DEPTH SEPARATIONS FOR MULTI-CLASS TREE CASCADES VIA MODULAR ADDITION

Anonymous authors
Paper under double-blind review

ABSTRACT

A central advantage of deep learning is its ability to transform representations progressively across layers. Motivated by this principle, deep forest brings layer-wise representation learning to tree ensembles by passing predictions from one layer to the next. This raises a basic question: when does such sequential representation make tree-based models fundamentally more efficient than simply growing a tree or adding more trees in parallel? We study this question through modular addition, a simple task that is difficult to resolve by spatial partitioning but remarkably easy to summarize sequentially. Large tree regions inevitably mix different residues, forcing single trees to use increasingly fine partitions. Adding trees in parallel can create more partitions, but does not provide the same reusable summary. By contrast, a cascade only needs to propagate the running residue, compressing exponentially many input prefixes into only p states. We prove that single trees require exponentially many leaves, and bounded-error forests remain stretched-exponentially large for even moduli, whereas a cascade computes the same target exactly with only linear complexity. Extensive experiments validate these theoretical findings, demonstrating that trained cascades recover the predicted running-residue representation and achieve near-perfect accuracy with substantially fewer leaves than single trees and random forests.

1 INTRODUCTION

Tree-based learning is widely used in many domains, such as biological data analysis (Basu et al., 2018; Guo et al., 2018), industrial fault diagnosis (Liu et al., 2019), and financial fraud detection (Zhang et al., 2019; Afriyie et al., 2023). The accuracy–efficiency trade-off of tree-based prediction remains an active research topic in tabular learning (Holzmüller et al., 2024; Hollmann et al., 2025). Recent methods improve tree-based prediction through supervised input transformations (Rauniyar, 2026) and semi-supervised cascade learning (Lyu et al., 2026). Inspired by the progressively transforming representations across layers in deep learning (LeCun et al., 2015), deep forest feeds predictions from one layer into next (Zhou & Feng, 2019). Compared with simply growing a tree or adding more trees in parallel, this creates a different way to organize computation: later layers can reuse information formed earlier. This raises a basic question: when can such sequential reuse make tree-based models fundamentally more efficient than size or width alone?

Recent theory has confirmed that depth can improve both generalization and representational efficiency in tree-based models (Arnould et al., 2021; Lyu et al., 2022; 2024). Most closely related to our setting, Lyu et al. (2024) studied binary classification using generalized parity functions. For single trees, they established an exponential separation under bounded approximation error. For majority-vote forests, their exponential separation concerns exact representation with zero error tolerance. From the theoretical perspective, the following two questions are still unclear.

Can tree-based models still benefit from width in more general multi-class problems?
Can tree cascades still be superior to forests when allowing positive error tolerance?

In this paper, we focus on modular addition, which is a multi-class problem for studying learned representations (Liu et al., 2022; Nanda et al., 2023; Zhong et al., 2023; Mohamadi et al., 2024; Li et al., 2025; McCracken et al., 2025; Moisescu-Pareja et al., 2026). Consider predicting the remainder of the sum of m discrete inputs modulo p. For a single tree, any large region of the input space inevitably mixes many different residues, so improving accuracy requires progressively

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

finer partitioning. Sequentially, however, the same task is simple for tree cascades: after processing the first d inputs, the remaining computation only needs their running residue modulo p. Thus, exponentially many possible prefixes can be summarized by only p states and reused by the next layer. Therefore, a tree must keep refining where an input lies, while a cascade only needs to retain what matters for the remaining computation.

We formalize this intuition through representation bounds for modular addition. For any fixed modulus, achieving any fixed nontrivial accuracy with a single tree requires exponentially many leaves, whereas a tree cascade computes the same target exactly with only linear complexity. The gap is therefore robust to approximation rather than an artifact of requiring exact representation. We further show that adding trees in parallel does not generally recover the same efficiency: for even moduli, any bounded-error forest still requires stretched-exponential total size. Our analysis turns the spatial difficulty above into quantitative lower bounds, while the cascade upper bound follows directly from reusing the running residue across layers. Finite-scale experiments further visualize these gaps and show that trained cascades can recover the same reusable residue representation.

Our contributions are summarized as follows:

- We use modular addition to reveal a simple mechanism behind depth advantage: a target can be difficult to resolve through spatial partitioning yet easy to summarize and reuse sequentially. This suggests a broader recipe for identifying problems that may benefit from depth.
- As summarized in Table 1, we prove that a tree cascade can compute the modular addition with leaf complexity linear in the input dimension m. In contrast, a single tree requires exponentially many leaves, while a forest also requires stretched-exponential total size for even moduli.
- Exact finite-domain experiments visualize the predicted complexity gaps, while constructive learning experiments show that trained cascades can recover the running-residue representation and achieve near-perfect accuracy with far fewer leaves than single trees and random forests.

Table 1: The leaf complexity of different tree-based models when approximating modular addition with a sufficiently small error tolerance, where m is the input dimension, and p denotes the modulus.

Model	Tree Cascades	Single Trees	Forests
Leaf Complexity	$O(mp^2)$ (Theorem 5)	$\Omega_p(p^m)$ (Theorem 6)	$\Omega\big(2^{\Theta(\sqrt{m})}\big)$ (Theorem 7)

2 Related Work

Tree ensembles and deep forests. Random forests aggregate randomized trees to improve prediction (Breiman, 2001), and their consistency has been studied theoretically (Biau, 2012; Scornet et al., 2015). Deep forest introduces a cascade that passes learned features across layers (Zhou & Feng, 2019). Analyses of deep forests cover margin distributions (Lyu et al., 2019), excess risk (Arnould et al., 2021), and consistency gains from prediction concatenation (Lyu et al., 2022). Our focus is the leaf complexity required to represent a target through cascades or parallel aggregation.

Depth separation. Depth separation results exhibit functions that compact deep networks can approximate but shallow networks require exponentially many units to match (Eldan & Shamir, 2016; Telgarsky, 2016; Daniely, 2017). For tree-based models, Lyu et al. (2024) studied binary classification with generalized parity functions. Their single-tree separation allows bounded approximation error, whereas their exponential lower bound for majority-vote forests only holds for exact representation. We extend this comparison to multi-class modular addition and prove a stretched-exponential leaf lower bound for plurality forests with even moduli at any fixed positive error tolerance $\epsilon \in (0, 1/2)$. The forest analysis draws on limitations of low-degree voting polynomials in approximating parity (Aspnes et al., 1991). The cascade construction reuses the running residue, compressing exponentially many input prefixes into only p states.

Modular arithmetic and learned representations. Studies of modular arithmetic connect generalization to the emergence of structured representations (Liu et al., 2022; Mohamadi et al., 2024). Fourier-based analyses characterize transformer circuits (Nanda et al., 2023) and representations for multi-input modular addition (Li et al., 2025). Zhong et al. (2023) distinguish the Clock and Pizza

mechanisms. Later analyses relate these mechanisms through approximate Chinese Remainder Theorem computation (McCracken et al., 2025) and equivalent representation geometry (Moisescu-Pareja et al., 2026). We study learned tree-cascade representations by measuring their agreement with the running residue across layers. These state diagnostics relate learned representations to the compact computation used in our upper bound.

3 PRELIMINARIES

For any integer $p \geqslant 2$, denote by $[p]_0$ the set $\{0, 1, \ldots, p-1\}$. Let $\mathcal{X}_{m,p} = [p]_0^m$ and $\mathcal{Y}_p = [p]_0$ indicate the input and output spaces, respectively. The target concept maps the input vector to the sum of its elements modulo p, i.e.,

$$c_{m,p}(\boldsymbol{x}) = \|\boldsymbol{x}\|_1 \bmod p \quad \text{for any} \quad \boldsymbol{x} \in \mathcal{X}_{m,p} \,. \tag{1}$$

For simplicity, we mainly focus on the uniform input distribution $\mathcal{U}_{m,p}$ on $\mathcal{X}_{m,p}$ in the theoretical analysis. Then for a possibly randomized predictor $f : \mathcal{X}_{m,p} \to \mathcal{Y}_p$, the expected risk is defined as

$$R(f) = \Pr_{\boldsymbol{x} \sim \mathcal{U}_{m,p}, \boldsymbol{\theta}} [f(\boldsymbol{x}; \boldsymbol{\theta}) \neq c_{m,p}(\boldsymbol{x})] \,, \tag{2}$$

where $\boldsymbol{\theta}$ denotes the internal randomness of f and is omitted for a deterministic predictor.

In this paper, we mainly focus on three models, i.e., trees, forests, and tree cascades (Lyu et al., 2024), which are reviewed and formally defined as follows.

Definition 1 (Trees). *In a tree h_{T}, each internal node queries $x_j \leqslant t$ for some dimension $j \in [m]$ and threshold $t \in [p-1]_0$. Every leaf outputs a label in $\mathcal{Y}_p$. Let leaf complexity $L(h_{\mathrm{T}})$ be the number of leaves, and $\mathcal{H}_{\mathrm{T}}$ denotes the class of trees with finite leaves.*

Definition 2 (Forests). *Let $h_1, h_2, \ldots, h_K$ be K trees with finite depth. A forest $h_{\mathrm{F}} = (h_1, \ldots, h_K)$ predicts a label having the largest number of tree votes, and ties are broken uniformly among the tied labels. More explicitly, the vote count for $r \in \mathcal{Y}_p$ is $V_r(\boldsymbol{x}) = \sum_{j=1}^{K} \mathbb{I}\{h_j(\boldsymbol{x}) = r\}$, and the prediction is a random element of $\operatorname{argmax}_{r \in \mathcal{Y}_p} V_r(\boldsymbol{x})$. The leaf complexity is $L(h_{\mathrm{F}}) = \sum_{j=1}^{K} L(h_j)$, and $\mathcal{H}_{\mathrm{F}}$ denotes the class of all such forests consisting of a finite number of trees.*

Note that the random tie rule is part of a forest predictor. It is equivalent to sampling a uniform permutation of the p labels and selecting the earliest label among those tied for the largest vote count. Consequently, exact representation implies that the correct label is the unique plurality winner at every input since a tied incorrect label would be selected with positive probability.

Definition 3 (Tree cascades). *A tree cascade of depth D consists of a first tree $h_1 : \mathcal{X}_{m,p} \to \mathcal{Y}_p$ and trees $h_d : \mathcal{X}_{m,p} \times \mathcal{Y}_p \to \mathcal{Y}_p$ for $2 \leqslant d \leqslant D$. It outputs $h_{\mathrm{C}}(\boldsymbol{x}) = h_D \oplus h_{D-1} \oplus \cdots \oplus h_1(\boldsymbol{x})$, where the cascade composition operation is defined by $g \oplus f(\boldsymbol{x}) = g(\boldsymbol{x}, f(\boldsymbol{x}))$. The leaf complexity is $L(H) = \sum_{d=1}^{D} L(h_d)$, and $\mathcal{H}_{\mathrm{C}}$ denotes the class of all such tree cascades with finite depth.*

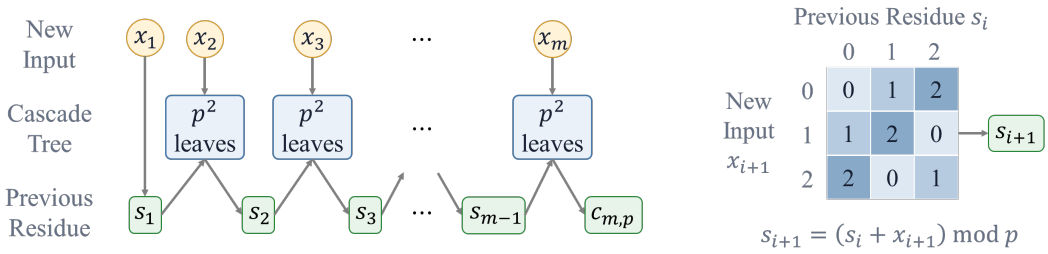
We proceed to introduce the concept of approximation complexity (Lyu et al., 2024), which is used to compare the representation efficiency of different tree-based models. The following definition focuses on the modular addition and uniform input distribution task, which is fixed in theoretical analysis.

Definition 4 (Approximation complexity). *Let $R(h)$ be the expected risk in Eq. (2). For a tree-based hypothesis class $\mathcal{H}$ and a risk tolerance $\varepsilon \in [0, 1]$, the approximation complexity is defined by*

$$\mathcal{C}(\mathcal{H}, \varepsilon) = \min\{L(h) \mid h \in \mathcal{H} \wedge R(h) \leqslant \varepsilon\},$$

where the approximation complexity takes the value $+\infty$ if the feasible set is empty.

In Definition 4, the approximation complexity $\mathcal{C}(\mathcal{H}, \varepsilon)$ quantifies the number of leaves required by models from the hypothesis class $\mathcal{H}$ to achieve at least $1-\epsilon$ accuracy in the uniform modular addition task. The approximation complexity measures the expressive capability of hypothesis spaces, with smaller complexity implying stronger expressive power and potentially better performance.



(a) Sequential propagation of partial residues with $m-1$ trees.

(b) Residue update with p^2 leaves.

Figure 1: The proof idea of Theorem 5, where $s_i = (\sum_{j=1}^{i} x_i) \bmod p$ denotes the residue of the first i inputs. (a) The m-dimensional modular addition can be split into $m-1$ sequential tasks and finished by $m-1$ cascade trees. (b) Each cascade tree can finish its task with at most p^2 leaves.

4 Main Results

In this section, we provide theoretical insights that tree cascades can be more efficient than trees and forests by depth separation results. Subsection 4.1 proves a polynomial upper bound for the approximation complexity of tree cascades. Then Subsections 4.2 and 4.3 show exponential lower bounds for the approximation complexity of trees and forests, respectively.

4.1 Approximation Complexity Upper Bound for Tree Cascades

From Eq. (1), it is observed that the target of modular addition has a compact sequential representation. After processing the first i coordinates, it is sufficient to retain only their partial residue modulo p, rather than the entire input prefix. This sequential structure can be exploited by a tree cascade, which can pass the current residue from one layer to the next and incorporate one additional coordinate at each layer. We formalize the above intuition in the following theorem.

Theorem 5. *For every $p \geqslant 2$ and $m \geqslant 2$, the approximation complexity of tree cascades satisfies*

$$\mathcal{C}(\mathcal{H}_{\mathrm{C}}, 0) \leqslant (m-1)p^2.$$

Theorem 5 shows that in general conditions, a tree cascade can represent modular addition with $O(mp^2)$ leaves, which grows only linearly with respect to the input dimension m and quadratically with respect to the modulus p. Note that the upper bound $O(mp^2)$ is derived for exact representation of the target using tree cascades. Thus, for any error tolerance ϵ, the approximation complexity $\mathcal{C}(\mathcal{H}_{\mathrm{C}}, \epsilon) = O(mp^2)$ trivially holds. Meanwhile, as we will see in the proof, depth enables each layer to reuse the partial residue produced by its predecessor. Thus, tree cascades can compress all input prefixes with the same residue into one of only p intermediate states, which is precisely the information needed to process the remaining coordinates.

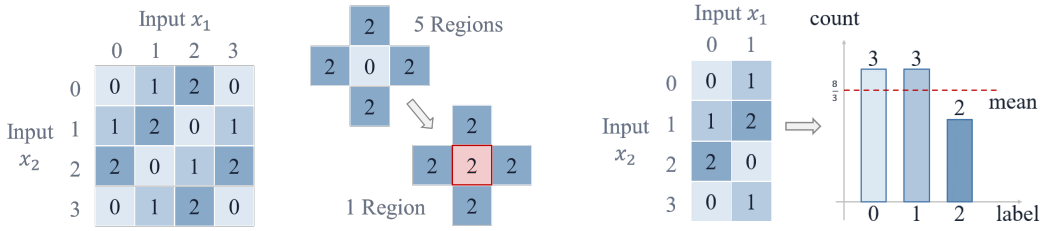
The proof of Theorem 5 mainly consists of two steps. Firstly, the m-dimensional modular addition can be decomposed into $m-1$ tasks, as shown in Figure 1a. The i-th task aims to use the new input and previous residue to obtain the current residue, i.e.,

$$s_{i+1} = (s_i + x_{i+1}) \bmod p \quad \text{with} \quad s_i = \left(\sum\nolimits_{j=1}^{i} x_i\right) \bmod p\,.$$

Then in the last task, one knows from $s_m = c_{m,p}(\boldsymbol{x})$ that the output is the target function. Secondly, as shown in Figure 1b, the new input x_{i+1} and previous residue s_i independently take p values, leading to p^2 values of the pair (s_i, x_{i+1}). Thus, the update $s_{i+1} = (s_i + x_{i+1}) \bmod p$ can be represented exactly by a tree with at most p^2 leaves. Finally, using $m-1$ such trees yields an exact representation with at most $(m-1)p^2$ leaves. Detailed proofs are available in Appendix B.

4.2 Approximation Complexity Lower Bound for Trees

In a single tree, the leaves partition the input space into axis-aligned rectangles, and each leaf must assign one label to all points in its rectangle. However, the modular addition partition the input space



(a) Reduction of target regions by misclassified points.

(b) Nearly balanced modular residues.

Figure 2: The proof idea of Theorem 6. (a) There are p^m target regions with distinct labels for adjacent ones, and each misclassified point can reduce the number of target regions by at most $2m$. (b) The modular residues are nearly balanced inside rectangles, which limits the number of correctly labeled points in one leaf.

$\mathcal{X}_{m,p}$ into p^m regions, and adjacent regions have different labels. Consequently, a single tree must use p^m leaves to perfectly express the modular addition. The following theorem extends the above intuition by quantifying the leaf complexity when allowing a positive error tolerance.

Theorem 6. *The following holds for the approximation complexity of trees.*

1. *For any $p \geqslant 2$ and $0 \leqslant \epsilon < 1/(2m)$, we have*
$$\mathcal{C}(\mathcal{H}_{\mathrm{T}}, \epsilon) \geqslant p^m(1 - 2m\epsilon) .$$
2. *For any $p \geqslant 3$ and $0 \leqslant \epsilon < (p-1)/p$, we have*
$$\mathcal{C}(\mathcal{H}_{\mathrm{T}}, \epsilon) \geqslant p^m\Big(1 - \frac{p\epsilon}{p-1}\Big)^{p^2 \ln p} .$$

Theorem 6 demonstrates that for sufficiently small error tolerances, the leaf complexity required by a single tree to approximate modular addition has a lower bound $\Omega_p(p^m)$, which increases exponentially with respect to the input dimension m. For exact representation, both conclusions recover the p^m leaf complexity by setting $\epsilon = 0$. When comparing the dependence on the input dimension m, the exponential requirement by single trees in Theorem 6 stands in contrast to the linear upper bound for tree cascades in Theorem 5. Intuitively, this difference occurs since a single tree cannot propagate a compact intermediate representation across layers, which can be finished by a tree cascade. Therefore, an exponential separation exists between the tree cascades and single trees, suggesting the importance of depth in the efficiency of representation.

In Theorem 6, the two cases characterize complementary parameter regimes. For the first case, the upper bound of error tolerance vanishes as the input dimension m increases, while the lower bound of approximation complexity is exponential with respect to m as long as the error tolerance is sufficiently small. Thus, the first conclusion applies in the regime of low error tolerance and arbitrary moduli. For the second case, the constraint on the error tolerance is negligible since the upper bound on ϵ becomes arbitrarily close to 1 as p increases. Meanwhile, one knows from $(1 - 1/n)^n < e^{-1}$ that the lower bound of approximation complexity is at most $p^{m-\epsilon p^2}$, which is exponential in m only if $p = O(\sqrt{m})$. Thus, the second conclusion implies exponential dependence in the regime of almost arbitrary error tolerance and small moduli.

We then provide the proof idea of the first conclusion in Theorem 6. As shown in Figure 2a, the modular addition target partitions the input space $\mathcal{X}_{m,p}$ into m^p target regions with distinct labels for adjacent ones. Meanwhile, each misclassified point of a tree is adjacent to at most $2m$ points and can reduce the number of target regions by at most $2m$. Thus, a single tree with error tolerance ϵ must partition the input space into at least $m^p(1 - 2m\epsilon)$ target regions, which is also the lower bound of the leaf complexity.

We proceed to introduce the proof idea of the second conclusion in Theorem 6. As shown in Figure 2b, the modular residues are nearly balanced inside rectangles. More formally, the number of points in any rectangle R with any label r satisfies

$$N_r(R) \leqslant \frac{|R|}{p} + \frac{p-1}{p}|R|^{1-\alpha_p} \quad \text{with} \quad \alpha_p \in (0, 1) . \tag{3}$$

Note that the right-hand side of Eq. (3) is strictly smaller than $|R|$ as long as $|R| > 1$. Consequently, a leaf covering many inputs cannot achieve accuracy substantially higher than $1/p$, and a tree must refine the input space into many small rectangles to attain a high classification accuracy. Specifically, by applying Eq. (3) to all L leaves of a tree, one knows from the concavity of $x^{1-\alpha_p}$ and Jensen's inequality that the expected risk satisfies $R(h_{\mathrm{T}}) \geqslant 1 - p^{-1}\left[1 + (p-1)\left(p^{-m}L\right)^{\alpha_p}\right]$. Requiring $R(h_{\mathrm{T}}) \leqslant \epsilon$ and rearranging the inequality gives the desired lower bound on the leaf complexity. Detailed proofs are available in Appendix C.

4.3 Approximation Complexity Lower Bound for Forests

A forest can combine many axis-aligned partitions through plurality voting, which may make it more expressive than a single tree. Nevertheless, all trees in a forest process the original inputs in parallel, and their predictions are combined only at the output. Thus, forests lack the capability of sequential representation, which is essential for tree cascades to deal with the modular addition task efficiently. The following theorem formalizes this intuition by providing a lower bound for the approximation complexity of forests.

Theorem 7. *Let $p \in 2\mathbb{N}^+$ be a positive even integer. For any $0 \leqslant \epsilon < 1/2$, the approximation complexity of forests satisfies*

$$\mathcal{C}(\mathcal{H}_{\mathrm{F}}, \epsilon) \geqslant \frac{1-2\epsilon}{8} 2^{\frac{1-2\epsilon}{12}\sqrt{m}} .$$

Theorem 7 shows that for a small error tolerance, a forest approximating modular addition necessitates a leaf complexity of at least $\Omega\big(2^{\Theta(\sqrt{m})}\big)$, which grows stretched-exponentially as the input dimension m increases. Comparing Theorems 5 and 7, we observe that the linear leaf complexity of tree cascades is much more efficient than the stretched-exponential requirement of forests, indicating that plurality voting cannot replace the representational benefit of sequential composition. Therefore, a stretched-exponential separation occurs between the tree cascades and forests, highlighting the necessity of depth in the efficiency of representation.

The proof idea of Theorem 7 mainly consists of three steps. First, we define a Boolean problem for any original input $\boldsymbol{x} \in \mathcal{X}_{m,p}$ as

$$\phi_{\boldsymbol{x}}(\boldsymbol{b})_i = x_i + a b_i \pmod p \quad \text{with} \quad a = p/2 \in \mathbb{N}^+ ,$$

where $\boldsymbol{b} \in \{-1, 1\}^m$ denotes an m-dimensional Boolean vector. Suppose that h_{F} is a forest with an expected risk of at most ϵ, and that both the original input $\boldsymbol{x} \in \mathcal{X}_{m,p}$ and the Boolean input $\boldsymbol{b} \in \{-1, 1\}^m$ follow uniform distributions in their domains. Then the expected risk of h_{F} in the Boolean problem is also bounded by ϵ. Thus, there exists a Boolean problem $\phi_{\tilde{\boldsymbol{x}}}$ with $\tilde{\boldsymbol{x}} \in \mathcal{X}_{m,p}$ denoting a particular input such that the expected risk of h_{F} on $\phi_{\tilde{\boldsymbol{x}}}$ is at most ϵ. Second, we construct a new forest $h_{\mathrm{F},d}$ by truncating the forest h_{F} to depth d. Note that as the depth d increases, it becomes exponentially less likely for uniformly sampled inputs to reach a leaf node. Then applying the union bound to all leaves, the truncation changes the predictions with probability at most $L2^{-d}$. Thus, it suffices to choose $d = \Theta(\log L)$ to control the error of the truncated forest $h_{\mathrm{F},d}$ on the Boolean problem $\phi_{\tilde{\boldsymbol{x}}}$. Third, the depth-$d$ forest $h_{\mathrm{F},d}$ coincides with the sign of a polynomial of degree at most d, while the Boolean problem $\phi_{\tilde{\boldsymbol{x}}}$ can be represented by the sign of a polynomial of degree at least m. Then by standard combinatorial inequalities, the depth d should satisfy $d = \Omega(\sqrt{m})$ to guarantee a low approximation error of $h_{\mathrm{F},d}$ in the Boolean problem $\phi_{\tilde{\boldsymbol{x}}}$ (Aspnes et al., 1991). Finally, combining the requirements on d in the second and third steps, we obtain the stretched-exponential dependence of L on m. Detailed proofs are available in Appendix D.

Note that Theorem 7 holds for any even modulus p. As mentioned in the above proof idea, this condition makes it possible to select two residues separated by $p/2$. Thereby, the modular addition problem can induce a Boolean problem with the parity target function, which enables the use of low-degree polynomial vote functions to analyze the leaf complexity. For an odd modulus p, the additive group $\mathbb{Z}_p$ no longer has a nonzero element of order two, and the proof by constructing a Boolean problem fails. Thus, the current analysis cannot be directly extended to odd moduli, and we leave the leaf complexity of forests in modular addition with odd moduli as future explorations.

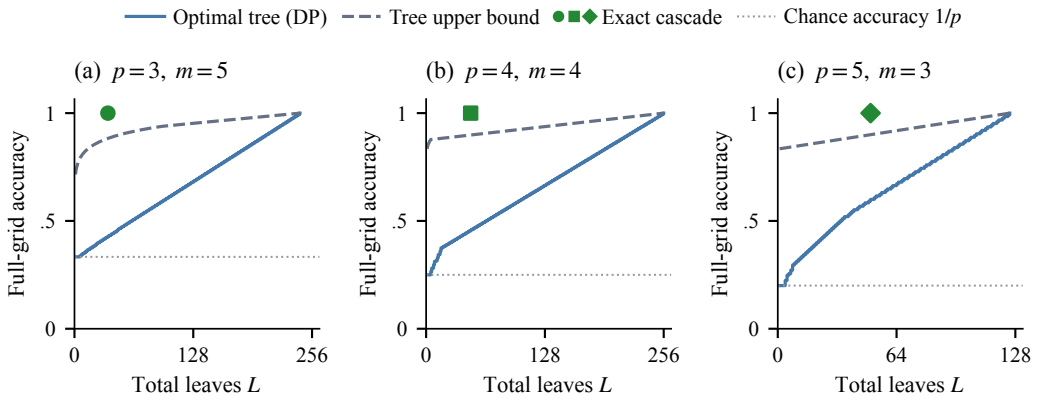


Figure 3: Finite-domain representation efficiency. (a–c) Solid step curves show optimal single-tree accuracy computed by exact dynamic programming; dashed curves show the smaller of the two accuracy upper bounds implied by Theorem 6. Filled markers denote the exact cascade from Theorem 5, and dotted lines indicate $1/p$. Circles, squares, and diamonds denote $p = 3$, $p = 4$, and $p = 5$, respectively. All results concern the uniform distribution on the full input domain and are deterministic; cascade construction sizes are upper bounds on the optimal cascade complexity.

5 Experiments

We examine the representation efficiency of tree cascades through two types of experiments: exact finite-domain computations under the uniform distribution and learning experiments under a constructed distribution. The finite-domain computations are deterministic, and the learning experiments are repeated with five random seeds. We report prediction accuracy, total leaf count, and agreement between intermediate predictions and the corresponding partial residues. For the learning experiments, we evaluate accuracy under the original nonuniform sampling and with equal weight on each input cell. Accuracy and leaf count characterize the accuracy–complexity trade-off, while intermediate-state agreement examines whether trained cascades recover the running-residue representation. Detailed experimental settings and evaluation protocols are presented in the following subsections. Formal metric definitions and aggregation rules are given in Appendix E.1.

5.1 Representation Efficiency on Finite Domains

We first examine the representation efficiency of tree cascades for modular addition under the uniform distribution on $\mathcal{X}_{m,p}$. We present exact computations for three instances, $(p, m) = (3, 5)$, $(4, 4)$, and $(5, 3)$, whose input domains contain 243, 256, and 125 points, respectively.

We compare the optimal single-tree accuracy with the accuracy upper bound implied by Theorem 6 and the accuracy of the cascade constructed in Theorem 5. For all three instances, exact dynamic programming over rectangular subdomains gives the optimal accuracy with at most L leaves for every $L = 1, \ldots, p^m$. The cascade contains $m - 1$ trees with $(m - 1)p^2$ leaves and is evaluated on all p^m inputs. The computation procedures are detailed in Appendix E.2.

Figure 3 presents the exact accuracy–complexity trade-offs for the three instances. The optimal single trees require at least 207, 219, and 107 leaves to reach 90% accuracy for $(p, m) = (3, 5)$, $(4, 4)$, and $(5, 3)$, respectively, while the constructed cascades achieve 100% accuracy using 36, 48, and 50 leaves. At these cascade budgets, the highest single-tree accuracies are 42.8%, 45.7%, and 60.0%, respectively. The theoretical accuracy upper bounds lie above the exact DP curves and are loose over much of the leaf-budget range. These finite-domain results illustrate that reusing partial residues can achieve exact predictions with substantially fewer leaves than an optimal single tree. Additional finite instances and error tolerances are presented in Appendix F.1.

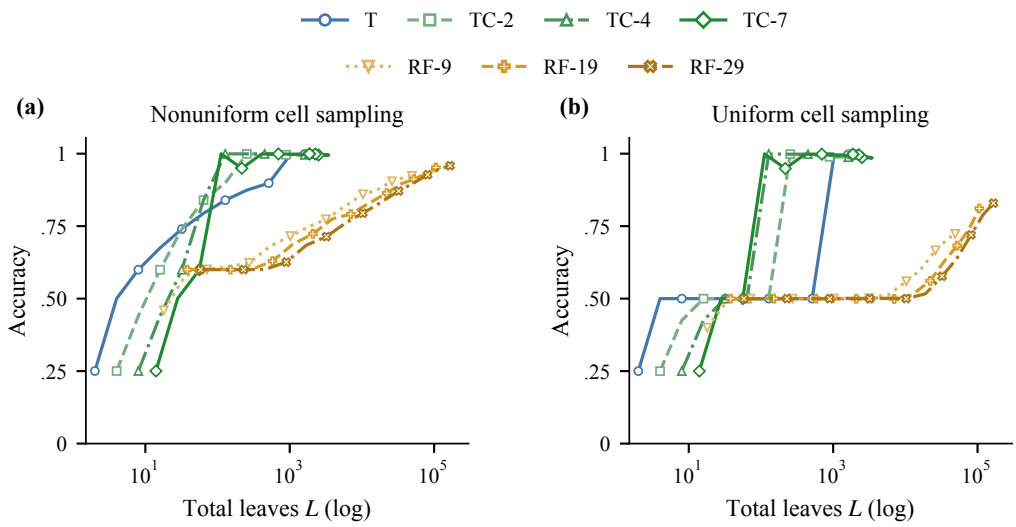


Figure 4: Accuracy–complexity trade-offs under (a) the original nonuniform sampling and (b) equal weighting of all 1,024 input cells. Curves show means over five seeds, connecting component-tree depths 1–15 in order.

5.2 Learning with Sequential Composition

We next evaluate the accuracy–complexity trade-off of learned tree cascades on an eight-class modular-addition task with $(p, m) = (8, 10)$. The first three grid coordinates are independently uniform on $\{0, 1\}$, $\{0, 2\}$, and $\{0, 4\}$, respectively. Independently, we draw a prefix length $J \in \{0, \ldots, 7\}$ with probabilities $(.20, .15, .13, .11, .09, .07, .05, .20)$ in order. The next J coordinates are independently uniform on $\{0, 4\}$, and the rest are zero. Short prefixes provide supervision for learning partial sums. We generate continuous inputs by adding independent uniform perturbations from $[-0.5, 0.5)$ to the sampled grid points (Lyu et al., 2024) and assign labels by summing the grid coordinates modulo 8. Each grid point and its perturbations form an input cell, giving 1,024 cells with unequal sampling probabilities. Following the sample-size and train–test split settings of Lyu et al. (2024), we generate 10^6 examples per seed and use 70% for training and 30% for testing, repeating the experiment with five random seeds. The full sampling procedure is given in Appendix E.3.

We compare a single decision tree (T) (Breiman et al., 1984), tree cascades with $D \in \{2, 4, 7\}$ layers (TC-D) (Lyu et al., 2024), and random forests with $K \in \{9, 19, 29\}$ trees (RF-K) (Breiman, 2001). For all models, we vary the maximum depth of each component tree from 1 to 15 and use the Gini splitting criterion in scikit-learn (Pedregosa et al., 2011). The forest sizes and maximum component-tree depth range follow Lyu et al. (2024). Each cascade layer is trained on the final class labels, and subsequent layers receive the original features together with the preceding layer's predicted class. Random forests use bootstrap samples and average class probabilities across trees. We compare accuracy against total leaf count using the same trained models under two evaluation distributions. Figure 4(a) uses the test split from the nonuniform distribution above; Figure 4(b) uses 500 uniform samples per cell, giving all 1,024 cells equal weight. This comparison tests whether high accuracy persists when rare input combinations receive equal weight. Implementation details and hyperparameters are listed in Appendix E.4.

Figure 4(a) shows the accuracy–complexity trade-offs of the learned models, and Table 2 summarizes the smallest observed leaf counts reaching 95% and 99% test accuracy. At the 99% threshold, the mean leaf count decreases from 1,024 for T to 254.2, 128, and 112 for TC-2, TC-4, and TC-7, respectively, with all five seeds reaching the threshold. TC-7 achieves a mean test accuracy of 99.968% with 112 leaves, giving a 9.14-fold reduction in leaf count relative to T at the same threshold. These results show that deeper cascades attain high accuracy with progressively smaller tree representations. Complete depth scans and per-seed threshold results are reported in Appendix F.2.

Increasing forest width also improves accuracy, but at much greater leaf complexity. RF-9 does not reach 95% accuracy, whereas RF-19 and RF-29 reach it in four and all five seeds, with mean leaf counts of 97,753 and 159,080.2, respectively. At the largest tested component depth, the mean accuracies are 92.32%, 95.23%, and 95.83% for RF-9, RF-19, and RF-29, and none of the tested forests reaches 99%. The cascades achieve higher accuracy using substantially fewer leaves than these parallel ensembles.

Table 2: Minimum observed leaves at each test-accuracy threshold (mean $\pm$ SD over successful seeds; $n = 5$, except †: $n = 4$; –: $n = 0$).

Model	$\geq 95\%$	$\geq 99\%$
T	$1{,}024 \pm 0$	$1{,}024 \pm 0$
TC-2	254.2 ± 1.1	254.2 ± 1.1
TC-4	128 ± 0	128 ± 0
TC-7	$\mathbf{112 \pm 0}$	$\mathbf{112 \pm 0}$
RF-9	—	—
RF-19	$97{,}753 \pm 10{,}083.5^{\dagger}$	—
RF-29	$159{,}080.2 \pm 21{,}352.4$	—

With equal cell weights in Figure 4(b), the mean accuracy of TC-7 at 112 leaves changes from 99.968% to 99.942%. For RF-29 at the largest tested component depth, it drops from 95.83% to 82.92%. Thus, the compact cascade maintains high accuracy when rare input combinations receive equal weight.

5.3 Recovery of Running-Residue States

We next examine whether the compact cascades recover the running-residue representation. We consider TC-7 with a maximum component-tree depth of 4, using the same five random seeds as in the preceding experiment. For layer $\ell \in \{1, \ldots, 7\}$, we compare its predicted class with the partial residue $s_{\ell+3}$ of the first $\ell + 3$ grid coordinates. Agreement is the fraction of matches over the 300,000 test examples for each seed. The measurement procedure and numerical results are given in Appendix E.5.

Figure 5 summarizes the mismatch rates for states $s_4, \ldots, s_{10}$ over five seeds. Mean mismatch rates range from 0.0136% to 0.0321% across states. Every layer uses 16 leaves, giving 112 leaves in total. Training on the final class labels thus yields a sequence of compact intermediate predictions that closely follows the running-residue computation in Theorem 5.

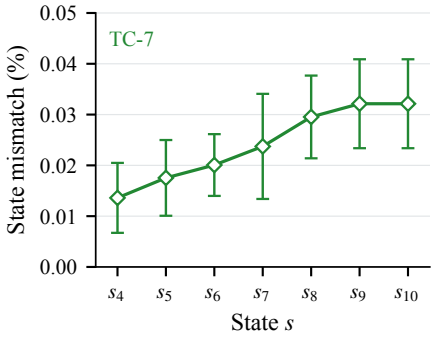


Figure 5: TC-7 state mismatch rate (mean $\pm$ one standard deviation over five seeds).

6 Conclusion and Future Work

Conclusion. We used modular addition to study how sequential reuse of intermediate states improves the representation efficiency of tree-based models. For fixed moduli, our leaf-complexity bounds separate cascades exponentially from single trees and, for even moduli, stretched-exponentially from plurality forests. Cascades compute the target exactly with linearly many leaves by propagating running residues. Exact finite-domain computations illustrate the representation gap, while trained cascades achieve near-perfect accuracy with fewer leaves and recover running-residue states. These results identify state reuse as a source of depth efficiency.

Future work includes extending the forest bounds to odd moduli and characterizing when cascades can learn running-residue states from final labels. We also plan to study how the training distribution and splitting criterion affect the recovery of these states. Extending the analysis to other multiclass tasks with compact sequential representations could further clarify when depth improves the representation efficiency of tree-based models.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript editing, technical consistency checking, and experimental code development. All AI-assisted content and code were reviewed and verified by the authors. The authors take full responsibility for all claims, results, and materials in this work.

ETHICS STATEMENT

This work is theoretical and methodological in nature and does not involve human participants, personal data, or real-world decision making. All experiments are conducted on synthetic modular-addition tasks or data generated from explicitly specified synthetic distributions. We therefore do not identify any specific ethical concerns associated with the experiments reported in this work.

REPRODUCIBILITY STATEMENT

The problem setting and model classes are formally defined in Section 3, and the main theoretical results are presented in Section 4, with complete proofs provided in Appendices B–D. Section 5 describes the finite-domain and learning experiments. Appendix E provides the evaluation metrics, exact dynamic-programming procedure, data-generation process, model configurations, training settings, random seeds, software environment, and intermediate-state diagnostics, while Appendix F reports additional finite-domain and learning results.

REFERENCES

Jonathan Kwaku Afriyie, Kassim Tawiah, Wilhemina Adoma Pels, Sandra Addai-Henne, Harriet Achiaa Dwamena, Emmanuel Odame Owiredu, Samuel Amening Ayeh, and John Eshun. A supervised machine learning algorithm for detecting and predicting fraud in credit card transactions. *Decision Analytics Journal*, 6:100163, 2023. doi: 10.1016/j.dajour.2023.100163. URL https://www.sciencedirect.com/science/article/pii/S2772662223000036.

Ludovic Arnould, Claire Boyer, and Erwan Scornet. Analyzing the tree-layer structure of Deep Forests. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 342–350. PMLR, 2021. URL https://proceedings.mlr.press/v139/arnould21a.html.

James Aspnes, Richard Beigel, Merrick Furst, and Steven Rudich. The expressive power of voting polynomials. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, STOC '91, pp. 402–409. Association for Computing Machinery, 1991. doi: 10.1145/103418.103461. URL https://dl.acm.org/doi/10.1145/103418.103461.

Sumanta Basu, Karl Kumbier, James B. Brown, and Bin Yu. Iterative random forests to discover predictive and stable high-order interactions. *Proceedings of the National Academy of Sciences*, 115(8):1943–1948, 2018. doi: 10.1073/pnas.1711236115. URL https://www.pnas.org/doi/10.1073/pnas.1711236115.

Gérard Biau. Analysis of a Random Forests Model. *Journal of Machine Learning Research*, 13(38):1063–1095, 2012. URL http://jmlr.org/papers/v13/biau12a.html.

Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324. URL https://link.springer.com/article/10.1023/A:1010933404324.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984. URL https://www.routledge.com/Classification-and-RegressionTrees/Breiman-Friedman-Stone-Olshen/p/book/9780412048418.

Amit Daniely. Depth Separation for Neural Networks. In *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pp. 690–696. PMLR, 2017. URL https://proceedings.mlr.press/v65/daniely17a.html.

Ronen Eldan and Ohad Shamir. The Power of Depth for Feedforward Neural Networks. In *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pp. 907–940. PMLR, 2016. URL https://proceedings.mlr.press/v49/eldan16.html.

Yang Guo, Shuhui Liu, Zhanhuai Li, and Xuequn Shang. BCDForest: a boosting cascade deep forest model towards the classification of cancer subtypes based on gene expression data. *BMC Bioinformatics*, 19(Suppl. 5):118, 2018. doi: 10.1186/s12859-018-2095-4. URL https://link.springer.com/article/10.1186/s12859-018-2095-4.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025. doi: 10.1038/s41586-024-08328-6. URL https://www.nature.com/articles/s41586-024-08328-6.

David Holzmüller, Léo Grinsztajn, and Ingo Steinwart. Better by default: Strong pre-tuned MLPs and boosted trees on tabular data. In *Advances in Neural Information Processing Systems*, volume 37, pp. 26577–26658. Curran Associates, Inc., 2024. doi: 10.52202/079017-0837. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/2ee1c87245956e3eaa71aaba5f5753eb-Paper-Conference.pdf.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539. URL https://www.nature.com/articles/nature14539.

Chenyang Li, Yingyu Liang, Zhenmei Shi, Zhao Song, and Tianyi Zhou. Fourier Circuits in Neural Networks and Transformers: A Case Study of Modular Arithmetic with Multiple Inputs. In *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 523–531. PMLR, 2025. URL https://proceedings.mlr.press/v258/li25b.html.

Xiaolian Liu, Yu Tian, Xiaohui Lei, Mei Liu, Xin Wen, Haocheng Huang, and Hao Wang. Deep forest based intelligent fault diagnosis of hydraulic turbine. *Journal of Mechanical Science and Technology*, 33(5):2049–2058, 2019. doi: 10.1007/s12206-019-0408-9. URL https://link.springer.com/article/10.1007/s12206-019-0408-9.

Ziming Liu, Ouail Kitouni, Niklas S Nolte, Eric Michaud, Max Tegmark, and Mike Williams. Towards Understanding Grokking: An Effective Theory of Representation Learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 34651–34663. Curran Associates, Inc., 2022. doi: 10.52202/068431-2511. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/dfc310e81992d2e4cedc09ac47eff13e-Paper-Conference.pdf.

Shen-Huan Lyu, Liang Yang, and Zhi-Hua Zhou. A Refined Margin Distribution Analysis for Forest Representation Learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/db5cea26ca37aa09e5365f3e7f5dd9eb-Paper.pdf.

Shen-Huan Lyu, Yi-Xiao He, and Zhi-Hua Zhou. Depth is More Powerful than Width with Prediction Concatenation in Deep Forest. In *Advances in Neural Information Processing Systems*, volume 35, pp. 29719–29732. Curran Associates, Inc., 2022. doi: 10.52202/068431-2155. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c017e92288b5056c578bb6b0b69d9e76-Paper-Conference.pdf.

Shen-Huan Lyu, Jin-Hui Wu, Qin-Cheng Zheng, and Baoliu Ye. The Role of Depth, Width, and Tree Size in Expressiveness of Deep Forest. In *ECAI 2024*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, pp. 2042–2049. IOS Press, 2024. doi: 10.3233/FAIA240721. URL https://journals.sagepub.com/doi/abs/10.3233/FAIA240721.

Shen-Huan Lyu, Jia-Le Xu, Yi-Xiao He, Yanyan Wang, Baoliu Ye, and Qingfu Zhang. A semi-supervised deep forest framework based on margin distribution optimization for tabular data. *Information Sciences*, 753:123615, 2026. doi: 10.1016/j.ins.2026.123615. URL https://www.sciencedirect.com/science/article/pii/S0020025526005463.

Gavin McCracken, Gabriela Moisescu-Pareja, Vincent Létourneau, Doina Precup, and Jonathan Love. Uncovering a Universal Abstract Algorithm for Modular Addition in Neural Networks. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 44291–44352. Curran Associates, Inc., 2025. doi: 10.52202/085713-1476. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/3edb234091dca2023308398dbf824850-Paper-Conference.pdf.

Mohamad Amin Mohamadi, Zhiyuan Li, Lei Wu, and Danica J. Sutherland. Why Do You Grok? A Theoretical Analysis on Grokking Modular Addition. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35934–35967. PMLR, 2024. URL https://proceedings.mlr.press/v235/mohamadi24a.html.

Gabriela Moisescu-Pareja, Gavin McCracken, Harley Wiltzer, Colin Daniels, Vincent Létourneau, Doina Precup, and Jonathan Love. On The Geometry and Topology of Representations: the Manifolds of Modular Addition. In *International Conference on Learning Representations*, volume 2026, pp. 402–422, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/011bc51517c79fe93db32623515d5049-Paper-Conference.pdf.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=9XFSbDPmdW.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12 (85):2825–2830, 2011. URL http://jmlr.org/papers/v12/pedregosa11a.html.

Sarwesh Rauniyar. Jacobian Aligned Random Forests. In *International Conference on Learning Representations*, volume 2026, pp. 103150–103173, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/a88763eb23936ced0c81603b9300c8d7-Paper-Conference.pdf.

Erwan Scornet, Gérard Biau, and Jean-Philippe Vert. Consistency of random forests. *The Annals of Statistics*, 43(4):1716–1741, 2015. doi: 10.1214/15-AOS1321. URL https://projecteuclid.org/journals/annals-of-statistics/volume-43/issue-4/Consistency-of-random-forests/10.1214/15-AOS1321.full.

Matus Telgarsky. benefits of depth in neural networks. In *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pp. 1517–1539. PMLR, 2016. URL https://proceedings.mlr.press/v49/telgarsky16.html.

Ya-Lin Zhang, Jun Zhou, Wenhao Zheng, Ji Feng, Longfei Li, Ziqi Liu, Ming Li, Zhiqiang Zhang, Chaochao Chen, Xiaolong Li, Yuan (Alan) Qi, and Zhi-Hua 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. doi: 10.1145/3342241. URL https://dl.acm.org/doi/10.1145/3342241.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The Clock and the Pizza: Two Stories in Mechanistic Explanation of Neural Networks. In *Advances in Neural Information Processing Systems*, volume 36, pp. 27223–27250. Curran Associates, Inc., 2023. doi: 10.52202/075280-1186. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/56cbfbf49937a0873d451343ddc8c57d-Paper-Conference.pdf.

Zhi-Hua Zhou and Ji Feng. Deep forest. *National Science Review*, 6(1):74–86, 2019. doi: 10.1093/nsr/nwy108. URL https://academic.oup.com/nsr/article/6/1/74/5123737.

A PRELIMINARIES

In this section, we first summarize frequently used notations in the following table.

Table 3: Frequently used notations.

Notation	Description
$c_{m,p}$	the target concept $c_{m,p}(\boldsymbol{x}) = \|\boldsymbol{x}\|_1 \bmod p$
$\mathcal{C}(\cdot,\cdot)$	the approximation complexity
$f(\cdot;\boldsymbol{\theta})$	a possibly randomized predictor with internal randomness $\boldsymbol{\theta}$
$h_{\mathrm{T}}, h_{\mathrm{F}}, h_{\mathrm{C}}$	the tree, forest, and tree cascade predictors
$\mathcal{H}_{\mathrm{T}}, \mathcal{H}_{\mathrm{F}}, \mathcal{H}_{\mathrm{C}}$	the class of trees, forests, and tree cascades with finite leaves
$\mathbb{I}(\cdot)$	the indicator function
$L(\cdot)$	the leaf complexity of a tree-based predictor
m	the input dimension
p	the modulus
$[p]$	the set $\{1, 2, \ldots, p\}$
$[p]_0$	the set $\{0, 1, \ldots, p-1\}$
$R(\cdot)$	the expected risk of a predictor
$\mathrm{sgn}(\cdot)$	the sign function
$\mathcal{U}_{m,p}$	the uniform distribution on $\mathcal{X}_{m,p}$
$\boldsymbol{x}$	an input vector from $\mathcal{X}_{m,p}$, and the i-th coordinate is x_i
$\mathcal{X}_{m,p}$	the input space $[p]_0^m$
$\mathcal{Y}_p$	the output space $[p]_0$
ω_p	the p-th root of unit $\omega_p = \mathrm{e}^{2\pi\mathrm{i}/p}$
$\oplus$	the cascade composition operation $g \oplus f = g(x, f(x))$

We then list several technical lemmas that will be used in later proofs.

Lemma 8. *Let $\omega_p = \mathrm{e}^{2\pi\mathrm{i}/p}$ be the p-th root of unit. Then for any integer $u \in \mathbb{Z}$, one has*

$$\mathbb{I}\big(u \equiv r \pmod p\big) = \frac{1}{p}\sum_{k=0}^{p-1} \omega_p^{k(u-r)} \,.$$

Proof. We prove this lemma by discussion as follows.

- Suppose $u \equiv r \pmod p$. Then one has $p \mid (u-r)$, which implies $\omega_p^{u-r} = 1$. Thus, we have
$$\frac{1}{p}\sum_{k=0}^{p-1} \omega_p^{k(u-r)} = \frac{1}{p}\sum_{k=0}^{p-1} 1^k = 1 \,,$$
which completes the proof of this case.
- Suppose $u \not\equiv r \pmod p$. Then one has $p \nmid (u-r)$, which implies $\omega_p^{u-r} \neq 1$. Meanwhile, we have $\omega_p^{p(u-r)} = (\omega_p^p)^{u-r} = 1$. Thus, one knows from the sum of a geometric series that
$$\frac{1}{p}\sum_{k=0}^{p-1} \omega_p^{k(u-r)} = \frac{1}{p}\cdot\frac{1-\omega_p^{p(u-r)}}{1-\omega_p^{u-r}} = 0 \,,$$
which completes the proof of this case.

Combining the cases above, we have completed the proof of the lemma. □

Lemma 9. *For any $p \geqslant 3$, $k \in [p-1]$, and $l \in [p]$, one has*

$$\left|\frac{\sin(kl\pi/p)}{\sin(k\pi/p)}\right| \leqslant l^{1-\frac{1}{p^2\ln p}}.$$

Proof. Let $\theta = k\pi/p$ for simplicity. Based on $k \in [p-1]$, one has $\pi/p \leqslant \theta \leqslant (p-1)\pi/p$.

We first prove by mathematical induction the following claim

$$|\sin(l\theta)| \leqslant l\sin\theta\cos(\pi/p) \quad \text{with} \quad 2 \leqslant l \leqslant p \,. \tag{4}$$

- Base case. For $l = 2$, one knows from the angle addition formula that
$$|\sin(2\theta)| = 2\sin\theta\cos\theta \overset{\text{(a)}}{\leqslant} 2\sin\theta\cos(\pi/p)\ ,$$
where inequality (a) holds from $\pi/p \leqslant \theta \leqslant (p-1)\pi/p$. Thus, we have proven the base case.
- Induction step. Suppose that the claim holds for $l = a$ with $a \geqslant 2$. Then for $l = a+1$, one knows from the angle addition formula and the triangle inequality that
$$\begin{aligned}
|\sin((a+1)\theta)| &\leqslant |\sin(a\theta)| \cdot |\cos\theta| + |\cos(a\theta)| \sin\theta \\
&\overset{\text{(a)}}{\leqslant} \big[a\cos(\pi/p) \cdot |\cos\theta| + |\cos(a\theta)|\big] \sin\theta \\
&\overset{\text{(b)}}{\leqslant} \big[a\cos^2(\pi/p) + 1\big] \sin\theta \\
&= (a+1)\sin\theta\cos(\pi/p) + \big[1 - a\cos(\pi/p)\big]\big[1 - \cos(\pi/p)\big]\sin\theta \\
&\overset{\text{(c)}}{\leqslant} (a+1)\sin\theta\cos(\pi/p)\ ,
\end{aligned}$$
where inequality (a) holds based on the induction hypothesis, inequality (b) holds because of $|\cos\theta| \leqslant \cos(\pi/p)$ for $\pi/p \leqslant \theta \leqslant (p-1)\pi/p$, and inequality (c) holds from $a\cos(\pi/p) \geqslant 1$ for $a \geqslant 2$ and $p \geqslant 3$. Thus, we have proven the inductive step.

Combining the results above, we have completed the proof of Eq. (4).

We proceed to prove the lemma by discussion.

- For $l = 1$, it is observed that both the left and right sides of the lemma are equal to 1. Thus, the lemma holds for $l = 1$.
- For $2 \leqslant l \leqslant p$, one knows from Eq. (4) that
$$\left|\frac{\sin(l\theta)}{\sin\theta}\right| \leqslant l\cos(\pi/p) = l^{1+\frac{\ln\cos(\pi/p)}{\ln l}} \overset{\text{(a)}}{\leqslant} l^{1+\frac{\ln\cos(\pi/p)}{\ln p}} \overset{\text{(b)}}{\leqslant} l^{1-\frac{1}{p^2\ln p}}\ ,$$
where inequality (a) holds based on $\ln\cos(\pi/p) < 0$ and $l \leqslant p$, and inequality (b) holds from $\ln\cos(\pi/p) \leqslant -1/p^2$. Recalling $\theta = k\pi/p$, we have proven the lemma for $2 \leqslant l \leqslant p$.

Combining the cases above, we have complete the proof. □

Lemma 10. *Let $y_0, y_1 \in \mathcal{Y}_p$ be two fixed labels, and $h_{\mathrm{F}} = (h_1, h_2, \ldots, h_K)$ is a forest, in which each tree h_j with $j \in [K]$ is truncated at depth d. Suppose that a deterministic priority order is used for breaking ties. Then there exists a polynomial $p : \mathcal{X}_{m,2} \to \mathbb{R}$ such that the following holds:*

- *The degree of p satisfies $\deg(p) \leqslant d$.*
- *For any input $\boldsymbol{b} \in \mathcal{X}_{2,p}$, it holds that $\operatorname{sgn}(p(\boldsymbol{b})) = \mathbb{I}(y_0$ ranks ahead of y_1 for $\boldsymbol{b})$.*

Proof. Let h_{T} be a tree with depth at most d, and $\mathcal{L}(h_{\mathrm{T}})$ denotes the set of leaves of h_{T}. For any leaf $l \in \mathcal{L}(h_{\mathrm{T}})$, let A_l and B_l denote the sets of coordinates that require the corresponding Boolean variables to be one and zero on the path to l, respectively. As the depth of h_{T} is at most d, we have $|A_l| + |B_l| \leqslant d$. Then we can construct a polynomial p_l at leaf $l \in \mathcal{L}(h_{\mathrm{T}})$ such that
$$p_l(\boldsymbol{b}) = \prod_{i \in A_l} b_i \prod_{i \in B_l} (1 - b_i) = \mathbb{I}(\boldsymbol{b} \text{ reaches leaf } l) \quad \text{with} \quad \deg(p_l) \leqslant d\ .$$
Let $\mathcal{L}_y(h_{\mathrm{T}}) = \{l \in \mathcal{L}(h_{\mathrm{T}}) \mid \mathrm{label}(l) = y\}$ denote the set of leaves of h_{T} that are labeled with y. Note that distinct leaves are disjoint and exactly one leaf is reached, we can construct a polynomial $p_{h_{\mathrm{T}},y}$ for the tree h_{T} and label y such that
$$p_{h_{\mathrm{T}},y}(\boldsymbol{b}) = \sum_{l \in \mathcal{L}_y(h_{\mathrm{T}})} p_l(\boldsymbol{b}) = \mathbb{I}(h_{\mathrm{T}}(\boldsymbol{b}) = y) \quad \text{with} \quad \deg(p_{h_{\mathrm{T}},y}) \overset{\text{(a)}}{\leqslant} d\ , \tag{5}$$
where inequality (a) holds since $p_{h_{\mathrm{T}},y}$ is the sum of polynomials with degree at most d. We then construct the polynomial p for the forest $h_{\mathrm{F}} = (h_1, h_2, \ldots, h_K)$ as follows
$$p(\boldsymbol{b}) = \sum_{j=1}^{K} \big[p_{h_j,y_0}(\boldsymbol{b}) - p_{h_j,y_1}(\boldsymbol{b})\big] + \mathbb{I}(\text{priority favors } y_0) - \frac{1}{2}.$$
Then it is observed that the following holds.

- Based on Eq. (5), one knows that p is the difference of two polynomials with degree at most d, which implies $\deg(p) \leqslant d$, which completes the proof of the first property.
- Recall that $V_y(\boldsymbol{b}) = \sum_{j=1}^{K} \mathbb{I}(h_j(\boldsymbol{b}) = y)$ is the vote count for label y. We then prove the second property by discussion.
  - Suppose that $V_{y_0}(\boldsymbol{b}) > V_{y_1}(\boldsymbol{b})$, i.e., y_0 ranks ahead of y_1 for $\boldsymbol{b}$. Then one has
  $$\sum_{j=1}^{K} \left[p_{h_j,y_0}(\boldsymbol{b}) - p_{h_j,y_1}(\boldsymbol{b})\right] \overset{\text{(a)}}{=} V_{y_0}(\boldsymbol{b}) - V_{y_1}(\boldsymbol{b}) \geqslant 1 \,,$$
  where equality (a) holds based on Eq. (5). Note that $\mathbb{I}(\text{priority favors } y_0) - 1/2 \geqslant -1/2$. Then we have $p(\boldsymbol{b}) \geqslant 1 - 1/2 = 1/2 > 0$, which implies $\operatorname{sgn}(p(\boldsymbol{b})) = 1$. Thus, the second property holds for this case.
  - Suppose that $V_{y_0}(\boldsymbol{b}) < V_{y_1}(\boldsymbol{b})$, i.e., y_1 ranks ahead of y_0 for $\boldsymbol{b}$. The proof of this case is similar to the first case.
  - Suppose that $V_{y_0}(\boldsymbol{b}) = V_{y_1}(\boldsymbol{b})$, i.e., y_0 and y_1 are tied for $\boldsymbol{b}$. Then one has
  $$p(\boldsymbol{b}) = \mathbb{I}(\text{priority favors } y_0) - \frac{1}{2} \,,$$
  which implies that $\operatorname{sgn}(p(\boldsymbol{b})) = 1$ if the priority favors y_0 and $\operatorname{sgn}(p(\boldsymbol{b})) = -1$ otherwise. Thus, the second property holds for this case.

Combining the cases above, we have completed the proof of the second property. □

Lemma 11 ((Aspnes et al., 1991), Theorem 2 and Lemma 3)**.** *Let p be a polynomial of degree d, and $A = \{\boldsymbol{b} \in \mathcal{X}_{m,2} \mid \operatorname{sgn}(p(\boldsymbol{b})) \neq (-1)^{\|\boldsymbol{b}\|_1}\}$. Then for $d < m$, we have*

$$|A| \geqslant \sum_{i=0}^{\lfloor (m-d-1)/2 \rfloor} \binom{m}{i} \,.$$

Lemma 12. *For any $m \geqslant 1$, $d \in [m-1]$, and any function $g : \mathcal{X}_{m,2} \to \{-1, 1\}$ that is the sign of a polynomial with degree at most d, the following holds*

$$\left| \mathbb{E}_{\boldsymbol{b}} \left[g(\boldsymbol{b})(-1)^{\|\boldsymbol{b}\|_1} \right] \right| \leqslant \frac{6d}{\sqrt{m}} \,.$$

Proof. We first consider the case of $\mathbb{E}_{\boldsymbol{b}} \left[g(\boldsymbol{b})(-1)^{\|\boldsymbol{b}\|_1} \right] \geqslant 0$. According to Lemma 11, we have

$$\Pr\left[g(\boldsymbol{b}) \neq (-1)^{\|\boldsymbol{b}\|_1} \right] \geqslant 2^{-m} \sum_{i=0}^{q} \binom{m}{i} \quad \text{with} \quad q = \left\lfloor \frac{m-d-1}{2} \right\rfloor \,, \tag{6}$$

where the probability is taken over $\boldsymbol{b} \sim \mathcal{U}_{m,2}$. Note that both functions g and $(-1)^{\|\boldsymbol{b}\|_1}$ take values in $\{-1, 1\}$. Then one knows from Eq. (6) that

$$\mathbb{E}_{\boldsymbol{b}} \left[g(\boldsymbol{b})(-1)^{\|\boldsymbol{b}\|_1} \right] = 1 - 2 \Pr\left[g(\boldsymbol{b}) \neq (-1)^{\|\boldsymbol{b}\|_1} \right] \overset{\text{(a)}}{\leqslant} 2^{-m} \sum_{i=q+1}^{m-q-1} \binom{m}{i} \,, \tag{7}$$

where inequality (a) holds from the combinatorial identities $\sum_{i=0}^{q} \binom{m}{i} = \sum_{i=m-q}^{m} \binom{m}{i}$ and $\sum_{i=0}^{m} \binom{m}{i} = 2^m$. We proceed the proof by discussing the parity of m.

- Suppose that $m = 2n$ for $n \in \mathbb{N}^+$. Then one knows from the inequality $1 - x \leqslant \mathrm{e}^{-x}$ that
$$2^{-2n} \binom{2n}{n} = \prod_{j=1}^{n} \left(1 - \frac{1}{2j}\right) \leqslant \mathrm{e}^{-\frac{1}{2}\sum_{j=1}^{n} \frac{1}{j}} \overset{\text{(a)}}{\leqslant} (n+1)^{-1/2} \,,$$
where inequality (a) holds based on $\sum_{j=1}^{n} \frac{1}{j} \geqslant \ln(n+1)$. Thus, we have
$$2^{-m} \sum_{i=q+1}^{m-q-1} \binom{m}{i} \leqslant 2^{-2n} \sum_{i=q+1}^{2n-q-1} \binom{2n}{n} \overset{\text{(a)}}{\leqslant} \frac{d+2}{\sqrt{n+1}} \,,$$
where inequality holds since there are $m - 2q - 1 \leqslant d + 2$ terms.

- Suppose that $m = 2n + 1$ for $n \in \mathbb{N}$. Similar to the proof of the first case, we have

$$2^{-m} \sum_{i=q+1}^{m-q-1} \binom{m}{i} \leqslant 2^{-2n-1} \sum_{i=q+1}^{2n-q} \binom{2n+1}{n} \leqslant 2^{-2n} \sum_{i=q+1}^{2n-q} \binom{2n}{n} \leqslant \frac{d+3}{\sqrt{n+1}} .$$

Substituting results of the above cases into Eq. (7), we obtain

$$\left| \mathbb{E}_{\boldsymbol{b}} \left[g(\boldsymbol{b})(-1)^{\|\boldsymbol{b}\|_1} \right] \right| = \mathbb{E}_{\boldsymbol{b}} \left[g(\boldsymbol{b})(-1)^{\|\boldsymbol{b}\|_1} \right] \leqslant \frac{d+3}{\sqrt{m/2}} \leqslant \frac{6d}{\sqrt{m}} ,$$

which completes the proof for $\mathbb{E}_{\boldsymbol{b}} \left[g(\boldsymbol{b})(-1)^{\|\boldsymbol{b}\|_1} \right] \geqslant 0$.

For $\mathbb{E}_{\boldsymbol{b}} \left[g(\boldsymbol{b})(-1)^{\|\boldsymbol{b}\|_1} \right] < 0$, applying the above proof to $-g$ completes the proof. □

B Complete Proofs for Theorem 5

We first introduce a useful lemma regarding the number of leaves required by trees to perfectly represent 2-dimensional modular additions.

Lemma 13. *Let the input dimension $m = 2$ be fixed. For the input space $\mathcal{X}_{m,p} = [p]_0^m$, the approximation complexity of trees satisfies*

$$\mathcal{C}(\mathcal{H}_{\mathrm{T}}, 0) = p^2.$$

Proof. For $m = 2$, let $g(s,t) = (s + t) \bmod p$ denote the target modular addition function, where the input $(s,t) \in \mathcal{X}_{m,p}$ has p^2 different values.

We first prove the upper bound. Consider a chain of queries $s \leqslant i$ with $i \in [p-2]_0$. It is observed that these queries isolate the p possible values of s. Then, below the branch for each fixed s, another chain of queries $z \leqslant j$ with $j \in [p-2]_0$ can isolate z. There are p outer branches for s and p leaves for z below each branch, generating exactly p^2 nonempty leaf rectangles. For each leaf (s,z), allocating a label $g(s,t)$ leads to a tree with zero error. Thus, we have proven $\mathcal{C}(\mathcal{H}_{\mathrm{T}}, 0) \leqslant p^2$.

We then show the lower bound. Let h_{T} be a tree with zero error. For any nonempty leaf of h_{T}, denote by $I \times J$ the corresponding leaf rectangle. Suppose that I contains two distinct values s and s'. Then one knows from $0 < |s - s'| < p$ that the target function outputs satisfy $g(s,t) \neq g(s',t)$ for any fixed $t \in J$, which contradicts zero error of h_{T}. Thus, we have $|I| = 1$. Similarly, one has $|J| = 1$. Therefore, each leaf of h_{T} is a single point, implying $\mathcal{C}(\mathcal{H}_{\mathrm{T}}, 0) \geqslant p^2$.

Combining the arguments above, we have completed the proof. □

Based on the above lemma, we are now ready to prove Theorem 5.

Proof of Theorem 5. Recalling that $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$ denote the input vector. Let h_1 be a tree such that $h_1(\boldsymbol{x}) = (x_1 + x_2) \bmod p$. According to Lemma 13, the tree h_1 exists and has p^2 leaves. For $i \in \{2, 3, \ldots, m-1\}$, define h_i as the tree satisfying $h_i(\boldsymbol{x}) = (h_{i-1}(\boldsymbol{x}) + x_{i+1}) \bmod p$. Based on Lemma 13, the tree h_i exists and has p^2 leaves. Then we construct a tree cascade $h_{\mathrm{C}}(\boldsymbol{x}) = h_{m-1} \oplus h_{m-2} \oplus \cdots \oplus h_1(\boldsymbol{x})$, which has $(m-1)p^2$ leaves and satisfies $h_{\mathrm{C}}(\boldsymbol{x}) = \|\boldsymbol{x}\|_1 \bmod p$. Thus, we obtain $\mathcal{C}(\mathcal{H}_{\mathrm{C}}, 0) \leqslant (m-1)p^2$, which completes the proof. □

C Complete Proofs for Theorem 6

We first count the number of points with any label in any rectangle and provide an upper bound in the following lemma, which is used to derive lower bounds for both trees and forests.

Lemma 14. *Let $R = I_1 \times \cdots \times I_m \subseteq \mathcal{X}_{m,p}$ be a nonempty rectangle. For any $r \in \mathcal{Y}_p$, define $N_r(R) = |\{\boldsymbol{x} \in R \mid \|\boldsymbol{x}\|_1 \equiv r \pmod p\}|$. Then for any $p \geqslant 3$, we have*

$$N_r(R) \leqslant \frac{|R|}{p} + \frac{p-1}{p} |R|^{1 - \frac{1}{p^2 \ln p}}.$$

Proof. According to the definition of $N_r(R)$, we have $N_r(R) = \sum_{\boldsymbol{x}\in R} \mathbb{I}\big(\|\boldsymbol{x}\|_1 \equiv r \pmod p\big)$. Then one knows from exchange of finite summation order and Lemma 8 with $u = \|\boldsymbol{x}\|_1$ that

$$N_r(R) = \frac{1}{p}\sum_{k=0}^{p-1} \omega_p^{-kr} \sum_{\boldsymbol{x}\in R} \prod_{i=1}^{m} \omega_p^{kx_i} \overset{\text{(a)}}{=} \frac{1}{p}\sum_{k=0}^{p-1} \omega_p^{-kr} \prod_{i=1}^{m} \sum_{x_i\in I_i} \omega_p^{kx_i} \,, \tag{8}$$

where equality (a) holds from the structural product and summation in the rectangle R. Note that in Eq. (8), the term with $k = 0$ equals $|R|/p$. For any $i \in [m]$, let $I_i = [a_i, b_i]$ with endpoints $a_i, b_i \in [p]_0$. Then one knows from the sum of a geometric series that

$$N_r(R) = \frac{|R|}{p} + \frac{1}{p}\sum_{k=1}^{p-1} \omega_p^{-kr} \prod_{i=1}^{m} \omega_p^{ka_i} \frac{1-\omega_p^{k|I_i|}}{1-\omega_p^{k}} \overset{\text{(a)}}{\leqslant} \frac{|R|}{p} + \frac{1}{p}\sum_{k=1}^{p-1} \prod_{i=1}^{m} \left|\frac{1-\omega_p^{k|I_i|}}{1-\omega_p^{k}}\right| \,, \tag{9}$$

where inequality (a) holds based on the triangle inequality and $|\omega_p^{-kr}| = |\omega_p^{ka_i}| = 1$. For any real number $\theta \in \mathbb{R}$, preliminary calculation implies

$$\left|1 - \mathrm{e}^{\theta \mathrm{i}}\right| = \sqrt{(1-\cos\theta)^2 + \sin^2\theta} = 2|\sin(\theta/2)| \,,$$

which, together with Eq. (9), indicates

$$\begin{aligned} N_r(R) &\leqslant \frac{|R|}{p} + \frac{1}{p}\sum_{k=1}^{p-1} \prod_{i=1}^{m} \left|\frac{\sin(k|I_i|\pi/p)}{\sin(k\pi/p)}\right| \\ &\overset{\text{(a)}}{\leqslant} \frac{|R|}{p} + \frac{1}{p}\sum_{k=1}^{p-1} \prod_{i=1}^{m} |I_i|^{1-\frac{1}{p^2\ln p}} \\ &= \frac{|R|}{p} + \frac{p-1}{p}|R|^{1-\frac{1}{p^2\ln p}} \,, \end{aligned}$$

where inequality (a) holds from $|I_i| \in [p]$ and Lemma 9. Thus, we have completed the proof. □

We then proceed to prove Theorem 6.

Proof of Theorem 6. We first prove the first conclusion. Let h_{T} be a tree with error at most ϵ. Note that the input space $\mathcal{X}_{m,p}$ has p^m points, and the input $\boldsymbol{x} \sim \mathcal{U}_{m,p}$ follows the uniform distribution. Thus, the number of misclassified points satisfies

$$|\{\boldsymbol{x} \mid h_{\mathrm{T}}(\boldsymbol{x}) \neq c_{m,p}(\boldsymbol{x})\}| \leqslant \lfloor \epsilon p^m \rfloor \,. \tag{10}$$

For the target concept $c_{m,p}$, the input space $\mathcal{X}_{m,p}$ is partitioned into p^m regions with distinct labels for adjacent ones. For the tree h_{T}, let N denote the number of regions that the input space is partitioned into such that adjacent regions have different labels. Then one knows from Eq. (10) that

$$N \overset{\text{(a)}}{\leqslant} p^m - 2m|\{\boldsymbol{x} \mid h_{\mathrm{T}}(\boldsymbol{x}) \neq c_{m,p}(\boldsymbol{x})\}| \leqslant p^m(1-2m\epsilon) \,,$$

where inequality (a) holds since each misclassified point is adjacent to at most $2m$ points and can reduce at most $2m$ partitioned regions. Thus, we have proven the first conclusion.

We proceed to prove the second conclusion. Let h_{T} be a tree with L nonempty leaves. For $l \in [L]$, define R_l and r_l as the rectangle and label corresponding to the l-th leaf, respectively. Recalling the definition of expected risk defined in Eq. (2), we have

$$R(h_{\mathrm{T}}) = 1 - \frac{1}{p^m}\sum_{l=1}^{L} N_{r_l}(R_l) \overset{\text{(a)}}{\geqslant} 1 - \frac{1}{p^m}\sum_{l=1}^{L}\left[\frac{|R_l|}{p} + \frac{p-1}{p}|R_l|^{1-\frac{1}{p^2\ln p}}\right] \,, \tag{11}$$

where inequality (a) holds by applying Lemma 14 to rectangles $\{R_l\}_{l=1}^{L}$. Note that the function $f(x) = x^{1-\frac{1}{p^2\ln p}}$ is a concave function of x. Then one knows from Jensen's inequality that Eq. (11) can be rewritten as

$$\begin{aligned} R(h_{\mathrm{T}}) &\geqslant 1 - \frac{1}{p^m}\left[\frac{1}{p}\sum_{l=1}^{L}|R_l| + \frac{p-1}{p}L^{\frac{1}{p^2\ln p}}\left(\sum_{l=1}^{L}|R_l|\right)^{1-\frac{1}{p^2\ln p}}\right] \\ &\overset{\text{(a)}}{=} 1 - \left[\frac{1}{p} + \frac{p-1}{p}\left(\frac{L}{p^m}\right)^{\frac{1}{p^2\ln p}}\right] \,, \end{aligned} \tag{12}$$

where equality (a) holds since the rectangles $\{R_l\}_{l=1}^{L}$ form a partition of the input space $\mathcal{X}_{m,p}$. Thus, to ensure $R(h_{\mathrm{T}}) \leqslant \epsilon$, one knows from Eq. (12) that $L \geqslant p^m\big(1 - p\epsilon/(p-1)\big)^{p^2 \ln p}$. Then by the arbitrariness of h_{T}, we have

$$\mathcal{C}(\mathcal{H}_{\mathrm{T}}, \epsilon) \geqslant p^m \left(1 - \frac{p\epsilon}{p-1}\right)^{p^2 \ln p} ,$$

which completes the proof of the second conclusion. □

D Complete Proofs for Theorem 7

Proof of Theorem 7. Let $h_{\mathrm{F}} \in \mathcal{H}_{\mathrm{F}}$ be a forest with L total leaves and risk $R(h_{\mathrm{F}}) \leqslant \epsilon$. The proof is divided into several steps.

Step 1: h_{F} is restricted to a Boolean task. As p is even, define $a = p/2 \in \mathbb{N}^+$. For any $\boldsymbol{x} \in \mathcal{X}_{m,p}$ and $\boldsymbol{b} \in \{0,1\}^m$, let $\phi_{\boldsymbol{x}} : \{0,1\}^m \to \mathcal{X}_{m,p}$ be a function of $\boldsymbol{b}$, of which the i-th component is defined as follows

$$\phi_{\boldsymbol{x}}(\boldsymbol{b})_i = x_i + ab_i \pmod p . \tag{13}$$

Combining Eqs. (1) and (13), one has

$$c_{m,p}(\phi_{\boldsymbol{x}}(\boldsymbol{b})) = c_{m,p}(\boldsymbol{x}) + a\|\boldsymbol{b}\|_1 \pmod p ,$$

which corresponds to a Boolean task with two possible labels, i.e., $c_{m,p}(\boldsymbol{x})$ and $c_{m,p}(\boldsymbol{x}) + a$, based on the parity of $\|\boldsymbol{b}\|_1$. Define the expected risk under the Boolean task as

$$R_{\phi_{\boldsymbol{x}}}(h_{\mathrm{F}}) = \Pr_{\boldsymbol{b}\sim\mathcal{U}_{m,2},\boldsymbol{\theta}}\big[h_{\mathrm{F}}(\phi_{\boldsymbol{x}}(\boldsymbol{b});\boldsymbol{\theta}) \neq c_{m,p}(\phi_{\boldsymbol{x}}(\boldsymbol{b}))\big] ,$$

where $\boldsymbol{\theta}$ is the randomness of h_{F}. Based on the uniform distribution of $\boldsymbol{b}$ and $\boldsymbol{x}$, it follows that $\phi_{\boldsymbol{x}}(\boldsymbol{b})$ is uniformly distributed over $\mathcal{X}_{m,p}$. Thus, we have

$$\mathbb{E}_{\boldsymbol{x}}\big[R_{\phi_{\boldsymbol{x}}}(h_{\mathrm{F}})\big] = \Pr_{\phi_{\boldsymbol{x}}(\boldsymbol{b})\sim\mathcal{U}_{m,p},\boldsymbol{\theta}}\big[h_{\mathrm{F}}(\phi_{\boldsymbol{x}}(\boldsymbol{b});\boldsymbol{\theta}) \neq c_{m,p}(\phi_{\boldsymbol{x}}(\boldsymbol{b}))\big] = R(h_{\mathrm{F}}) \leqslant \epsilon .$$

Then by the definition of expectation, there exists a specific $\tilde{\boldsymbol{x}} \in \mathcal{X}_{m,p}$ such that

$$R_{\phi_{\tilde{\boldsymbol{x}}}}(h_{\mathrm{F}}) \leqslant \mathbb{E}_{\boldsymbol{x}}\big[R_{\phi_{\boldsymbol{x}}}(h_{\mathrm{F}})\big] \leqslant \epsilon . \tag{14}$$

Furthermore, the Boolean forest, i.e., h_{F} under the Boolean task, has at most L leaves.

Step 2: h_{F} is truncated to depth d. Let $h_{\mathrm{F},d}$ be the forest obtained by truncating h_{F} to depth d, i.e., for each tree in forest h_{F}, replacing each subtree rooted at depth d by an arbitrary leaf. Since the input $\boldsymbol{b} \in \{0,1\}^m$ is a Boolean vector, each coordinate of $\boldsymbol{b}$ can be queried at most once along any path from the root to a leaf. Thus, one knows from the uniform distribution of $\boldsymbol{b}$ that each node at depth d is reached with probability 2^{-d}. Since the truncation does not generate new leaves, the truncated forest $h_{\mathrm{F},d}$ has at most L leaves. Note that the output can change only if the input reaches one of the depth-d nodes. Then the union bound indicates

$$\Pr_{\boldsymbol{b}\sim\mathcal{U}_{m,2}}\big[h_{\mathrm{F},d}(\phi_{\tilde{\boldsymbol{x}}}(\boldsymbol{b})) \neq h_{\mathrm{F}}(\phi_{\tilde{\boldsymbol{x}}}(\boldsymbol{b}))\big] \overset{\text{(a)}}{\leqslant} L2^{-d} \leqslant \delta , \tag{15}$$

where $\delta = (1-2\epsilon)/4$ and inequality (a) holds by setting $d = \lceil \log_2(L/\delta) \rceil$.

Step 3: we prove the lower bound for $L(h_{\mathrm{F}})$. We prove the theorem by discussion.

- Suppose that $d \geqslant m$. Then one knows that

$$L \overset{\text{(a)}}{\geqslant} \frac{1-2\epsilon}{4}2^{m-1} = \frac{1-2\epsilon}{8}2^m \geqslant \frac{1-2\epsilon}{8}2^{(1-2\epsilon)\sqrt{m}} ,$$

  where inequality (a) holds according to the choices of δ and d in Eq. (15).
- Suppose that $d < m$. Define a function $g : \mathcal{X}_{m,2} \to \{-1, 1\}$ as follows

$$g(\boldsymbol{b}) = \begin{cases} 1, & h_{\mathrm{F},d} \text{ ranks } c_{m,p}(\tilde{\boldsymbol{x}}) \text{ ahead of } c_{m,p}(\tilde{\boldsymbol{x}}) + a, \\ -1, & \text{otherwise.} \end{cases}$$

By Lemma 10, we have that g is the sign of a polynomial with degree at most d. Meanwhile, one knows from the union bound that

$$\begin{aligned}\Pr\left[g(\boldsymbol{b}) \neq (-1)^{\|\boldsymbol{b}\|_1}\right] &= \Pr\left[h_{\mathrm{F},d}(\phi_{\tilde{\boldsymbol{x}}}(\boldsymbol{b})) \neq h_{\mathrm{F}}(\phi_{\tilde{\boldsymbol{x}}}(\boldsymbol{b})) \vee h_{\mathrm{F}}(\phi_{\tilde{\boldsymbol{x}}}(\boldsymbol{b})) \neq c_{m,p}(\phi_{\tilde{\boldsymbol{x}}}(\boldsymbol{b}))\right] \\ &\leqslant \Pr\left[h_{\mathrm{F},d}(\phi_{\tilde{\boldsymbol{x}}}(\boldsymbol{b})) \neq h_{\mathrm{F}}(\phi_{\tilde{\boldsymbol{x}}}(\boldsymbol{b}))\right] + R_{\phi_{\tilde{\boldsymbol{x}}}}(h_{\mathrm{F}}) \\ &\overset{\text{(a)}}{\leqslant} (1+2\epsilon)/4\,,\end{aligned} \tag{16}$$

where the probability is taken with respect to $\boldsymbol{b} \sim \mathcal{U}_{m,2}$, inequality (a) holds based on Eq. (14), Eq. (15), and the choice of δ after Eq. (15). Note that both g and $(-1)^{\|\boldsymbol{b}\|_1}$ are functions with values in $\{-1, 1\}$. Then one knows from Eq. (16) that

$$\mathbb{E}\Big[g(\boldsymbol{b})(-1)^{\|\boldsymbol{b}\|_1}\Big] = 1 - 2\Pr\left[g(\boldsymbol{b}) \neq (-1)^{\|\boldsymbol{b}\|_1}\right] \geqslant \frac{1}{2} - \epsilon\,, \tag{17}$$

which, together with Lemma 12, implies that

$$\frac{1}{2} - \epsilon \leqslant \mathbb{E}\Big[g(\boldsymbol{b})(-1)^{\|\boldsymbol{b}\|_1}\Big] \leqslant \frac{6d}{\sqrt{m}}\,. \tag{18}$$

Meanwhile, one knows from the choice of d after Eq. (15) that $d \leqslant \log_2(L/\delta) + 1$, which, together with Eq. (18), implies that

$$\frac{\sqrt{m}}{12}(1-2\epsilon) \leqslant d \leqslant \log_2(L/\delta) + 1\,.$$

Rearranging the above inequality, one knows from the choice of δ after Eq. (15) that

$$L \geqslant \frac{\delta}{2} 2^{\frac{1}{12}(1-2\epsilon)\sqrt{m}} = \frac{1-2\epsilon}{8} 2^{\frac{1}{12}(1-2\epsilon)\sqrt{m}}\,.$$

Combining the above two cases, we obtain

$$L \geqslant \frac{1-2\epsilon}{8} 2^{\frac{1-2\epsilon}{12}\sqrt{m}}\,.$$

Then by the arbitrariness of h_{F}, we have

$$\mathcal{C}(\mathcal{H}_{\mathrm{F}}, \epsilon) \geqslant \frac{1-2\epsilon}{8} 2^{\frac{1-2\epsilon}{12}\sqrt{m}}\,,$$

which completes the proof. □

E Experimental Details

E.1 Metrics and Evaluation Protocols

We specify the metrics used in Section 5. For a test set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, prediction accuracy is

$$\widehat{\mathsf{acc}}(h) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\{h(\boldsymbol{x}_i) = y_i\}.$$

In the finite-domain experiments, the test set is the entire grid $\mathcal{X}_{m,p}$, so this quantity equals uniform-distribution accuracy. For nonuniform cell sampling in Figure 4(a), accuracy uses the $300{,}000$ held-out examples drawn from the learning distribution for each seed. For uniform cell sampling in Figure 4(b), we sample 500 independent uniform perturbations within each of the $1{,}024$ cells defined in Appendix E.3 and average correctness over all $512{,}000$ points. The same evaluation set, generated with seed 20260831, is shared across models and data seeds, giving every cell equal weight.

Leaf complexity is the sum of the realized leaf counts of all component trees. For each model and maximum component depth, Figure 4 reports the arithmetic mean of leaf counts and accuracies over five seeds. Table 8 reports medians and interquartile ranges; the quartiles are the second and fourth ordered observations, and the median is the third. For Table 2, we select the smallest leaf count reaching each accuracy threshold within each seed's depth scan; ties favor the smaller component depth. We then report the mean and sample standard deviation across successful seeds, together with their number. The per-seed selections are listed in Table 9.

E.2 Exact Finite-Domain Computations

We detail the exact computations in Section 5.1. For a rectangular grid region R and a budget $L \geq 1$, let $V(R, L)$ be the maximum number of correctly classified points attainable with at most L leaves. Let $M(R) = \max_{r \in \mathcal{Y}_p} N_r(R)$ be the majority-class count. Let $\mathcal{A}(R)$ contain all pairs of nonempty child regions obtained by a coordinate threshold split. The single-leaf value is $V(R, 1) = M(R)$, and larger budgets satisfy

$$V(R, L) = \max \left\{ M(R), \max_{\substack{(R^-, R^+) \in \mathcal{A}(R) \\ 1 \leq a < L}} \left[V(R^-, a) + V(R^+, L - a) \right] \right\}. \tag{19}$$

Values saturate once the budget exceeds the region's number of points. We evaluate Eq. (19) on smaller regions first and return $V(\mathcal{X}_{m,p}, L)/p^m$ for $L = 1, \ldots, p^m$. Translating a rectangle only permutes residue labels, and permuting its coordinates preserves the admissible splits. We therefore cache states by sorted side lengths. This implementation agrees with the full rectangular-state computation at all 1,820 evaluated budgets.

For the accuracy curves, write $u = L/p^m$. The two upper bounds used in Figure 3 are

$$U_1(L) = 1 - \frac{1 - u}{2m}, \qquad U_2(L) = \frac{1}{p} + \frac{p - 1}{p} u^{1/(p^2 \ln p)}.$$

We plot their minimum for $p \geq 3$ and use U_1 for $p = 2$. For dimension sweeps, we take the maximum of one leaf and the applicable complexity lower bounds in Theorem 6: the first requires $\epsilon < 1/(2m)$, and the second requires $p \geq 3$ and $\epsilon < (p - 1)/p$. Threshold leaf counts are obtained from integer correct counts: accuracy q requires at least $\lceil qp^m \rceil$ correct predictions.

The explicit cascade first computes $(x_1 + x_2) \bmod p$ and then incorporates one coordinate at each subsequent layer. Each component partitions the p^2 pairs of incoming values, giving $m - 1$ trees and $(m - 1)p^2$ leaves. All three main-text instances are evaluated exhaustively. Across $p \in \{2, 3, 4\}$ and $m = 2, \ldots, 30$, we also check all inputs when $p^m \leq 65{,}536$ and 10,000 uniformly sampled inputs otherwise. These checks cover 31 exhaustive and 56 sampled cases, with no errors among 867,013 evaluated inputs. The sampling seed is $20260925 + 100p + m$.

E.3 Construction of the Learning Distribution

We give the complete distribution used in Section 5.2. Sample a prefix length $J \in \{0, \ldots, 7\}$ with the probabilities in Table 4, independently of ten fair Bernoulli variables $b_1, \ldots, b_{10}$. The first three grid coordinates are $z_1 = b_1$, $z_2 = 2b_2$, and $z_3 = 4b_3$. For $k = 1, \ldots, 7$, set $z_{3+k} = 4b_{3+k}\mathbb{I}\{k \leq J\}$. We form $\boldsymbol{x} = \boldsymbol{z} + \boldsymbol{\eta}$, where the coordinates of $\boldsymbol{\eta}$ are independent and uniform on $[-0.5, 0.5)$, and set $y = (\sum_{j=1}^{10} z_j) \bmod 8$. The first three bits provide all eight residues, while the random prefix determines how many further coordinates contribute to the sum.

Table 4: Probabilities of the active suffix-prefix length.

j	0	1	2	3	4	5	6	7
$\mathbb{P}(J = j)$	.20	.15	.13	.11	.09	.07	.05	.20

The grid support is $\mathcal{S} = \{0, 1\} \times \{0, 2\} \times \{0, 4\}^8$, containing 1,024 points. Each $\boldsymbol{z} \in \mathcal{S}$ defines a support cell $\boldsymbol{z} + [-0.5, 0.5)^{10}$ with a constant target label. All cells have positive probability because $\mathbb{P}(J = 7) > 0$. For each seed in $\{0, 1, 2, 3, 4\}$, NumPy's `default_rng` samples the prefix lengths, binary variables, and perturbations in that order for one million examples. Inputs are stored in double precision. We use `train_test_split` with the same seed to shuffle the indices and obtain 700,000 training and 300,000 test examples, without stratification. Every model evaluated under a given seed uses this same split.

E.4 Models and Training Settings

We summarize the implementations used in Section 5.2 in Table 5. The single tree and cascade components use CART (Breiman et al., 1984); the random forests follow Breiman (2001), as imple-

mented in scikit-learn (Pedregosa et al., 2011). The component-depth scan and forest widths follow Lyu et al. (2024). We evaluate all 7×15 configurations for each of the five datasets, giving 525 completed runs.

Table 5: Training settings. T and TC use all input features at each split; RF samples the square root of the feature count.

Setting	T / TC	RF
Splitting criterion	Gini	Gini
Split search	Best	Best
Maximum component depth	$1, \ldots, 15$	$1, \ldots, 15$
Minimum samples to split / leaf	2 / 1	2 / 1
Features considered per split	All	$\lfloor \sqrt{10} \rfloor = 3$
Class weights	None	None
Cost-complexity pruning α	0	0
Number of components	1 / 2, 4, 7	9, 19, 29
Bootstrap sampling	No	Yes, 700,000 draws
Prediction	Final tree's class	Mean class probabilities

A cascade's first tree receives the ten original features. For each later layer, we append the preceding tree's hard class prediction as one scalar feature to the original input, producing eleven features. Every layer is trained against the final labels y. The additional training feature is computed from the preceding layer's predictions on the training examples themselves. At evaluation, predictions are propagated through the same sequence, and the final tree supplies the output. Thus, later layers always retain access to the original input and the immediately preceding prediction.

For data seed s, maximum component depth d, and family code c, the model seed is $10^6 s + 100c + d$. The family codes are $c = 1$ for T, $c = 10 + D$ for TC-D, and $c = 100 + K$ for RF-K. Cascade layer ℓ, indexed from one, uses ten times the model seed plus ℓ. The saved learning environment uses Python 3.11.11, NumPy 2.4.6, SciPy 1.17.1, and scikit-learn 1.9.0 on Windows. Training uses CPU implementations; random forests use `n_jobs=-1`. All remaining parameters use the library defaults.

E.5 Running-Residue Diagnostics

We detail the measurement in Section 5.3. For each data seed, we fit TC-7 with maximum component depth four using the split and model seed above, and extract all seven layers' predictions on the test set. For example i, the reference state is $s_{i,\ell+3} = (\sum_{j=1}^{\ell+3} z_{ij}) \bmod 8$, computed from the unperturbed grid coordinates. If $\widehat{s}_{i,\ell}$ is the prediction of layer ℓ, its agreement is

$$\frac{1}{300{,}000} \sum_{i=1}^{300{,}000} \mathbb{I}\{\widehat{s}_{i,\ell} = s_{i,\ell+3}\}.$$

Classes are compared directly, without permuting labels. Table 6 gives the numerical values underlying Figure 5. Figure 5 reports the mean and sample standard deviation of the mismatch rate over five seeds.

All fitted components have 16 leaves. This count matches the compact updates available on the constructed support: the first four binary coordinates admit 16 combinations, and each later update combines eight incoming residues with a coordinate in $\{0, 4\}$. The seven layers therefore correspond to $s_4, \ldots, s_{10}$, with a total of 112 leaves. The measured agreement shows how closely the fitted predictions follow these states.

F Additional Experimental Results

F.1 Additional Finite-Domain Results

Figures 6 and 7 provide additional finite-domain curves for $p \in \{2, 3, 4\}$, and Table 7 gives exact single-tree leaf requirements at three thresholds for all thirteen computed instances. At $p = 2, m =$

Table 6: Layerwise state agreement (%) for TC-7 at maximum component depth four. Each entry uses 300,000 test examples; every component has 16 leaves.

Layer	State	Seed 0	Seed 1	Seed 2	Seed 3	Seed 4
1	s_4	99.9887	99.9910	99.9827	99.9933	99.9763
2	s_5	99.9860	99.9870	99.9750	99.9903	99.9740
3	s_6	99.9843	99.9820	99.9740	99.9863	99.9730
4	s_7	99.9837	99.9803	99.9700	99.9860	99.9613
5	s_8	99.9707	99.9770	99.9660	99.9793	99.9593
6	s_9	99.9650	99.9757	99.9623	99.9783	99.9580
7	s_{10}	99.9650	99.9757	99.9623	99.9783	99.9580

4, the optimal tree requires 15 leaves for 90% accuracy, while the exact cascade uses 12. As the dimension increases to nine, the former rises to 414 leaves while the latter uses 32. The multi-class instances show the same growing advantage of sequential composition with dimension. Figure 8 shows how the single-tree lower bound and cascade construction size vary with dimension for $p \in \{3, 4\}$, $m = 2, \ldots, 30$, and $\epsilon \in \{0, 0.05, 0.2\}$. For both moduli, the single-tree lower bound grows exponentially with dimension at each fixed tolerance, while the exact cascade size grows linearly.

Table 7: Minimum single-tree leaf counts from exact DP and the leaf count of the exact cascade construction. Cascade sizes are construction upper bounds.

			Optimal single tree			Exact cascade
p	m	p^m	90%	95%	100%	$(m-1)p^2$
2	4	16	15	16	16	12
2	5	32	28	31	32	16
2	6	64	54	60	64	20
2	7	128	106	118	128	24
2	8	256	209	234	256	28
2	9	512	414	465	512	32
3	3	27	24	26	27	18
3	4	81	69	75	81	27
3	5	243	207	225	243	36
4	2	16	15	16	16	16
4	3	64	55	60	64	32
4	4	256	219	238	256	48
5	3	125	107	116	125	50

F.2 Detailed Learning Results

We report the full numerical results for Section 5.2 in Table 8. Each row reports the median and interquartile range across the five runs at one model and component depth. Both evaluation distributions are included for all 105 configurations. Table 9 lists each seed's smallest observed leaf count at 95% and 99% test accuracy, together with the selected component depth. All five seeds attain both thresholds for T and each cascade, whereas the forests reach 95% in zero, four, and five seeds for widths 9, 19, and 29, respectively. None of the evaluated forests reaches 99%.

Table 8: Complete learning depth scans. Entries are median [first quartile, third quartile] over five seeds. Accuracies are percentages; d is the maximum component-tree depth.

Model	d	Total leaves	Test accuracy (%)	Support-uniform accuracy (%)
T	1	2 [2, 2]	25.032 [25.025, 25.033]	25.000 [25.000, 25.000]
T	2	4 [4, 4]	50.074 [49.974, 50.076]	50.000 [50.000, 50.000]
T	3	8 [8, 8]	59.990 [59.988, 59.990]	50.001 [49.999, 50.001]
T	4	16 [16, 16]	67.523 [67.485, 67.562]	49.998 [49.998, 49.998]
T	5	32 [32, 32]	73.987 [73.929, 74.016]	50.000 [50.000, 50.001]

Continued on next page

Table 8 continued

Model	d	Total leaves	Test accuracy (%)	Support-uniform accuracy (%)
T	6	64 [64, 64]	79.407 [79.404, 79.461]	49.998 [49.998, 49.999]
T	7	128 [128, 128]	83.969 [83.947, 83.979]	50.005 [50.004, 50.008]
T	8	256 [256, 256]	87.443 [87.436, 87.461]	49.994 [49.994, 49.997]
T	9	512 [512, 512]	89.796 [89.794, 89.861]	50.004 [50.002, 50.009]
T	10	1,024 [1,024, 1,024]	99.659 [99.659, 99.664]	99.085 [99.076, 99.089]
T	11	1,608 [1,596, 1,615]	99.893 [99.891, 99.898]	99.697 [99.690, 99.728]
T	12	1,797 [1,795, 1,804]	99.926 [99.924, 99.929]	99.777 [99.767, 99.782]
T	13	1,827 [1,825, 1,830]	99.928 [99.927, 99.929]	99.777 [99.771, 99.786]
T	14	1,830 [1,827, 1,831]	99.929 [99.928, 99.929]	99.781 [99.774, 99.786]
T	15	1,830 [1,828, 1,831]	99.930 [99.929, 99.934]	99.780 [99.772, 99.785]
TC-2	1	4 [4, 4]	25.032 [25.025, 25.033]	25.000 [25.000, 25.000]
TC-2	2	8 [8, 8]	40.154 [39.996, 49.968]	37.501 [37.500, 50.000]
TC-2	3	16 [16, 16]	59.990 [59.988, 59.990]	50.001 [49.999, 50.001]
TC-2	4	32 [32, 32]	73.990 [73.929, 74.021]	50.000 [49.999, 50.001]
TC-2	5	64 [64, 64]	83.989 [83.967, 84.003]	50.005 [50.005, 50.005]
TC-2	6	128 [128, 128]	89.905 [89.902, 89.964]	49.999 [49.997, 50.006]
TC-2	7	254 [254, 254]	99.913 [99.913, 99.915]	99.834 [99.833, 99.837]
TC-2	8	456 [456, 462]	99.831 [99.819, 99.833]	99.596 [99.578, 99.616]
TC-2	9	891 [887, 892]	99.617 [99.617, 99.624]	98.871 [98.869, 98.924]
TC-2	10	1,617 [1,612, 1,632]	99.604 [99.575, 99.606]	98.827 [98.737, 98.848]
TC-2	11	1,892 [1,888, 1,896]	99.876 [99.870, 99.881]	99.650 [99.649, 99.680]
TC-2	12	1,871 [1,871, 1,877]	99.921 [99.921, 99.925]	99.771 [99.766, 99.774]
TC-2	13	1,838 [1,837, 1,843]	99.928 [99.927, 99.929]	99.774 [99.771, 99.790]
TC-2	14	1,838 [1,837, 1,839]	99.931 [99.928, 99.932]	99.778 [99.777, 99.784]
TC-2	15	1,838 [1,836, 1,839]	99.930 [99.929, 99.932]	99.783 [99.781, 99.788]
TC-4	1	8 [8, 8]	25.032 [25.025, 25.033]	25.000 [25.000, 25.000]
TC-4	2	16 [16, 16]	40.154 [39.996, 49.968]	37.501 [37.500, 50.000]
TC-4	3	32 [32, 32]	59.990 [59.988, 59.990]	50.001 [49.999, 50.001]
TC-4	4	64 [64, 64]	83.991 [83.960, 84.004]	50.002 [50.002, 50.003]
TC-4	5	128 [128, 128]	99.945 [99.943, 99.950]	99.881 [99.866, 99.896]
TC-4	6	249 [246, 249]	99.915 [99.914, 99.926]	99.837 [99.806, 99.848]
TC-4	7	459 [451, 465]	99.913 [99.912, 99.914]	99.825 [99.822, 99.826]
TC-4	8	852 [852, 863]	99.812 [99.802, 99.815]	99.563 [99.531, 99.568]
TC-4	9	1,610 [1,609, 1,610]	99.577 [99.576, 99.589]	98.744 [98.727, 98.787]
TC-4	10	2,570 [2,556, 2,593]	99.538 [99.528, 99.554]	98.648 [98.559, 98.672]
TC-4	11	2,188 [2,170, 2,193]	99.858 [99.854, 99.860]	99.610 [99.603, 99.619]
TC-4	12	1,894 [1,886, 1,896]	99.921 [99.919, 99.922]	99.764 [99.750, 99.769]
TC-4	13	1,854 [1,853, 1,859]	99.929 [99.926, 99.931]	99.774 [99.770, 99.779]
TC-4	14	1,854 [1,853, 1,855]	99.928 [99.928, 99.934]	99.772 [99.772, 99.787]
TC-4	15	1,854 [1,852, 1,855]	99.931 [99.930, 99.932]	99.775 [99.774, 99.790]
TC-7	1	14 [14, 14]	25.032 [25.025, 25.033]	25.000 [25.000, 25.000]
TC-7	2	28 [28, 28]	50.074 [49.974, 50.076]	50.000 [50.000, 50.000]
TC-7	3	56 [56, 56]	59.990 [59.988, 59.990]	50.001 [49.999, 50.001]
TC-7	4	112 [112, 112]	99.965 [99.962, 99.976]	99.946 [99.942, 99.946]
TC-7	5	220 [216, 221]	99.943 [99.942, 99.947]	99.865 [99.863, 99.872]
TC-7	6	417 [401, 422]	99.913 [99.908, 99.918]	99.829 [99.800, 99.833]
TC-7	7	732 [685, 743]	99.906 [99.900, 99.908]	99.817 [99.814, 99.820]
TC-7	8	1,380 [1,254, 1,448]	99.791 [99.776, 99.792]	99.496 [99.482, 99.545]
TC-7	9	2,533 [2,462, 2,545]	99.536 [99.532, 99.559]	98.639 [98.616, 98.708]
TC-7	10	3,539 [3,403, 3,542]	99.508 [99.475, 99.510]	98.548 [98.397, 98.549]
TC-7	11	2,258 [2,252, 2,284]	99.854 [99.850, 99.862]	99.593 [99.577, 99.595]
TC-7	12	1,916 [1,910, 1,920]	99.920 [99.918, 99.923]	99.758 [99.750, 99.773]
TC-7	13	1,878 [1,877, 1,883]	99.929 [99.927, 99.934]	99.772 [99.768, 99.797]
TC-7	14	1,878 [1,877, 1,879]	99.930 [99.928, 99.932]	99.788 [99.776, 99.789]
TC-7	15	1,878 [1,876, 1,879]	99.931 [99.930, 99.933]	99.789 [99.777, 99.793]
RF-9	1	18 [18, 18]	50.042 [30.020, 59.846]	49.994 [25.006, 50.000]
RF-9	2	36 [36, 36]	59.987 [59.849, 59.987]	49.999 [49.999, 50.000]
RF-9	3	72 [71, 72]	60.025 [59.987, 60.129]	50.001 [49.998, 50.011]
RF-9	4	141 [141, 143]	59.746 [59.713, 59.966]	50.001 [49.999, 50.011]

Continued on next page

Table 8 continued

Model	d	Total leaves	Test accuracy (%)	Support-uniform accuracy (%)
RF-9	5	276 [271, 280]	61.534 [61.532, 63.257]	50.013 [49.989, 50.021]
RF-9	6	535 [523, 542]	67.864 [66.566, 70.195]	50.052 [49.824, 50.124]
RF-9	7	1,029 [1,018, 1,046]	72.053 [70.440, 72.231]	50.133 [49.909, 50.142]
RF-9	8	1,788 [1,724, 1,907]	75.551 [75.118, 75.720]	50.197 [50.109, 50.214]
RF-9	9	3,216 [3,068, 3,246]	77.841 [76.443, 78.223]	50.234 [50.172, 50.247]
RF-9	10	5,899 [5,727, 6,111]	81.765 [81.425, 83.244]	51.597 [51.106, 51.660]
RF-9	11	9,794 [9,640, 11,057]	85.824 [85.813, 86.539]	56.296 [55.148, 57.161]
RF-9	12	16,106 [15,304, 17,329]	87.450 [86.337, 87.761]	59.832 [57.817, 59.857]
RF-9	13	26,285 [24,752, 27,346]	90.399 [90.168, 90.753]	67.158 [65.690, 67.621]
RF-9	14	37,428 [36,838, 37,499]	91.978 [91.478, 92.467]	70.755 [69.358, 72.222]
RF-9	15	46,830 [46,188, 53,803]	92.224 [91.603, 93.377]	71.597 [70.851, 74.763]
RF-19	1	38 [38, 38]	59.989 [59.985, 59.990]	49.999 [49.997, 50.000]
RF-19	2	76 [76, 76]	59.990 [59.987, 59.991]	49.999 [49.999, 50.000]
RF-19	3	151 [150, 152]	59.989 [59.919, 60.015]	50.001 [50.000, 50.002]
RF-19	4	295 [294, 297]	59.798 [59.525, 59.924]	50.005 [49.998, 50.011]
RF-19	5	580 [577, 583]	63.047 [61.082, 64.510]	50.069 [50.031, 50.080]
RF-19	6	1,085 [1,080, 1,134]	68.375 [68.243, 71.046]	49.911 [49.903, 49.942]
RF-19	7	2,103 [2,101, 2,107]	72.211 [71.897, 73.448]	50.000 [49.940, 50.066]
RF-19	8	3,951 [3,921, 4,019]	77.654 [76.863, 77.678]	50.148 [50.131, 50.314]
RF-19	9	7,145 [6,540, 7,253]	79.982 [77.494, 80.769]	49.821 [49.801, 49.992]
RF-19	10	12,403 [12,162, 12,527]	82.943 [82.416, 83.252]	51.193 [51.178, 51.327]
RF-19	11	22,072 [21,009, 22,661]	86.574 [85.768, 86.918]	56.312 [54.956, 57.000]
RF-19	12	35,281 [35,096, 36,668]	89.653 [89.266, 89.713]	63.256 [62.900, 63.461]
RF-19	13	51,074 [49,468, 55,183]	90.868 [90.550, 91.892]	67.114 [66.887, 70.203]
RF-19	14	76,690 [71,372, 77,931]	92.371 [92.227, 93.023]	71.607 [71.494, 73.535]
RF-19	15	104,700 [102,068, 109,267]	95.430 [95.330, 95.515]	81.674 [81.231, 81.995]
RF-29	1	58 [58, 58]	59.989 [59.986, 59.990]	50.000 [49.999, 50.001]
RF-29	2	116 [116, 116]	59.989 [59.988, 59.990]	50.000 [49.998, 50.001]
RF-29	3	229 [229, 230]	59.989 [59.988, 59.990]	50.000 [50.000, 50.000]
RF-29	4	456 [453, 460]	59.993 [59.981, 60.121]	50.011 [50.003, 50.013]
RF-29	5	899 [895, 901]	62.550 [60.043, 63.158]	49.979 [49.946, 50.013]
RF-29	6	1,692 [1,692, 1,736]	68.698 [67.804, 68.876]	50.035 [50.032, 50.258]
RF-29	7	3,206 [3,156, 3,276]	71.198 [70.633, 71.923]	49.935 [49.904, 49.956]
RF-29	8	5,959 [5,940, 6,241]	76.269 [75.908, 76.347]	50.172 [50.155, 50.206]
RF-29	9	9,953 [9,854, 10,333]	79.520 [78.753, 80.299]	50.030 [49.877, 50.336]
RF-29	10	18,615 [18,405, 19,740]	83.193 [83.142, 83.332]	52.046 [51.346, 52.135]
RF-29	11	32,380 [31,155, 32,598]	87.271 [86.878, 87.739]	57.337 [56.601, 59.222]
RF-29	12	50,832 [50,580, 55,573]	89.437 [89.224, 90.682]	63.554 [62.433, 65.849]
RF-29	13	80,811 [80,551, 84,423]	93.426 [93.086, 93.435]	73.686 [73.042, 74.163]
RF-29	14	119,572 [115,579, 122,178]	94.739 [94.731, 94.748]	78.784 [78.203, 78.929]
RF-29	15	164,060 [163,521, 168,361]	95.819 [95.671, 95.982]	82.710 [82.528, 83.917]

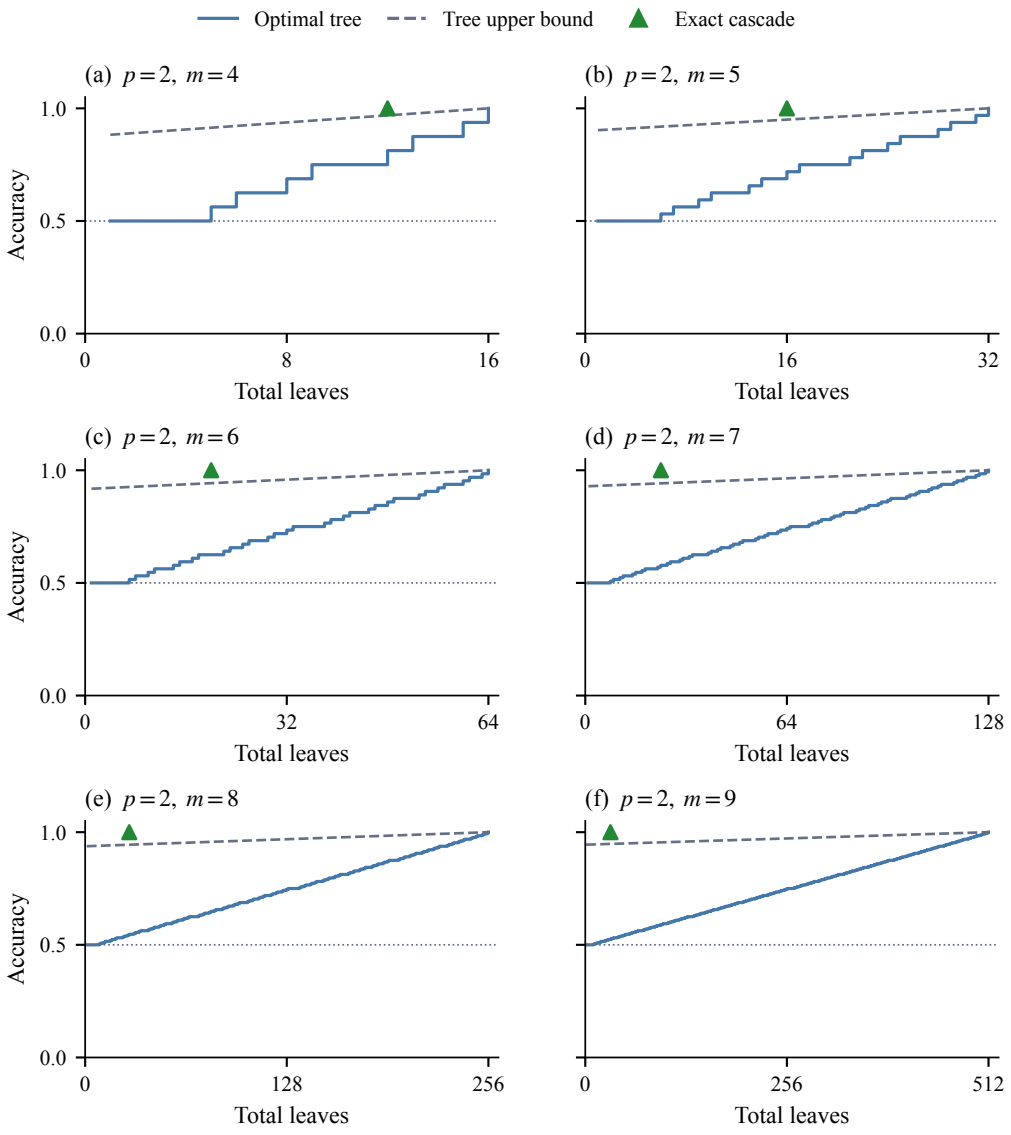


Figure 6: Exact finite-domain results for $p = 2$ and $m = 4, \ldots, 9$. Solid steps show optimal single-tree accuracy; dashed curves show the upper bound from the first part of Theorem 6. Green triangles mark the exact cascade at $(m-1)p^2$ leaves, and dotted lines indicate chance accuracy $1/p$. All results are deterministic and evaluated uniformly on the full grid.

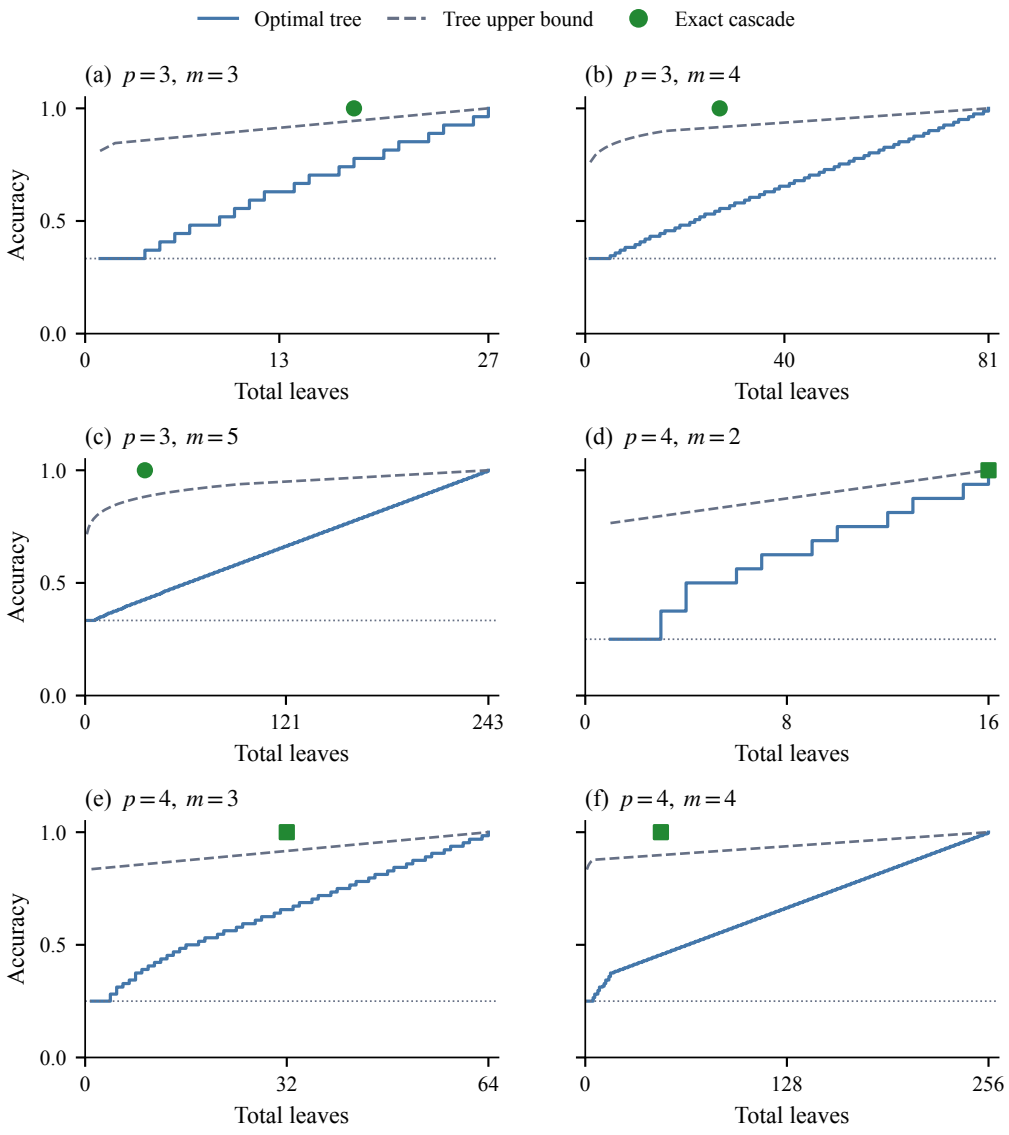


Figure 7: Exact finite-domain results for $p \in \{3, 4\}$. Solid steps show optimal single-tree accuracy, and dashed curves show the smaller of the two theoretical accuracy upper bounds. Green circles ($p = 3$) and squares ($p = 4$) mark exact cascades; dotted lines indicate $1/p$. Each panel uses the uniform distribution on its full input grid.

Table 9: Per-seed threshold selections underlying Table 2. A dash means no tested depth reached the threshold.

		Test accuracy $\geq 95\%$		Test accuracy $\geq 99\%$	
Model	Seed	Depth	Leaves	Depth	Leaves
T	0	10	1,024	10	1,024
T	1	10	1,024	10	1,024
T	2	10	1,024	10	1,024
T	3	10	1,024	10	1,024
T	4	10	1,024	10	1,024
TC-2	0	7	256	7	256
TC-2	1	7	253	7	253
TC-2	2	7	254	7	254
TC-2	3	7	254	7	254
TC-2	4	7	254	7	254
TC-4	0	5	128	5	128
TC-4	1	5	128	5	128
TC-4	2	5	128	5	128
TC-4	3	5	128	5	128
TC-4	4	5	128	5	128
TC-7	0	4	112	4	112
TC-7	1	4	112	4	112
TC-7	2	4	112	4	112
TC-7	3	4	112	4	112
TC-7	4	4	112	4	112
RF-9	0	—	—	—	—
RF-9	1	—	—	—	—
RF-9	2	—	—	—	—
RF-9	3	—	—	—	—
RF-9	4	—	—	—	—
RF-19	0	14	82,775	—	—
RF-19	1	15	101,469	—	—
RF-19	2	15	104,700	—	—
RF-19	3	15	102,068	—	—
RF-19	4	—	—	—	—
RF-29	0	15	168,361	—	—
RF-29	1	14	122,178	—	—
RF-29	2	15	163,521	—	—
RF-29	3	15	164,060	—	—
RF-29	4	15	177,281	—	—

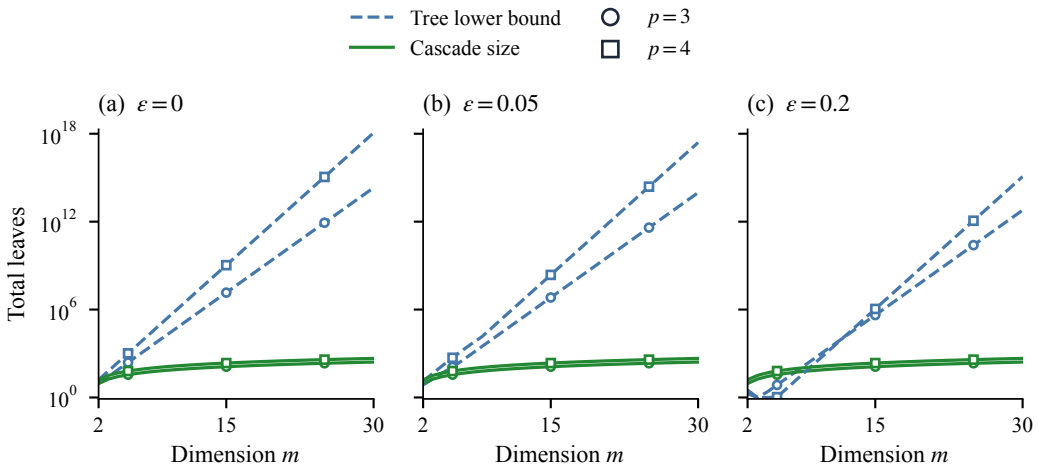


Figure 8: Dimension dependence at additional error tolerances. Blue dashed curves show the strongest applicable single-tree lower bound, including one leaf; green solid curves show the zero-error cascade construction size. Circles and squares distinguish $p = 3$ and $p = 4$. All panels share a logarithmic leaf-count axis.