Free Lunch in the Forest: Functionally-Identical Pruning of Boosted Tree Ensembles

Youssouf Emine[1,2], **Alexandre Forel**[1, 3], **Idriss Malek**[1, 3], **Thibaut Vidal**[1, 3]
[1]Department of Mathematics and Industrial Engineering, Polytechnique Montréal
[2]Canada Excellence Research Chair in Data-Science for Real-time Decision-Making (CERC)
[3]CIRRELT & SCALE-AI Chair in Data-Driven Supply Chains
{youssouf.emine, alexandre.forel, thibaut.vidal}@polymtl.ca, idriss.malek@polytechnique.edu

Abstract

Tree ensembles, including boosting methods, are highly effective and widely used for tabular data. However, large ensembles lack interpretability and require longer inference times. We introduce a method to prune a tree ensemble into a reduced version that is "functionally identical" to the original model. In other words, our method guarantees that the prediction function stays unchanged for any possible input. As a consequence, this pruning algorithm is lossless for any aggregated metric. We formalize the problem of functionally identical pruning on ensembles, introduce an exact optimization model, and provide a fast yet highly effective method to prune large ensembles. Our algorithm iteratively prunes considering a finite set of points, which is incrementally augmented using an adversarial model. In multiple computational experiments, we show that our approach is a "free lunch", significantly reducing the ensemble size without altering the model's behavior. Thus, we can preserve state-of-the-art performance at a fraction of the original model's size.

Package — www.github.com/eminyous/fipe
Code — www.github.com/eminyous/fipe-experiments
Paper with appendices — arxiv.org/abs/2408.16167

1 Introduction

Ensembles remain one of the most effective and commonly utilized machine learning models. Among them, tree ensembles such as random forests and boosting are known to achieve state-of-the-art performance on tabular data (Grinsztajn, Oyallon, and Varoquaux 2022; McElfresh et al. 2024). They also offer greater interpretability compared to large neural networks. Theory and practice indicate that superior performance is achieved with large ensembles, i.e., forests with many trees (Biau and Scornet 2016; Probst and Boulesteix 2017; Buschjäger and Morik 2021). However, large ensembles lead to large memory requirements and inference times, which can quickly become a bottleneck when embedding models into hardware such as microcontrollers or smartphones. Further, large ensemble models lack interpretability due to the complex interaction of their components.

Ensemble pruning denotes the process of reducing the size of a trained ensemble model. This can be understood as removing nodes from the trees or removing trees entirely from the forest. Early approaches studied how to prune random forests made of shallow trees, i.e., trees that were themselves pruned beforehand (see, e.g., Margineantu and Dietterich 1997; Martínez-Muñoz and Suárez 2006). These works showed that pruning could improve the test error and out-of-sample generalization. However, this is not true with boosted ensembles, since the trees are typically already shallow. Hence, most recent studies seek a trade-off between the amount of pruning and the impact on test error (Liu and Mazumder 2023; Amee et al. 2023). This trade-off may be profitable in some situations but there is no guarantee that the model will perform well on new data.

In this paper, we completely avoid this trade-off by pruning ensemble models *without any change in their behavior*. We call this a "functionally-identical" pruning since the prediction functions of the pruned ensemble and the original one are identical. This has two main advantages. First, as demonstrated in this paper, the resulting optimization problems can be very efficiently solved. Second, such pruned models give a "free lunch": they achieve the exact same performance as the original model at a fraction of its size. This provides multiple advantages regarding memory, inference time, and interpretability. An example ensemble that can be pruned without any change in its prediction function is shown in Figure 1.

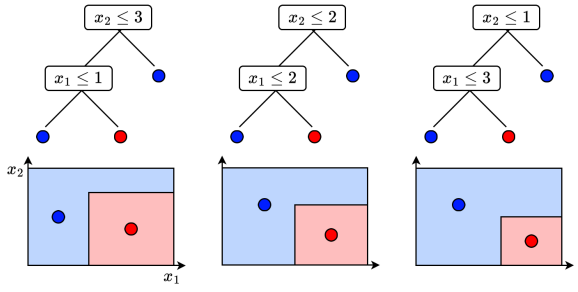

Figure 1: A small ensemble made of three trees with equal weights. This ensemble can be pruned without any change in its prediction function by removing the first and third trees.

This paper introduces `FIPE`, a general algorithm for

Functionally **I**dentical **P**runing of **E**nsembles. FIPE iterates between a *pruning model* and a *separation oracle*. The pruning model identifies candidate pruned ensembles on a finite set of points. The separation oracle checks whether the pruned model is functionally identical to the original model on the entire feature space. If this is not the case, new points are added to the pruning set. Our iterative algorithm is such that the final pruned ensemble is certifiably equivalent to the original one on the entire feature space.

We make the following contributions:

(i) We formalize the problem of functionally identical pruning of tree ensembles and characterize its computational complexity.

(ii) We present FIPE, our iterative algorithm and its two main components: the pruner, and the oracle. We prove that the algorithm terminates in a finite number of steps.

(iii) We investigate two variants of the pruning model: an exact combinatorial optimization model based on the $\|\cdot\|_0$ norm that guarantees that the pruned model is of minimal size, and a fast approximate algorithm based on the $\|\cdot\|_1$ norm that is not necessary of minimal size but scales to large datasets. We further provide an efficient pruning oracle inspired by recent works on counterfactual explanations.

(iv) We apply FIPE on boosted ensembles AdaBoost, LightGBM and XGBoost as well as random forests. Our extensive experiments demonstrate that significantly smaller ensembles can be identified across various real-world datasets. A central finding of this study is identifying that many base learners of boosted ensembles are superfluous. Consequently, an integrated pruning and reweighting approach can substantially reduce their size while maintaining their predictive performance.

2 Problem Statement

Let $\{(x_i, y_i)\}_{i=1}^n$ be a labeled set of observations, where $x \in \mathcal{X} \subseteq \mathbb{R}^p$ is a feature vector. We focus on classification problems with C classes. Denote by $[\![a, b]\!]$ the set of integers between a and b. Any class y belongs to $[\![1, C]\!]$.

2.1 Classification ensembles

A classification ensemble is a weighted set of classifiers $\{(h_m, \alpha_m)\}_{m=1}^M$ where each classifier $h_m : \mathcal{X} \to [0,1]^C$ provides a score vector for any input x, and α_m is the weight of classifier m. The ensemble prediction function is a map $H : \mathcal{X} \times \mathbb{R}^M_{\geq 0} \to [\![1, C]\!]$ that assigns a class to any input x using a voting scheme, expressed generally as:

$$H(x; \alpha) = \underset{c \in [\![1,C]\!]}{\operatorname{argmax}} \sum_{m=1}^{M} \alpha_m h_m^{(c)}(x), \qquad (1)$$

where $h_m^{(c)}(x)$ is the score predicted by classifier of index m for class c. Any deterministic tie-breaking rule can be used to break the ties in Equation (1) if the argmax is set-valued.

The computation of the score of a tree depends on the nature of the ensemble. For instance, AdaBoost (Hastie et al. 2009) and the original random forest algorithm of Breiman (2001) use the so-called hard-voting criterion in which each classifier prediction $h_m^{(c)}(x) \in \{0, 1\}$ is binary.

2.2 Functionally-identical pruning

We study how to prune classification ensembles while guaranteeing that the pruned model is functionally identical to the original model. This is formalized in the following definition.

Definition 2.1. A pruned ensemble with weights w is *functionally identical* to the original ensemble on the entire feature space $\mathcal{X}$ if $H(x; w) = H(x; \alpha)$ for all $x \in \mathcal{X}$.

The pruned ensemble with minimum size is obtained by solving:

$$\min_{w \geq 0} \ \|w\|_0 \ : \ H(x; w) = H(x; \alpha), \ \forall x \in \mathcal{X}. \qquad (2)$$

Since any learner with a zero weight is effectively pruned, Problem (2) minimizes the number of active learners. The minimizer of Problem (2) is not only certifiably functionally identical, but it is also of minimal size. That is, it is guaranteed to be the smallest reweighted ensemble that is functionally identical to the original one.

The constraint in Problem (2) ensures that the pruned model is functionally identical to the original model on the entire space. Functionally-identical pruning is also known as "faithful" pruning in Vidal, Pacheco, and Schiffer (2020). We use equivalently both words in the rest of the paper.

Proposition 2.2. *For additive tree ensembles, i.e., a broad class including random forests and boosting methods, Problem* (2) *is NP-hard.*

The proof is given in Appendix A (see extended version). Proposition 2.2 states that optimal faithful pruning is a difficult task in general. Nevertheless, this paper shows that Problem (2) can be solved efficiently for large ensembles on real-world datasets, and provides effective heuristics otherwise. Our algorithms are presented in the next section.

3 Pruning Algorithms

To solve the faithful pruning problem, FIPE iterates between solving a *pruning problem* on a finite set of points and a *separation oracle*. This is illustrated in Figure 2.

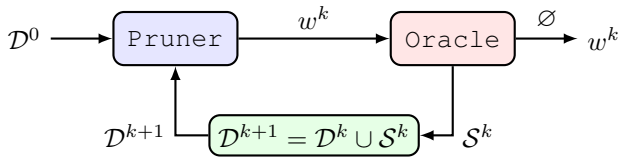

Figure 2: FIPE iterates between the pruning model and the separation oracle until it returns the set of weights of the pruned model.

FIPE is thus made of two essential components: (i) a faithful pruner Pruner that returns the set of weights $\{w_m\}_{m=1}^M$ of the pruned ensemble faithful to the original ensemble on a given finite set of points $\mathcal{D} \subset \mathcal{X}$, and (ii) a separation oracle Oracle that returns a set of point $\mathcal{S} \subset \mathcal{X}$ for which the pruned ensemble with weights w and the original ensemble with weights α differ. If the oracle cannot find

Algorithm 1: Faithful pruning algorithm

$\mathcal{D}^0 \leftarrow$ An arbitrary finite set of points of $\mathcal{X}$
$w^0 \leftarrow \texttt{Pruner}\left(\mathcal{D}^0\right)$
$k \leftarrow 0$
while $\texttt{Oracle}\left(w^k\right)$ is not $\varnothing$ **do**
  $\mathcal{S}^k \leftarrow \texttt{Oracle}\left(w^k\right)$
  $\mathcal{D}^{k+1} \leftarrow \mathcal{D}^k \cup \mathcal{S}^k$
  $w^{k+1} \leftarrow \texttt{Pruner}\left(\mathcal{D}^{k+1}\right)$
  $k \leftarrow k+1$
end while
return w^k

a separating point, the two ensembles are functionally identical over the entire feature space. The algorithm then returns the set of weights of the pruned ensemble. This algorithm is presented in Algorithm 1.

Theorem 3.1. *For tree ensembles,* FIPE *terminates after a finite number of calls to the* Oracle.

The proof is given in Appendix A (see extended version).

3.1 Pruning on a finite set of points

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ a finite set of points $\mathcal{D} \subset \mathcal{X}$. The pruning model returns the set of weights of the pruned ensemble faithful to the original ensemble on $\mathcal{D}$.

Denote by $c_i = H(x_i, \alpha)$ the class of the point x_i given by the original ensemble with weights α. Note that the classes c_i are not necessarily equal to the original classes y_i depending on the training of the ensemble H. Pruning on a finite set of points is a special case of Problem (2) when $\mathcal{X}$ is finite. The constraint of Problem (2) can be reformulated to ensure that the pruned ensemble is faithful to the original ensemble on $\mathcal{D}$. It can be written as:

$$\sum_{m=1}^M w_m \left(h_m^{(c_i)}(x_i) - h_m^{(c)}(x_i)\right) > 0. \tag{3}$$

It ensures that the pruned ensemble gives a higher weighted score for the class c_i than for the class c for each point x_i and each class $c \neq c_i$.

Equation (3) is difficult to handle numerically since it involves a strict inequality. However, using the same idea as in a support vector machine, we express it equivalently as:

$$\sum_{m=1}^M w_m \left(h_m^{(c_i)}(x_i) - h_m^{(c)}(x_i)\right) \geq 1. \tag{4}$$

Minimal-size faithful pruner. The faithful pruning problem on a finite set of points is a combinatorial optimization problem given by:

$$\min_{w \geq 0,\, u \in \{0,1\}^M} \sum_{m=1}^M u_m \tag{5a}$$

$$w_m \leq W u_m,\ \forall m \in [\![1, M]\!], \tag{5b}$$

$$\sum_{m=1}^M w_m \left(h_m^{(c_i)}(x_i) - h_m^{(c)}(x_i)\right) \geq 1, \tag{5c}$$
$$\forall i \in [\![1, N]\!],\ \forall c \in [\![1, C]\!],\ c \neq c_i.$$

This problem minimizes the $\|\cdot\|_0$ norm of the weights. It uses the binary variables u_m to indicate whether the estimator m is active or not, and a parameter W, which is an upper bound on the weights of the active estimators. A minimizer of Problem (5) is both faithful to the original ensemble on $\mathcal{D}$ and of certifiably minimal size according to the $\|\cdot\|_0$ norm.

Efficient approximation. While Problem (5) ensures finding the sparsest model, it might be expensive to solve numerically since its objective function with $\|\cdot\|_0$ is hard to linearize. Hence, we propose an efficient approximation by replacing the $\|\cdot\|_0$ norm with the $\|\cdot\|_1$ norm. This problem is a linear programming problem given by:

$$\min_{w \geq 0} \|w\|_1 = \sum_{m=1}^M w_m \tag{6a}$$

$$\sum_{m=1}^M w_m \left(h_m^{(c_i)}(x_i) - h_m^{(c)}(x_i)\right) \geq 1, \tag{6b}$$
$$\forall i \in [\![1, N]\!],\ \forall c \in [\![1, C]\!],\ c \neq c_i.$$

Problem (6) can be solved very efficiently using standard linear programming solvers. The faithfulness constraint of the problem is unchanged so that any solution of Problem (6) is guaranteed to remain faithful to the original ensemble on $\mathcal{D}$.

3.2 Separation oracle for tree ensembles

This section presents the separation oracle used in FIPE. The goal of the oracle is twofold: (i) it certifies that a pruned model is functionally equivalent to the original model on the entire feature space, or (ii) it provides a set of points for which the two models differ. These two goals can be achieved jointly by formulating an optimization model that maximizes the misclassification between the pruned and original model over the feature space.

Let c and y be two fixed classes. The separation problem over these two classes can be formulated as:

$$\max_{x \in \mathcal{X}} \sum_{m=1}^M w_m \left(h_m^{(c)}(x) - h_m^{(y)}(x)\right) \ :\ y = H(x, \alpha). \tag{7}$$

This problem maximizes the difference between the score of class c and class y according to the pruned ensemble. The constraint of Problem (7) ensures that the original ensemble predicts class y for the point x. If the value of Problem (7) is positive, the pruned ensemble predicts class c whereas the original ensemble predicts y. Hence, a separating point is such that it has a positive value according to Problem (7).

Remark 3.2. Since Problem (7) maximizes the difference in prediction between two ensembles, it can be interpreted as a variation of the adversarial example or counterfactual explanation problem.

If the objective function is non-positive across all points in the feature space, both models are certified to be equivalent on the entire feature space. In practice, we solve the separation problem for each pair of classes c and y. Depending on the algorithm used to solve Problem (7), several points may be found that have a positive score difference. In FIPE, we add all points with a positive score difference $\mathcal{S}$. When $\mathcal{S}$ is empty, the algorithm terminates.

The complexity of Problem (7) lies in representing the classification function of the trees with linear variables and constraints. To ensure that the original ensemble predicts

class y for point x, we can replace the constraint of Problem (7) by the following inequality:

$$\sum_{m=1}^{M} \alpha_m \left(h_m^{(y)}(x) - h_m^{(y')}(x) \right) > 0, \forall y' \neq y. \quad (8)$$

These constraints ensure that the original ensemble gives a higher score for class y than for any other class $y' \neq y$, making y the predicted class for point x by the original ensemble. Again, this constraint is ill defined because of the strict inequality. By introducing a sufficiently small $\varepsilon > 0$, the separation problem can be equivalently formulated as:

$$\max_{x \in \mathcal{X}} \sum_{m=1}^{M} w_m \left(h_m^{(c)}(x) - h_m^{(y)}(x) \right) \quad (9a)$$

$$\sum_{m=1}^{M} \alpha_m \left(h_m^{(y)}(x) - h_m^{(y')}(x) \right) \geq \varepsilon, \forall y' \neq y. \quad (9b)$$

Separation for tree ensembles. Problem (9) is nonlinear in general. However, for additive tree ensembles, we can linearize it efficiently using a procedure similar to the one used in Parmentier and Vidal (2021). This consists in introducing a set of variables and constraints to track the path of the separating point x in all trees of the ensembles.

For each tree m, let $\mathcal{V}_m$ be the set of its internal nodes, $\mathcal{L}_m$ the set of its leaves, and d_m its maximum depth. For each node $v \in \mathcal{V}_m$, let $\mathbf{left}(v)$ and $\mathbf{right}(v)$ be its left and right children respectively, and let $\mathbf{depth}(v)$ be its depth in the tree m. Let $\mathbf{root}(m)$ denote the root of tree m. Let $\mathrm{f}_{m,v}$ be a binary variable indicating if node $v \in \mathcal{V}_m \cup \mathcal{L}_m$ is in the path of the point x in the tree m.

To linearize the tree prediction functions, we introduce a binary variable $\lambda_{m,d}$ to indicate whether point x goes to the left child at depth d of the tree m. The path consistency constraints in the tree m can then be expressed as:

$$\mathrm{f}_{m,\mathbf{root}(m)} = 1, \quad (10a)$$

$$\mathrm{f}_{m,\mathbf{left}(v)} + \mathrm{f}_{m,\mathbf{right}(v)} = \mathrm{f}_{m,v}, \ \forall v \in \mathcal{V}_m, \quad (10b)$$

$$\sum_{v \in \mathcal{V}_m : \mathbf{depth}(v) = d} \mathrm{f}_{m,\mathbf{left}(v)} = \lambda_{m,d}, \forall d \in [\![0, d_m]\!]. \quad (10c)$$

Constraint (10a) ensures that the root of tree m is in the path of point x. Constraint (10b) ensures that one of the children of node v is in the path of point x if node v is in the path. It also ensures that the path of point x in tree m at depth d goes to the left child if $\lambda_{m,d} = 1$. Using these variables, we can formulate Problem (9) equivalently as:

$$\max_{\mathrm{f} \geq 0} \sum_{m=1}^{M} \sum_{v \in \mathcal{L}_m} w_m \left(h_m^{(c)}(x) - h_m^{(y)}(x) \right) \mathrm{f}_{m,v} \quad (11)$$

$$\sum_{m=1}^{M} \sum_{v \in \mathcal{L}_m} \alpha_m \left(h_m^{(y)}(x) - h_m^{(y')}(x) \right) \mathrm{f}_{m,v} \geq \varepsilon, \forall y' \neq y.$$

Remark 3.3. Problem (11) is not optimizing over the variable x anymore, but only over the variables that track the path over the trees. Indeed, for tree ensembles, the prediction function is piecewise-constant. Hence, it is sufficient to identify the set of leaves to separate the pruned and original ensembles. A separating point is directly determined by taking the center of the cell defined by the leaf variables.

Further, observe that the binary variables $\mathrm{f}_{m,v}$ can be defined as continuous in $[0, 1]$, as they will be forced to binary values due to Constraints (10a)–(10c).

Feature consistency. To ensure the value of x is consistent across the trees, we need to add feature consistency variables and constraints. Each feature type has to be handled separately. We explain below how to ensure the consistency of continuous features, and in Appendix A (see extended version) how to handle categorical and binary features.

Let j be a continuous feature and let $\{t_{j,1}, t_{j,2}, \ldots, t_{j,R_j}\}$ be the set of thresholds that the nodes of the original ensemble are splitting on. We consider without loss of generality that this set is sorted. For each continuous feature j and $r \in [\![1, R_j]\!]$, we define the continuous auxiliary variables $\mu_{j,r} \in [0, 1]$. These variables indicate whether the new point is on the left or right-side of a level. They are constrained such that $\mu_{j,r} = 0 \Leftrightarrow x_j \in]-\infty, t_{j,r}]$ by imposing:

$$\mu_{j,r} \geq \mu_{j,r+1}, \forall r \in [\![1, R_j - 1]\!] \quad (12)$$

For each tree m, let $\mathcal{V}_m^{j,r}$ be the set of nodes that split on feature j at threshold $t_{j,r}$ for $r \in [\![1, R_j]\!]$. The following constraints ensure that the value of $\mu_{j,r}$ is consistent across the nodes of tree m:

$$\mathrm{f}_{m,\mathbf{left}(v)} \leq 1 - \mu_{j,r}, \forall r \in [\![1, R_j]\!], \forall v \in \mathcal{V}_m^{j,r}, \quad (13a)$$

$$\mathrm{f}_{m,\mathbf{right}(v)} \leq \mu_{j,r}, \quad \forall r \in [\![1, R_j]\!], \forall v \in \mathcal{V}_m^{j,r}. \quad (13b)$$

Constraint (13a) ensures that if the value of x_j is in the interval $]t_{j,r}, \infty[$, then at node v we cannot go to the left child. Similarly, constraint (13b) ensures that if the value of x_j is in the interval $]-\infty, t_{j,r}]$, then at node v we can not go to the right child.

Our separation oracle resembles closely the problem of finding counterfactual explanations (Parmentier and Vidal 2021) or classifier evasions (Kantchelian, Tygar, and Joseph 2016). Yet, optimizing over the set of leaves (see Remark 3.3) instead of the continuous variable x offers numerical benefits. Determining the exact value of x, as needed in counterfactual explanation, requires checking for strict inequalities when going right on a threshold.

4 Computational Experiments

We now study the value of `FIPE` for pruning large training ensembles while maintaining faithfulness. We investigate how much `FIPE` can prune ensembles and its scalability over several datasets. We analyze the behavior of `FIPE` in terms of what trees are removed, and how pruning is impacted by the size of the original ensemble. Finally, we compare our approach to baselines from the recent literature.

In this section, we focus our experiments on `AdaBoost` classifiers. Experiments with other boosted ensembles, such as `XGBoost` and `LightGBM`, are provided in Appendix B (see extended version) with essentially the same conclusions. We also study random forests in this appendix. We perform our analyses across 11 datasets commonly used in previous studies. In each experiment, we split the data set into a training dataset (80%) and a test dataset (20%) to measure faithfulness and test accuracy. This process is repeated

Dataset	FIPE$-\|\cdot\|_0$ $\|w\|_0$	K	FI	Time (s)	FIPE$-\|\cdot\|_1$ $\|w\|_0$	K	FI	Time (s)	ACC
Breast-Cancer	20.2	32	100%	166	20.2	25	100%	13	95.33%
COMPAS	12.6	16	100%	72	12.6	16	100%	14	66.66%
ELEC2	12.4	2	100%	3	12.4	2	100%	7	77.30%
FICO	16.2	22	100%	456	16.2	19	100%	30	71.42%
HTRU2	9.4	18	100%	25	9.4	13	100%	26	97.75%
House-16H	34.8	42	100%	643	34.8	40	100%	235	85.57%
Ionosphere	26.6	57	100%	4 475	26.6	47	100%	132	90.70%
Pima-Diabetes	25.0	28	100%	189	25.0	24	100%	16	77.92%
PoL	15.6	6	100%	6	15.6	6	100%	8	90.90%
Seeds	8.8	6	100%	5	9.8	4	100%	4	89.05%
Spambase	26.4	48	100%	765	26.4	43	100%	118	93.38%

Table 1: Comparison of FIPE$-\|\cdot\|_0$ and FIPE$-\|\cdot\|_1$ for pruning AdaBoost ensembles with 50 learners. The table shows the number of learners in the pruned ensemble $\|w\|_0$, test faithfulness to the original ensemble FI, test accuracy ACC, and rounded number of oracle calls K. All results are averaged over five repetitions with different train/test splits.

over five different random seeds. The characteristics of the datasets are presented in Table 2: their number of samples n, number of features p including the number of numerical p_N and binary p_B features, and number of classes C.

Dataset	n	p	p_N	p_B	C
Seeds	210	7	7	0	3
Ionosphere	351	33	32	1	2
Breast-Cancer-Wisconsin	683	9	9	0	2
Pima-Diabetes	768	8	8	0	2
Spambase	4 601	57	57	0	2
COMPAS-ProPublica	6 907	12	0	12	2
PoL	10 082	26	26	0	2
FICO	10 459	17	0	17	2
House-16H	13 488	16	16	0	2
HTRU2	17 898	8	8	0	2
ELEC2	38 474	7	7	0	2

Table 2: Characteristics of the data sets

All experiments are implemented in python. We used scikit-learn for training the ensembles. All optimization problems are solved to global optimality using the commercial solver Gurobi v11.0. The experiments are conducted on a computing grid. Each experiment utilizes a single core of an Intel Xeon Gold 6258R CPU running at 2.7 GHz and is provided with 4 GB RAM.

4.1 Main results

First, we evaluate how much FIPE is able to prune tree ensembles while certifying faithfulness to the original model. We compare both the certifiable minimal-size and fast but approximate, respectively denoted by FIPE$-\|\cdot\|_0$ and FIPE$-\|\cdot\|_1$. We measure the number of learners in the pruned ensemble, the faithfulness to the original ensemble on the test set denoted FI, the test accuracy ACC, and the number of oracle calls rounded to its upper integer denoted by K. All performance metrics are average over the five repetitions.

The results are presented in Table 1 for ensembles of 50 learners. They show that FIPE is consistently able to prune AdaBoost ensembles while maintaining faithfulness. Further, we observe that FIPE$-\|\cdot\|_1$ achieves the same pruned size as FIPE$-\|\cdot\|_0$ for all but one dataset, while having significantly faster pruning times.

Result 1. Using $\|\cdot\|_1$ in FIPE typically yields pruning performance similar to $\|\cdot\|_0$ but with significantly faster runtimes.

Because FIPE$-\|\cdot\|_0$ struggles to scale to large datasets when the ensemble size increases, we restrict our study to FIPE$-\|\cdot\|_1$ and investigate its scalability to large ensembles. Table 3 shows that FIPE$-\|\cdot\|_1$ is consistently able to reduce the number of active estimators while maintaining the fidelity of the original ensemble. Further, the results show that it scales well to large datasets and ensemble sizes. Another remarkable result shown by Table 3 is that the number of oracle calls K is relatively small, even for large ensembles. Indeed, it is possible to certify that the pruned model is functionally identical to the original one on the entire feature space with less than 200 calls.

Result 2. FIPE$-\|\cdot\|_1$ consistently provides small pruned ensembles even when the dataset and original ensemble are large.

4.2 Analysis of the pruned ensembles

Our previous experiment shows that close to 90% of the base learners of AdaBoost ensembles are superfluous. That is, these learners can be effectively pruned without any change in the prediction function, if the remaining learners are reweighted adequately. We now study how FIPE is able to prune by jointly removing and reweighting the base learners. Figure 3 shows the weights of the active estimators in the original and pruned ensembles. It shows that some learners that seem to have a low predictive power in the original ensemble because they are created later in the training process (Figure 3 left) in fact have a large predictive power after pruning and reweighting (Figure 3 right). For instance, tree index 180 has a weight of around 30 in the original ensemble and a weight of around 1300 in the pruned model. Hence,

Dataset	$M = 100$					$M = 200$				
	$\|w\|_0$	K	FI	ACC	Time (s)	$\|w\|_0$	K	FI	ACC	Time (s)
Breast-Cancer	26.4	34	100%	95.33%	37	31.2	42	100%	95.47%	110
COMPAS	12.8	16	100%	66.70%	39	13.0	17	100%	66.73%	99
ELEC2	20.2	6	100%	79.77%	120	39.6	32	100%	80.40%	1 272
FICO	18.0	24	100%	71.54%	107	18.0	24	100%	71.61%	208
HTRU2	29.0	34	100%	97.77%	267	39.6	46	100%	97.78%	752
House-16H	55.4	66	100%	86.29%	1 520	86.0	116	100%	86.86%	6 114
Ionosphere	41.6	66	100%	91.55%	835	59.8	85	100%	90.70%	2 184
Pima-Diabetes	35.8	38	100%	76.49%	61	48.8	57	100%	76.62%	282
PoL	18.2	9	100%	92.97%	40	26.6	21	100%	93.16%	194
Seeds	9.8	4	100%	89.05%	7	9.8	4	100%	89.05%	11
Spambase	42.2	72	100%	93.64%	1 624	62.6	92	100%	94.31%	2 867

Dataset	$M = 500$					$M = 1000$				
	$\|w\|_0$	K	FI	ACC	Time (s)	$\|w\|_0$	K	FI	ACC	Time (s)
Breast-Cancer	39.8	53	100%	95.18%	384	45.4	62	100%	95.33%	904
COMPAS	13.0	15	100%	66.73%	218	13.0	16	100%	66.73%	616
ELEC2	79.0	70	100%	81.10%	8 762	104.4	97	100%	81.20%	26 493
FICO	18.0	26	100%	71.63%	593	18.0	25	100%	71.63%	1 434
HTRU2	70.2	92	100%	97.80%	5 517	99.0	128	100%	97.81%	18 637
House-16H	123.8	155	100%	87.18%	14 381	150.6	172	100%	87.12%	27 127
Ionosphere	90.8	130	100%	91.27%	4 890	113.8	156	100%	90.42%	7 014
Pima-Diabetes	70.4	88	100%	76.49%	1 264	96.0	126	100%	74.68%	4 145
PoL	52.4	61	100%	93.17%	1 840	74.8	88	100%	93.64%	8 059
Seeds	9.8	4	100%	89.05%	22	9.8	4	100%	89.05%	44
Spambase	103.2	124	100%	94.55%	6 294	147.8	204	100%	94.64%	19 040

Table 3: Pruning with FIPE$-\|\cdot\|_1$ on AdaBoost ensembles with 100, 200, 500, and 1000 base learners.

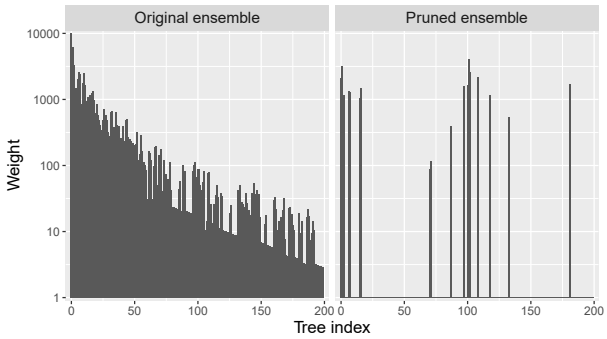

Figure 3: Weights of learners in the original and pruned ensembles on the FICO dataset with $M = 200$.

it is possible to prune learners created early in the training process by assigning a large weight to later learners.

It is also interesting to observe that the size of the pruned ensemble grows significantly slower than the size of the original ensemble. This can be seen in Table 3, which shows the size of the pruned ensemble for varying sizes of the original ensemble. Yet, the test accuracy tends to increase with the size of the ensemble. This suggests that only a small percentage of the newly created learners have a significant predictive power.

Result 3. As boosted ensembles grow, their accuracy increases but they tend to have more superfluous base learners.

4.3 Comparison with "non-faithful" baselines

We compare FIPE with several baselines from other recent studies. To the best of our knowledge, we are the first to study functionally-identical pruning of ensembles. Hence, all existing baselines cannot guarantee faithfulness to the original ensemble. We implement the method of Guo et al. (2018) denoted as IMD, Lu et al. (2010) denoted as IC and Li, Yu, and Zhou (2012) denoted as DREP. We use the implementation from the pypruning package (Buschjäger and Morik 2021). A key difference with our approach is that these methods require the size of the pruned ensemble as a hyperparameter. To ensure a fair comparison, we set the value of this hyperparameter as the size of the pruned ensemble identified by FIPE.

Table 4 shows the faithfulness FI of the pruned model to the original model as well as the test accuracy ACC of the pruned model. The results show that the baselines have significantly lower accuracy and faithfulness to the original model on average, regardless of the size of the original ensemble. Hence, FIPE outperforms all baselines w.r.t. both faithfulness and accuracy.

Result 4. Functionally identical pruning guarantees that there is no decrease in test accuracy. In contrast, existing pruning methods have no guarantee and decrease the test accuracy in practice.

M	$\|w\|_0$		FI	ACC
100	28.13	FIPE	**100.00%**	**85.55%**
		DREP	87.56%	81.68%
		IC	87.50%	80.56%
		IMD	37.08%	41.08%
200	39.55	FIPE	**100.00%**	**85.70%**
		DREP	86.71%	81.25%
		IC	87.33%	80.77%
		IMD	32.92%	37.57%
500	60.95	FIPE	**100.00%**	**85.83%**
		DREP	85.98%	80.72%
		IC	86.40%	80.29%
		IMD	28.06%	32.75%
1000	79.33	FIPE	**100.00%**	**85.66%**
		DREP	85.72%	80.70%
		IC	82.85%	79.04%
		IMD	34.95%	38.30%

Table 4: Comparison of FIPE$-\|\cdot\|_1$ and non-faithful baselines for varying size of the original ensemble M. Results are averaged over all datasets and repetitions.

5 Related Literature

Many approaches have been proposed for ensemble pruning. We provide an overview of the closely related literature in this section, and refer an interested reader to Tsoumakas, Partalas, and Vlahavas (2009) and Chapter 6 of Zhou (2012) for a more exhaustive review.

Pruning tree ensembles. We can distinguish three main frameworks for pruning: ordering-based, clustering-based, and optimization-based. The first two are approximate and therefore do not guarantee that the pruned ensembles are of minimal size. Early approaches rank the elements of the ensemble and select them in a greedy fashion (Margineantu and Dietterich 1997). This can be regularized by encouraging the diversity of the selected classifiers (Lu et al. 2010; Guo et al. 2018). Cluster-based pruning is performed in two steps: first, cluster the classifiers in representative groups, then select a representative element from these groups (see e.g. Giacinto, Roli, and Fumera 2000).

A few optimization-based approaches have also been proposed. They are based on mixed-integer quadratic formulations, which typically do not scale to large models and datasets. Zhang et al. (2006) present a quadratic formulation and propose a relaxation based on semi-definite programming. Li, Yu, and Zhou (2012) and Cavalcanti et al. (2016) extend this idea with diversity measures.

Finally, an interesting stream of literature focuses on building models with small memory requirements (Kumar, Goyal, and Varma 2017). Painsky and Rosset (2019) study how to compress random forests by encoding the trees into binary codes. Nakamura and Sakurada (2022) simplify tree ensembles by reducing the number of distinct splitting rules and sharing the rules across trees.

Optimal pruning. Optimal methodologies have demonstrated effectiveness across several pruning tasks. Sherali, Hobeika, and Jeenanunta (2009) present an optimal algorithm for pruning nodes of decision trees. Liu and Mazumder (2023) show how to jointly prune nodes of random forest ensembles using a coordinated-descent algorithm. The tree-based structure of tree ensembles is especially suitable for combinatorial optimization methods, allowing to adapt the training algorithm or post-process ensembles to improve fairness or interpretability (Carrizosa, Molero-Río, and Romero Morales 2021). Combinatorial optimization methods have also proven successful for pruning large deep neural networks (Serra, Kumar, and Ramalingam 2020; Yu et al. 2022; Benbaki et al. 2023; Meng et al. 2024).

Faithfulness. Functionally identical pruning introduces a significant shift in perspective compared to existing works. Traditionally, the question asked was "*given a target size, what is the best subset I should select from my ensemble?*". In contrast, we look for the smallest reweighted subset of trees that provide the same prediction function. The closest work to ours is likely Vidal, Pacheco, and Schiffer (2020), who show how to transform a tree ensemble into a decision tree with an identical prediction function on the entire feature space. However, this born-again tree is computationally expensive to obtain and might grow exponentially large in some cases as some ensembles are not efficiently representable as single trees

6 Conclusion

This paper proposed new methodological tools to prune additive tree ensembles while guaranteeing that their prediction function remains unchanged. Through extensive experimental analyses, we have demonstrated that boosted ensembles can be significantly compressed. This suggests that boosted ensembles have many superfluous learners, which can be removed entirely from the ensemble on the condition that the remaining learners are adequately reweighted. A significant advantage of such pruning is that, contrary to existing pruning methods, it guarantees that the test accuracy of the pruned ensemble does not decrease.

This work opens numerous research perspectives. First, while we focused on being certifiably identical to the original model on the entire feature space, it is worthwhile to study how to impose faithfulness on a subspace, such as the space of plausible observations as Parmentier and Vidal (2021). This might allow stronger compression. For most practical situations, it is possible to avoid using the combinatorial optimization oracle. Using an approximate oracle may lead to a faster algorithm that lacks the faithfulness certificate but might empirically be very close to the original model. Last but not least, pursuing a disciplined analysis of model compression with faithfulness guarantees for other machine learning models is a promising direction for future work.

Acknowledgements

This work was supported by the Canada Excellence Research Chair in Data Science for Real-Time Decision-Making and the SCALE-AI Research Chair in Data-Driven

Supply Chains. The authors would like to thank the anonymous reviewers for their valuable suggestions.

References

Amee, A. H.; Hossain, M. I.; Khan, S. F.; Farid, D. M.; et al. 2023. A novel pessimistic decision tree pruning approach for classification. In *International Conference on Electrical Information and Communication Technology*, 1–6. IEEE.

Benbaki, R.; Chen, W.; Meng, X.; Hazimeh, H.; Ponomareva, N.; Zhao, Z.; and Mazumder, R. 2023. Fast as CHITA: Neural network pruning with combinatorial optimization. In *International Conference on Machine Learning*, 2031–2049. PMLR.

Biau, G.; and Scornet, E. 2016. A random forest guided tour. *Test*, 25(2): 197–227.

Breiman, L. 2001. Random forests. *Machine Learning*, 45(1): 5–32.

Buschjäger, S.; and Morik, K. 2021. Improving the accuracy-memory trade-off of random forests via leaf-refinement. *arXiv preprint arXiv:2110.10075*.

Carrizosa, E.; Molero-Río, C.; and Romero Morales, D. 2021. Mathematical optimization in classification and regression trees. *TOP*, 29(1): 5–33.

Cavalcanti, G. D.; Oliveira, L. S.; Moura, T. J.; and Carvalho, G. V. 2016. Combining diversity measures for ensemble pruning. *Pattern Recognition Letters*, 74: 38–45.

Giacinto, G.; Roli, F.; and Fumera, G. 2000. Design of effective multiple classifier systems by clustering of classifiers. In *International Conference on Pattern Recognition*, volume 2, 160–163. IEEE.

Grinsztajn, L.; Oyallon, E.; and Varoquaux, G. 2022. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*.

Guo, H.; Liu, H.; Li, R.; Wu, C.; Guo, Y.; and Xu, M. 2018. Margin & diversity based ordering ensemble pruning. *Neurocomputing*, 275: 237–246.

Hastie, T.; Rosset, S.; Zhu, J.; and Zou, H. 2009. Multi-class AdaBoost. *Statistics and its Interface*, 2(3): 349–360.

Kantchelian, A.; Tygar, J. D.; and Joseph, A. 2016. Evasion and hardening of tree ensemble classifiers. In *International Conference on Machine Learning*, 2387–2396. PMLR.

Kumar, A.; Goyal, S.; and Varma, M. 2017. Resource-efficient machine learning in 2 kb ram for the internet of things. In *International Conference on Machine Learning*, 1935–1944. PMLR.

Li, N.; Yu, Y.; and Zhou, Z.-H. 2012. Diversity regularized ensemble pruning. In *Machine Learning and Knowledge Discovery in Databases: European Conference*, 330–345. Springer.

Liu, B.; and Mazumder, R. 2023. ForestPrune: Compact depth-pruned tree ensembles. In *International Conference on Artificial Intelligence and Statistics*, 9417–9428. PMLR.

Lu, Z.; Wu, X.; Zhu, X.; and Bongard, J. 2010. Ensemble pruning via individual contribution ordering. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 871–880.

Margineantu, D. D.; and Dietterich, T. G. 1997. Pruning adaptive boosting. In *International Conference on Machine Learning*, 211–218.

Martínez-Muñoz, G.; and Suárez, A. 2006. Pruning in ordered bagging ensembles. In *International Conference on Machine Learning*, 609–616.

McElfresh, D.; Khandagale, S.; Valverde, J.; Prasad C, V.; Ramakrishnan, G.; Goldblum, M.; and White, C. 2024. When do neural nets outperform boosted trees on tabular data? *Advances in Neural Information Processing Systems*, 36.

Meng, X.; Ibrahim, S.; Behdin, K.; Hazimeh, H.; Ponomareva, N.; and Mazumder, R. 2024. OSSCAR: One-shot structured pruning in vision and language models with combinatorial optimization. In *International Conference on Machine Learning*.

Nakamura, A.; and Sakurada, K. 2022. Simplification of Forest Classifiers and Regressors. *arXiv preprint arXiv:2212.07103*.

Painsky, A.; and Rosset, S. 2019. Lossless compression of random forests. *Journal of Computer Science and Technology*, 34: 494–506.

Parmentier, A.; and Vidal, T. 2021. Optimal counterfactual explanations in tree ensembles. In *International Conference on Machine Learning*, 8422–8431. PMLR.

Probst, P.; and Boulesteix, A.-L. 2017. To tune or not to tune the number of trees in random forest. *Journal of Machine Learning Research*, 18(1): 6673–6690.

Serra, T.; Kumar, A.; and Ramalingam, S. 2020. Lossless compression of deep neural networks. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 417–430. Springer.

Sherali, H. D.; Hobeika, A. G.; and Jeenanunta, C. 2009. An optimal constrained pruning strategy for decision trees. *INFORMS Journal on Computing*, 21(1): 49–61.

Tsoumakas, G.; Partalas, I.; and Vlahavas, I. 2009. An ensemble pruning primer. *Applications of Supervised and Unsupervised Ensemble Methods*, 1–13.

Vidal, T.; Pacheco, T.; and Schiffer, M. 2020. Born-again tree ensembles. In *International Conference on Machine Learning*, 9743–9753. PMLR.

Yu, X.; Serra, T.; Ramalingam, S.; and Zhe, S. 2022. The combinatorial brain surgeon: Pruning weights that cancel one another in neural networks. In *International Conference on Machine Learning*, 25668–25683. PMLR.

Zhang, Y.; Burer, S.; Nick Street, W.; Bennett, K. P.; and Parrado-Hernández, E. 2006. Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research*, 7(7).

Zhou, Z.-H. 2012. *Ensemble methods: Foundations and algorithms*. CRC press, 1st edition.