LETTER **Communicated by Siyu Zhou**

Comparison of the Representational Power of Random Forests, Binary Decision Diagrams, and Neural Networks

So Kumano
soir22@icloud.com
Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan

Tatsuya Akutsu
takutsu@kuicr.kyoto-u.ac.jp
Bioinformatics Center, Institute for Chemical Research, Kyoto University, Kyoto 611-0011, Japan

In this letter, we compare the representational power of random forests, binary decision diagrams (BDDs), and neural networks in terms of the number of nodes. We assume that an axis-aligned function on a single variable is assigned to each edge in random forests and BDDs, and the activation functions of neural networks are sigmoid, rectified linear unit, or similar functions. Based on existing studies, we show that for any random forest, there exists an equivalent depth-3 neural network with a linear number of nodes. We also show that for any BDD with balanced width, there exists an equivalent shallow depth neural network with a polynomial number of nodes. These results suggest that even shallow neural networks have the same or higher representation power than deep random forests and deep BDDs. We also show that in some cases, an exponential number of nodes are required to express a given random forest by a random forest with a much fewer number of trees, which suggests that many trees are required for random forests to represent some specific knowledge efficiently.

1 Introduction

Selecting an appropriate learning model, that is, selecting an appropriate function family, is important in machine learning. If the function family is unnecessarily large, it will cause problems of excessive computation costs and overfitting. But if the function family is too small and thus does not include the target functions, desired prediction results will not be obtained.

In order to obtain effective clues in selecting such a function family, studies on the representational ability of learning models have been conducted. For example, the universal approximation theorem, which claims that an

Neural Computation **34, 1019–1044** (2022)
https://doi.org/10.1162/neco_a_01486

arbitrary Borel measurable function can be approximated by a depth-2 neural network, is well known (Hornik, Stinchcombe, & White, 1989). However, there exist many cases for which a large number of nodes are required using depth-2 neural networks. In recent years, it has been claimed that the representational power of neural networks varies significantly by the number of layers. We note in section 2 that many studies show the existence of function families, which cannot be expressed by shallow neural networks using a small number of nodes while deep neural networks can (Chatziafratis, Nagarajan, Panageas, & Wang, 2020; Delalleau & Bengio, 2011; Hanin & Rolnick, 2019; Montufar, Pascanu, Cho, & Bengio, 2014; Raghu, Poole, Kleinberg, Ganguli, & Dickstein, 2017; Szymanski & McCane, 2014; Telgarsky, 2015). The property of the deep neural network that allows functions to be expressed efficiently is called *depth efficiency*. Indeed, some studies show evidence that the representational power of neural networks increases exponentially by their depth (Montufar et al., 2014; Raghu et al., 2017).

Tree models and their extensions, specifically, decision trees, random forests, and binary decision diagrams (BDDs), have been widely used for representation of a variety of information, as well as for various prediction tasks (Breiman, Friedman, Olshen, & Stone, 1984; Breiman, 2001; Bryant, 1986; Ho, 1995; Meinel & Theobald, 1998). Although many theoretical studies have been conducted on the representational power of neural networks, not many have been done on that of random forests and binary decision diagrams (BDDs) (de Mello, Manapragada, & Bifet, 2019; Zhou & Mentch, 2021; Xu, He, Xie, & Li, 2018), which we explain in section 2. Therefore, in this letter, we study the representational power of random forests and BDDs via comparison with neural networks, focusing on theoretical aspects.

This letter is organized as follows. We briefly review related work and give necessary definitions in sections 2 and 3, respectively. Section 4 shows that for any decision tree and any random forest with n nodes, there exists an equivalent depth-3 neural network with $O(n)$ nodes. Although these results are already known for Heaviside functions, sigmoidal functions, and hyperbolic tangent functions (Bengio, Delalleau, & Simard, 2010; Biau, Scornet, & Welbl, 2019; Sethi, 1990), we present the proofs for the ReLU (rectified linear unit) and related functions. Section 5 shows that for any depth-D BDD with $O(n/D)$ nodes at each depth, there exists a depth-$(D+1)$ neural network with $O(n)$ nodes and a depth-$(2\lceil \log_2 D \rceil + 1)$ neural network with $O(n^3/D^2)$ nodes that simulate the binary decision diagram. Regarding the efficiency due to the number of trees, which is peculiar to random forests, section 6 shows that there exists an n-trees random forest with n nodes for which an equivalent T-trees random forest with $T < n$ requires $\Omega((2^n/\sqrt{n})^{(2/T+1)})$ nodes. To our knowledge, all of these results are new and thus are the contributions of this letter. We conclude with future work.

2 Related Work

It is well known as the universal approximation theorem that an arbitrary Borel measurable function can be expressed by using a depth-2 neural network (Hornik et al., 1989). However, the required number of nodes cannot be derived from the theorem. Thus, it is meaningful to know how large the model should be for each task. Actually, there are studies on what function families can be expressed efficiently by using what architectures, that is, whether they can be expressed with a small number of nodes. Several studies have shown that the depth of the network plays an important role. In particular, the expression capacity of neural networks is thought to increase exponentially with the depth. For example, Montufar et al. (2014) showed that when a piecewise linear function is used as an activation function, the number of linear regions represented by the network increases not only in the polynomial order of width but in the exponential order of depth. Raghu et al. (2017) also argued that the representational power of a neural network increases exponentially with depth by showing that the length of the network's output trajectory also increases exponentially with depth. In fact, many studies show that particular function families can or cannot be expressed efficiently on particular architecture. Telgarsky (2015) proved the existence of a function family that can be expressed by deep neural networks with nodes of the linear order while the nodes of the exponential order are required with shallow neural networks. Chatziafratis et al. (2020) extended his result and gave general lower bounds for the width needed to represent a periodic function as a function of the depth, using Sharkovsky's theorem in dynamical systems. Szymanski and McCane (2014) showed that the input periodicity is one of the factors that cause the depth efficiency in deep neural networks. In other words, a function with the periodicity can be expressed more efficiently by using a deep network. Delalleau and Bengio (2011) showed that there is a function family that can be expressed more efficiently by using deep networks than shallow networks in the context of the sum-product network that calculates a linear sum or a product between nodes at each layer. On the other hand, Hanin and Rolnick (2019) showed that the average number of linear regions in a neural network with piecewise linear activation functions along any curve grows linearly if parameters are initialized at random, which suggests that the number of regions does not grow rapidly in practice.

A random forest is a set of decision trees (Breiman, 2001; Ho, 1995). Since the decision tree is an old model, the limitations of the model have been shown both theoretically and experimentally. For example, Grigoriev, Karpinski, and Yao (1998) showed that decision trees need an exponential number of nodes to represent the max function. Perez and Rendell (1996) showed that decision trees cannot generalize areas that do not appear in the training data. Bengio et al. (2010) also showed that decision trees cannot generalize the data with many variations. Cucker and Grigoriev (1999)

pointed out the importance of the depth in a decision tree and showed that a certain depth is needed to express a function with an arbitrary error. In order to analyze the model complexity of specific decision tree models, Yildiz (2015) developed algorithms for computing lower bounds of the VC dimension for univariate decision trees, and de Mello et al. (2019) developed an algorithm for measuring the Shattering coefficient of decision tree models. Mansour (1997) showed that the lower bound of the VC dimension of the binary decision tree with N nodes and input dimension d is $\Omega(N)$ and the upper bound is $O(N \log_2 d)$. As for the expressive ability of random forests, the upper bound of the VC dimension of ensemble learning using T classifiers with VC dimension d is known to be $O(dT \log_2 dT)$ (Shalev-Shwartz & Ben-David, 2014). Oshiro, Perez, and Baranauskas (2012) defined the density of the data and confirmed the relationship between the density and the optimal number of trees by experiments. Zhou and Mentch (2021) suggested that tree depth should be seen as a natural form of regularization across the entire procedure and showed via computational experiments that random forests with shallow trees are advantageous when the signal-to-noise ratio in the data is low. It seems from these studies that the representational power of random forests does not dramatically increase by increasing the number of trees. However, the trade-off between the number of trees and the number of nodes in random forests is still unclear with respect to the representational power. Therefore, this letter studies this trade-off in random forests, focusing on classification random forests among two major types of random forests (classification and regression ones). As for relations with neural networks, it is known that decision trees (Bengio et al., 2010; Sethi, 1990) and random forests (Biau et al., 2019) are efficiently embedded into neural networks using Heaviside functions, sigmoidal functions, and hyperbolic tangent functions as activation functions. However, it is not clear whether similar results hold for ReLU and related activation functions.

Binary decision diagrams (BDDs) are another important data structure to represent various areas of knowledge (Bryant, 1986). BDDs are considered an extension of binary decision trees in which rooted, directed acyclic graphs are used instead of binary trees. BDDs have been extensively applied to design of logic circuits, and a number of studies have been done on efficient design of BDDs (Meinel & Theobald, 1998). For relations with neural networks, Prasad, Assi, and Beg (2007) used neural networks to estimate the space complexity of BDDs, and Xu et al. (2018) used neural networks to reduce the size of ordered BDDs. However, to our knowledge, there is no work on comparison of the representational power of BDDs and neural networks. Therefore, this letter also studies relations between BDDs and neural networks with respect to representational power.

3 Definitions

In this section, we define and analyze models. First, we define tree models, specifically, decision trees and random forests. We consider normal

axis-aligned binary decision trees. That is, each node in the decision tree has exactly two child nodes and splits the input space by a hyperplane that is perpendicular to an axis and parallel to the other axes. A random forest is defined as a collection of decision trees, where the decision is made by majority vote. Next, we define binary decision diagrams as an extension of binary decision trees. Binary decision diagrams use directed acyclic graphs instead of trees. Finally, we define neural networks. We consider feedforward neural networks. In the following, R and R^d denote the set of real numbers and the d-dimensional Euclidean space, respectively. C denotes a set of target classes, where we assume that C is finite and thus each target class is represented by a positive integer. For a vector $\mathbf{x} \in R^d$, $\mathbf{x}_i$ (or, $(\mathbf{x})_i$) denotes the ith element of $\mathbf{x}$.

3.1 Decision Trees. The decision tree is one of the fundamental machine learning models (Breiman et al., 1984). It is represented as a rooted binary tree in which the input region is split into two regions at each nonleaf node. Since the decision tree is simple and has high explainability, it has been widely utilized. However, it has several drawbacks. In fact, some studies show performance limitations in both theoretical and experimental ways (Vilalta, Blix, & Rendell, 1997; Grigoriev et al., 1998).

Definition 1 *(Heaviside Function). A function $H : R \to \{0, 1\}$ is called the Heaviside function if it is expressed as*

$$H(x) = \begin{cases} 1 & (x \geq 0), \\ 0 & (x < 0). \end{cases} \tag{3.1}$$

Definition 2 *(Axis Aligned Function). A function $f : R^d \to \{0, 1\}$ is called an axis aligned function if f is expressed as $f(\mathbf{x}) = H(x_i - a)$, $f(\mathbf{x}) = 1 - H(\mathbf{x}_i - a)$, $f(\mathbf{x}) = H(a - \mathbf{x}_i)$, or $f(\mathbf{x}) = 1 - H(a - \mathbf{x}_i)$.*

An axis aligned function splits input space into two regions by using a hyperplane that is perpendicular to one axis and parallel to the other axes.

Definition 3 *(Decision Tree). A decision tree T is a rooted binary tree with a decision function $T : R^d \to C$ defined by*

$$T(\mathbf{x}) = \sum_{l \in L(T)} c_l \prod_{e \in ancestor\_edges(l)} f_e(\mathbf{x}). \tag{3.2}$$

Note that we use T to denote both a tree and the corresponding decision function. Here $L(T)$ is the set of all leaves in a binary tree T, $ancestor\_edges(l)$ is the set of all edges in the path from the root to a leaf l, f_e is an axis aligned function assigned to an edge e, and c_l is a positive integer that represents the class value assigned to l (see Figure 1). Moreover, each nonleaf node in T has two outgoing edges e and e' to which axis aligned functions f and $1 - f$ are assigned, respectively.

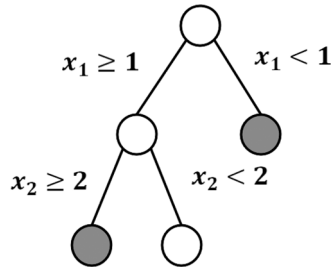

Figure 1: Example of a decision tree. White and gray leaves represent class 1 and class 2 leaves, respectively.

In a decision tree, for any input vector $\mathbf{x} \in R^d$, exactly one leaf l satisfies

$$\prod_{e \in ancestor\_edges(l)} f_e(\mathbf{x}) = 1.$$

Thus, we use $reach(\mathbf{x})$ to denote such a leaf l. In addition, we use L_c to denote the set of leaves with class label c.

3.2 Random Forests. The random forest is one of the machine learning models that uses multiple trees. Random forests are classified into two major types, classification random forests and regression random forests, where the former and latter use decision trees and regression trees, respectively. As mentioned in section 1, this letter considers classification random forests only. A classification random forest determines the result by majority vote of decision trees where each tree uses a randomly selected subset of samples and a randomly selected subset of features. Ho (1995) introduced the concept of random decision forests in which trees are built using randomly selected subspaces. Then Breiman (2001) established the concept of random forests by introducing the use of a randomly selected subset of features as candidates at each split in each tree. In construction of a random forest, trees can be computed in parallel.

Notice that a key component of random forests is the randomization injected into the learning procedure—not only randomness in drawing resamples but also that in building individual trees. However, since we focus on the representational power of prediction models, we ignore the randomness and simply treat a random forest as a collection of decision trees.

Definition 4 *(Indicator Function). An indicator function* $1_S : R^d \to \{0, 1\}$ *is defined by*

$$1_S(x) = \begin{cases} 1 & (x \in S), \\ 0 & (x \notin S). \end{cases} \tag{3.3}$$

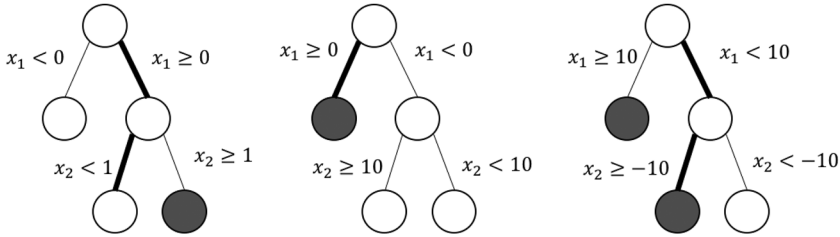

Figure 2: Example of a random forest. White and gray leaves represent class 1 and class 2 leaves, respectively. In this case, $RF(\mathbf{x}) = 2$ for $\mathbf{x} = (5, -5)$ because $T_1(\mathbf{x}) = 1$ whereas $T_2(\mathbf{x}) = T_3(\mathbf{x}) = 2$.

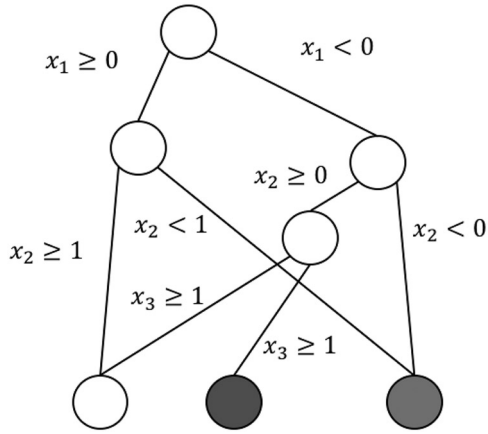

Figure 3: Example of a binary decision diagram. The bottom nodes are the output nodes.

Definition 5 *(Random Forest). A random forest is a collection of trees* $\mathcal{T} = \{T_1, \ldots, T_{|\mathcal{T}|}\}$ *with a decision function* $RF : R^d \to C$ *defined by*

$$RF(\mathbf{x}) = \arg\max_{c \in C} \sum_{T_i \in \mathcal{T}} 1_{\{c\}}(T_i(\mathbf{x})). \tag{3.4}$$

Thus, a random forest outputs the result of majority vote of decision trees (see Figure 2). In the following, a random forest consisting of n nodes in total is called a random forest with n nodes.

3.3 Binary Decision Diagrams. We define binary decision diagrams (BDDs) as an extension of decision trees. These diagrams use directed acyclic graphs instead of trees, as shown in Figure 3. Thus, binary decision

diagrams have much stronger representative power than decision trees have. For example, a d-bit parity function can be expressed by a binary decision diagram with $O(d)$ nodes, whereas a decision tree needs $\Omega(2^d)$ nodes. Moreover, each class c has $O(|L_c|)$ regions in a decision tree, whereas each class has $O((2W)^D)$ regions in a binary decision diagram where D is the depth of the binary decision diagram and each layer has $O(W)$ nodes.

Definition 6 *(Binary Decision Diagram, BDD). A binary decision diagram is a binary directed acyclic graph with a decision function $B : R^d \to C$ defined by*

$$B(\mathbf{x}) = \sum_{l \in L(B)} c_l \sum_{p \in paths(l)} \prod_{e \in edges(p)} f_e(\mathbf{x}). \tag{3.5}$$

Here $L(B)$ is the set of all nodes whose out-degree is 0 (these nodes are called *output nodes*), $paths(l)$ is the set of all paths that connect the root and an output node l, and $edges(p)$ is the set of all edges that are included in a path p. In addition, each internal node has exactly two outgoing edges to which axis aligned functions $f, 1 - f$ are assigned, respectively.

3.4 Neural Networks. In this letter, we consider feedforward neural networks. At each depth (i.e., at each layer), an input vector (from the preceding depth) is multiplied by a weight matrix, a bias vector is added, and then nonlinear activation functions are applied.

Definition 7 *(Piecewise Linear Function). A piecewise linear function $f : R \to R$ is a function such that the input domain R is divided into a finite set of ranges in each of which f is a linear function.*

Definition 8 *(Sigmoidal Function). A function $\sigma : R \to R$ is called a sigmoidal function if there exist some constants a, b $(a \neq b)$ such that*

$$\lim_{x \to -\infty} \sigma(x) = a, \quad \lim_{x \to \infty} \sigma(x) = b. \tag{3.6}$$

Definition 9 *(F_{pl} Function Family). F_{pl} is a function family such that each $f \in F_{pl}$ is a piecewise linear function $f : R \to R$ that satisfies*

$$\lim_{x \to -\infty} f'(x) \neq \lim_{x \to \infty} f'(x). \tag{3.7}$$

Definition 10 *(F_{sigpl} Function Family). F_{sigpl} is a function family such that each $f \in F_{sigpl}$ is a sigmoidal function or an element of F_{pl}.*

Function family F_{sigpl} includes almost all of practical activation functions such that like sigmoid, tanh, ReLU (Nair & Hinton, 2010), and Leaky ReLU (Trottier, Gigu, & Chaib-draa, 2017) (see Figure 4).

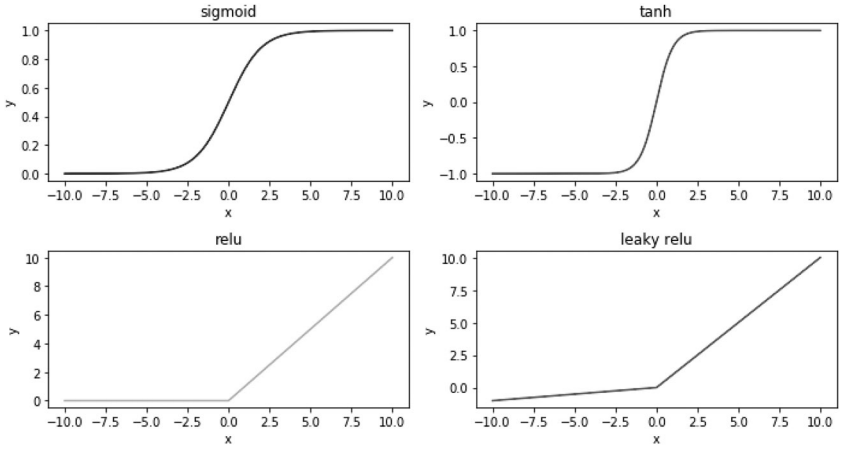

Figure 4: Examples of functions in F_{sigpl}.

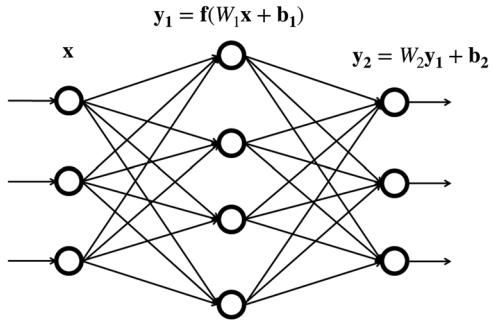

Figure 5: Example of a depth-2 neural network.

Definition 11 *(Neural Network). Let* $\mathbf{y}_0 = \mathbf{x}$ *be an input vector and* u_k *be the number of depth-k nodes. Then the output vector of depth-k nodes in a neural network is defined as*

$$\mathbf{y}_k = \mathbf{f}(W_k\mathbf{y}_{(k-1)} + \mathbf{b}_k). \tag{3.8}$$

The output vector of the depth-d neural network is defined as

$$\mathbf{y}_d = W_d\mathbf{y}_{(d-1)} + \mathbf{b}_d. \tag{3.9}$$

Here, W_k is a $u_k \times u_{(k-1)}$ matrix, $\mathbf{b}_k$ is a u_k-dimensional vector (a vector of biases), and $\mathbf{f}$ is a list of the identical activation functions (see Figure 5).

Table 1: Summary of Symbols Introduced in Section 3.

Symbol	Meaning
$H(x)$	Heaviside function
$1_S(x)$	Indicator function
$\sigma(x)$	Sigmoid function
F_{pl}	Family of piecewise linear functions satisfying equation 3.7
F_{sigpl}	Family of functions in F_{pl} and sigmoid functions
f_{parity}	Parity function
$T(\mathbf{x})$	Decision tree ($\mathbf{x}$: input data)
$RF(\mathbf{x})$	Random forest
$B(\mathbf{x})$	Binary decision diagram
$L(T)$	Set of leaves in tree T
$ancestor\_edges(l)$	Set of edges in the path from root to leaf l
f_e	Axis aligned function assigned to edge e
c_l	Class label (integer) assigned to leaf l
$reach(\mathbf{x})$	Leaf corresponding to input vector $\mathbf{x}$
$paths(l)$	Set of paths from root to output node l in BDD
$edges(p)$	Set of edges included in path p

In this letter, we consider only classification problems. Neural networks have exactly the same number of output nodes as the number of classes, and their output class is determined as the class whose corresponding output node has the highest value.

3.5 Parity Function. Here, we define the *parity function* that we use as an example function to show lower bounds in latter sections. The parity function outputs 1 if and only if the sum of the elements in the input is odd. Two-bit parity function is the same function as *XOR*.

Definition 12 *(Parity Function). d-bit parity function* $f_{parity} : \{0,1\}^d \to \{0,1\}$ *is defined as*

$$f_{parity}(\mathbf{x}) = \begin{cases} 1 & (\textit{the number of one in } \mathbf{x} \textit{ is odd}), \\ 0 & (\textit{the number of one in } \mathbf{x} \textit{ is even}). \end{cases} \tag{3.10}$$

Table 1 summarizes the symbols introduced in this section.

4 Comparison of Tree Models and Neural Networks

In this section, we compare the representational power of tree models and neural networks when both of them have $O(n)$ nodes. It is already known that a decision tree has shallow depth in the context of neural networks; for example, Sethi (1990) and Bengio et al. (2010) showed that the architectural depth of a decision tree is 2 by analogy with the disjunctive normal form

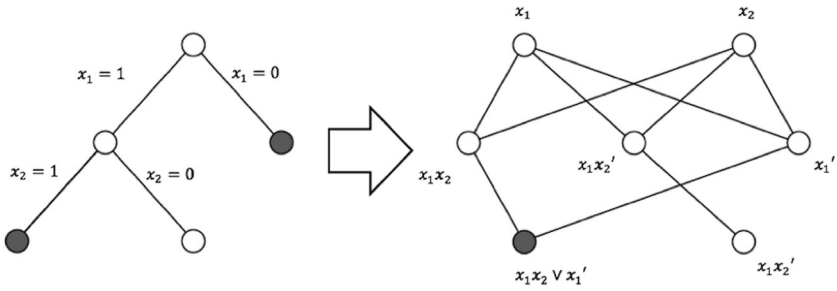

Figure 6: Transformation of a decision tree to a depth-2 network.

(see Figure 6). In depth-1, each node of the network corresponds to a leaf in the decision tree and takes a product of conditional expressions assigned to edges in the path from the root to the leaf. In depth-2, each node corresponds to a class and takes a sum of the values of depth-1 nodes. In the case of a random forest, Biau et al. (2019) showed that it is enough to add another layer to sum up the output values of depth-2. They also showed the sum-up can be done by directly connecting the nodes corresponding to the leaves of decision trees to the output node.

From the above discussion, we can speculate that random forests have less representational power than fixed-depth neural networks, and random forests do not have the same nature as the depth efficiency. However, most recent results about the depth efficiency are shown in the case of neural networks whose activation functions are piecewise linear functions. Thus, we begin with comparison of the representational power of decision trees and neural networks with more general activation functions. We will prove that random forests have no more representational power than depth-3 neural networks whose activation functions are in F_{sigpl}. Recall that F_{sigpl} is the function family that was defined in section 3. Thus, the statements in section 4 hold for activation functions that are often used in practice, like ReLU, sigmoid, and tanh.

4.1 Decision Trees and Neural Networks. To prove random forests have no more representational power than depth-3 neural networks, we compare the representational power of decision trees and neural networks and prove that decision trees have no more representational power than depth-3 neural networks. First, we consider neural networks whose activation functions are Heaviside functions.

Hereafter, we assume that the domain of input data is a finite set $S \subseteq R^d$ because each input datum is represented as a vector of finite precision numbers in practice and thus the number of possible input vectors is finite for fixed dimensions d.

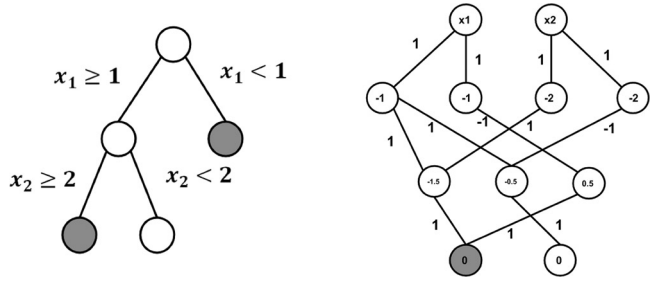

Figure 7: These decision tree, and neural networks are equivalent. Each constant on the edge of the neural network is the weight, and each constant on the node is the bias. Weight zero edges are omitted. Activation functions are Heaviside functions.

Lemma 1. *For any decision tree with n nodes $T : S \to C$, there is an equivalent depth-3 neural network with $O(n)$ nodes using Heaviside functions as the activation functions.*

Proof. For a given decision tree T, we construct a depth-3 neural network that always outputs the same class as the decision tree does (see Figure 7). First, we construct depth-1 nodes. For each edge e in T, we construct a depth-1 node v_e so that v_e outputs 1 if and only if the condition assigned to e is satisfied. To this end, for an edge e with $\mathbf{x}_i \geq a$, we assign $H(\mathbf{x}_i - a)$ to v_e as an activation function, and for an edge e with $\mathbf{x}_i > a$, we assign $1 - H(a - \mathbf{x}_i)$ to v_e. Although $1 - H(x)$ is not a Heaviside function, we can modify the network by adjusting the weights of edges and biases so that only Heaviside functions are used. Identically, for an edge e with $\mathbf{x}_i \leq a$, we assign $H(a - \mathbf{x}_i)$ to v_e, and for an edge e with $\mathbf{x}_i < a$, we assign $1 - H(\mathbf{x}_i - a)$ to v_e.

Next, for each leaf l in T, we construct a depth-2 node v_l in the neural network. Let D_l be the depth of l and V_l be the set of depth-1 neural network nodes corresponding to edges in *ancestor_edges*(l) (recall that *ancestor_edges*(l) denotes the set of edges in the path from the root to a leaf l in the decision tree). Then we assign

$$H\left(\left(\sum_{v \in V_l} v\right) - D_l + 0.5\right) \tag{4.1}$$

to v_l. Notice that we use v to denote both a node and its output value.

Finally, for each output class $c \in C$ for T, we construct a depth-3 node (i.e., an output node) v_c. Let I_c be the set of depth-2 nodes corresponding to leaves with label c (i.e., $I_c = \{v_l | l \in L_c\}$). Then we assign

$$\sum_{v \in I_c} v \tag{4.2}$$

to v_c. It is straightforward to see that v_c takes value 1 for an input vector $\mathbf{x}$ if and only if the class label of a leaf $l = reach(\mathbf{x})$ is c. Since the number of nodes in the constructed neural network is $O(n)$, the lemma holds. □

Using this lemma, we can prove that decision trees have no more representational power than depth-3 neural networks with $O(n)$ nodes using any activation function in F_{sigpl}.

Theorem 1. *Let f be an arbitrary function in F_{sigpl}. Then for any decision tree with n nodes $T : S \to C$, there exists an equivalent depth-3 neural network with $O(n)$ nodes using f as the activation functions.*

Proof. Since S is a finite set of samples, we can assume without loss of generality (w.l.o.g.) that there are no samples on a decision boundary of a decision tree. We will prove that for any neural network with n nodes using Heaviside functions as activation functions, there exists an equivalent neural network with $O(n)$ nodes using $f \in F_{sigpl}$ as activation functions.

Let g be a sigmoidal function. Thus, g satisfies

$$\exists a, b \in R \; s.t. \; a \neq b, \; \lim_{x \to \infty} g(x) = a, \; \lim_{x \to -\infty} g(x) = b. \tag{4.3}$$

Then we can express H by g for $x \neq 0$. In fact,

$$H(x) = \lim_{\epsilon \to 0} \left(g\left(\frac{x}{\epsilon}\right) - b \right) \frac{1}{a-b} = \frac{1}{a-b} \lim_{\epsilon \to 0} g\left(\frac{x}{\epsilon}\right) - \frac{b}{a-b}. \tag{4.4}$$

Of course, we need to use some contact $\epsilon > 0$ in practice, and thus there is a difference between $H(x)$ and its representation by using $g(x)$. However, the final decisions can be the same if we use sufficiently small constant $\epsilon > 0$.

Next, let h be an arbitrary function included in F_{pl}. Thus, h satisfies

$$\begin{aligned} &\exists c, d, c', d' \in R \; s.t. \; c \neq c', \\ &\lim_{x \to \infty} h(x) = cx + d, \; \lim_{x \to -\infty} h(x) = c'x + d'. \end{aligned} \tag{4.5}$$

Then, since

$$\lim_{x \to \infty} h(x+1) - h(x) = c, \; \lim_{x \to -\infty} h(x+1) - h(x) = c', \tag{4.6}$$

we can express H by using $h(x+1)$ and $h(x)$ (i.e., two nodes using h as activation functions) for $x \neq 0$ (see Figure 8).

Therefore, for any $f \in F_{sigpl}$ and for any depth-3 neural network with $O(n)$ nodes using H as activation functions, there exists an equivalent depth-3 neural network with $O(n)$ nodes using f as activation functions. Thus, the theorem follows from lemma 1. □

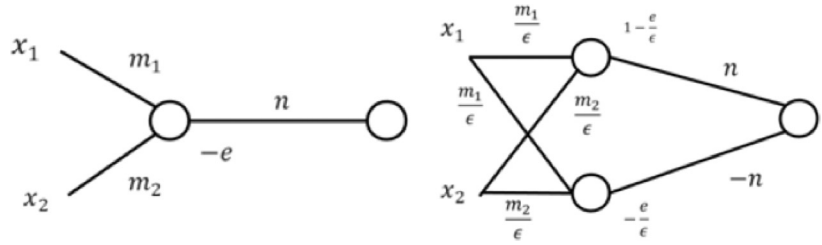

Figure 8: These two neural networks are equivalent. A constant on each edge is a weight, and weight zero edges are omitted. A constant near each node is a bias. The left neural network uses Heaviside activation functions, and the right neural network uses ReLU activation functions. ϵ is a small, positive value. The left network represents $nH(m_1x_1 + m_2x_2 - e)$, and the right network represents $nReLU(m_1x_1/\epsilon + m_2x_2/\epsilon + 1 - e/\epsilon) - nReLU(m_1x_1/\epsilon + m_2x_2/\epsilon - e/\epsilon)$. These two expressions can be the same for the finite set of inputs if we use sufficiently small constant $\epsilon > 0$.

4.2 Random Forests and Neural Networks. From the results in section 4.1, we can easily prove that random forests have no more representational power than depth-3 neural networks.

Theorem 2. *Let f be an arbitrary function in F_{sigpl}. Then for any random forest with n nodes $RF : S \to C$, there exists an equivalent depth-3 neural network with $O(n)$ nodes using f as the activation functions.*

Proof. We prove the theorem only for the case of the Heaviside function. The other cases can be proven as in theorem 1.

First, for each decision tree T_i $(i = 1, \ldots, |\mathcal{T}|)$, we construct a depth-3 neural network. Here, each depth-2 node v_l, which corresponds to a leaf l in some T_i, outputs 1 for an input vector $\mathbf{x}$ if and only if $reach(\mathbf{x}) = l$ in T_i, and otherwise outputs 0.

Next, we remove all depth-3 nodes, and then for each class $c \in C$, we construct a depth-3 node v_c. Let L_c be the set of all leaves (in all T_is) having label c and $I_c = \{v_l | l \in L_c\}$. We assign

$$\sum_{v \in I_c} v \tag{4.7}$$

to v_c.

Then the output node with the highest value corresponds to the class determined by RF. Furthermore, this neural network clearly satisfies the requirement of the theorem. □

This result holds also for weighted random forests in which the decision is made by weighted vote in place of majority vote.

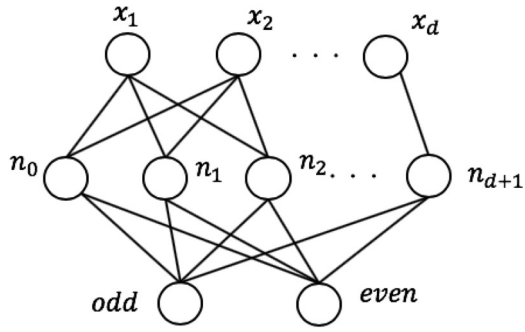

Figure 9: This neural network represents parity function.

4.3 Efficiency for the Number of Nodes in Tree Models. In this section, we compare the representational power of tree models and neural networks. In the context of circuits, it is obvious that tree models have shallow depth; in fact, their depth is 2. However, most results of the efficiency on the number of nodes in neural networks have been proved with neural networks whose activation functions are piecewise linear functions. Thus, we compare decision trees and neural networks whose activation functions are in the function family F_{sigpl}, which includes most practical activation functions.

It is seen from theorem 2 that tree models (i.e., decision trees and random forests) with n nodes can be expressed with depth-3 neural networks with $O(n)$ nodes whose activation functions are in the function family F_{sigpl}. That is, the functions for which the depth works well in neural networks (e.g., periodic function) cannot be expressed efficiently with tree models, since the function family whose elements can be expressed efficiently by a decision tree is included by the function family whose elements can be expressed efficiently by a depth-3 neural network. In contrast, there are many functions that can be efficiently expressed by a depth-3 neural network but require many nodes in tree models. For example, depth-3 neural networks can express the d bit-parity function with $O(d)$ nodes (Bengio & Lecun, 2007), while decision trees need an exponential number of nodes. Figure 9 shows the neural network that expresses the d-bit parity function. Each depth-1 node v_i in the neural network calculates whether the number of ones in $x = (x_1, x_2, \ldots, x_d)$ is larger than i or not. This can be established by simple summation and bias $-i$. Depth-2 output nodes also take simple summation as $\sum_{\text{even } i} v_i - v_{i-1}$ for the output node "odd" and $\sum_{\text{odd } i} v_i - v_{i-1}$ for the output node "even." This neural network calculates the d-bit parity function since $v_i - v_{i-1} = 1$ holds if and only if exactly i of d input bits are 1.

Thus, the function family corresponding to n-nodes tree models (resp., n-nodes decision trees) is a subset (resp., a proper subset) of the function

family corresponding to depth-3, $O(n)$ nodes neural networks. Therefore, we can conclude that tree models have no more representational power than depth-3 neural networks.

5 Comparison of Binary Decision Diagrams and Neural Networks

In the previous section, we compared the representational power of neural networks and tree models and showed that tree models have no more representational power than depth-3 neural networks. In this section, we prove analogous results for binary decision diagrams. Recall that we defined the binary decision diagram as an extension of the binary decision tree. Only the difference between binary decision diagrams and decision trees is that binary decision diagrams use binary-directed acyclic graphs instead of binary trees. Binary decision diagrams have higher representational power than decision trees because it is obvious that n nodes decision trees can be simulated by n nodes binary decision diagrams. Moreover, a decision tree with n nodes outputs class c when

$$\bigvee_{l_i \in L(c)} l_i(\mathbf{x}) \tag{5.1}$$

is satisfied, while a depth- D binary decision diagram with n nodes outputs class c when

$$\bigvee_{p_i \in P(c)} p_i(\mathbf{x}) \tag{5.2}$$

is satisfied. Here $\mathbf{x}$ is an input vector, and l_i and p_i are conditional expressions corresponding to the conjunctions of the conditional expressions from the root to the leaf and output node, respectively. $L(c)$ is a set of the conjunctions corresponding to paths from the root to leaves with class label c in the decision tree, and $P(c)$ is a set of conjunctions corresponding to paths from the root to output nodes with class label c in the binary decision diagram. Then, assuming that each layer in the binary decision diagram has $O(\frac{n}{D})$ nodes, it is seen that $|L(c)|$ is $O(n)$, whereas $|P(c)|$ is $O((\frac{2n}{D})^D)$. Since $O((\frac{2n}{D})^D)$ is much larger than $O(n)$, this suggests that binary decision diagrams have much higher representational power than decision trees have.

Actually, there are functions that can be expressed efficiently with binary decision diagrams, while they need many nodes in decision trees. For example, a decision tree needs an exponential number of nodes to express the parity function, while a binary decision diagram needs only a linear number of nodes. Figure 10 shows the 2-bit parity function represented by a decision tree and a binary decision diagram.

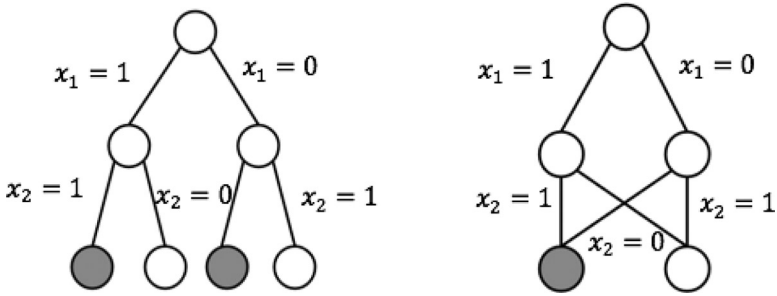

Figure 10: Two-bit parity function expressed by a decision tree and a binary decision diagram. In the bottom, gray nodes and white nodes represent even and odd classes, respectively.

5.1 Binary Decision Diagrams and Neural Networks. Here we show that for any depth-D binary decision diagram with n nodes, there exists an $O(n)$ nodes neural network that can express the binary decision diagram.

Theorem 3. *Let f be an arbitrary function in F_{sigpl}. Then for any binary decision diagram with n nodes $B : S \to C$, there exists an equivalent depth-$(D+1)$ neural network with $O(n)$ nodes using f as the activation functions. The network has skip connections.*

Proof. We prove the theorem only for the case of the Heaviside function. The other cases can be proven as in theorem 1.

First, for each edge e in the binary decision diagram, we construct a depth-1 neural network node v_e where v_e outputs 1 if and only if e is activated. This can be done as in the proof of lemma 1. For example, for an edge e with $x_i \geq a$, we assign $H(\mathbf{x}_i - a)$ to v_e as an activation function.

Next, for each depth-s $(1 \leq s \leq D-1)$ binary decision diagram edge e, we construct a depth-$(s+1)$ neural network node $v_e^{(s+1)}$. Suppose that e has $e_1, e_2, \ldots, e_k$ as input edges. As before, we use v to denote both a node and its output value. Then $v_e^{(s+1)}$ is determined by

$$v_e^{(s+1)} = H((k+1)v_e + v_{e_1}^{(s)} + \cdots + v_{e_k}^{(s)} - (k+1)). \tag{5.3}$$

Note that a neural network edge from v_e to $v_e^{(s+1)}$ corresponds to a skip connection (see Figure 11).

Finally, for each output node v with class label c in the binary decision diagram, we construct a depth-$(D+1)$ neural network node $v_c^{(D+1)}$. Suppose that v has $e_1, e_2, \ldots, e_k$ as input edges. Then $v_c^{(D+1)}$ is determined by

$$v_c^{(D+1)} = v_{e_1}^{(D)} + \cdots + v_{e_k}^{(D)}. \tag{5.4}$$

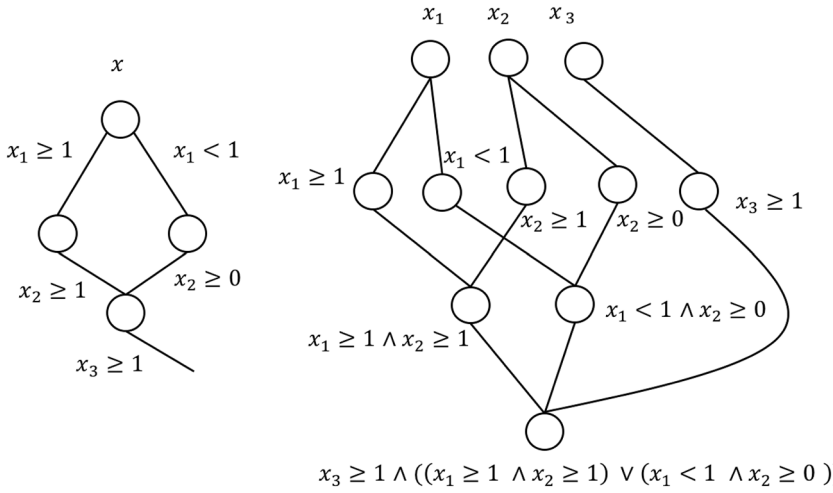

Figure 11: Example of transformation from a binary decision diagram to a neural network. For each edge in the binary decision diagram, we construct a depth-2 neural network node corresponding to the path from the root to the edge. Depth-1 neural network nodes correspond to edges in the binary decision diagram.

Furthermore, since each node in the binary decision diagram has two output edges, the number of nodes in the resulting neural network is $O(n)$. □

Theorem 4. *Let f be an arbitrary function in F_{sigpl}, let D be the depth of a given binary decision diagram, and assume that the diagram has no skip connections. Then for any binary decision diagram with n nodes $B : S \to C$ and a positive integer b, there exists an equivalent depth-$\lceil \frac{2D}{b} + 1 \rceil$ neural network with $O(bn^b)$ nodes using f as the activation functions, where the network has skip connections.*

Proof. We construct neural network nodes that correspond to binary decision diagram nodes whose depths are multiples of b. For each $k = 1, \ldots, \lceil \frac{D}{b} \rceil$, we construct network nodes corresponding to decision diagram nodes between depth-$b(k-1)$ and depth-bk.

This construction needs two layers per k, as shown in Figure 12. In the first layer, for each path p from depth-$b(k-1)$ to depth-bk in the binary decision diagram, we construct a node v_p representing the conjunction of the conditions given by p and the condition given by the beginning node of p. In the second layer, for each depth-bk binary decision diagram node u, we construct a node v_u with inputs from nodes v_ps corresponding to paths ps ending at u (see Figure 13).

Clearly the binary decision diagram and the constructed neural network always output the same class. Since the number of paths between

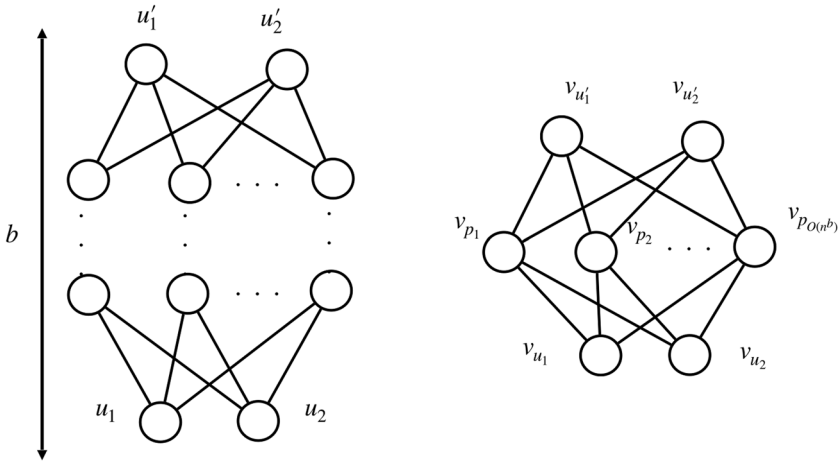

Figure 12: For each sub-binary decision diagram with depth-b, we construct the corresponding depth-2 neural subnetwork.

depth-$b(k-1)$ nodes and depth-bk nodes in the binary decision diagram is $O(n^b)$, the number of nodes per k is $O(n+n^b)$. Therefore, the constructed neural network has depth-$\lceil \frac{2D}{b} + 1 \rceil$ and $O(bn^b)$ nodes, where it has skip connections. □

As shown below, the depth of the network can be significantly reduced without significantly increasing the number of nodes.

Theorem 5. *Let f be an arbitrary function in F_{sigpl}, let D be the depth of a given binary decision diagram, and assume that the diagram has no skip connections and has $O(\frac{n}{D})$ nodes in each layer. Then for any binary decision diagram with n nodes $B : S \to C$, there exists an equivalent depth-$(2\lceil \log_2 D \rceil + 1)$ neural network with $O(\frac{n^3}{D^2})$ nodes using f as the activation functions, where the network has skip connections.*

Proof. For a specific pair of nodes (s,t) in the binary decision diagram, we construct a neural network node $v_{(s,t)}$ such that $v_{(s,t)}$ takes value 1 if and only if at least one path between s and t is activated.

For an integer $S \in 1, \ldots, D$, let V_S denote the set of depth-S nodes in the binary decision diagram. For integers S, T such that $S < T$, we define $V_{(S,T)}$ by

$$V_{(S,T)} = \{v_{(s,t)} | s \in V_S, t \in V_T\}. \tag{5.5}$$

We construct $v_{(s,t)}$s and the corresponding activation functions in a bottom-up manner. Suppose that we have $V_{(S,T)}$ and $V_{(T,U)}$ at the same depth d along with activation functions where $1 \leq S < T < U \leq D$. Then we

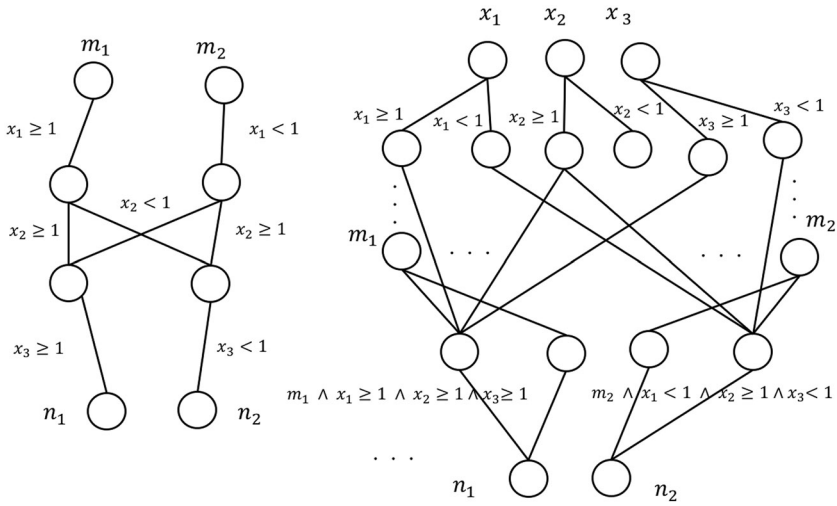

Figure 13: Example of transformation from a binary decision diagram to a neural network for $b = 3$. Depth-1 nodes in the neural network correspond to edges in the binary decision diagram. For each path from m_1 or m_2 to n_1 or n_2, we construct a corresponding neural network node.

construct neural network nodes in $V_{(S,U)}$ at depth $d + 2$ such that the output value of $v_{(s,u)} \in V_{(S,U)}$ is determined by

$$v_{(s,u)} = \bigvee_{t \in V_T} (v_{(s,t)} \wedge v_{(t,u)}). \tag{5.6}$$

This can be done adding nodes (at depth $d + 1$) representing $v_{(s,t)} \wedge v_{(t,u)}$ for each pair $(v_{(s,t)}, v_{(t,u)})$, which we call conjunction nodes. Therefore, to construct nodes in $V_{(S,U)}$, we need two additional layers and $O(|V_S||V_T||V_U|) = O((n/D)^3)$ additional nodes because we assume that the binary decision tree has $O(n/D)$ nodes in each layer.

The bottom-up construction is done as follows. First, for depth-1 neural network nodes, we construct nodes in $V_{(0,1)}, V_{(1,2)}, \ldots, V_{(D-1,D)}$, that is, we construct nodes corresponding to edges in the binary decision diagram. We can see that at most $2n$ nodes are at depth 1 because each (nonoutput) node has exactly two output edges in the binary decision diagram. Next, for depth-3 neural network nodes, we construct nodes in $V_{(0,2)}, V_{(2,4)}, \ldots, V_{(D-2,D)}$ along with conjunction nodes at depth 2. Then, at depth-5 neural network nodes, we construct nodes in $V_{(0,4)}, V_{(4,8)}, \ldots,$ $V_{(D-4,D)}$ along with conjunction nodes at depth 4. We repeat this procedure until nodes in $V_{(0,D)}$ are constructed. Although $D = 2^h$ for some h is assumed here, we can modify the construction procedure for the case of $D \neq 2^h$.

From the construction, we can see that each node $v_{(s,t)}$ takes value 1 if and only if at least one path between s and t is activated. Finally, we can see that the order of the number of nodes in the constructed neural network is bounded by

$$2n + \frac{D}{2}\frac{n^3}{D^3} + \frac{D}{4}\frac{n^3}{D^3} + \cdots \leq 2n + \frac{n^3}{D^2}, \tag{5.7}$$

and its depth is $2\lceil \log_2 D \rceil + 1$. Therefore, the theorem holds. □

6 The Number of Trees and Nodes in Random Forests

In this section, we compare random forests consisting of different numbers of trees.

6.1 Lower Bound of the Number of Nodes. In the previous section, we proved that tree models do not have the same efficiency on the number of nodes as neural networks. However, random forests may have specific efficiency on the number of nodes. We show that in some cases, the efficiency on the number of nodes is generated from the number of trees by proving a lower bound of the number of nodes to express an n-trees random forest with n nodes by a T-trees random forest with $T < n$. Let $(\mathbf{x}_i, c_i)$ be a pair of a vector and its class. A leaf l is called a *support leaf* for $(\mathbf{x}_i, c_i)$ if $reach(\mathbf{x}_i) = l$ and $c(l) = c_i$. First, we show that if any pair of two distinct samples in the same class has fewer than $(T+1)/2$ common support leaves, then the total number of nodes in a T-trees random forest must be large.

Lemma 2. *Let (X, c) be a pair of a set of vectors and a classification function satisfying the following property and $M = |X|$. Then any T-trees random forest $RF : S \to \{1, 2\}$ that can correctly classify all elements in X (according to c) has $\Omega(M^{\frac{2}{T+1}})$ nodes, where T is a positive odd integer.*

For every T-trees random forest that correctly classifies X,

$$\begin{aligned} &\forall \mathbf{x}_1 \neq \mathbf{x}_2 \in X, c(\mathbf{x}_1) = c(\mathbf{x}_2) = c_i \to \\ &|\{l | c(l) = c_i, reach(\mathbf{x}_1) = reach(\mathbf{x}_2) = l\}| \\ &< \frac{T+1}{2}, \end{aligned} \tag{6.1}$$

holds, where $c(\mathbf{x})$ is a class of $\mathbf{x}$, $reach(\mathbf{x})$ is the leaf that $\mathbf{x}$ reaches, and $c(l)$ is a class label of a leaf l.

Proof. The above property states that any two different inputs $\mathbf{x}_1$ and $\mathbf{x}_2$ from the same class c_i do not share more than half of the leaf nodes having class label c_i in any random forest that recognizes X.

Let $L_i = \{l | c(l) = i, l \text{ is a leaf}\}$, $h_i = |L_i|$. Thus, h_i is the number of leaves that have class label i. Let $M_i = |\{\mathbf{x} \in X | c(\mathbf{x}) = i\}|$. That is, M_i denotes the number of class i samples in X. Let $R_i(\mathbf{x}) = \{l | reach(\mathbf{x}) = l, c(l) = i\}$.

Let $\mathbf{x}_1$ be a vector in X such that $c(\mathbf{x}_1) = 1$. Let S_1 be an arbitrary subset of $R_1(\mathbf{x}_1)$ such that $|S_1| = \frac{T+1}{2}$. It is seen from the property that for any vector $\mathbf{x}_2$ in X such that $c(\mathbf{x}_2) = 1$ and $\mathbf{x}_2 \neq \mathbf{x}_1$, there exists $S_2 \subseteq R_1(\mathbf{x}_2)$ such that $|S_2| = \frac{T+1}{2}$ and $S_2 \neq S_1$. Furthermore, for any vector $\mathbf{x}_3$ in X such that $c(\mathbf{x}_3) = 1$, $\mathbf{x}_3 \neq \mathbf{x}_1$, and $\mathbf{x}_3 \neq \mathbf{x}_2$, there exists $S_3 \subseteq R_1(\mathbf{x}_3)$ such that $|S_3| = \frac{T+1}{2}$, $S_3 \neq S_1$, and $S_3 \neq S_2$. We can repeat this procedure until all $\mathbf{x}_i$s in M_1 are tested. This means that for each $\mathbf{x}_i \in M_1$, there exists a unique subset S_i such that $|S_i| = \frac{T+1}{2}$. Thus,

$$\binom{h_1}{\frac{T+1}{2}} \geq M_1 \tag{6.2}$$

holds. Similarly,

$$\binom{h_2}{\frac{T+1}{2}} \geq M_2 \tag{6.3}$$

holds. Hence, we have

$$h_1^{\frac{T+1}{2}} \geq M_1, \tag{6.4}$$

$$h_2^{\frac{T+1}{2}} \geq M_2. \tag{6.5}$$

Therefore, the number of leaves in this random forest is

$$h_1 + h_2 \geq M_1^{\frac{2}{T+1}} + M_2^{\frac{2}{T+1}} \geq M^{\frac{2}{T+1}}, \tag{6.6}$$

from which the theorem follows. □

With this lemma, we prove a lower bound of the number of nodes to express an n-trees random forest with n nodes by a T-trees random forest. Specifically, we show that there exists a function that satisfies the condition in the lemma. This function only needs a linear number of nodes in an n-trees random forest; however, it needs a large number of nodes in a T-trees random forest with $T < n$.

Theorem 6. *There exists an n-trees random forest with $O(n)$ nodes $RF : \{0,1\}^n \to \{1,2\}$ such that any T-trees random forest needs $\Omega((\frac{2^n}{\sqrt{n}})^{\frac{2}{T+1}})$ nodes to represent it, where T is any odd integer such that $0 < T < n$.*

Proof. Let $c(\mathbf{x})$ be a function such that it takes value 1 if and only if the sum of the elements of $\mathbf{x}$ is at least $\frac{n+1}{2}$, where we assume w.l.o.g. that n is an odd number. That is, $c(\mathbf{x})$ is a kind of majority function. Let X be the set of vectors

$\mathbf{x}$ such that the sum of the elements of $\mathbf{x}$ is either $\frac{n+1}{2}$ or $\frac{n+1}{2} - 1$. Note that for each $\mathbf{x} \in X$, $c(\mathbf{x}) = 1$ if this sum is $\frac{n+1}{2}$ and $c(\mathbf{x}) = 2$ otherwise. Clearly, $c(\mathbf{x})$ can be represented by an n-trees random forest with $O(n)$ nodes: T_i outputs 1 if $(\mathbf{x})_i = 1$, and 2 otherwise.

We will prove by contradiction that for any $T(T < n)$, X satisfies the condition in lemma 2. Assume that there exists a T-trees random forest that classifies X correctly and (X, c) does not satisfy the condition in lemma 2. Then we can assume w.l.o.g. that there exist two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ such that $c(\mathbf{x}_1) = c(\mathbf{x}_2) = 1$ holds and $\frac{T+1}{2}$ or more leaves satisfy $reach(\mathbf{x}_1) = reach(\mathbf{x}_2) = l$ and $c(l) = 1$.

Since $\mathbf{x}_1 \neq \mathbf{x}_2$, we can assume w.l.o.g. that there exists at least one index k such that $(\mathbf{x}_1)_k = 1$ and $(\mathbf{x}_2)_k = 0$ hold. Let $\mathcal{T}'$ be the set of trees T_i such that $reach(\mathbf{x}_1) = reach(\mathbf{x}_2) = l$ and $c(l) = 1$, from which it is seen that the path from the root to a leaf l in any $T_i \in \mathcal{T}'$ is not affected by the kth element of input vectors.

Let $\mathbf{x}'_1$ be the vector obtained by inverting the kth bit of $\mathbf{x}_1$. Then, $reach(\mathbf{x}_1) = reach(\mathbf{x}'_1) = l$ holds for all trees in $\mathcal{T}'$ because the kth element of an input vector does not affect the path from the root to $l = reach(\mathbf{x}_1)$. However, $\mathbf{x}'_1$ should be classified as class 2 because the sum of elements in $\mathbf{x}'_1$ is equal to $\frac{n+1}{2} - 1$. This fact contradicts the assumption that X is classified correctly by a T-trees random forest. Therefore, (X, c) satisfies the condition in lemma 2.

Furthermore, the size of X is given by

$$|X| = 2\binom{n}{\frac{n+1}{2}} \geq \frac{2^n}{\sqrt{n}}. \tag{6.7}$$

Therefore, from lemma 2, the number of required nodes is

$$\Omega\left(\left(\frac{2^n}{\sqrt{n}}\right)^{\frac{2}{T+1}}\right). \tag{6.8}$$

□

7 Conclusion

In this letter, we have investigated the relationship on the representational power of machine learning models under some restriction of the number of nodes. Without this restriction, discussions in this letter would be meaningless because neural networks can express any Borel measurable function and decision trees can express any function whose domain is finite.

Based on other studies (Sethi, 1990; Bengio et al., 2010; Biau et al., 2019), we showed that for any random forest with n nodes, there exists an equivalent depth-3 neural network with $O(n)$ nodes for the activation function

family F_{sigpl} that includes almost all practical activation functions. This result shows that the function family corresponding to tree models is a subset of the function family corresponding to depth-3 neural networks. Furthermore, since there exist functions that can be expressed efficiently by depth-3 neural networks but need a lot of nodes in decision trees, the inclusionship is proper for decision trees.

We also showed that for any depth-D binary decision diagram with $O(n/D)$ nodes at each depth, there exists an equivalent depth-$(D+1)$ neural network with $O(n)$ nodes and an equivalent depth-$(2\lceil \log_2 D \rceil + 1)$ neural network with $O(n^3/D^2)$ nodes. The latter result suggests that neural networks need much smaller depth to express any functions than binary decision diagrams need under the condition that a polynomial number of nodes are used.

Finally, we obtained an $\Omega((\frac{2^n}{\sqrt{n}})^{\frac{2}{T+1}})$ lower bound of the number of nodes to express an n-trees random forest with $O(n)$ nodes by a T-trees random forest with $T < n$. This result implies that there are some cases in which the efficiency on the number of nodes is caused by the number of trees in random forests.

In section 5, we have not yet shown whether analogous results as in section 4 hold. Thus, it is left as an open problem to decide whether binary decision diagrams with an arbitrary depth can be expressed by fixed-depth neural networks using a polynomial number of nodes. It is also left as an open problem to find a function family that can be efficiently represented by neural networks but needs an exponential number of nodes in binary decision diagrams. In section 6, we only proved a lower bound of the number of nodes in random forests. Therefore, showing nontrivial upper bounds of the number of nodes in random forests is also an open problem.

Acknowledgments

T.A. was partially supported by grant-in-aid 18H04113 from JSPS, Japan.

References

Bengio, Y., Delalleau, O., & Simard, C. (2010). Decision trees do not generalize to new variations. *Computational Intelligence*, *26*, 449–467. 10.1111/j.1467-8640.2010.00366.x

Bengio, Y., & Lecun, Y. (2007). Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste, & J. Weston (Eds.), *Large-scale kernel machines* (pp. 1–41). Cambridge, MA: MIT Press.

Biau, G., Scornet, E., & Welbl, J. (2019). Neural random forests. *Sankhya A*, *81*, 347–386. 10.1007/s13171-018-0133-y

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32. 10.1023/A:1010933404324

Breiman, L., Friedman, J., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Boca Raton, FL: Chapman & Hall/CRC.

Bryant, R. E. (1986). Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, *C-35*, 677–691. 10.1109/TC.1986.1676819

Chatziafratis, V., Nagarajan, S. G., Panageas, I., & Wang, X. (2020). *Depth-width tradeoffs for ReLU networks via Sharkovsky's theorem*. arXiv:1912.04378.

Cucker, F., & Grigoriev, D. (1999). Complexity lower bounds for approximation algebraic computation trees. *Journal of Complexity*, *15*, 499–512. 10.1006/jcom.1999.0519

de Mello, R. F., Manapragada, C., & Bifet, A. (2019). Measuring the shattering coefficient of decision tree models. *Expert Systems with Applications*, *137*, 443–452. 10.1016/j.eswa.2019.07.012

Delalleau, O., & Bengio, Y. (2011). Shallow vs. deep sum-product networks. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *24* (pp. 666–674). Red Hook, NY: Curran.

Grigoriev, D., Karpinski, M., & Yao, A. (1998). An exponential lower bound on the size of algebraic decision trees for Max. *Computational Complexity*, *7*, 193–203. 10.1007/s000370050010

Hanin, B., & Rolnick, D. (2019). Complexity of linear regions in deep networks. In *Proceedings of the 36th International Conference on Machine Learning* (pp. 2596–2604).

Ho, T. K. (1995). Random decision forests.In *Proceedings of Third International Conference on Document Analysis and Recognition* (pp. 278–282). Washington, DC: IEEE Computer Society.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*, 359–366. 10.1016/0893-6080(89)90020-8

Mansour, Y. (1997). Pessimistic decision tree pruning based on tree size. In *Proceedings of the 14th International Conference on Machine Learning* (pp. 195–201).

Meinel, C., & Theobald, T. (1998). *Algorithms and data structures in VLSI-design: OBDD—Foundations and applications*. Berlin: Springer-Verlag.

Montufar, G. F., Pascanu, R., Cho, K., & Bengio, Y. (2014). On the number of linear regions of deep neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *27* (pp. 2924–2932). Red Hook, NY: Curran.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning* (pp. 807–814).

Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (2012). How many trees in a random forest? In *Proceedings of International Workshop on Machine Learning and Data Mining in Pattern Recognition* (pp. 154–168). Berlin: Springer.

Perez, E., & Rendell, L. A. (1996). Learning despite concept variation by finding structure in attribute-based data. In *Proceedings of the 13th International Conference on Machine Learning* (pp. 391–399).

Prasad, P. W. C., Assi, A., & Beg, A. (2007). Binary decision diagrams and neural networks. *Journal of Supercomputing*, *39*, 301–320. 10.1007/s11227-006-0010-7

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., & Dickstein, J. S. (2017). On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning* (pp. 2847–2854).

Sethi, I. K. (1990). Entropy nets: From decision trees to neural networks. In *Proceedings of the IEEE*, *78*, 1605–1613. 10.1109/5.58346

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. New York: Cambridge University Press.

Szymanski, L., & McCane, B. (2014). Deep networks are effective encoders of periodicity. *IEEE Transactions on Neural Networks and Learning Systems*, *25*, 1816–1827. 10.1109/TNNLS.2013.2296046, PubMed: 25291735

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

Trottier, L., Gigu, P., & Chaib-draa, B. (2017). Parametric exponential linear unit for deep convolutional neural networks. In *Proceedings of the 16th IEEE International Conference on Machine Learning and Applications* (pp. 207–214). Piscataway, NJ: IEEE.

Vilalta, R., Blix, G., & Rendell, L. (1997). Global data analysis and the fragmentation problem in decision tree induction. In *Proceedings of the 9th European Conference on Machine Learning* (pp. 312–326). Berlin: Springer.

Xu, F., He, F., Xie, E., & Li, L. (2018). *Fast OBDD reordering using neural message passing on hypergraph*. arXiv:1811.02178.

Yildiz, O. T. (2015). VC-dimension of univariate decision trees. *IEEE Transactions on Neural Networks and Learning Systems*, *26*, 378–387. 10.1109/TNNLS.2014.2385837, PubMed: 25594983

Zhou, S., & Mentch, L. (2021). *Trees, forests, chickens, and eggs: When and why to prune trees in a random forest*. arXiv:2103.16700.

Received June 10, 2021; accepted December 15, 2021.