The Role of Depth, Width, and Tree Size in Expressiveness of Deep Forest

Shen-Huan Lyu[1,2,†], **Jin-Hui Wu**[3,4,†], **Qin-Cheng Zheng**[3,4] **and Baoliu Ye**[3,‡]

[1]Key Laboratory of Water Big Data Technology of Ministry of Water Resources, Hohai University, China
[2]College of Computer Science and Software Engineering, Hohai University, China
[3]National Key Laboratory for Novel Software Technology, Nanjing University, China
[4]School of Artificial Intelligence, Nanjing University, China
lvsh@hhu.edu.cn {wujh,zhengqc}@lamda.nju.edu.cn yebl@nju.edu.cn

Abstract. Random forests are classical ensemble algorithms that construct multiple randomized decision trees and aggregate their predictions using naive averaging. Zhou and Feng [51] further propose a deep forest algorithm with multi-layer forests, which outperforms random forests in various tasks. The performance of deep forests is related to three hyperparameters in practice: depth, width, and tree size, but little has been known about its theoretical explanation. This work provides the first upper and lower bounds on the approximation complexity of deep forests concerning the three hyperparameters. Our results confirm the distinctive role of depth, which can exponentially enhance the expressiveness of deep forests compared with width and tree size. Experiments validate these theoretical findings. The detailed proof and code are available in the full version [31].

1 Introduction

Random forests [8] and neural networks [2] are regarded as two contrasting approaches to learning. The former is considered more suitable for modeling categorical and mixed data, such as medical diagnosis analysis [4] and financial anomaly detection [1]. In contrast, the latter is better suited for modeling numerical data, such as computer vision [25] and natural language processing [15]. Random forests are favored for the tree-based intuitive inference, which makes them easier for users to understand and utilize [8]. On the other hand, neural networks are renowned for their distinguishing performance when dealing with complex data, even though they are often perceived as opaque black-box models that are difficult to comprehend [34].

Recently, deep learning [26] has significantly improved the expressiveness and performance of neural networks. Expressiveness of a model implies its hypothesis class complexity, and higher expressiveness implies higher approximation efficiency. Numerous studies have demonstrated that, within the neural network architecture, depth plays a crucial role in efficiently representing complex data, surpassing width exponentially [13, 17, 22, 40]. Neural networks rely on gradient propagation for training, but their performance on many categorical datasets is often inferior to that of traditional tree-based learning algorithms. Therefore, many real-world tasks require algorithms composed of non-differentiable modules, such as random forests [8, 19], GBDTs [10, 24], etc.

† Equal contribution.
‡ Corresponding author.

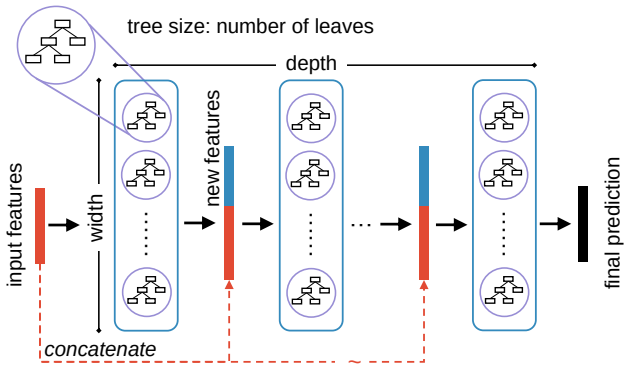


Figure 1: Illustration of the deep forest architecture.

By realizing that the essence of deep learning lies in *layer-by-layer processing*, *in-model feature transformation*, and *sufficient model complexity*, Zhou and Feng [51] propose the first non-differentiable deep model based on decision trees, known as deep forest. In practice, deep forests outperform various ensemble algorithms based on decision trees and have been involved in real applications such as biomedicine [21], smart water management [27], and financial risk assessment [48], etc. Various adaptations of deep forests have excelled in diverse learning scenarios [39, 41, 46], and efforts are ongoing to improve the learning efficiency and reduce the computational cost for large-scale deep forests [29, 35, 50].

The success of deep forests in practice has attracted attention for its theoretical analysis. Regarding *layer-by-layer processing*, Lyu et al. [28] prove that deep forests can optimize sample margin distributions layer by layer, thereby alleviating the overfitting risk. The axis-aligned structure of decision trees is considered to prove that *layer-by-layer processing* significantly improves the consistency rate of random forests [3, 30]. In terms of *in-model feature transformation*, a line of work shows that new features based on predictions can easily cause overfitting risk, and propose a novel feature representation method based on decision rules [11, 29, 35]. However, there is still a lack of theoretical explanation for *sufficient model complexity*.

As shown in Figure 1, a single decision tree serves as the unit of a deep forest, then the number of parameters in a decision tree is referred to as the 'tree size'. Each layer of deep forests consists of a certain number of decision trees. This is known as the 'width' of deep

forests. Additionally, predictions from each layer are transmitted to the next layer as augmented features. This iterative process continues for a specified number of layers, which is referred to as the 'depth' of deep forests. These three hyperparameters impact the complexity of deep forests in practice. Notably, the depth distinguishes the deep forests from individual decision trees and random forests. However, the theoretical advantage of depth remains unclear.

Contributions. In this paper, we show the advantages of depth over width and tree size from the perspective of expressiveness (refer to approximation complexity in Definition 1), focusing on the generalized parity functions and a simplified architecture of deep forest. The main contributions can be summarized as follows:

- We show that depth is more powerful than tree size and width from the aspect of expressiveness. Specifically, we prove the advantage of depth over tree size by separation results of expressing parity functions and the worst-case guarantees, as shown in Theorems 1 and 2, respectively. We show that depth can be more efficient than width by separation results, as shown in Theorem 3.
- We further demonstrate the power of depth from the aspect of learning in experiments. Specifically, we construct a product distribution to learn parity functions and verify our theoretical findings via simulation. Real-world experiments also support the efficiency of depth in deep forests.

Organization. The rest of this work is organized as follows: Section 2 introduces related work. Section 3 provides basic notations and definitions. Section 4 compares depth, width, and tree size in deep forests via expressiveness. Section 5 presents experiments to confirm the theoretical results. Section 6 concludes with future work.

2 Related work

Random forests aggregate multiple decision trees along the width dimension to improve performance. Therefore, theoretical properties of width have attracted significant interest [5, 14]. Breiman [8] offers an upper bound on the generalization error of random forests in terms of correlation and accuracy of individual trees. In recent years, various theoretical works [6, 5, 23, 18, 52] have been performed, analyzing the consistency of various simplified forests, and moving ever closer to practice. Scornet et al. [37] prove the first $\mathbb{L}_2^2$-consistency of Breiman's original random forests with CART-split criterion [9]. Wang et al. [42] show that a larger width can enhance the stability of random forests. Curth et al. [12] present the adaptive smoothing behavior of random forests over decision trees. Additionally, tree size plays a crucial role in random forests. Scornet et al. [37] prove that limiting the size of decision trees can lead to insufficient diversity, which slows down the consistency rate.

Although deep forests achieve satisfactory performance in many tasks, there lacks theoretical discussions on how the depth, width, and tree size impact the capability of deep forests. Approximation complexity reflects the capability of approximating certain function classes and is widely studied in neural networks to compare the impact of different hyperparameters on approximation efficiency. Many attractive conclusions are derived from analyzing approximation complexity of neural networks, such as depth is more powerful than width and activation complexity [33, 17, 38, 13, 40, 20], and complex-valued networks are more powerful than real-valued networks [47, 44]. This paper takes the first step towards investigating approximation capability for tree-based models and proves that depth can be more powerful than width and tree size in achieving sufficient model complexity using fewer number of leaves.

3 Preliminaries

Let $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_n$ denote the n-dimensional discrete input space, where $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n \subset \mathbb{R}$ are finite sets. For simplicity, these finite sets are supposed to share the same domain, i.e., $\mathcal{X}_1 = \cdots = \mathcal{X}_n = [p] := \{1, 2, \ldots, p\}$. We consider the binary classification problem in this paper, i.e., the target concept c is a mapping from the input space $\mathcal{X}$ to the output space $\mathcal{Y} = \{-1, +1\}$. Let $\boldsymbol{v}_i$ indicate the i-the coordinate of the vector $\boldsymbol{v}$, and $\dim(\cdot)$ denotes the dimension of a vector or a space.

This work focus on the expressiveness of decision trees, forests, and deep trees, where trees in forests and deep trees have a restricted number of leaves. These three models have one unrestricted and two restricted hyperparameters, as shown in Table 1. We proceed to provide formal definitions for the size of the three mentioned models. Let $\mathcal{P}_{\mathrm{T}}$ denote the parameter space of binary decision trees, which satisfies the following requirements: i) $\mathcal{Y} \subset \mathcal{P}_{\mathrm{T}}$; ii) $(i; x_i; \Theta_L; \Theta_R) \in \mathcal{P}_{\mathrm{T}}$ holds for any $i \in [\dim(\mathcal{X})]$, $x_i \in \mathcal{X}_i$, and $\Theta_L, \Theta_R \in \mathcal{P}_{\mathrm{T}}$. Then any parameter $\Theta \in \mathcal{P}_{\mathrm{T}}$ determines a decision tree as follows

$$h_\Theta(\boldsymbol{x}) = \begin{cases} \Theta_1, & \dim(\Theta) = 1, \\ h_{\Theta_L}(\boldsymbol{x}), & \dim(\Theta) > 1 \text{ and } \boldsymbol{x}_{\Theta_1} \leqslant \Theta_2, \\ h_{\Theta_R}(\boldsymbol{x}), & \dim(\Theta) > 1 \text{ and } \boldsymbol{x}_{\Theta_1} > \Theta_2. \end{cases}$$

All such trees form the hypothesis space of decision trees

$$\mathcal{H}_{\mathrm{T}}(\mathcal{X}, \mathcal{Y}) = \{h_\Theta : \mathcal{X} \to \mathcal{Y} \mid \Theta \in \mathcal{P}_{\mathrm{T}}\}.$$

Table 1: Hyperparameters of tree-based models.

	Tree Size	Width	Depth
Tree	Unrestricted	Restricted	Restricted
Forest	Restricted	Unrestricted	Restricted
Deep Tree	Restricted	Restricted	Unrestricted

Define $\dim(h_\Theta) = \dim(\Theta)$ as the size of a tree $h_\Theta \in \mathcal{H}_{\mathrm{T}}$. A decision tree with m parent nodes satisfies $\dim(h_\Theta) = 3m + 1$. We are also interested in decision trees of restricted size. More specially, the maximal number of parent nodes is set as twice the input size for simplicity, or equivalently,

$$\mathcal{H}_{\mathrm{T}^\circ}(\mathcal{X}, \mathcal{Y}) = \{h_\Theta \in \mathcal{H}_{\mathrm{T}} \mid \dim(h_\Theta) \leqslant 6\dim(\mathcal{X}) + 1\}.$$

Let $M(\boldsymbol{v})$ represent the mode (or majority vote) of the vector $\boldsymbol{v}$. When the mode is not unique, we set $M(\boldsymbol{v})$ by uniformly randomly choosing one from all modes. We focus on the forests composed of trees with restricted size, whose hypothesis space is

$$\begin{aligned}\mathcal{H}_{\mathrm{F}}(\mathcal{X}, \mathcal{Y}) = \{h \mid \exists\, N \in \mathbb{N}^+, h_{\Theta_1}, \ldots, h_{\Theta_N} \in \mathcal{H}_{\mathrm{T}^\circ}(\mathcal{X}, \mathcal{Y}), \\ \text{s.t. } h(\boldsymbol{x}) = M(h_{\Theta_1}(\boldsymbol{x}), \ldots, h_{\Theta_N}(\boldsymbol{x}))\},\end{aligned}$$

and $\dim(h) = \dim(h_{\Theta_1}) + \cdots + \dim(h_{\Theta_N})$ is the size of a forest $h \in \mathcal{H}_{\mathrm{F}}$. Let $f : \mathcal{X} \to \mathcal{Y}$ and $g : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be two mappings. Define the cascade composition of g and f as $g \oplus f = g(x, f(x))$. Then the hypothesis space of restricted-tree-size deep trees is

$$\begin{aligned}\mathcal{H}_{\mathrm{DT}}(\mathcal{X}, \mathcal{Y}) = \{h \mid \exists\, D \in \mathbb{N}^+, h_{\Theta_1} \in \mathcal{H}_{\mathrm{T}^\circ}(\mathcal{X}, \mathcal{Y}), \\ h_{\Theta_2}, \ldots, h_{\Theta_D} \in \mathcal{H}_{\mathrm{T}^\circ}(\mathcal{X} \times \mathcal{Y}, \mathcal{Y}), \\ \text{s.t. } h = h_{\Theta_D} \oplus \cdots \oplus h_{\Theta_1}\},\end{aligned}$$

and $\dim(h) = \dim(h_{\Theta_1}) + \cdots + \dim(h_{\Theta_D})$ is the size of a deep tree $h \in \mathcal{H}_{\mathrm{DT}}$. We then introduce approximation complexity, which is used to compare the expressiveness of different models.

Definition 1. *Let $\mathcal{H}$ denote a hypothesis space, c represents a target concept, $\epsilon \in [0,1]$ indicates the approximation error, and $\mathcal{D}$ is a distribution on the input space $\mathcal{X}$. Define the approximation complexity $\mathcal{C}(\mathcal{H}, c, \mathcal{D}, \epsilon)$ as*

$$\mathcal{C}(\mathcal{H}, c, \mathcal{D}, \epsilon) = \min_{h \in \mathcal{H}} \{\dim(h) \mid \Pr_{\boldsymbol{x} \sim \mathcal{D}}[h(\boldsymbol{x}) \neq c(\boldsymbol{x})] \leqslant \epsilon\}.$$

The approximation complexity $\mathcal{C}(\mathcal{H}, c, \mathcal{D}, \epsilon)$ measures the number of parameters required to express target concept c using models from the hypothesis space $\mathcal{H}$, under the distribution $\mathcal{D}$ and within an error tolerance ϵ. It shows the expression capability of the hypothesis space, which is the premise of promising performance. A summary of mathematical symbols is provided in full version [31].

4 Main results

In this section, we demonstrate theoretical advantages of depth over tree size and width from the perspective of approximation complexity. In Section 4.1, we provide the definition and several useful properties of generalized parity functions. Sections 4.2 and 4.3 show the advantage of depth over tree size and width, respectively, when approximating generalized parity functions.

4.1 Generalized parity functions

We first define the generalized parity functions

$$c(\boldsymbol{x}) = (-1)^{\|\boldsymbol{x}\|_1} \quad \text{for} \quad \boldsymbol{x} \in [p]^n,$$

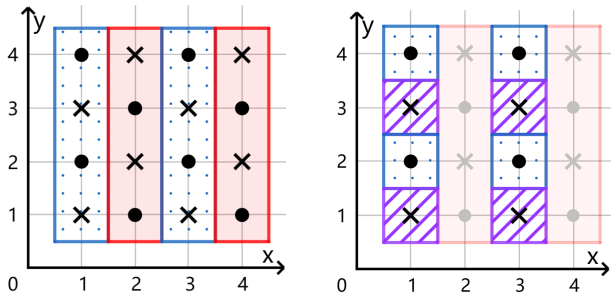
which degenerates to the parity function when $p = 2$. We proceed to introduce the notion of label-connected sets, which is used to characterize the complexity of parity functions.

Definition 2. *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ denote two input vectors, $r > 0$ is a positive real number, and $f : \mathcal{X} \to \{-1, 1\}$ represents a classification mapping. Vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are (r, f)-label-connected if there exist n input vectors $\boldsymbol{z}_1 = \boldsymbol{x}, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n = \boldsymbol{y} \in \mathcal{X}$, such that $\|\boldsymbol{z}_{i+1} - \boldsymbol{z}_i\|_1 \leqslant r$ and $f(\boldsymbol{z}_{i+1}) = f(\boldsymbol{z}_i)$ holds for any $i \in [n-1]$. The (r, f)-label-connected set of vector $\boldsymbol{x}$ is defined as $C_{r,f}(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathcal{X} \mid \boldsymbol{x}$ and $\boldsymbol{y}$ are (r, f)-label connected$\}$.*

We mostly focus on the $(1, f)$-label-connected sets throughout this paper. When the input dimension is 2, imagine the input space $\mathcal{X}$ as a chess board, the points with the same label to $\boldsymbol{x}$ as pools, and the other points as mountains. Two adjacent pools are connected and otherwise severed. Then the $(1, f)$-label-connected set of $\boldsymbol{x}$ depicts the water area containing $\boldsymbol{x}$. From this intuitive explanation, it is observed that either two points share the same water area (label-connected set) or their water areas are disjoint. We formally claim this property for general label-connected sets in the following lemma and provide rigorous proof in the full version [31].

Lemma 1. *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ denote two input vectors, $r > 0$ is a positive real number, and $f \in \mathcal{F}_{\mathcal{X}}$ represents a classification mapping. Then either $C_{r,f}(\boldsymbol{x}) = C_{r,f}(\boldsymbol{y})$ or $C_{r,f}(\boldsymbol{x}) \cap C_{r,f}(\boldsymbol{y}) = \varnothing$.*

Lemma 1 shows that the label-connected sets of two inputs are either the same or disjoint. Then label-connected sets of all inputs form a partition of the input space by the connectivity of labels. The cardinality of the partition reflects the complexity of the function since each element of the partition corresponds to a constant piece of the function. For the simplest constant function, the cardinality equals 1.



(a) Step 1. Split the plane into strips, where the same pattern have the same pseudo label.

(b) Step 2. Assign final labels after gathering strips with the same pattern using the pseudo labels.

Figure 2: A 2-dimensional demonstration of the construction of the deep tree expressing the parity function. Circles and crosses at integral points are positive and negative classes, respectively. Rectangles indicate tree leaves.

For parity functions, the function values are different on any two adjacent inputs, leading to an exponential cardinality when considering $(1, f)$-label-connected sets. The following lemma formally presents the cardinality for parity functions and is proved in the full version [31].

Lemma 2. *Let $\mathcal{X} = [p]^n$ be the input space, and $c(\boldsymbol{x}) = (-1)^{\|\boldsymbol{x}\|_1}$ is the generalized parity function with $\boldsymbol{x} \in \mathcal{X}$. Then we have*

1. $|\{C_{1,c}(\boldsymbol{x})\}| = p^n$.
2. $\big||\{C_{1,c}(\boldsymbol{x}) \mid c(\boldsymbol{x}) = 1\}| - |\{C_{1,c}(\boldsymbol{x}) \mid c(\boldsymbol{x}) = -1\}|\big| \leqslant 1$.

Lemma 2 demonstrates two properties of parity functions. The first one implies that the cardinality of all label-connected sets is exponential of input dimension n, and the second one shows that the cardinality of positive label-connected sets is almost the same as that of negative ones. Thus, a parity function is a piecewise constant function with exponential pieces, and both positive and negative pieces are exponential. This makes parity functions hard to approximate using piecewise constant functions with polynomial pieces, including decision trees with polynomial leaves. We formally present the hardness of approximation in the following lemma and provide detailed proof in the full version [31].

Lemma 3. *Let $h_{\mathrm{T}} \in \mathcal{H}_{\mathrm{T}}$ represent a decision tree with L leaves, c denotes the parity function. Define $\mathcal{E}(h_{\mathrm{T}}, c) = \{\boldsymbol{x} \in \mathcal{X} \mid h_{\mathrm{T}}(\boldsymbol{x}) \neq c(\boldsymbol{x})\}$ and $\mathcal{P}(h_{\mathrm{T}}, c) = \{\boldsymbol{x} \in \mathcal{X} \mid h_{\mathrm{T}}(\boldsymbol{x}) = c(\boldsymbol{x})\}$ as the error set and proper set of the decision tree h_{T}, respectively. Then we have $|\mathcal{E}(h_{\mathrm{T}}, c)| \geqslant (p^n - L)/2$ and $|\mathcal{P}(h_{\mathrm{T}}, c)| \leqslant (p^n + L)/2$.*

Lemma 3 provides a lower bound of the cardinality of misclassified inputs when approximating parity functions using decision trees. As long as the number of leaves in a decision tree is polynomial with respect to the input dimension, the lower bound is dominated by $\Omega(p^n)$. Thus, decision trees suffer an $\Omega(1)$ classification error under the uniform distribution unless the number of leaves is exponential.

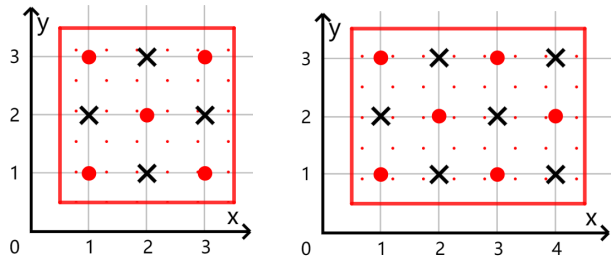
4.2 Depth is more powerful than tree size

In this subsection, we compare the efficiency of depth and tree size, beginning with the advantage of depth.

Theorem 1. *For any input space $\mathcal{X}$, there exist a concept c and a distribution $\mathcal{D}$ over $\mathcal{X}$, such that*

$$\mathcal{C}(\mathcal{H}_{\mathrm{DT}}, c, \mathcal{D}, \epsilon) \leqslant 10pn \quad \text{and} \quad \mathcal{C}(\mathcal{H}_{\mathrm{T}}, c, \mathcal{D}, \epsilon) \geqslant p^n/2$$

holds for any $\epsilon \in [0, 1/4]$.



(a) Case 1: One more correct point than mistaken points.

(b) Case 2: The same number of correct points as mistaken points.

Figure 3: A 2-dimensional demonstration of the relationship between the number of correct points and the number of mistaken points in a tree leaf, where rectangles, circles, and crosses represent tree leaves, correct points, and mistaken points, respectively.

Theorem 1 shows that there exists a classification mapping such that restricting the depth requires increasing the tree size polynomially (with respect to the feature complexity p) or exponentially (with respect to the input dimension n). This indicates the efficiency advantage of depth over tree size in approximating particular functions.

The key observation of proof is that each leaf of a decision tree corresponds to a continuous area with the same labels, while each leaf of a deep tree may assign the same label to many disjoint areas. This phenomenon motivates us to construct the parity function, i.e., $c(\boldsymbol{x}) = (-1)^{\|\boldsymbol{x}\|_1}$, which maximizes the number of disjoint areas since any two points with the same label are not connected.

For deep trees, the upper bound is proven by construction. Take the 2-dimensional case as an example. As shown in Figure 2a, the first part of the deep tree splits the plane into strips using the first coordinate of input. There are only two patterns among all strips since the target concept c is the parity function. The leaves of the first part assign pseudo labels to these strips according to their patterns. Then as shown in Figure 2b, the second part utilizes the pseudo labels to gather strips with the same pattern and assigns the final labels. Since strips are gathered according to their patterns, each leaf of the second part can label half a line of points, which reduces the complexity dramatically. This intuitive construction can be directly extended to arbitrary input dimension and feature complexity, which leads to a deep tree with n parts and $O(p)$ leaves in each part, i.e., the approximation complexity is $O(pn)$.

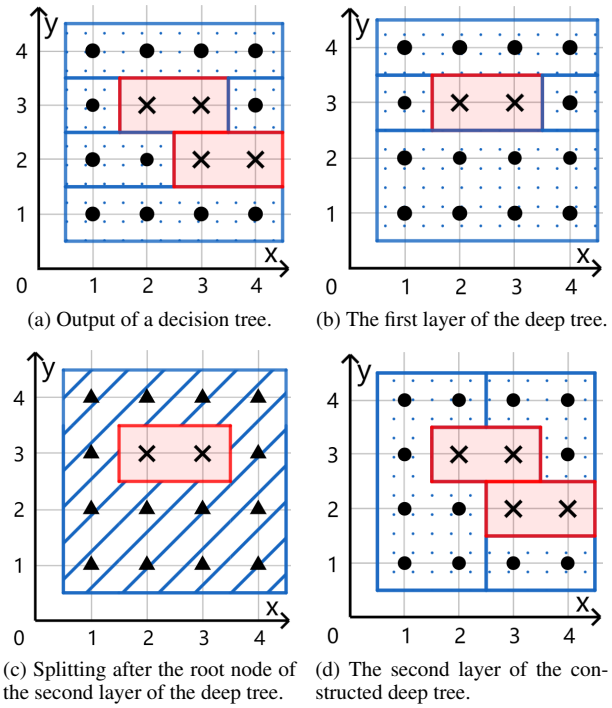
For decision trees expressing the parity function, a basic observation is that the number of correctly labeled points cannot exceed that of mistakenly labeled in any leaf plus 1. As shown in Figure 3a, the number of correctly labeled points is one more than that of mistakenly labeled points when all widths of the leaf are odd numbers and the leaf receives the dominant label. In the other case as shown in Figure 3b, the number of correctly labeled points equals that of mistakenly labeled points when some width of the leaf is an even number. Thus, a decision tree must grow one more leaf to increase the number of correctly labeled points by 1, which only promotes the accuracy by $1/p^n$. Therefore, a decision tree requires at least $\Omega(p^n)$ leaves to enhance the accuracy by a constant.

We then study the dual problem of Theorem 1, i.e., does decision tree possess an exponential efficiency advantage over deep trees when expressing suitable concepts? The next theorem provides a negative answer for this question.

Theorem 2. *For any input space $\mathcal{X}$, any concept c, any distribution $\mathcal{D}$ over $\mathcal{X}$, and any $\epsilon \in [0, 1]$, one has*

$$\mathcal{C}(\mathcal{H}_{\mathrm{DT}}, c, \mathcal{D}, \epsilon) \leqslant (4n + 1)\mathcal{C}(\mathcal{H}_{\mathrm{T}}, c, \mathcal{D}, \epsilon).$$

Theorem 2 provides the worst-case guarantee for deep trees by showing that for all classification mappings, the approximation complexity of deep trees is no more than that of decision trees multiplying a factor linear in the input dimension n. Although decision trees might prevail over deep trees, the advantage of decision trees cannot transcend an upper bound linear in n, which is exponentially smaller than the superiority of deep trees over decision trees as demonstrated in Theorem 1. This tremendous gap between exponentiality and linearity indicates that deep trees outperform decision trees consistently from the perspective of approximation complexity.



(a) Output of a decision tree.

(b) The first layer of the deep tree.

(c) Splitting after the root node of the second layer of the deep tree.

(d) The second layer of the constructed deep tree.

Figure 4: A demonstration of expressing a decision tree using a deep tree. Circles, crosses, and triangles represent positive, negative, and unlabeled points, respectively. Rectangles indicate tree leaves.

The main idea of the proof is that a deep tree can represent a decision tree leaf by leaf, i.e., each layer of the deep tree depicts a leaf of the decision tree and the cascade structure gathers these leaves together. Take a 2-dimensional case as an example. Figure 4a exhibits the output of a decision tree and we aim to find a deep tree with the same pattern as the given decision tree. It is observed that the decision tree assigns negative labels to two leaves, which motivates us to build a 2-layer deep tree. As shown in Figure 4b, the first layer of the deep tree simply divides the input space along the boundary of a positive leaf. Then this positive leaf obtains a positive pseudo label, and the other leaves receive negative ones. Figure 4c illustrates the root node of the second layer of the deep tree, which utilizes the cascaded dimension to provide positive labels for points with positive pseudo labels and remains the rest points unlabeled. Then as pictured by Figure 4d, the rest nodes of the second layer directly isolate the input space along the frontier of the other positive leaf, labeling this leaf as a positive leaf and the rest ones as negative ones. One can immediately extend this construction to general input dimensions and decision trees. The number of layers of the constructed deep tree does not surpass the number of leaves of the decision tree, which possesses the same order of the approximation complexity as deep trees. Meanwhile, each layer of the deep tree just separates a hyperrectangle from the input space, which requires $O(n)$ parameters. Thus, the approximation complexity of deep trees is at most of order n times the approximation complexity of decision trees.

4.3 *Depth can be more powerful than width*

In this subsection, we compare the efficiency of depth and width.

Theorem 3. *For any input space $\mathcal{X}$, there exist a concept c and a distribution $\mathcal{D}$ over $\mathcal{X}$, such that*

$$\mathcal{C}(\mathcal{H}_{\mathrm{DT}}, c, \mathcal{D}, 0) \leqslant 10pn \quad \textit{and} \quad \mathcal{C}(\mathcal{H}_{\mathrm{F}}, c, \mathcal{D}, 0) \geqslant p^n.$$

Theorem 3 shows that there exists a classification mapping such that depth undertakes a more important role than width to efficiently express this mapping. The construction of the concept inherits the idea in Theorem 1, i.e., the concept c is the parity function. For deep trees, the proof remains the same as that of Theorem 1. For forests, we prove the lower bound of approximation complexity by analyzing the total counts of correctly labeled points. In order to label a point properly, there should be at least one more tree assigning the correct label than the wrong label. Thus, each point contributes at least one more count to the total counts of correctly labeled points than the total counts of mistakenly labeled points. As shown in Figure 3 and its explanations, one leaf cannot cause a distinction larger than 1 between the number of correctly labeled and mistakenly labeled points. Thus, the number of leaves is no less than the number of points in the input space, leading to $\Omega(p^n)$ approximation complexity.

5 Experiments

In this section, we verify the power of depth in experiments. Section 5.1 introduces the construction of a product distribution, which is important to learn parity functions in simulation. Section 5.2 verifies theoretical findings via simulation. Section 5.3 shows the advantage of depth in real-world experiments.

5.1 *Product distributions*

As shown in Section 4, generalized parity functions can be efficiently expressed by deep trees under any distribution. But from the aspect of learning, it is known that parity functions with uniform distributions are hard to learn using decision trees since there is no impurity gain at early splits [7, 49, 32]. This motivates us to investigate the existence of a specific distribution to demonstrate the power of depth in both approximation and learning. When considering learning tree-based models in experiments, we focus on the hypercubic input space $[p]^n$ with $p = 4$ and the following probability mass function

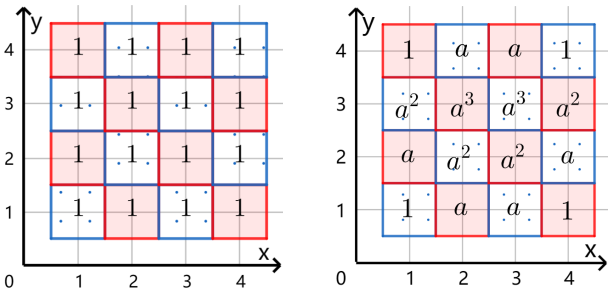
$$p_n(\boldsymbol{x}) = \prod_{i=1}^{n} f_i(x_i) \quad \forall \boldsymbol{x} \in [p]^n,$$

where $f_i : [p] \to \mathbb{R}$ denotes the probability mass function of a 1-dimensional random variable defined by

$$f_i(1) = \frac{1}{b_i}, \quad f_i(2) = \frac{a}{b_i}, \quad f_i(3) = \frac{a^i}{b_i}, \quad f_i(4) = \frac{1}{b_i},$$

where $b_i = 2 + a + a^i$ denotes the normalization coefficient, and a represents a constant. The constructed distribution is a product distribution parameterized by the constant a. The constant a controls the extent of asymmetry and should be large enough to support efficient learning of parity functions using deep trees. In experiments, it suffices to choose $a = 3$ while it fails when $a = 2$.

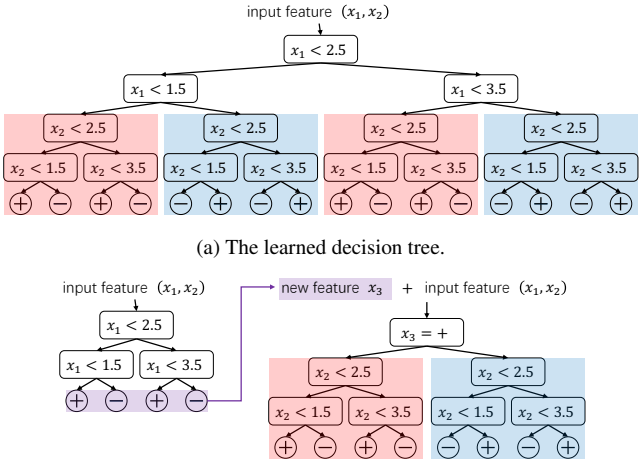
The intuition behind the construction is that the distribution should be highly asymmetric to meet two requirements. Firstly, the root node splits at the midpoint of a feature, and its child nodes split at the midpoint of the same feature until this feature can no longer be split. Secondly, all nodes choose the same feature as the next splitting feature and repeat the first requirement after one feature is split completely. These two requirements lead to highly symmetric decision paths and help deep trees learn parity functions efficiently.



(a) The uniform distribution. The root node splits at random, and there is no impurity change.

(b) The constructed product distribution. The root node splits at $x = 2.5$, and impurity decreases.

Figure 5: A 2-dimensional demonstration of parity functions with the uniform distribution and the constructed product distribution. Grids with red shadow have negative labels, and grids with blue dots have positive labels. The number in each grid represents the relative magnitude of the probability mass function, and $a = 3$ is a constant.



(a) The learned decision tree.

(b) The learned deep tree.

Figure 6: The decision tree and deep tree learned from the 2-dimensional parity function with the constructed product distribution. Subtrees with shadows of the same color are the same. The deep tree merges identical subtrees and uses fewer leaves.

The constructed product distribution makes learning parity functions more efficient, compared with the uniform distribution. We demonstrate 2-dimensional examples of parity functions with the uniform distribution and the constructed product distribution in Figure 5. Consider the root node of a decision tree. For the uniform distribution, the probability mass function is mirror symmetric along both $x = 2.5$ and $y = 2.5$. Thus, there is no impurity change to split any feature at any point, which leads to a random split at the root node. While for the constructed product distribution, the probability mass distribution is mirror symmetric only along $x = 2.5$. Thus, a vertical split breaks the symmetry and brings impurity decrease. In n-dimensional spaces, the uniform distribution has n axes of symmetry, and it takes at least n layers in a decision tree to start the reduction of impurity. While the constructed product distribution only has 1 axis of symmetry since only f_1 is symmetric. Thus, the

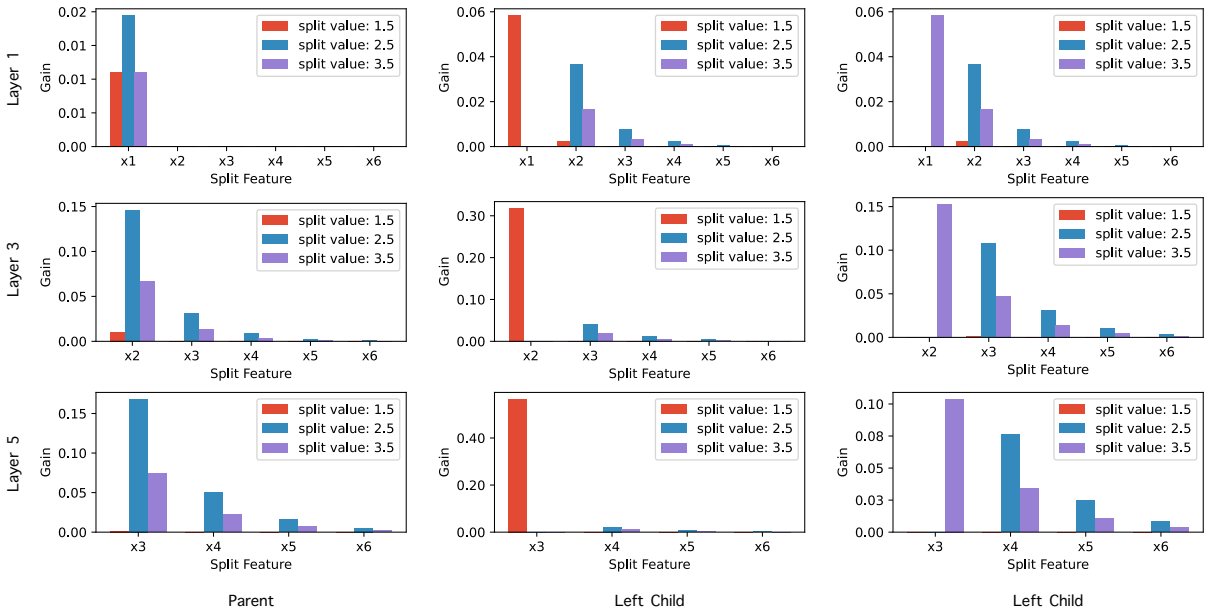


Figure 7: The purity gain in a decision tree with 6 layers when learning 6-dimensional parity functions with the constructed product distribution. The first column demonstrates gains in the first, third, and fifth layers. The second and third columns show gains in the left and right child nodes of the parent node in the first column, respectively. By the principle of maximizing purity gain, each parent node splits at the midpoint of a feature, and its child node splits at midpoints of the same feature.

root node splits using x_1, and the impurity decreases in each layer.

Furthermore, the constructed product distribution induces symmetric decision paths in a decision tree and makes deep trees more efficient than decision trees. We demonstrate the decision tree and deep tree learned from the 2-dimensional parity function with the constructed product distribution in Figure 6. When learning using decision trees, the learned tree is symmetric in the sense that the two red subtrees are the same, and the two blue subtrees are the same. When learning using deep trees with a suitable depth, the output of the first tree, which is also the new feature in the second tree, automatically merges identical subtrees. For high-dimensional parity functions with the constructed product distribution, numerical calculation verifies that the distribution satisfies the two requirements mentioned above when the input dimension $n \leqslant 8$. Due to the limited space, we demonstrate the verification of $n = 6$ in Figure 7 and provide verification of all $n \leqslant 8$ in the supplementary materials. We use Gini index as the measure of impurity and plot the purity gain in a decision tree with 6 layers when learning 6-dimensional parity functions with the constructed product distribution. As shown in the first row, the first and second layers split at midpoints of the same feature x_1. Then x_1 only takes one value in each leaf and is removed from the horizontal axis. In the third layer, the distributions in the 4 leaves are the same since all distributions are product distributions. Thus, it suffices to consider the purity gain in one leaf. We repeat this procedure and find that all nodes split at the midpoint of one feature, and nodes in even-numbered layers split at the same feature as their parent nodes. Therefore, the learned decision tree is highly symmetric, and the number of identical subtrees is exponential with respect to n. Then deep trees can reduce the model complexity exponentially. The numerical verification of higher dimensions is difficult since the 4^n-dimensional probability mass matrix exceeds storage limits.

5.2 *Numerical simulation*

In this subsection, we verify our theories by comparing the performance of trees, deep trees, and random forests via simulation.

Datasets. We randomly sample 10^6 points from the probability mass function p_n. Inputs are generated by perturbing all points with random noises from the uniform distribution over $[-0.5, 0.5]^n$. The label of an input is the same as the output of the parity function on its nearest integer lattice point., i.e.,

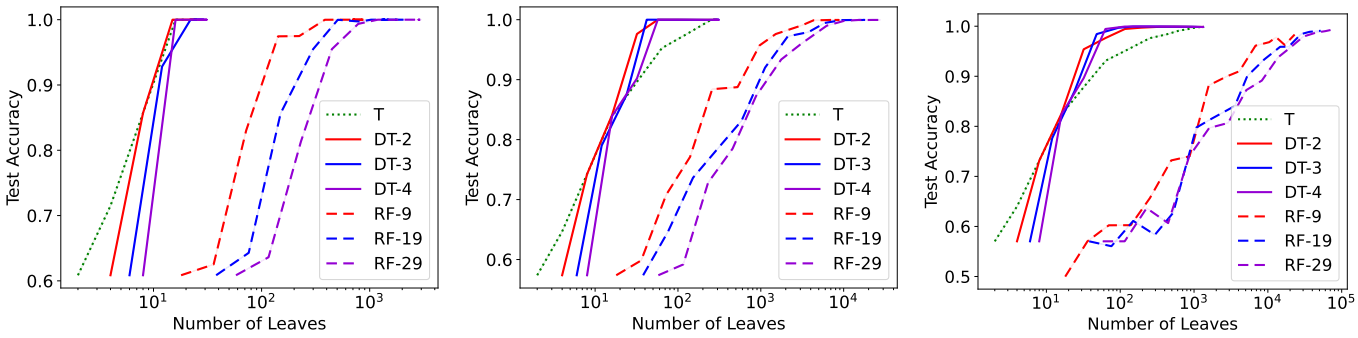
$$y(\boldsymbol{x}) = c(\boldsymbol{x}_0) \quad \text{with} \quad \boldsymbol{x}_0 = \operatorname{argmin}_{\boldsymbol{x}' \in [p]^n} \|\boldsymbol{x}' - \boldsymbol{x}\|^2,$$

where $c(\boldsymbol{x}) = (-1)^{\|\boldsymbol{x}\|_1}$ is the parity function. When some coordinate of $\boldsymbol{x}$ is 0.5, which happens with 0 probability and does not impact on the performance, the minimizer is not unique, and the label is randomly chosen. These 1,000,000 pairs of inputs and labels form a dataset, of which 70% are used as a training set, and the remaining 30% are used as a test set.

Training and testing. We use the training set to train decision trees (T), deep trees with depth 2 (DT-2), 3 (DT-3), and 4 (DT-4), and random forests with 9 (RF-9), 19 (RF-19), and 29 (RF-29) trees. The maximum depth of trees in these models varies from 1 to 15. The Gini index is used as the splitting criterion. Other parameters use the default values implemented by scikit-learn [36]. We test these models on the test set and record the test accuracy.

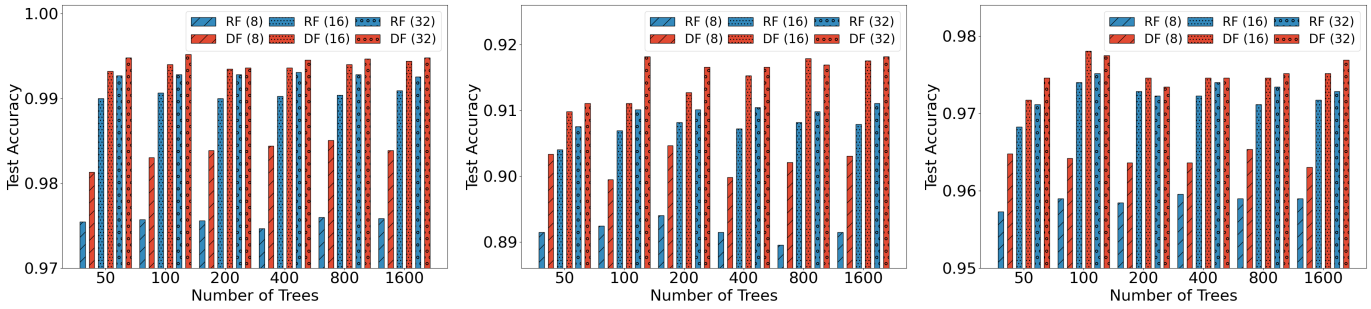
Simulation results. In Figure 8, we plot the test accuracy with respect to the total number of leaves, which equals the summation of the number of leaves over all trees. We also provide the total number of leaves needed to achieve a test accuracy of 99% in Table 2. Deep trees achieve high accuracy with the fewest number of leaves. As the input dimension n increases, deeper deep trees become more efficient, and the gap between deep trees and other models becomes larger. These results support the power of depth in theories.

It is observed that random forests require a much larger number of leaves in the simulation since there are plenty of labeled samples without label noise, which can be done with a single tree. In such a task, diversity from random forests only takes more leaves and hardly helps training. However, these results do not eliminate the importance of width and diversity in deep forests since real-world data usually has noise and a restricted number of samples.



(a) $n = 2$. (b) $n = 4$. (c) $n = 8$.

Figure 8: Test accuracy of learning n-dimensional parity functions with the constructed product distribution using trees, deep trees of different depths, and random forests of different numbers of trees. The test accuracy is plotted as a function of total number of leaves. Deep trees can achieve the same performance using fewer leaves, which is more obvious when dealing with high-dimensional inputs.



(a) Pendigits dataset. (b) Satimage dataset. (c) Segment dataset.

Figure 9: Test accuracy of learning three benchmark datasets with random forests (RF) and deep forests (DF) of different numbers of trees and different tree sizes of base learners. '$(\cdot)$' denotes the tree size of base learners.

Table 2: The total number of leaves needed to achieve a test accuracy of 99% using different models, where a horizontal line means failure of achieving an accuracy of 99%, and the bold number indicates the fewest number of leaves. Deep trees use the fewest number of leaves.

	T	DT-2	DT-3	DT-4	RF-9	RF-19	RF-29
$n = 2$	16	**15**	22	16	385	507	783
$n = 4$	251	57	**42**	57	3,142	5,918	6,255
$n = 8$	633	116	95	**63**	—	51,972	59,795

5.3 Real-world experiments

In this subsection, we verify our theories by comparing the performance of deep forests and random forests with different tree sizes on three real-world datasets.

Datasets. We select three widely used benchmark datasets of classification tasks from the UCI Machine Learning Repository [16]. The datasets vary in size: from 2310 to 10992 instances, from 16 to 36 features, and from 6 to 10 classes. From the literature, these datasets come pre-divided into training and testing sets. Therefore in our experiments, we use them in their original format.

Training and testing. We use the training set to train random forests with width in $\{50, 100, 200, 400, 800, 1600\}$ and deep forests with depth 2 and width in $\{25, 50, 100, 200, 400, 800\}$, so that the total numbers of trees in both equal. We also set all the tree sizes in $\{8, 16, 32\}$ to show the influence of different sizes of trees as base learners on the ensembles. Other parameters use the default values implemented by scikit-learn [36]. We test these models on the test set and record the test accuracy.

Benchmark results. We plot the test accuracy of random forests and two-layer deep forests with different tree sizes and widths in Figure 9. Under equivalent tree size and number of trees, deep forests consistently outperform random forests. Even when the width of random forests significantly exceeds that of deep forests, its performance struggles to match that of deep forests. These results demonstrate that depth, compared to width, provides greater efficiency. It is observed that the tree size of base learners plays a crucial role in practice. While individual decision trees with larger tree sizes perform poorly, random forests and deep forests constructed from these trees outperform those built with smaller tree sizes.

6 Conclusion

This paper provides the first comparison of tree size, width, and depth from the aspect of expressiveness. We theoretically prove that depth is more powerful than tree size and width when approximating parity functions. Experiments show that our theoretical results are valid in many objective function classes other than parity functions. In the future, it is promising to investigate the learnability advantage [43, 45] of depth in deep forests and analyze the robustness of deep forests when dealing with noisy data.

Acknowledgements

S.-H. Lyu was supported by the National Natural Science Foundation of China (62306104), Jiangsu Science Foundation (BK20230949), China Postdoctoral Science Foundation (2023TQ0104), Jiangsu Excellent Postdoctoral Program (2023ZB140). J.-H. Wu was supported by the Program for Outstanding PhD Candidates of Nanjing University (202401A13).

References

[1] J. K. Afriyie, K. Tawiah, W. A. Pels, S. Addai-Henne, H. A. Dwamena, E. O. Owiredu, S. A. Ayeh, and J. Eshun. A supervised machine learning algorithm for detecting and predicting fraud in credit card transactions. *Decision Analytics Journal*, 6:100163, 2023.

[2] M. Anthony, P. L. Bartlett, P. L. Bartlett, et al. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

[3] L. Arnould, C. Boyer, and E. Scornet. Analyzing the tree-layer structure of deep forests. In *Proccedings of the 37th International Conference on Machine Learning*, pages 342–350, 2021.

[4] S. Basu, K. Kumbier, J. B. Brown, and B. Yu. Iterative random forests to discover predictive and stable high-order interactions. *Proceedings of the National Academy of Sciences*, 115(8):1943–1948, 2018.

[5] G. Biau. Analysis of a random forests model. *Journal of Machine Learning Research*, 13(1):1063–1095, 2012.

[6] G. Biau, L. Devroye, and G. Lugosi. Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 9 (9), 2008.

[7] G. Blanc, J. Lange, M. Qiao, and L. Tan. Decision tree heuristics can fail, even in the smoothed setting. In *Proceedings of the 25th Approximation, Randomization, and Combinatorial Optimization. Proceedings of the 24th Algorithms and Techniques*, pages 45:1–45:16, 2021.

[8] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[9] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Chapman & Hall/CRC, 1984.

[10] T. Chen and C. Guestrin. XGboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[11] Y.-H. Chen, S.-H. Lyu, and Y. Jiang. Improving deep forest by exploiting high-order interactions. In *Proceedings of the 21st IEEE International Conference on Data Mining*, pages 1030–1035, 2021.

[12] A. Curth, A. Jeffares, and M. van der Schaar. Why do random forests work? Understanding tree ensembles as self-regularizing adaptive smoothers. *CoRR*, abs/2402.01502, 2024.

[13] A. Daniely. Depth separation for neural networks. In *Proceedings of 30th Conference on Learning Theory*, pages 690–696, 2017.

[14] M. Denil, D. Matheson, and N. De Freitas. Narrowing the gap: Random forests in theory and in practice. In *Proccedings of the 30th International Conference on Machine Learning*, pages 665–673, 2014.

[15] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 19th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, 2019.

[16] D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017.

[17] R. Eldan and O. Shamir. The power of depth for feedforward neural networks. In *Proceedings of 29th Conference on Learning Theory*, pages 907–940, 2016.

[18] R. Genuer. Variance reduction in purely random forests. *Journal of Nonparametric Statistics*, 24(3):543–562, 2012.

[19] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[20] A. Goujon, A. Etemadi, and M. Unser. On the number of regions of piecewise linear neural networks. *Journal of Computational and Applied Mathematics*, 441:115667, 2024.

[21] Y. Guo, S. Liu, Z. Li, and X. Shang. Bcdforest: A boosting cascade deep forest model towards the classification of cancer subtypes based on gene expression data. *BMC Bioinformatics*, 19:1–13, 2018.

[22] Q. Hu, H. Zhang, F. Gao, C. Xing, and J. An. Analysis on the number of linear regions of piecewise linear neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):644–653, 2022.

[23] H. Ishwaran and U. B. Kogalur. Consistency of random survival forests. *Statistics and Probability Letters*, 80(13-14):1056–1064, 2010.

[24] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30*, pages 3146–3154, 2017.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

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

[27] X. Liu, Y. Tian, X. Lei, M. Liu, X. Wen, H. Huang, and H. Wang. Deep forest based intelligent fault diagnosis of hydraulic turbine. *Journal of Mechanical Science and Technology*, 33:2049–2058, 2019.

[28] S.-H. Lyu, L. Yang, and Z.-H. Zhou. A refined margin distribution analysis for forest representation learning. In *Advances in Neural Information Processing Systems 32*, pages 5531–5541, 2019.

[29] S.-H. Lyu, Y.-H. Chen, and Z.-H. Zhou. A region-based analysis for the feature concatenation in deep forests. *Chinese Journal of Electronics*, 31(6):1072–1080, 2022.

[30] S.-H. Lyu, Y.-X. He, and Z.-H. Zhou. Depth is more powerful than width with prediction concatenation in deep forests. In *Advances in Neural Information Processing Systems 35*, pages 29719–29732, 2022.

[31] S.-H. Lyu, J.-H. Wu, Q.-C. Zheng, and B. Ye. The role of depth, width, and tree size in expressiveness of deep forest, 2024. URL https://arxiv.org/abs/2407.05108.

[32] R. Mazumder and H. Wang. On the convergence of CART under sufficient impurity decrease condition. In *Advances in Neural Information Processing Systems 36*, pages 57754–57782, 2023.

[33] 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 27*, pages 2924–2932, 2014.

[34] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019.

[35] M. Pang, K.-M. Ting, P. Zhao, and Z.-H. Zhou. Improving deep forest by confidence screening. *IEEE Transactions on Knowledge and Data Engineering*, 34(9):4298–4312, 2022.

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[37] E. Scornet, G. Biau, and J.-P. Vert. Consistency of random forests. *Annals of Statistics*, 43(4):1716–1741, 2015.

[38] M. Telgarsky. Benefits of depth in neural networks. In *Proceedings of the 24th Conference on Learning Theory*, pages 1517–1539, 2016.

[39] L. V. Utkin and M. A. Ryabinin. Discriminative metric learning with deep forest. *International Journal on Artificial Intelligence Tools*, 28 (2):1950007:1–1950007:19, 2019.

[40] G. Vardi, G. Yehudai, and O. Shamir. Width is less important than depth in relu neural networks. In *Proceedings of 35th Conference on Learning Theory*, pages 1249–1281, 2022.

[41] Q. Wang, L. Yang, and Y. Li. Learning from weak-label data: A deep forest expedition. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pages 6251–6258, 2020.

[42] Y. Wang, H. Wu, and D. Nettleton. Stability of random forests and coverage of random-forest prediction intervals. In *Advances in Neural Information Processing Systems 36*, 2023.

[43] J.-H. Wu, S.-Q. Zhang, Y. Jiang, and Z.-H. Zhou. Complex-valued neurons can learn more but slower than real-valued neurons via gradient descent. In *Advances in Neural Information Processing Systems 36*, pages 23714–23747, 2023.

[44] J.-H. Wu, S.-Q. Zhang, Y. Jiang, and Z.-H. Zhou. Theoretical exploration of flexible transmitter model. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[45] W. Xu and S. Du. Over-parameterization exponentially slows down gradient descent for learning a single neuron. In *Proceedings of the 36th Annual Conference on Learning Theory*, pages 1155–1198, 2023.

[46] L. Yang, X. Wu, Y. Jiang, and Z. Zhou. Multi-label learning with deep forest. In *Proceedings of the 24th European Conference on Artificial Intelligence*, pages 1634–1641, 2020.

[47] S.-Q. Zhang, W. Gao, and Z.-H. Zhou. Towards understanding theoretical advantages of complex-reaction networks. *Neural Networks*, 151: 80–93, 2022.

[48] Y. Zhang, J. Zhou, W. Zheng, J. Feng, L. Li, Z. Liu, M. Li, Z. Zhang, C. Chen, X. Li, Y. A. Qi, and Z. Zhou. Distributed deep forest and its application to automatic detection of cash-out fraud. *ACM Transactions on Intelligent Systems and Technology*, 10(5):55:1–55:19, 2019.

[49] Q.-C. Zheng, S.-H. Lyu, S.-Q. Zhang, Y. Jiang, and Z.-H. Zhou. On the consistency rate of decision tree learning algorithms. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, pages 7824–7848, 2023.

[50] M. Zhou, X. Zeng, and A. Chen. Deep forest hashing for image retrieval. *Pattern Recognition*, 95:114–127, 2019.

[51] Z.-H. Zhou and J. Feng. Deep forest. *National Science Review*, 6(1): 74–86, 2019.

[52] R. Zhu, D. Zeng, and M. R. Kosorok. Reinforcement learning trees. *Journal of the American Statistical Association*, 110(512):1770–1784, 2015.