study of effective model complexity helps the exploration of various aspects of deep models, such as understanding the optimization algorithms [81], improving model selection strategies [72] and improving model compression [25].

Expressive capacity and effective model complexity are closely related but are two different concepts. Expressive capacity describes the expressive power of the hypothesis space of a deep model. Effective model complexity explores the complexity of a specific hypothesis within the hypothesis space. Let us use an example to demonstrate the distinction and relationship between model expressive capacity and effective model complexity.

Example 1 (Difference between expressive capacity and effective complexity)**.** Consider unary polynomial function $f(x) = ax^2+bx+c$. The expressive capacity of $f(x)$ is unary quadratic. In other words, $f(x)$ cannot express a function more complicated than a unary quadratic polynomial. When assigning different values to the parameters a, b and c, the corresponding effective complexity may be different. With parameters $a = 0$, $b = 1$ and $c = 1$, for example, $f(x) = x + 1$, the effective complexity becomes linear, which is obviously lower than the expressive capacity.

Denote by $\mathcal{H}$ the hypothesis space of a fixed deep learning model structure, and by $h \in \mathcal{H}$ a hypothesis (i.e., a deep learning model) in $\mathcal{H}$. The effective model complexity is the complexity of a specific hypothesis, written as $EMC(h)$. The expressive power of the deep model can be written in the form of

$$\sup\{EMC(h) : h \in \mathcal{H}\}$$

This reflects the relationship between model expressive capacity and effective model complexity.

Because of the complex, over-parameterized architectures, deep learning models usually have high expressive capacity [70, 87]. However, a series of

studies [5, 37, 81] find that the effective complexity of a trained deep model may be much lower than the expressive capacity.

Informally and intuitively, a deep learning model can be regarded as a "container" of knowledge learned from data. The same model architecture as a "container" may contain different amounts of knowledge by learning from different data and thus equipped with different parameters. The expressive capacity can be regarded as the upper bound of the amount of knowledge that a model architecture can hold. The effective model complexity is concerned about, for a specific model, a specific training dataset, how much knowledge it holds.

2.2 Important Factors of Deep Learning Model Complexity

Bonaccorso [17] points out that a deep learning model consists of a static structure part and a dynamic parametric part. The static structure part is always determined before the learning process by the model selection principle, then stays immutable once decided. The parametric part is the objective of the optimization and is determined by a learning process. Both the static and dynamic parts contribute to model complexity. We refine this division and summarize four aspects affecting model complexity, including both expressive capacity and effective complexity.

Model framework The choice of model framework affects model complexity. The factor of model framework includes model type (e.g., FCNN, CNN), activation function (e.g., Sigmoid [29], ReLU [73]), and others. Different model frameworks may require different complexity measure criteria and may not be directly comparable to each other.

Model size The size of deep model affects model complexity. Some commonly used measures of model size include the number of parameters, the number of hidden layers, the width of hidden layers, the number of filters, and the filter size. Under the same model framework, the complexities of models with different sizes can be quantified by the same complexity measure criteria and thus become comparable [54].

Optimization process The optimization process affects model complexity, including the form of objective functions, the selection of learning algorithms, and the setting of hyperparameters.

Data complexity The data on which a model is trained affect model complexity, too. The major factors include the data dimensionality, data distribution [69], information volume measured by Kolmogorov complexity [22, 59], and some others.

Table 1: Summarize the aspects affecting the expressive capacity and effective complexity, respectively.

		Model Framework	Model Size	Learning Process	Data Complexity
Expressive Capacity	Depth Efficiency		✓		
	Width Efficiency		✓		
	Expressible Functional Space	✓	✓	✓	✓
Effective Complexity	Effective Complexity Measure	✓	✓	✓	✓
	High-capacity Low-reality Phenomenon	✓	✓	✓	✓

Among these four aspects, model framework and model size mainly affect the static structural part of a deep model, and optimization process and data complexity mainly affect the dynamic parametric part.

The effect of these four aspects on expressive capacity can be understood through the influence on the hypothesis space. A selected model framework corresponds to a hypothesis space $\mathcal{H}$. Each hypothesis $h \in \mathcal{H}$ represents a model with the given framework. Once the model size is determined, the hypothesis space shrinks to a subset of $\mathcal{H}$. For example, suppose we set up two models with depth l_1 and l_2 ($l_1 \leq l_2$), respectively, and both with width m, the corresponding hypothesis spaces are $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively. We have $\mathcal{H}_1 \subset \mathcal{H}_2$. This is because, for each $h \in \mathcal{H}_1$, where h is a hypothesis and thus a deep network model in this context, there exists $h' \in \mathcal{H}_2$ whose first l_1 layers are identical to those in h and the subsequent layers are identical mappings. It is easy to show that the expressive capacity of $\mathcal{H}_1$ will not exceed the expressive capacity of $\mathcal{H}_2$. Recently, model framework and size and their effect on expressive capacity of deep models have been well explored [13, 51, 64, 89].

The choice of data distribution and optimization algorithm will further reduce the scope of the hypothesis space, thereby affecting the expression capacity. For example, Rademacher complexity is a data-dependent expressive capacity measure taking into account the effect of data distribution [12, 69]. However, to the best of our knowledge, the effect of data complexity and optimization process on expressive capacity of deep learning models is still rarely explored.

These four aspects also affect the effective model complexity. In general, model framework and model size decide the available range of effective model complexity. The effective complexity of a model with fixed parameters is a value in this range selected by optimization process and training data. The optimization process affects the effective complexity, for example, adding L^1 regularization constrains the degrees of freedom in a deep neural network, and thus constrains the effective model complexity [44, 46]. The training data affects the effecive complexity, for example, using the same model and the same optimization process, the effective complexity of a model trained on linearly classifiable data is much lower than that trained on the ImageNet [31, 59].

Specifically, the effect of training data complexity and optimization process on effective complexity are reflected in the values of model parameters.

In Table 1 we list the corresponding aspects affecting expressive capacity and effective complexity, respectively. In Table 2 we list some representative studies on the two major problems, expressive capacity and effective complexity.

2.3 Existing Studies on Deep Learning Model Complexity: An Overview

The literature on deep model complexity can be categorized from two different angles. The first angle focuses on whether an approach is model-specific or cross-model. The second angle focuses on whether an approach develops an explicit complexity measure. These two grouping angles are applicable to both expressive capacity and effective complexity.

2.3.1 Model-Specific vs. Cross-Model

Based on whether an approach focuses on one type of models or crossing multiple types of models, the existing studies on model complexity can be divided into two groups, model-specific approaches and cross-model approaches.

A **model-specific** approach focuses on a certain type of models, and explores complexity based on structural characteristics. For example, Bianchini *et al.* [14, 15] and Hanin *et al.* [37] study model complexity of FCNNs, Bengio and Delalleau [13, 30] focus on model complexity of sum-product networks. Moreover, some studies further propose constraints on the activation functions to constrain the nonlinearity properties. For example, Liang *et al.* [60] assume that activation functions $\sigma(\cdot)$ satisfy $\sigma(z) = \sigma'(z)z$.

An approach is **cross-model** when it covers multiple types of models rather than one specific type of models, and thus can be applied to compare two or more different types of models. For example, Khrulkov *et al.* [51] compare the complexity of general RNNs, CNNs and shallow FCNNs by building connections among these network architectures and tensor decompositions.

2.3.2 Measure-Based vs. Reduction-Based

According to whether an explicit measure is designed, the state-of-the-art model complexity approaches can be divided into two groups, measure-based approaches and reduction-based approaches.

A deep model complexity study is **measure-based** if it defines an appropriate quantitative representation of model complexity. For example, the number of linear regions is used to represent the complexity of FCNNs with piecewise linear activation functions [37, 70, 81, 89].

Table 2: Overview of a collection of studies on model complexity of deep neural networks.

		References	*Model-specific*	*Cross-model*	*Measure-based*	*Reduction-based*	Objective Model	Measure Metric
Expressive Capacity	Depth Efficiency	[13, 30]	•			•	SPN	-
		[70]	•		•		FCNN with $\sigma(z) = \{ReLU, Maxout\}$	#linear regions
		[14, 15]		•	•		FCNN	Sum of Betti number
		[67]	•			•	Binary tree hierarchical network, deep Gaussian network	#trainable parameters
		[54]	•			•	FCNN with $\sigma(z) = \{Signum, Heaviside\}$	Variational norm
	Width Efficiency	[64]	•			•	FCNN with $\sigma(z) = \{ReLU\}$	-
	Expressible Functional Space	[51]		•		•	FCNN, CNN, RNN	Rank of tensor decomposition
		[3]	•			•	FCNN with $\sigma(z) = \{ReLU\}$	#linear regions
		[36]	•			•	FCNN with $\sigma(z) = \{ReLU\}$	Depth, #neurons, #parameters
		[52]	•		•		FCNN with $\sigma(z) = \{\sigma(z) = z^r\}$	Dimension of functional variety
	VC Dimension and Rademacher Complexity	[66]	•		•		FCNN with linear threshold gates	VC dimension
		[11]	•		•		FCNN with $\sigma = \{piecewise\ polynomial\}$	VC dimension
		[10]	•		•		Networks with $\sigma = \{piecewise\ linear\}$	VC dimension
		[9]	•		•		Networks with $\sigma = \{ReLU\}$	Rademacher complexity
		[77]	•		•		Two-layer FCNN with $\sigma = \{ReLU\}$	Rademacher complexity
Effective Complexity	Effective Complexity Measure	[89]	•		•		Networks with $\sigma = \{piecewise\ linear\}$	#linear regions, trajectory length
		[81]	•		•		Networks with $\sigma = \{piecewise\ linear\}$	Jacobian norm, trajectory length
		[60]	•		•		FCNN with $\sigma(z) = \sigma'(z)z$	Fisher-Rao norm
		[44]	•		•		FCNN with $\sigma(z) = \{curve\ activation\}$	#approximated linear regions
		[74]	•		•		Any deep models	#samples
	High-Capacity Low-Reality Fact	[37]	•		•		FCNN with $\sigma(z) = \{ReLU\}$	#linear regions, volumn of region boundaries

A complexity approach is **reduction-based** if it investigates a model complexity problem by reducing deep networks to some known problems and functions, and does not define any explicit complexity measure. For example, Arora *et al.* [3] build connections between deep networks with ReLU activation functions and Lebesgue spaces. Khrulkov *et al.* [51] connect deep neural networks with several types of tensor decompositions.

3 Expressive Capacity

Expressive capacity of deep models is also known as representation capacity, expressive power, and complexity capacity [60, 86]. It describes how well a deep learning model can approximate complex problems. Informally, the expressive capacity describes the upper bound of the complexity in a parametric family of models.

Expressive capacity of deep learning models has been mainly explored from four aspects.

- **Depth efficiency** analyzes how deep learning models gain performance (e.g., accuracy) from the depth of architectures.
- **Width efficiency** analyzes how the widths of layers in deep learning models affect model expressive capacity.
- **Expressible functional space** investigates the functions that can be expressed by a deep model with a specific framework and specified size using different parameters.
- **VC Dimension and Rademacher Complexity** are two classic measures of expressive capacity in machine learning.

In this section, we review these four groups of studies.

3.1 Depth Efficiency

A series of recent studies demonstrate that deep architectures significantly outperform shallow ones [13, 67]. Depth efficiency [64], which is the effectiveness of depth of deep models, has attracted a lot of interest. Specifically, the studies on depth efficiency analyze why deep architectures can obtain good performance and measures the effects of model depth on expressive capacity. We divide the studies of depth efficiency into two subcategories: model reduction methods and expressive capacity measures.

3.1.1 Model Reduction

A way to study the expressive capacity of deep learning models is to reduce deep learning models to understandable problems and functions for analysis.

To investigate depth efficiency, one intuitive idea is to compare the representation efficiency between deep networks and shallow ones. Bengio and Delalleau [13, 30] investigate the depth efficiency problem on deep sum-product networks (SPN for short). An SPN consists of neurons computing either products or weighted sums of their inputs. They consider two families of functions built from deep sum-product networks. The first family of functions is $F = \cup_{n\geq 4} F_n$, where F_n is the family of functions represented by deep SPNs with $n = 2^k$ inputs and depth k. The second family of functions is $G = \cup_{n\geq 2, i\geq 0} G_{i,n}$, where $G_{i,n}$ is the family of functions represented by SPNs with n inputs and depth $2i + 1$.

Then, Bengio and Delalleau establish the lower bounds on the number of hidden neurons required by a shallow sum-product network to represent those families of functions. To approach a function $f \in F$, a one-hidden-layer shallow SPN needs at least $2^{\sqrt{n}-1}$ hidden neurons and at least $2^{\sqrt{n}-1}$ product neurons. Similarly, to approach a function $g \in G$, a one-hidden-layer shallow SPN needs at least $(n-1)^i$ hidden neurons. The comparison of deep and shallow sum-product networks representing the same function indicates that, to represent the same functions, the number of neurons in a shallow network has to grow exponentially but only a linear growth is needed for deep networks.

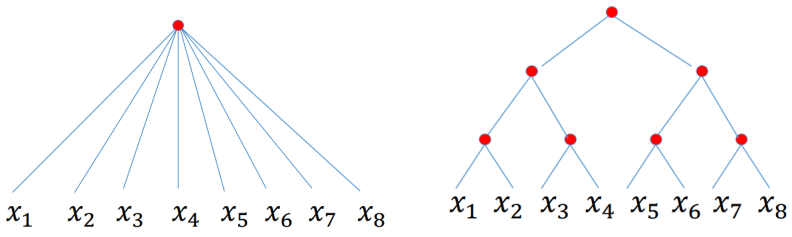
Mhaskar *et al.* [67] study functions representable by hierarchical binary tree networks, comparing with shallow networks. Fig. 1 demonstrates shallow networks and hierarchical binary tree networks. Denote by S_m the class of shallow networks with m neurons, in the form of

$$x \to \sum_{k=1}^{m} a_k \sigma(w_k \cdot x + b_k)$$

where $w_k \in \mathbb{R}^d$, $b_k, a_k \in \mathbb{R}$ are the parameters for the k-th hidden neuron, σ is the activation function. Denote by $W_{r,d}^{NN}$ the class of functions with continuous partial derivatives of order $\leq r$ with certain assumptions (see [67] for detail). Correspondingly, a hierarchical binary tree network is a network with the structure of

$$\begin{aligned} f(x_1, \ldots, x_d) = & h_{l1}(h_{(l-1),1}(\ldots(h_{11}(x_1, x_2), \\ & h_{12}(x_3, x_4), \ldots), h_{(l-1),2}(\ldots)), \end{aligned} \tag{1}$$

where each h_{ij} is in S_m, l is the depth of the network, h_{l1} corresponds to the root node of the tree structure (Fig. 1(b)). Denote by D_m the class of hierarchical binary tree networks. Let $W_{H,r,d}^{NN}$ be the class of functions with the structure of



(a) (b)

Figure 1: 8 input dimensions fed into a shallow network (a) and a hierarchical binary tree network (b), studied by Mhaskar *et al.* [67].

Eq. (1), where each h_{ij} is in $W_{r,2}^{NN}$. Define by $\inf_{P \in V} ||f - P||$ the approximation degree of V to function f, where $V = \{S_m, D_m\}$.

Mhaskar *et al.* [67] demonstrate that, to approximate a function $f \in W_{r,d}^{NN}$ to approximation degree ϵ, a shallow network in S_m requires $O(\epsilon^{-d/r})$ trainable parameters. Meanwhile, to approximate a function $f \in W_{H,r,d}^{NN}$ to the same approximation degree, a network in D_m requires only $O(\epsilon^{-2/r})$ trainable parameters. Then, Mhaskar *et al.* [67] compare shallow Gaussian networks with hierarchical binary tree structures (Eq. (1)) where each h_{ij} computes a shallow Gaussian network, and obtain a similar conclusion. They demonstrate that functions with a designed compositional structure can be approximated by both deep and shallow networks to the same degree of approximation. However, the numbers of parameters of deep networks are much less than those of shallow networks.

Arora *et al.* [3] investigate the importance of depth on deep neural networks with ReLU activation function. First, they investigate neural networks with one-dimensional input and one-dimensional output. They prove that given any natural numbers $k \geq 1$ and $w \geq 2$, there exists a family of functions that can be represented by a ReLU neural network with k hidden layers each of width w. However, to represent this family of functions, a network with $k' < k$ hidden layers requires at least $\frac{1}{2}k'w^{\frac{k}{k'}} - 1$ hidden neurons. Then, they investigate ReLU neural networks with d input dimensions. They prove that, given natural numbers $k, m, d \geq 1$ and $w \geq 2$, there exists a family of $\mathbb{R}^d \to \mathbb{R}$ functions that can be represented by a ReLU network with $k + 1$ hidden layers and $2m + wk$ neurons. This family of functions is constructed using the zonotope theory from the polyhedral theory [101]. However, to represent this family of functions, the minimum number of hidden neurons that a ReLU neural network with $k' \leq k$

hidden layers requires is

$$\max\{\frac{1}{2}(k'w^{\frac{k}{k'd}})(m-1)^{(1-\frac{1}{d})\frac{1}{k'}}-1, k'(\frac{w^{\frac{k}{k'}}}{d^{\frac{1}{k'}}})\}.$$

To investigate when deep neural networks are provably more efficient than shallow ones, Kuurkova [54] analyzes the limitation of expressive capacity of shallow neural networks with Signum activation function. The Signum activation function is defined as

$$sgn(z) = \begin{cases} -1 & \text{for} \quad z < 0 \\ 1 & \text{for} \quad z \geq 0 \end{cases}$$

Kuurkova proves that there exist functions that cannot be L^1 sparsely represented by one-hidden-layer Signum neural networks, which have a limited number of neurons and a limited sum of absolute values of output weights (i.e., L^1-norm). Such functions should be nearly orthogonal to any function f from the class of Signum perceptrons $\{sgn(vx+b) : X \to \{-1,1\} | v \in \mathbb{R}^d, b \in \mathbb{R}\}$. The functions generated by Hadamard matrices are such examples. A Hadamard matrix of order n is a $n \times n$ matrix with entries in $\{-1,1\}$ such that any two distinct rows or columns are orthogonal. Kuurkova [54] proves that the functions induced by $n \times n$ Hadamard matrices cannot be computed by shallow Signum networks with both the number of hidden neurons and the sum of absolute values of output weights smaller than $\frac{\sqrt{n}}{\lceil \log_2 n \rceil}$.

To further illustrate the limitation of shallow networks, Kuurkova [54] compares the representation capability of one and two hidden layer networks with Heaviside activation function. The Heaviside activation function is defined as

$$\sigma(z) = \begin{cases} 0 & \text{for} \quad z < 0 \\ 1 & \text{for} \quad z \geq 0 \end{cases} \tag{2}$$

Let $S(k)$ be a $2^k \times 2^k$ Sylvester-Hadamard matrix, which is constructed by starting from $S(2) = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ and then recursively iterating $S(l+1) = S(2) \otimes S(l)$. Kuurkova [54] shows that, to represent a function induced by $S(k)$, a two-hidden-layer Heaviside network requires k neurons in each hidden layer. However, to represent such a function, a one-hidden-layer Heaviside network requires at least $\frac{2^k}{k}$ hidden neurons, or the absolute value of some output weights is no less than $\frac{2^k}{k}$.

In summary, the model reduction approaches reduce neural networks to some kind of functions [3, 13, 54, 67], and investigate the effects of model depth on the capacity to express function families.

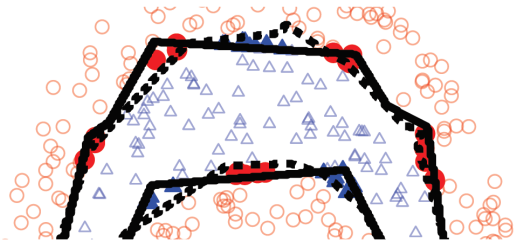


Figure 2: An example from Montufar *et al.* [70] illustrating the advantage of deep models. A deep ReLU network (the dotted line) captures the boundary more accurately by approximating it with a larger number of linear regions than a shallow one (the solid line).

3.1.2 Expressive Capacity Measures

To investigate depth efficiency, another idea is to develop an appropriate measure of expressive capacity and study how expressive capacity changes when the depth and layer width of a model increase.

Montufar *et al.* [70] focus on fully-connected feed forward neural networks (FCNNs) with piecewise linear activation functions (e.g., ReLU [73] and Maxout [35]), and propose to use the number of linear regions as a representation of model complexity (Figure 2). The basic idea is that an FCNN with piecewise linear activation functions divides the input space into a large number of linear regions. Each linear region corresponds to a linear function. The number of linear regions can reflect the flexibility and complexity of the model.

Montufar *et al.* [70] investigate FCNNs with two types of piecewise linear activation functions: ReLU [73] and Maxout [35]. They prove that, under certain parameter settings, the model expressive capacity of a ReLU network $\mathcal{N}$, which is represented by the maximum number of linear regions and denoted by $MEC(\mathcal{N})$, is bounded by

$$MEC(\mathcal{N}) \geq \left(\prod_{i=1}^{l-1} \lfloor \frac{m_i}{m_0} \rfloor^{m_0}\right) \sum_{j=0}^{m_0} \binom{m_l}{j} \tag{3}$$

where l is the number of hidden layers, m_i is the width of i-th hidden layer, $m_0 = d$ is the dimensionality of the input. Based on this bound, they show that a ReLU network with l hidden layers and layer width $m_i \geq m_0$ is able to approximate any piecewise linear function that has $\Omega((\frac{m}{m_0})^{(l-1)m_0} m^{m_0})$ linear regions.

Montufar *et al.* [70] also prove that, for the rank-k Maxout activation function, the expressive capacity of a one-hidden-layer Maxout network

with m neurons is bounded by $MEC(\mathcal{N}) \geq k^{\min(d,m)}$ and $MEC(\mathcal{N}) \leq \min\{\sum_{j=1}^{d} \binom{k^2 m}{j}, k^m\}$. A rank-$k$ Maxout network, which consists of l hidden layers and the width of each layer equals m, can compute any piecewise linear function with $\Omega(k^{l-1}k^m)$ linear regions. In conclusion, the maximum number of linear regions generated by a FCNN with piecewise linear activation functions increases exponentially with model depth.

Montufar *et al.* [70] provide an explanation for the depth efficiency. They suggest that the intermediary layer of a deep model is able to map several pieces of the inputs into the same output. As the number of layers increases, the layer-wise compositions of the functions re-use lower-level computation exponentially. This allows deep models to compute highly complex functions, even with relatively fewer parameters.

Serra *et al.* [92] improve the bounds of the maximum number of linear regions proposed by Montufar *et al.* [70] (Eq. (3)). Given a deep ReLU neural network with l layers, let m_i be the width of i-th hidden layer with $m_i \geq 3d$, d is the input dimension. Serra *et al.* [92] prove that the maximal number of linear regions of this neural network is lower bounded by

$$MEC(\mathcal{N}) \geq (\prod_{i=1}^{l-1} (\lfloor \frac{m_i}{d} \rfloor + 1)^d) \sum_{j=0}^{d} \binom{m_l}{j}$$

and is upper bounded by

$$MEC(\mathcal{N}) \leq \sum_{(j_1,\ldots,j_{l+1}) \in J} \prod_{i=1}^{l+1} \binom{m_i}{j_i}$$

where $J = \{(j_1, \ldots, j_{l+1}) \in \mathbb{Z}^{l+1} : 0 \leq j_i \leq \min\{d, m_1 - j_1, \ldots, m_{i-1} - j_{i-1}, m_i\}\}$.

Bianchini and Scarselli [14, 15] design a topological measure of model complexity for deep neural networks. Given an FCNN with single output, denoted by $\mathcal{N} : \mathbb{R}^d \to \mathbb{R}$, they define $S = \{x \in \mathbb{R}^d | \mathcal{N}(x) \geq 0\}$ the set of instances being classified by $\mathcal{N}$ to the positive class. The model complexity of neural network $\mathcal{N}$ is measured by $B(S)$, the sum of the Betti numbers of set S, that is,

$$MEC(\mathcal{N}) = B(S) = \sum_{i=0}^{d-1} b_i(S)$$

where $b_i(S)$ denotes the i-th Betti number that counts the number of $(i+1)$-dimensional holes in S. The Betti number is generally used to distinguish between spaces with different characteristics in algebraic topology [18]. S contains instances positively classified by the network $\mathcal{N}$. Therefore, $B(S)$ can be used

Table 3: Upper and lower bounds of $B(S)$ given by Bianchini and Scarselli [14, 15], for networks with M hidden units, d inputs and l hidden layers.

Inputs	Hidden Layers	Activation Function	Bound
		Upper Bounds	
d	1	threshold	$O(M^d)$
d	1	arctan	$O((d+M)^{d+2})$
d	1	polynomial, degree r	$\frac{1}{2}(2+r)(1+r)^{d-1}$
1	1	arctan	M
d	many	arctan	$2^{M(2M-1)}O((dl+d)^{d+2M})$
d	many	tanh	$2^{M(M-1)/2}O((dl+d)^{d+M})$
d	many	polynomial, degree r	$\frac{1}{2}(2+r^l)(1+r^l)^{d-1}$
		Lower Bounds	
d	1	any sigmoid	$(\frac{M-1}{d})^d$
d	many	any sigmoid	2^{l-1}
d	many	polynomial, degree $r \geq 2$	2^{l-1}

to investigate how S is affected by the architecture of $\mathcal{N}$ and can represent the model complexity.

Bianchini and Scarselli [14, 15] report upper and lower bounds of $B(S)$ for a series of network architectures (Table 3). In particular, Bianchini and Scarselli [14, 15] demonstrate that $B(S)$ of a single-hidden-layer network grows polynomially with respect to the hidden layer width. That is, $B(S) \in O(m^d)$, where m is the width of the hidden layer. They also prove that $B(S)$ of a deep neural network grows exponentially in the total number of hidden neurons. That is, $B(S) \in O(2^M)$, where M is the total number of hidden neurons. This indicates that deep neural networks have higher expressive capacity, and therefore are able to learn more complex functions than shallow ones.

Bianchini and Scarselli [14, 15] suggest that the layer-wise composition mechanism of a deep model allows the model to replicate the same behavior in different regions of the input space, thus makes depth more effective than width.

3.2 Width Efficiency

In addition to depth efficiency, the effect of width on expressive capacity, namely width efficiency, is also worth exploring. Width efficiency analyzes how width affects the expressive capacity of deep learning models [64]. Width efficiency is important for fully understanding expressive capacity and helps to validate the insights obtained from depth efficiency [64].

Lu *et al.* [64] investigate the width efficiency of neural networks with ReLU activation function. They extend the universal approximation theorem [7, 43] to width-bounded deep ReLU neural networks. The classical universal approxima-

tion theorem [7, 43] states that, one-hidden-layer neural networks with certain activation functions (e.g., ReLU) can approximate any continuous functions on a compact domain to any desired performance. Lu *et al.* [64] show that, for any Lebesgue-integrable function $f : \mathbb{R}^d \to \mathbb{R}$ and any $\epsilon > 0$, there exists a ReLU network $\mathcal{N} : \mathbb{R}^d \to \mathbb{R}$ that can approximate f to ϵ L^1-distance. That is,

$$\int_{\mathbb{R}^d} |f(x) - F_{\mathcal{N}}(x)|dx < \epsilon.$$

Here $F_{\mathcal{N}}$ is the function represented by the neural network $\mathcal{N}$. The width m_i of each hidden layer of $\mathcal{N}$ satisfies $m_i \leq d + 4$.

Moreover, to explore the role of layer width in expressive capacity quantitatively, Lu *et al.* [64] raise the dual question of depth efficiency. That is, whether there exists wide, shallow ReLU networks that cannot be approximated by any narrow, deep neural network whose size is not substantially increased. Denote by $F_{\mathcal{A}} : \mathbb{R}^d \to \mathbb{R}$ the function represented by a ReLU neural network $\mathcal{A}$ whose depth $h = 3$ and whose width of each layer is $2k^2$, where k is an integer such that $k \geq d + 4$. Let $F_{\mathcal{B}} : \mathbb{R}^d \to \mathbb{R}$ be the function represented by a ReLU neural network $\mathcal{B}$ whose depth $h \leq k + 2$ and whose width of each hidden layer $m_i \leq k^{3/2}$, where the parameter values of $\mathcal{B}$ are bounded by $[-b, b]$. For any constant $b > 0$, there is $\epsilon > 0$ such that $F_{\mathcal{A}}$ can never be approximated by $F_{\mathcal{B}}$ to ϵ L^1-distance. That is,

$$\int_{\mathbb{R}^d} |F_{\mathcal{A}}(x) - F_{\mathcal{B}}(x)|dx \geq \epsilon.$$

This indicates that there exists a family of shallow ReLU neural networks which cannot be approximated by narrow networks whose depth is constrained by polynomial bounds.

The polynomial lower bound for width efficiency is smaller than the exponential lower bound for depth efficiency [13, 15, 70]. That is, to approximate a deep model whose depth increases linearly, a shallow model requires at least an exponential increase in width. To approximate a shallow, wide model whose width increases linearly, a deep, narrow model requires at least a polynomial increase in depth. However, Lu *et al.* [64] point out that depth cannot be strictly proved to be more effective than width since a polynomial upper bound for width is still lacking. The polynomial upper bound ensures that to approximate a shallow, wide model, a deep, narrow one requires at most a polynomial increase in depth.

3.3 Expressible Functional Space

In addition to the studies of depth efficiency and width efficiency, the third line of works explores the classes of functions that can be expressed by deep learning models with specific frameworks and specified size. This line of works explores the expressible functional space of deep learning models. In general, there are two subcategories of this line: model-specific approaches and cross-model approaches, according to our grouping criterion in Section 2.3.1.

Arora *et al.* [3] investigate the family of functions representable by deep neural networks with ReLU activation function. They prove that every piecewise linear function $f : \mathbb{R}^d \to \mathbb{R}$ can be represented by a ReLU neural network that consists of at most $\lceil \log_2(d+1) \rceil$ hidden layers. The family of piecewise linear functions is dense in the family of compactly supported continuous functions, and the family of compactly supported continuous functions is dense in Lebesgue space $L^p(\mathbb{R}^d)$ [3, 23]. The Lebesgue space $L^p(\mathbb{R}^d)$ is defined as the the family of Lebesgue intergrable functions f for which $\int_{\mathbb{R}^d} |f| < +\infty$ [23]. Define L^p norm as $||f||_p = [\int_{\mathbb{R}^d} |f|^p]^{1/p}$ [23]. Then, the above conclusion can be extended to the $L^p(\mathbb{R}^d)$-space. That is, every function $f \in L^p(\mathbb{R}^d)$ can be approximated to arbitrary L^p norm by a ReLU neural network which consists of at most $\lceil \log_2(d+1) \rceil$ hidden layers.

Gühring *et al.* [36] study the expressive capacity of deep neural networks with ReLU activation function in Sobolev space [1]. Given $\Omega \subset \mathbb{R}^d$, $p \in [1, \infty]$, $n \in \mathbb{N}$, $L^p(\Omega)$ is the Lebesgue space on Ω, the Sobolev space [84] is defined as

$$W^{n,p}(\Omega) = \{f \in L^p(\Omega) : D^\alpha f \in L^p(\Omega) \text{ for } \forall \alpha \in \mathbb{N}_0^d, |\alpha| \leq n\},$$

where $D^\alpha f$ is the α-th order derivative of f. Sobolev norm is defined as

$$||f||_{W^{n,p}(\Omega)} = (\sum_{0 \leq |\alpha| \leq n} ||D^\alpha f||^p_{L^p(\Omega)})^{1/p}.$$

Gühring *et al.* [36] analyze the effect of ReLU neural networks in approximating functions from Sobolev space, and establish upper and lower bounds on the sizes of models to approximate functions in the Sobolev space. Specifically, define a subset of Sobolev space f as

$$\mathcal{F}_{n,d,p,B} = \{f \in W^{n,p}((0,1)^d) : ||f||_{W^{n,p}((0,1)^d)} \leq B\}.$$

The upper bound shows that, for any $d \in \mathbb{N}$, $n \in \mathbb{N}, n \geq 2$, $\varepsilon > 0$, $0 \leq s \leq 1$, $1 \leq p \leq +\infty$, and $B > 0$, for any function $f \in \mathcal{F}_{n,d,p,B}$, there exists a neural network $\mathcal{N}_\varepsilon$ and a choice of weights w_f such that

$$||\mathcal{N}_\varepsilon(\cdot|w_f) - f(\cdot)||_{W^{s,p}((0,1)^d)} \leq \varepsilon$$

where $\mathcal{N}_\varepsilon$ represents a ReLU neural network consisting of at most $c\log_2(\varepsilon^{-\frac{n}{n-s}})$ layers and $c\varepsilon^{-\frac{d}{n-s}}\log_2(\varepsilon^{-\frac{n}{n-s}})$ neurons, and with non-zero parameters. The value of constant c depends on the value of d, p, n, s and B.

Besides, the lower bound [36] shows that, for any $d \in \mathbb{N}$, $n \in \mathbb{N}, n \geq 2$, $\varepsilon > 0$, $B > 0$, $k \in \{0, 1\}$, with $p = +\infty$, for any function $f \in \mathcal{F}_{n,d,p,B}$, there exists a ReLU neural network $\mathcal{N}_\varepsilon$ such that

$$||\mathcal{N}_\varepsilon(\cdot|w_f) - f(\cdot)||_{W^{k,\infty}((0,1)^d)} \leq \varepsilon$$

where $\mathcal{N}_\varepsilon$ has at least $c'\varepsilon^{-\frac{d}{2(n-1)}}$ non-zero weights.

Kileel *et al.* [52] explore the functional space of deep neural networks with polynomial activation functions. A polynomial activation function $\rho_r(z)$ raises z to the power of r. They suggest that, with polynomial activation functions, the study of model complexity can be benefitted from the application of powerful mathematical machinery of algebraic geometry. In addition, polynomials can approximate any continuous activation function, thus help to explore other deep learning models.

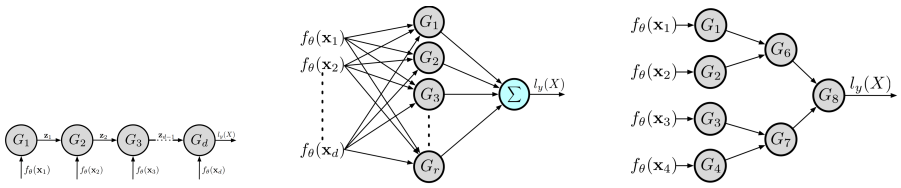
Given a deep polynomial neural network $\mathcal{N} : \mathbb{R}^d \to \mathbb{R}^c$ with depth h and polynomial degree r, let $m = \{m_0, m_1, \ldots, m_h\}$ represent the architecture of $\mathcal{N}$ with the width m_i of layer i and $m_0 = d$, $m_h = c$. Let $\mathcal{F}_{m,r}$ be the functional space of $\mathcal{N}$. Kileel *et al.* [52] define the functional variety $\mathcal{V}_{m,r}$ as $\mathcal{V}_{m,r} = \overline{\mathcal{F}_{m,r}}$, which is the Zariski closure of the functional space $\mathcal{F}_{m,r}$. "The Zariski closure of a set X is the smallest set containing X that can be described by polynomial equations." [52] The functional variaty $\mathcal{V}_{m,r}$ can be much larger than the functional space $\mathcal{F}_{m,r}$. Kileel *et al.* [52] propose to use the dimensionality of functional variety, denoted by $\dim \mathcal{V}_{m,r}$, as the representation of the expressive capacity of deep polynomial neural networks, written as

$$MEC(\mathcal{N}_{m,r}) = \dim \mathcal{V}_{m,r}.$$

To measure $\dim \mathcal{V}_{m,r}$, Kileel *et al.* [52] build connections between deep polynomial networks and tensor decompositions. Specifically, polynomial networks with $h = 2$ are connected to CP tensor decompositions [56], and deep polynomial networks are connected to an iterated tensor decomposition [65]. Based on decompositions, they prove that, for any fixed m, there exists $\tilde{r} = r(m)$ such that for any $r > \tilde{r}$, $\dim \mathcal{V}_{m,r}$ is bounded by

$$\dim \mathcal{V}_{m,r} \leq \min\left(m_h + \sum_{i=1}^{h}(m_{i-1} - 1)m_i, m_h\binom{m_0 + r^{h-1} + 1}{r^{h-1}}\right).$$

Besides, Kileel *et al.* [52] prove a bottleneck property of deep polynomial networks. That is, a too narrow layer is a bottleneck and may "chock" the



(a) TT-decomposition (b) CP-decomposition (c) HT-decomposition

Figure 3: Examples of networks corresponding to various tensor decompositions [51], from the left are TT decomposition, CP-decomposition, HT-decomposition.

polynomial network such that the network can never fill the ambient space. The ambient space $Sym_r(\mathbb{R}^d)$ is the space of homogeneous polynomials of degree r in d variables. A polynomial network filling the ambient space satisfies $\mathcal{F}_{m,r} = Sym_{r^{h-1}}(\mathbb{R}^d)^c$. They show that, network architectures filling the ambient space can be helpful for optimization and training.

In addition to model-specific approaches, expressible functional space can be investigated in a cross-model manner. Specifically, Khrulkov *et al.* [51] study the expressive capacity of recurrent neural networks (RNNs). They investigate the connections between network architectures and tensor decompositions, then make comparison between the expressive capacity of RNNs, CNNs, and shallow FCNNs.

Let $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$ be a d-dimensional tensor. The Tensor Train (TT) decomposition [83] of a tensor $\mathcal{X}$ is computed by

$$\mathcal{X}^{i_1 i_2 \ldots i_d} = \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_{d-1}=1}^{r_{d-1}} G_1^{i_1 \alpha_1} G_2^{\alpha_1 i_2 \alpha_2} \ldots G_d^{\alpha_{d-1} i_d}$$

where the tensors $G_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ are called TT-cores. Khrulkov *et al.* [51] introduce bilinear units to represent TT-cores. Given $x \in \mathbb{R}^m, y \in \mathbb{R}^n$ and $G \in \mathbb{R}^{m \times n \times k}$, a bilinear unit performs a map $G : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^k$, written as $G(x, y) = z$. Based on the bilinear units, they show that a recurrent neural network realizes the TT-decomposition of the weight tensor (Figure 3(a)). Similarly, Khrulkov *et al.* [51] show that the Canonical (CP) decomposition [24], with the form of

$$\mathcal{X}^{i_1 i_2 \ldots i_d} = \sum_{\alpha=1}^{r} v_{1,\alpha}^{i_1} v_{2,\alpha}^{i_2} \ldots v_{d,\alpha}^{i_d},$$

corresponds to a one-hidden-layer FCNN (Figure 3(b)). Each unit of the network is denoted by $G_\alpha = v_{1,\alpha}^{i_1} v_{2,\alpha}^{i_2} \ldots v_{d,\alpha}^{i_d}$, where $v_{i,\alpha} \in \mathbb{R}^{n_i}$. The Hierarchical Tucker (HT) decomposition [27] corresponds to an CNN structure (Figure 3(c)).

Table 4: Comparison of the expressive power of various network architectures given by Khrulkov *et al.* [51]. Each column represents a specific network with width r, rows show the upper bounds on the width of other types of networks to achieve equivalent expressive capacity.

	TT-Network	HT-Network	CP-Network
TT-Network	r	$r^{\log_2(d)/2}$	r
HT-Network	r^2	r	r
CP-Network	$\geq r^{d/2}$	$\geq r^{d/2}$	r

Khrulkov *et al.* [51] propose the rank of tensor decomposition as a measure of neural network complexity, since the rank of decompositions corresponds to the width of networks. Based on the correspondence relationships between neural networks and tensor decompositions, they compare the model complexity of RNNs, CNNs, and shallow FCNNs. The main conclusions are summarized in Table 4. In particular, given a random d-dimensional tensor whose TT-decomposition is with rank r and mode size n, this tensor has exponentially large ranks of CP-decomposition and HT-decomposition. That tells, to approximate a recurrent neural network, a shallow FCNN or CNN requires an exponentially larger width.

3.4 VC Dimension and Rademacher Complexity

The VC dimension and Rademacher complexity are widely used to analyze the expressive capacity and generalization of classical parametric machine learning models [8, 12, 26, 69, 95]. A series of works investigate the VC dimension and Rademacher complexity of deep learning models.

The VC dimension is an expressive capacity measure that reflects the largest number of samples that can be shattered by the hypothesis space [69]. A higher VC dimension means the model can shatter a larger number of samples and thus the model has a higher expressive capacity. Maass [66] studies the VC dimension of feedforward neural networks with linear threshold gates. The linear threshold gate means that each neuron is composed of a weighted sum function followed by a Heaviside activation function (Eq. (2)). Let W be the number of parameters in the network and L be the depth of the network. Maass [66] proves that for $L \geq 3$, the VC dimension of such networks is $\Theta(W \log W)$.

Bartlett *et al.* [11] investigate the VC dimension of feedforward neural networks with piecewise polynomial activation functions. A piecewise polynimial activation function with p pieces has the form $\sigma(z) = \phi_i(z)$, where $z \in [t_{i-1}, t_i)$, $i \in 1, \ldots, p+1$, and $t_{i-1} < t_i$. Each ϕ_i is a polynomial function of degree no more than r. Let W be the number of parameters in the network and L be the

depth of the network. Bartlett *et al.* [11] prove that the upper bound for the VC dimension of such networks is $O(WL^2 + WL\log WL)$ and the lower bound for the VC dimension is $\Omega(WL)$. Later, Bartlett *et al.* [10] improve this lower bound to $\Omega(WL\log(W/L))$.

Bartlett *et al.* [10] study the VC dimension for deep neural networks with piecewise linear activation functions (e.g., ReLU). Given a deep neural network with L layers and W parameters, they prove that the lower bound for VC dimension of such networks is $\Omega(WL\log(W/L))$ and the upper bound for VC dimension is $O(WL\log W)$.

Rademacher complexity captures the capacity of a hypothesis space to fit random labels as a measure of the expressive capacity [69]. A higher Rademacher complexity means that the model can fit a larger number of random labels and thus the model has a higher expressive capacity. Bartlett *et al.* [9] investigate the Rademacher complexity of deep neural networks with ReLU activation function. Given a deep ReLU neural network with L layers, let A_i be the parameter matrix of layer i, and $X \in \mathbb{R}^{n\times d}$ the data matrix, where n is the number of instances and d is the input dimension. Bartlett *et al.* [9] prove that the lower bound for the Rademacher complexity of such networks is $\Omega(||X||_F \prod_i ||A_i||_\sigma)$, where $||\cdot||_\sigma$ is the Spectral norm, and $||\cdot||_F$ is the Frobenius norm. Neyshabur *et al.* [77] prove a tighter lower bound for two-layer ReLU neural networks. Suppose $||A_1||_\sigma \leq s_1$, $||A_2||_\sigma \leq s_2$, and $s_1 s_2$ is the Lipschitz bound of the function represented by the network. They prove that the Rademacher complexity is lower bounded by $\Omega(s_1 s_2\sqrt{m}||X||_F/n)$, where m is the width of the hidden layer. This lower bound improves the bound [9] by a factor of $\sqrt{m}$.

Yin *et al.* [99] study the Rademacher complexity of adversarial trained neural networks. Given a feedforward neural network with ReLU activation function, denoted by f, let L be the depth of the network, and A_i the parameter matrix of layer i. The function family represented by f with adversarial loss function can be written as

$$\mathcal{F} = \{f_A : \min_{x'\in\mathbb{B}(\epsilon)} y f_A(x), \prod_{i=1}^{L} ||A_i||_\sigma \leq r\}$$

where $\mathbb{B}(\epsilon) = \{x' \in \mathbb{R}^d : ||x' - x|| \leq \epsilon\}$ is the set of samples perturbed around x with l_∞ distance $\leq \epsilon$. Yin *et al.* [99] prove that the lower bound for the Rademacher complexity of $\mathcal{F}$ is $\Omega(||X||_F/n + \epsilon\sqrt{d/n})$. This lower bound exhibits an explicit dependence on the input dimension d.

Some studies [75, 100] suggest that deep learning models are often over-parameterized in practice and have significantly more parameters than samples. In this case, the VC dimension and Rademacher complexity of deep learning models are always too high, so the practical guidance they can provide is weak.

4 Effective Complexity of Deep Learning Models

Effective complexity of deep learning models is also known as practical complexity, practical expressivity, and usable capacity [37, 81]. It reflects the complexity of the functions represented by deep models with specific parameterizations [89].

Effective complexity of deep learning models has been mainly explored from two different aspects.

- **General measures of effective complexity** design quantitative measures for effective complexity of deep learning models.
- Investigations into the **high-capacity low-reality phenomenon** find that effective complexity of deep learning models may be far lower than their expressive capacity.

In this section, we review these two groups of studies.

4.1 General Measures of Effective Complexity

Compared to expressive capacity, the study of effective model complexity has stronger requirements for sensitive and precise complexity measures. This is because the effective complexity cannot be directly derived from the model structure alone [44]. Different parameter values of the same model structure may lead to different effective complexity. An effective complexity measure is expected to be sensitive to different parameter values used in models with the same structure.

A series of works devote to proposing feasible measures for effective complexity of deep learning models. A major group of the complexity measures depends on the linear region splitting of piecewise linear neural networks in the input space. We start with those methods and then discuss the others.

4.1.1 Piecewise Linear Property

It is well known that a neural network with piecewise linear activation functions generates a finite number of linear regions in the input space [44, 70, 89]. This property is called the *piecewise linear property* and is demonstrated in Fig. 4. The number of linear regions as well as the density of such regions can usually reflect the effective complexity. Therefore, a series of studies on effective complexity starts from piecewise linear activation functions (e.g., ReLU, Maxout) or are based on the piecewise linear property.

Raghu *et al.* [89] propose two interrelated effective complexity measures for deep neural networks with piecewise linear activation functions, especially ReLU

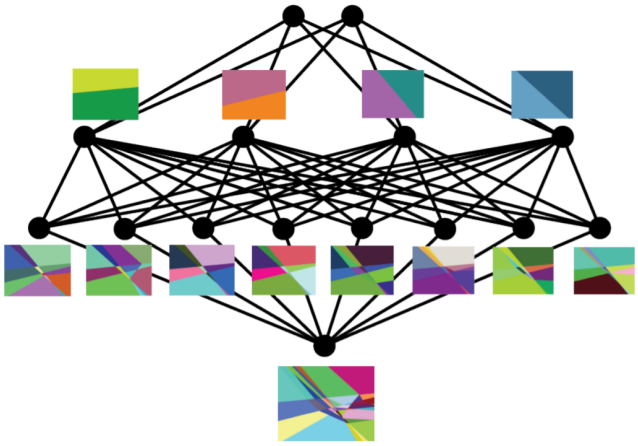


Figure 4: A 2-hidden-layer ReLU network divides the 2-dimensional input space into a number of linear regions, studied by Hanin and Rolnick [37]. Upon each hidden neuron shows the linear regions divided by the current neuron.

and hard Tanh [82]. Specifically, they define trajectory $x(t)$ between two input points $x_1, x_2 \in \mathbb{R}^d$, which is a curve parametrized by a scalar $t \in [0, 1]$, with $x(0) = x_1$ and $x(1) = x_2$. The first effective complexity measure they propose is the number of linear regions in the input space when sweeping an input point through a trajectory path $x(t)$, that is, the effective complexity $EMC(\mathcal{N})$ of model $\mathcal{N}$ is

$$EMC(\mathcal{N}) = \mathcal{T}(\mathcal{N}(x(t); W))$$

where $\mathcal{T}$ is the number of linear regions passing through the trajectory $x(t)$ and W is a specific set of model parameters. The second effective complexity measure they propose is the trajectory length $l(x(t))$ defined as

$$l(x(t)) = \int_t ||\frac{dx(t)}{dt}||dt$$

which is the standard *arc length* of the trajectory $x(t)$. They prove the proportional relationship between these two complexity measures. To obtain the estimated value of effective complexity, Raghu *et al.* [89] bound the expected value of trajectory length of any layer in a deep ReLU neural network. Specifically, given a deep ReLU neural network whose weights are initialized by $N(0, \sigma_w^2/m)$ and the biases are initialized by $N(0, \sigma_b^2)$. Let $z^{(i)}(x(t))$ denote the new trajectory obtained after the transformation of the first i hidden layers of the trajectory

Figure 5: A circular trajectory (the left most) is fed into a Tanh neural network, then the following images show the trajectory after the transformation of each hidden layer in turn [89]. It shows the increase of the length of the trajectory after the layer transformation.

$x(t)$. The expected value of its trajectory length can be bounded by

$$\mathbb{E}[l(z^{(i)}(x(t)))] \geq O(\frac{\sigma_w\sqrt{m}}{\sqrt{m+1}})^i l(x(t))$$

where m denotes the hidden layer width. Similarly, for a hard Tanh neural network under the same initialization, the trajectory length is bounded by

$$\mathbb{E}[l(z^{(i)}(x(t)))] \geq O\left(\frac{\sigma_w\sqrt{m}}{\sqrt{\sigma_w^2+\sigma_b^2+m\sqrt{\sigma_w^2+\sigma_b^2}}}\right)^i l(x(t)).$$

Using these two complexity measures, Raghu *et al.* [89] explore the performance of deep neural networks and report some findings. First, the effective complexity grows exponentially with respect to the depth of the model and polynomially with respect to the width. Fig. 5 is an example showing the evolution of a trajectory after each hidden layer of a deep network. Second, how the parameters are initialized affects the effective complexity. Third, injecting perturbations to a layer leads to exponentially larger perturbations at the remaining layers. Last, regularization approaches, such as Batch Normalization [45], help to reduce trajectory length. This explains why Batch Normalization helps model stability and generalization.

Novak *et al.* [81] empirically investigate the relationship between model complexity and generalization of neural networks with piecewise linear activation functions. They propose to use model sensitivity to measure effective complexity. Model sensitivity, also known as robustness, reflects the capacity of a model to distinguish between different inputs at small distances. Novak *et al.* [81] introduce two sensitivity metrics, input-output Jacobian norm and trajectory length [89].

First, based on the piecewise linear property, they propose the Jacobian norm to measure the local sensitivity when the input is perturbed within the

same linear region, further represent the effective model complexity. That is,

$$EMC(\mathcal{N}) = \mathbb{E}_{x\sim D}[||J(x)||_F]$$

where $J(x) = \partial f_{\mathcal{N}}(x)/\partial x^T$ is the Jacobian of class probabilities, $f_{\mathcal{N}}$ is the function represented by the network $\mathcal{N}$, D is the data distribution, and $||\cdot||_F$ is the Frobenius norm.

Second, they propose a trajectory length metric similar to what developed by Raghu *et al.* [89] as a sensitivity measure when the input is perturbed to other linear regions, further as a measure of effective complexity, written as

$$EMC(\mathcal{N}) = \mathbb{E}_{x\sim D}[t(x)]$$

where $t(x)$ is a trajectory length defined by the number of linear regions passing through a trajectory $\tau(x)$, that is,

$$\begin{aligned} t(x) &= \sum_{i=0}^{k-1} ||c(z_i) - c(z_{(i+1)\%k})||_1 \\ &\approx \oint_{z\in\tau(x)} ||\frac{\partial c(z)}{\partial(dz)}||_1 dz \end{aligned}$$

where $z_0, \ldots, z_{k-1}$ are k equidistant points on the trajectory $\tau(x)$, $c(z)$ is the status encoding of all hidden neurons at point z.

Using these two complexity measures, Novak *et al.* [81] study the correlation between complexity and generalization. They demonstrate that neural networks have strong robustness in the vicinity of the training data manifold where deep models have good generalization capability. They also show that the factors associated with poor generalization (e.g., full-batch training, random labels) correspond to weaker robustness, and the factors associated with good generalization (e.g., data augmentation, ReLU) correspond to stronger robustness.

To develop a measure of effective complexity for general smooth activation functions, Hu *et al.* [44] propose an effective complexity measure for deep neural networks with curve activation functions (e.g., Sigmoid [53], Tanh [48]). Motivated by the piecewise linear property, they suggest that, using a piecewise linear function with a minimal number of linear regions to approximate a given network, the number of linear regions of the approximation function can be a measure of effective complexity of the given network. They learn a piecewise linear approximation of a deep neural network with curve activation functions, which is called the Linear Approximation Neural Network (LANN for short). The LANN is constructed by learning a piecewise linear approximation function for the curve activation function on each hidden neuron. Specifically, Hu *et al.* [44] define the approximation error of a LANN g to the target network

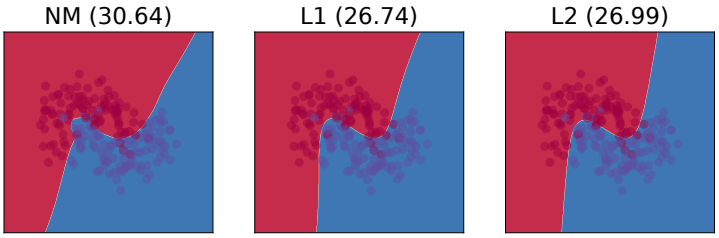


Figure 6: Influence of regularization approaches on effective complexity [44]. This figure shows that the decision boundaries of models trained on the Moon dataset. NM, L1, L2 are short for normal train, train with L^1, and L^2 regularization, respectively. In brackets are the value of effective complexity measure.

f as $\mathcal{E}(g; f) = \mathbb{E}[|g(x) - f(x)|]$. They analyze how the approximation error at a certain layer is propagated through the remaining hidden layers, and obtain the influence of the approximation error of each hidden neuron on $\mathcal{E}(g; f)$. That is,

$$\mathcal{E}(g; f) = \sum_{i,j} \frac{1}{c} \sum (\ |V_o| \prod_{q=L}^{i+1} \mathbb{E}[|J_q|] \ )_{*,j} (\mathbb{E}[e_{i,j}] + \mathbb{E}[|\hat{\epsilon}_{i,j}|]) \tag{4}$$

where J_q is the Jacobian matrix of layer q of the network f, $e_{i,j}$ is the approximation error of g on a specific neuron $\{i, j\}$, V_o is the weight matrix of the output layer, and $\hat{\epsilon}_{i,j}$ is the negligible estimation error on neuron $\{i, j\}$. Given an approximation degree λ, the LANN with the smallest number of linear regions is constructed to meet the requirement $\mathcal{E}(g; f) \leq \lambda$. The approximated number of linear regions of the LANN is used as the effective complexity measure, that is,

$$EMC(f)_\lambda = d \sum_{i=1}^{L} \log(\sum_{j=1}^{m_i} k_{i,j} - m_i + 1) \tag{5}$$

where $k_{i,j}$ is the number of linear pieces of the approximation function on neuron $\{i, j\}$, m_i is the width of layer i, and L is the depth of f.

Using the complexity measure, Hu *et al.* [44] investigate the trend of model complexity in the training process. They show that the effective complexity increases with respect to the number of training iterations. They also demonstrate that the occurrence of overfitting is positively correlated with the increase of effective complexity, while regularization methods (e.g., L^1, L^2 regularizations) suppress the increase of model complexity (see Fig. 6).

The piecewise linear property may provide novel opportunities for capturing model complexity in deep learning. In addition to the above studies on effective complexity, piecewise linear neural networks or the piecewise linear property can also help the exploration of expressive capacity (see Section 2, [4, 36, 64, 70]).

Piecewise linear activation functions, especially ReLU, are popular and effective activation functions in many tasks and applications [58, 70]. The local linear characteristic and a finite number of regional divisions facilitate quantifying and analyzing neural network model complexity with piecewise linear activation functions.

4.1.2 Other Measure Metrics

There are other effective complexity measures based on ideas other than the piecewise linear property.

Nakkiran *et al.* [74] introduce an effective complexity measure by investigating the double descent phenomenon. The double descent phenomenon of deep neural networks is that, as the model size, training epochs, or size of training data increases, the test performance often first decreases and then increases. They suggest that to help capture the double descent effect in training, a complexity measure should be sensitive to training processes, data distribution, and model architectures. They propose such an effective complexity measure, that is, the maximum number of samples on which the model can be trained to close to zero training error, written as

$$EMC_{D,\epsilon}(\mathcal{T}) = \max \{n | \mathbb{E}_{S\sim D}[Error_S(\mathcal{T}(S))] \leq \epsilon\} \tag{6}$$

where D is the data distribution, ϵ is the threshold of training error, $\mathcal{T}$ is the training procedure, $Error_S$ is the mean error of the model on the training set S, and n the size of training set S. Based on the effective complexity measure $EMC_{D,\epsilon}(\mathcal{T})$, they investigate the evolution of the training process and show that, if $EMC_{D,\epsilon}(\mathcal{T})$ is sufficiently smaller or sufficiently larger than n, the size of the training set, then any perturbation of $\mathcal{T}$ that increases its effective complexity $EMC_{D,\epsilon}(\mathcal{T})$ helps to improve the test performance. However, if $EMC_{D,\epsilon}(\mathcal{T}) \approx n$, then any perturbation of $\mathcal{T}$ that increases its effective complexity $EMC_{D,\epsilon}(\mathcal{T})$ hurts the test performance.

To approach the generalization problem of deep learning models, Liang *et al.* [60] introduce a new notion of model complexity measure, the Fisher-Rao norm. Their study focuses on deep fully connected neural networks whose activation function σ satisfies $\sigma(z) = \sigma'(z)z$. The model complexity represented by Fisher-Rao norm is given by

$$EMC(\mathcal{N}) = ||\theta||_{fr}^2 = \langle \theta, I(\theta)\theta \rangle \tag{7}$$

where θ is the set of parameters (i.e., weights, bias) of neural network $\mathcal{N}$, $\langle \cdot, \cdot \rangle$ is the inner product, and $I(\theta)$ is the Fisher information matrix, written as

$$I(\theta) = \mathbb{E}[\nabla_\theta \ell(\mathcal{N}(X), Y) \otimes \nabla_\theta \ell(\mathcal{N}(X), Y)] \tag{8}$$

where $\ell(\cdot, \cdot)$ is the loss function and $\otimes$ is the tensor product. Liang *et al.* [60] introduce the geometric invariance property that a complexity measure should satisfy in order to study generalization capability. The invariance property essentially says that, since many different continuous operations may lead to exactly the same prediction, thus the generalization should only depend on the equivalence classes obtained by these continuous transformations. Specific parameterization should not affect the generalization. The complexity measure used to investigate generalization should satisfy the invariance property. They demonstrate that this Fisher-Rao norm satisfies the invariance property. Let θ_1, θ_2 denote two parameter settings of a model f. If $f_{\theta_1} = f_{\theta_2}$, then their Fisher-Rao norms are equal, that is, $||\theta_1||_{fr} = ||\theta_2||_{fr}$. In particular, the Fisher-Rao norm remains invariant during the node-wise rescaling of a model.

Effective complexity can be measured by the size of training samples that a model achieves zero training error [74], or by the Fisher-Rao metric [60]. More effective complexity measures along the line may be developed.

4.2 High-Capacity Low-Reality Phenomenon

Several studies explore the gap between the effective complexity and the expressive capacity of deep learning models.

Ba and Caruana [5] show that shallow fully connected neural networks can learn complex functions previously learned by deep neural networks, sometimes even only requiring the same number of parameters as the deep networks. Specifically, given a well-trained deep model, they propose to train a shallow model based on the outputs of the deep model, to mimic the deep model. They show that, the shallow mimic model can achieve an accuracy as high as the deep model. However, the shallow model cannot be trained directly on the original labeled training data to achieve the same accuracy. This is also well recognized as knowledge distillation [41].

Based on this phenomenon, Ba and Caruana [5] conjecture that the strength of deep learning may arise in part from a good match between deep architectures and current training algorithms. That is, compared with shallow architectures, deep architectures may be easier to train by the current optimization techniques. Moreover, they propose that when it is able to mimic the function learned by a deep complex model using a shallow model, the function learned by the deep model is not really too complicated to learn. This study suggests that there may be a big gap between the practical effective complexity of a deep learning model and the theoretical bound of its expressive capacity. We call this the *high-capacity low-reality phenomenon*.

Hanin and Rolnick [37] investigate the high-capacity low-reality phenomenon in fully connected neural networks with piecewise linear activation functions, especially ReLU. They propose two effective complexity measures using the num-

ber of linear regions in the input space and the volume of boundaries between these linear regions.

First, they investigate ReLU neural networks whose dimensionality of input and that of output are both equal to 1, and use the number of linear regions as the effective complexity measure. They show that the average number of linear regions grows linearly with respect to the total number of neurons, far below the exponential upper bound. Specifically, given a ReLU neural network $\mathcal{N} : \mathbb{R} \to \mathbb{R}$ whose weights and biases of neurons z are randomly initialized and are bounded by $\mathbb{E}[||\nabla z(x)||] \leq C$ for some $C > 0$, the average number of linear regions is proportional to the product of the number of neurons and the size of training set, that is,

$$\mathbb{E}[\#\{linear\ regions\ in\ S\}] \approx |S| \cdot T \cdot M \tag{9}$$

where S is the training dataset, T is the number of breakpoints in the activation function (for ReLU, $T = 1$), and M is the total number of hidden neurons.

Second, they investigate ReLU neural networks whose input dimensionality exceeds 1, denoted by $\mathcal{N} : \mathbb{R}^d \to \mathbb{R}$ $(d > 1)$, and use the volume of boundaries between linear regions in the input space as an estimation of model complexity. That is,

$$EMC(\mathcal{N}) = \frac{volume_{d-1}(B_{\mathcal{N}} \cap K)}{volume_d(K)} \tag{10}$$

where $B_{\mathcal{N}} = \{x|\nabla\mathcal{N}(x)$ is not continuous at $x\}$ represents the boundaries of the linear regions formed by $\mathcal{N}$, $K \in \mathbb{R}^d$ is the data distribution. They prove that, under the same parameter initialization assumption as $d = 1$, the expected value of the volume of linear region boundaries is approximately equal to

$$\mathbb{E}[EMC(\mathcal{N})] \approx T \cdot M \tag{11}$$

This demonstrates that the average size of the region boundaries depends only on the number of neurons, not on the depth of the model. They conclude that the effective complexity of deep neural networks may be much lower than the theoretical bound. That is, the function learned by deep neural networks may not be more complex than that learned by shallow ones.

4.3 Discussion

Effective model complexity is a relatively new, promising and useful problem in deep learning. Detecting effective model complexity during training helps to investigate the usefulness of optimization algorithms [47], the role of regularizations [44, 89], and generalization capability [60, 81]. Furthermore, effective model complexity can be used to describe model compression ratio, since ef-

fective model complexity can be considered as a reflection of the information volume in the model [32]. Effective complexity can also be used for model selection and design to balance resource utilization and model performance.

In addition to the effective complexity measures and the high-capacity low-reality phenomenon, there are a series of interesting problems about effective complexity of deep learning models. For example, the cross-model comparison of effective complexity is worth exploring. That is, how to compare the effective complexity of multiple models with different architectures, and how the effective complexity is affected by the choice of different model architectures. Moreover, can one specify the granularity of effective complexity measures? Different cases may have different requirements for effective complexity. Correspondingly, the application scopes and granularities of effective complexity measures should be specified and clarified. Typically, effective complexity measure by the number of non-zero parameters is obviously not sufficient to study the optimization process.

5 Application Examples of Deep Learning Model Complexity

Model complexity of deep learning has many applications. In this section, we review three interesting applications of deep learning model complexity, namely understanding model generalization capability, model optimization, and model selection and design.

5.1 Model Complexity in Understanding Model Generalization Capability

Deep learning models are always over-parameterized, that is, they have far more model parameters than the optimal solutions and the number of training samples. However, it is often found that large over-parameterized neural networks exhibit good generalization capability [49, 76, 100]. Some studies even find that larger and more complex networks usually generalize better [81]. This observation is in contradiction with the classical notion of function complexity [81] and the well-known Occam's razor [90], which prefer simple models. What leads to the good generalization capability of over-parameterized deep learning models?

In statistical learning theory, expressive capacity (i.e., hypothesis space complexity) is used to bound generalization error [69]. Specifically, let $\mathcal{F}$ be the set of functions representable by a certain model structure. Let $f_{A(D)}$ be a function $f \in \mathcal{F}$ learned by algorithm A on training dataset D. Let $E_D(f_{A(D)})$ be the emperical error of $f_{A(D)}$ and $E(f_{A(D)})$ the generalization error of $f_{A(D)}$. The

gap between generalization error and emperical error is bounded by

$$E(f_{A(D)}) - E_D(f_{A(D)}) \leq \sup_{f \in \mathcal{F}} \{E(f) - E_D(f)\} \tag{12}$$

The right-hand side can be quantified by analyzing the expressive capacity (e.g., Rademacher complexity) [49].

A series of studies investigate model complexity measures that can explain generalization capability of deep learning models [2, 60, 76, 81]. Neyshabur *et al.* [76] suggests that, from the perspective of generalization, a complexity measure should satisfy the following property: a learned model with lower complexity generalizes better. In particular, they list several requirements which are summarized from observed empirical phenomena and are expected to be satisfied by complexity measures.

- With zero training error, a network trained on real labels, which leads to good generalization, is expected to have much lower complexity than a network trained on random labels.
- Increasing the number of hidden units or the number of parameters, which leads to a decreased generalization error, is expected to decrease the complexity measure.
- When training the same architecture on the same training dataset using two different optimization algorithms, if both lead to zero training errors, the model with better generalization is expected to have lower complexity.

Based on these desiderata, Neyshabur *et al.* [76] investigate several complexity measures including norms [79], robustness [97], and sharpness [50]. They show that, these measures can meet some of the above requirements, but not all.

Novak *et al.* [81] define two complexity measures from the perspective of model sensitivity, and identify an empirical correlation between the complexity measures and model generalization capability. They show that operations that lead to poor generalization, such as full batch training, correspond to high sensitivity, and in turn imply high effective complexity. Similarly, operations that lead to good generalization, such as data augmentation, correspond to low sensitivity, and thus imply low effective complexity.

Liang *et al.* [60] define a complexity measure using the Fisher-Rao norm to investigate model generalization capability. They suggest that a complexity measure used to study generalization should satisfy the invariance property. The invariance property requires that the generalization capacity depends on the equivalence classes obtained by deep models. In other words, many different parameterizations may lead to the same prediction. Thus, the specific

parameterization of deep models should not affect the generalization and the complexity measure. They show that the Fisher-Rao norm honors this invariance property and thus is able to explain the generalization capability of deep learning models.

5.2 Model Complexity in Optimization

Model optimization is concerned about how and why a neural network model can be successfully trained [86, 94]. Specifically, the optimization of a deep model is to determine model parameters to minimize a loss function in general non-convex. The loss function is typically designed based on the understanding of a problem and the requirements of the model, and thus generally includes a performance measure, which is evaluated on the training set, and other constraint terms [34].

Model complexity is widely used to provide a metric to make optimization traceable. For example, a measure metric of the effective model complexity of neural networks helps to monitor the changes of a model during the optimization process and understand how the optimization process progresses [44, 47, 74, 89]. Such a metric also helps to verify the effectiveness of new improvements of optimization algorithms [39]. For example, Nakkiran *et al.* [74] investigate the double descent phenomenon during training using effective complexity measured by the maximal size of the dataset on which zero training error can be achieved. They show that the double descent phenomenon can be represented as a function of the effective complexity. Raghu *et al.* [89] and Hu *et al.* [44] propose new regularization methods and demonstrate the effectiveness of these regularizations through their impact on complexity.

The study of model complexity inspires explorations on the effectiveness of optimization approaches. Hanin and Rolnick [37] use the boundary volumes of linear regions as the complexity measure of ReLU neural networks, and find that during the training, the average boundary volume is always linearly proportional to the number of neurons, irrelevant to the depth of the model. This demonstrates that deep models do not always learn more complex functions than shallow ones, the success of deep learning may be related to optimization algorithms. Ba and Caruana [5] suggest that the great performance of deep learning may be due to the fact that deep models are easier to train than shallow architectures using the current optimization algorithms [37, 81]. This calls for further exploration of the effectiveness of optimization approaches and the relationship with model structures.

5.3 Model Complexity in Model Selection and Design

Given a specific learning task, how can we determine a feasible model structure for the task? Given a variety of models with different architectures and different complexity, how can we pick the best model from them? This is the model selection and design problem [71].

In general, model selection and design is based on the tradeoff between prediction performance and model complexity [61, 72]. On one hand, making predictions with high accuracy is the essential goal of learning a model [69]. A model is expected to be able to capture the underlying patterns hidden in the training data and achieve predictions of accuracy as high as possible. In order to represent a large amount of knowledge and obtain high accuracy, a model with a high expressive capacity, a large degree of freedom and a large training set is required [13]. To this extent, a model with more parameters and higher complexity is favored. On the other hand, an overly complex model may be difficult to train and may incur unnecessary resource consumption, such as storage, computation and time cost [72]. Unnecessary resource consumption should be avoided particularly in practical large scale applications [42]. To this extent, a simpler model with comparable accuracy is preferred than a more complicated one.

To maintain a good tradeoff between accuracy and complexity, a model selected is expected to be complex enough to fit the given data and achieve high accuracy. At the same time, the model should not be highly over-complicated. Understanding model complexity and developing an effective complexity measure are the premise for good model selection strategies. For instance, Michel and Nouy [68] propose a model selection strategy for tree tensor networks (i.e., sum-product neural networks). Their method combines the empirical risk minimization and model complexity penalty to select a model from a family of models.

Neural architecture search (NAS) is a popular solution to deep learning model selection [57, 62, 63, 102], which automatically selects a good neural network architecture for a given learning task. Since an overly complex model may take too-long training time and thus may become a serious obstacle of neural architecture search [57, 62], the accuracy-complexity tradeoff is an important consideration in neural architecture search. Liu *et al.* [62] propose Progressive Neural Architecture Search, which searches for convolutional neural network architectures in the increasing order of model complexity. Therefore, Progressive Neural Architecture Search favors low complexity models that meet the requirement on prediction accuracy. Laredo *et al.* [57] propose Automatic Model Selection, which searches for fully connected neural networks that yield a good balance between prediction accuracy and model complexity.

Radosavovic *et al.* [88] investigate the network design spaces of model selec-

tion and design approaches. They propose to compare design spaces by contrasting the estimated distributions of model complexity in the network design spaces. Using their proposed method of comparing the distributions of model complexity of network design space, they investigate several popular NAS approaches, such as ENAS [85] and DARTS [63], and find that there are significant differences between the network design spaces of these approaches.

6 Conclusions and Future Directions

In this paper, we survey model complexity in deep learning. We summarize four aspects affecting deep learning model complexity, and two angles to overview existing studies on deep learning model complexity. We discuss the two major problems of deep learning model complexity, namely the model expressive capacity and effective model complexity. We overview the state-of-the-art studies on the expressive capacity from four aspects: depth efficiency, width efficiency, expressible functional space, and VC dimension and Rademacher complexity. We overview the state-of-the-art studies on the effective complexity from two aspects: general measures of effective complexity and the high-capacity low-reality phenomenon. We discuss the application of deep learning model complexity, especially in generalization capability, optimization, model selection and design.

Model complexity of deep learning is still in its infant stage. There are many interesting challenges for future works.

Expressive capacity of deep learning models is a challenging problem. For example, in most cases, deep learning models are over-parameterized and have sufficient expressive power for given tasks and data. A natural question is what expressive capacity is sufficient for a given task. In other words, can we obtain a lower bound of expressive capacity of deep learning models that are sufficient for a given task? Does a narrow layer limit the expressive capacity of a model even if the model itself has a large number of parameters?

Several studies explore the bottleneck of model size (i.e., depth, width) in the expressive capacity. That is, when the model size may become a bottleneck that restricts the expressive capacity. For example, Hanin and Sellke [38] and Lu *et al.* [64] discover that any deep ReLU network with width constrained by the input dimensionality has very limited expressive power. Serra *et al.* [92] find that smaller widths in the first few layers of a deep ReLU network cause a bottleneck on the expressive power. Kileel *et al.* [52] identify a bottleneck of layer width of deep polynomial networks. In a deep polynomial network, if there is a very narrow layer, no matter how wide the other layers are, the network can never correspond to a convex functional space. A convex functional space benefits the optimization and makes the model easier to train. Research on the

bottleneck of expressive capacity may help to tackle many other problems, such as model design, model selection, model compression, and pruning.

Though some progress has been made, effective complexity measures are still a largely under-developed direction in deep learning. Comparing to expressive capacity of deep learning models, measuring effective model complexity is even more challenging. Effective complexity measures have to be capable of capturing fine granularity differences between two models, such as the same model architecture with two different optimization algorithms. Several previous studies [44, 60, 74, 81, 89] define and explore effective complexity measures mainly from the perspective of piecewise linear property [44, 81, 89], Fisher-Rao metric [60], or the size of trainable samples [74]. However, the effective complexity measure is still a largely unexplored and valuable direction.

Last, cross-model complexity comparison is a promising direction. Given several models with different model frameworks and different model sizes, how can we compare their expressive capacity? After these models are trained sufficiently on the same dataset, such as obtaining zero training error, how can we compare their effective complexity and further understand their generalization capability? In these cases, the cross-model complexity comparison is useful. Comparing model complexity crossing different deep models can in general help many problems of deep learning, in particular model selection and design. However, the exploration of cross-model comparison of expressive capacity or effective complexity of deep learning models is still very limited. Khrulkov *et al.* [51] compare expressive capacity between shallow FCNNs, CNNs, and RNNs by connecting network architectures to tensor decompositions. However, many more sophisticated models are not involved and need to be further explored.

References

[1] R. A. Adams and J. J. Fournier. *Sobolev spaces*. Elsevier, 2003.

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

[3] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*, 2018.

[4] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[5] J. Ba and R. Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[6] V. Balasubramanian. Statistical inference, occam's razor, and statistical mechanics on the space of probability distributions. *Neural computation*, 9(2):349–368, 1997.

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

[8] P. L. Bartlett, S. Boucheron, and G. Lugosi. Model selection and error estimation. *Machine Learning*, 48(1-3):85–113, 2002.

[9] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017.

[10] P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *J. Mach. Learn. Res.*, 20(63):1–17, 2019.

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

[12] P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[13] Y. Bengio and O. Delalleau. On the expressive power of deep architectures. In *International Conference on Algorithmic Learning Theory*, pages 18–36. Springer, 2011.

[14] M. Bianchini and F. 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.

[15] M. Bianchini and F. Scarselli. On the complexity of shallow and deep neural network classifiers. In *ESANN*, 2014.

[16] M. Bohanec and I. Bratko. Trading accuracy for simplicity in decision trees. *Machine Learning*, 15(3):223–250, 1994.

[17] G. Bonaccorso. *Machine learning algorithms*. Packt Publishing Ltd, 2017.

[18] G. E. Bredon. *Topology and geometry*, volume 139. Springer Science & Business Media, 2013.

[19] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[20] H. Buhrman and R. De Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

[21] N. Bulso, M. Marsili, and Y. Roudi. On the complexity of logistic regression models. *Neural computation*, 31(8):1592–1623, 2019.

[22] J.-R. Cano. Analysis of data complexity measures for classification. *Expert systems with applications*, 40(12):4820–4831, 2013.

[23] N. L. Carothers. *Real analysis*. Cambridge University Press, 2000.

[24] J. D. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[25] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, 2018.

[26] V. Cherkassky, X. Shao, F. M. Mulier, and V. N. Vapnik. Model complexity control for regression using vc generalization bounds. *IEEE transactions on Neural Networks*, 10(5):1075–1089, 1999.

[27] N. Cohen and A. Shashua. Convolutional rectifier networks as generalized tensor decompositions. In *International Conference on Machine Learning*, pages 955–963, 2016.

[28] S. Cook, C. Dwork, and R. Reischuk. Upper and lower time bounds for parallel random access machines without simultaneous writes. *SIAM Journal on Computing*, 15(1):87–97, 1986.

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

[30] O. Delalleau and Y. Bengio. Shallow vs. deep sum-product networks. In *Advances in Neural Information Processing Systems*, pages 666–674, 2011.

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[32] J. Du. The "weight" of models and complexity. *Complexity*, 21(3):21–35, 2016.

[33] B. Frieden. Science from fisher information: A unification–cambridge univ. *Press.–2004*, 2004.

[34] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[35] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *International conference on machine learning*, pages 1319–1327. PMLR, 2013.

[36] I. Gühring, G. Kutyniok, and P. Petersen. Complexity bounds for approximations with deep relu neural networks in sobolev norms. 2019.

[37] B. Hanin and D. Rolnick. Complexity of linear regions in deep networks. In *International Conference on Machine Learning*, pages 2596–2604. PMLR, 2019.

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

[39] S. Hayou, A. Doucet, and J. Rousseau. On the selection of initialization and activation function for deep neural networks. *arXiv preprint arXiv:1805.08266*, 2018.

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

[41] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *stat*, 1050:9, 2015.

[42] M. Höge, T. Wöhling, and W. Nowak. A primer for model selection: The decisive role of model complexity. *Water Resources Research*, 54(3):1688–1715, 2018.

[43] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[44] X. Hu, W. Liu, J. Bian, and J. Pei. Measuring model complexity of neural networks with curve activation functions. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1521–1531, 2020.

[45] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 448–456, 2015.

[46] S. M. Kakade, K. Sridharan, and A. Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. In *Advances in neural information processing systems*, pages 793–800, 2009.

[47] D. Kalimeris, G. Kaplun, P. Nakkiran, B. Edelman, T. Yang, B. Barak, and H. Zhang. Sgd on neural networks learns functions of increasing complexity. In *Advances in Neural Information Processing Systems*, pages 3491–3501, 2019.

[48] B. L. Kalman and S. C. Kwasny. Why tanh: choosing a sigmoidal function. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 4, pages 578–581. IEEE, 1992.

[49] K. Kawaguchi, L. P. Kaelbling, and Y. Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

[50] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *International Conference on Learning Representations*, 2017.

[51] V. Khrulkov, A. Novikov, and I. Oseledets. Expressive power of recurrent neural networks. In *International Conference on Learning Representations*, 2018.

[52] J. Kileel, M. Trager, and J. Bruna. On the expressive power of deep polynomial neural networks. In *Advances in Neural Information Processing Systems*, pages 10310–10319, 2019.

[53] J. Kilian and H. T. Siegelmann. On the power of sigmoid neural networks. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 137–143, 1993.

[54] V. Kuurkova. Constructive lower bounds on model complexity of shallow perceptron networks. *Neural Computing and Applications*, 29(7):305–315, 2018.

[55] G. Lample, M. Ott, A. Conneau, L. Denoyer, and M. Ranzato. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049, 2018.

[56] J. M. Landsberg. Tensors: geometry and applications. *Representation theory*, 381(402):3, 2012.

[57] D. Laredo, S. F. Ma, G. Leylaz, O. Schütze, and J.-Q. Sun. Automatic model selection for fully connected neural networks. *International Journal of Dynamics and Control*, 8(4):1063–1079, 2020.

[58] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[59] L. Li. *Data complexity in machine learning and novel classification algorithms*. PhD thesis, California Institute of Technology, 2006.

[60] T. Liang, T. Poggio, A. Rakhlin, and J. Stokes. Fisher-rao metric, geometry, and complexity of neural networks. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 888–896. PMLR, 2019.

[61] T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine learning*, 40(3):203–228, 2000.

[62] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[63] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. *International Conference on Learning Representations*, 2019.

[64] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. The expressive power of neural networks: A view from the width. In *Advances in neural information processing systems*, pages 6231–6239, 2017.

[65] S. Lundqvist, A. Oneto, B. Reznick, and B. Shapiro. On generic and maximal k-ranks of binary forms. *Journal of Pure and Applied Algebra*, 223(5):2062–2079, 2019.

[66] W. Maass. Neural nets with superlinear vc-dimension. *Neural Computation*, 6(5):877–884, 1994.

[67] H. Mhaskar, Q. Liao, and T. Poggio. Learning functions: when is deep better than shallow. *arXiv preprint arXiv:1603.00988*, 2016.

[68] B. Michel and A. Nouy. Learning with tree tensor networks: complexity estimates and model selection. *arXiv preprint arXiv:2007.01165*, 2020.

[69] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[70] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

[71] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[72] I. J. Myung. The importance of complexity in model selection. *Journal of mathematical psychology*, 44(1):190–204, 2000.

[73] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[74] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2020.

[75] B. Neyshabur. Implicit regularization in deep learning. *arXiv preprint arXiv:1709.01953*, 2017.

[76] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956, 2017.

[77] B. Neyshabur, Z. Li, S. Bhojanapalli, Y. LeCun, and N. Srebro. The role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations*, 2018.

[78] B. Neyshabur, R. Tomioka, and N. Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

[79] B. Neyshabur, R. Tomioka, and N. Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pages 1376–1401, 2015.

[80] N. Nisan and M. Szegedy. On the degree of boolean functions as real polynomials. *Computational complexity*, 4(4):301–313, 1994.

[81] R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018.

[82] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[83] I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[84] I. Pérez Arribas. Sobolev spaces and partial differential equations. 2017.

[85] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient neural architecture search via parameters sharing. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4095–4104, 2018.

[86] T. Poggio, K. Kawaguchi, Q. Liao, B. Miranda, L. Rosasco, X. Boix, J. Hidary, and H. Mhaskar. Theory of deep learning iii: explaining the non-overfitting puzzle. *arXiv preprint arXiv:1801.00173*, 2017.

[87] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pages 3360–3368, 2016.

[88] I. Radosavovic, J. Johnson, S. Xie, W.-Y. Lo, and P. Dollár. On network design spaces for visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1882–1890, 2019.

[89] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. S. Dickstein. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2847–2854. JMLR, 2017.

[90] C. E. Rasmussen and Z. Ghahramani. Occam's razor. In *Advances in neural information processing systems*, pages 294–300, 2001.

[91] P. Rebentrost, B. Gupt, and T. R. Bromley. Quantum computational finance: Monte carlo pricing of financial derivatives. *Physical Review A*, 98(2):022321, 2018.

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

[93] D. J. Spiegelhalter, N. G. Best, B. P. Carlin, and A. Van Der Linde. Bayesian measures of model complexity and fit. *Journal of the royal statistical society: Series b (statistical methodology)*, 64(4):583–639, 2002.

[94] R. Sun. Optimization for deep learning: theory and algorithms. *arXiv preprint arXiv:1912.08957*, 2019.

[95] Y. Tan and J. Wang. A support vector machine with a hybrid kernel and minimal vapnik-chervonenkis dimension. *IEEE Transactions on knowledge and data engineering*, 16(4):385–395, 2004.

[96] V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[97] H. Xu and S. Mannor. Robustness and generalization. *Machine learning*, 86(3):391–423, 2012.

[98] A. C.-C. Yao. Decision tree complexity and betti numbers. *journal of computer and system sciences*, 55(1):36–43, 1997.

[99] D. Yin, R. Kannan, and P. Bartlett. Rademacher complexity for adversarially robust generalization. In *International Conference on Machine Learning*, pages 7085–7094. PMLR, 2019.

[100] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations*, 2017.

[101] G. M. Ziegler. Lectures on polytopes. 1993.

[102] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *International Conference on Learning Representations*, 2016.