Genetic Adversarial Training of Decision Trees

Francesco Ranzato
Dipartimento di Matematica, University of Padova
Padova, Italy
ranzato@math.unipd.it

Marco Zanella
Dipartimento di Matematica, University of Padova
Padova, Italy
mzanella@math.unipd.it

ABSTRACT

We put forward a novel learning methodology for ensembles of decision trees based on a genetic algorithm that is able to train a decision tree for maximizing both its accuracy and its robustness to adversarial perturbations. This learning algorithm internally leverages a complete formal verification technique for robustness properties of decision trees based on abstract interpretation, a well-known static program analysis technique. We implemented this genetic adversarial training algorithm in a tool called MetaSilvae and we experimentally evaluated it on some standard reference datasets used in adversarial training. The experimental results show that MetaSilvae is able to train robust models that compete with and often improve on the current state-of-the-art of adversarial training of decision trees while being much more compact and therefore interpretable and efficient tree models.

CCS CONCEPTS

• **Computing methodologies** → **Classification and regression trees**; **Machine learning algorithms**; **Genetic algorithms**.

KEYWORDS

Genetic Algorithm, Adversarial Machine Learning, Decision Trees, Abstract Interpretation

ACM Reference Format:
Francesco Ranzato and Marco Zanella. 2021. Genetic Adversarial Training of Decision Trees. In *Proceedings of 2021 Genetic and Evolutionary Computation Conference (GECCO '21).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3449639.3459286

1 INTRODUCTION

Adversarial machine learning [27, 34] is a hot topic studying vulnerabilities of machine learning (ML) models in adversarial scenarios. Adversarial examples have been found in diverse application fields of ML, ranging from image classification to malware detection, and the current defense techniques include adversarial model training, input validation, testing and automatic verification of learning algorithms. A ML classifier is defined to be robust for a (typically very small) perturbation of its input samples, which represents an adversarial attack, when it assigns the same correct class to all the samples within that perturbation, so that unnoticeable malicious alterations of input objects should not deceive a robust classifier.

This work focuses on the robustness of ML classifiers consisting of decision tree ensembles, such as random forests and gradient boosted decision trees, which are well known for being both accurate and interpretable ML models and are widely used in adversarial scenarios. It has been amply shown that decision trees can be very non-robust [30, Section 8.1.4], although it is only recently that robustness verification and adversarial training of tree models started to be an active subject of investigation [2, 12, 14, 15, 21, 44, 51, 52].

Main Contributions. Genetic algorithms (GAs) [28, 50] provide a widespread effective search technique which computes the next set of hypotheses by repeatedly mutating and then combining parts of the best currently known hypotheses. A number of successful ML methodologies for decision trees are based on GAs [4, 22, 25, 31, 40, 53]. Recently, some GAs have also been investigated for adversarial training of neural networks [17, 54]. To the best of our knowledge, the use of GAs for adversarial training of ensembles of decision trees is still an unexplored topic. In this work we design and experimentally evaluate an adversarial training algorithm for decision tree ensembles based on a genetic algorithm, that we called MetaSilvae[1] and aims at maximizing both accuracy and robustness of decision trees. MetaSilvae (MS) relies on an open source verification method of the robustness of ensembles of decision trees called Silva [44]. This robustness verification algorithm Silva performs an abstract interpretation-based static analysis [19, 46] of a decision tree classifier which is able to abstractly compute the exact set of leaves of a decision tree which are reachable from an adversarial region. By exploiting this robustness information provided by Silva, MS is designed as a genetic algorithm that maximizes an objective performance function which is a linear combination of accuracy and robustness. MS is based on well established design choices of GAs: (1) elitist selection strategy; (2) roulette wheel selection; (3) single-point crossover; (4) offspring mutation. MS has been implemented in C and experimentally evaluated on the reference datasets for adversarial training of decision tree ensembles. MS has been compared with random forests [8] and with the current state-of-the-art of adversarially trained gradient boosted decision trees [2, 14]. Overall, the experimental results show that MS trained models significantly increase their robustness over natural random forests on average of, resp., 3.4× (with an average absolute gain of +54.3%) while at the same time preserving a comparable accuracy with an expected slight drop (on average −1.8%). Moreover, MS models compete with and often improve on the state-of-the-art of adversarially trained gradient boosted decision trees while

GECCO '21, July 10–14, 2021, Lille, France
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8350-9/21/07…$15.00
https://doi.org/10.1145/3449639.3459286

[1]Latin for *beyond forests* and acronym of *Magister Efficiens Temperat Arbore Silvae*, translating to "*the efficient master mixes the trees of the forest*".

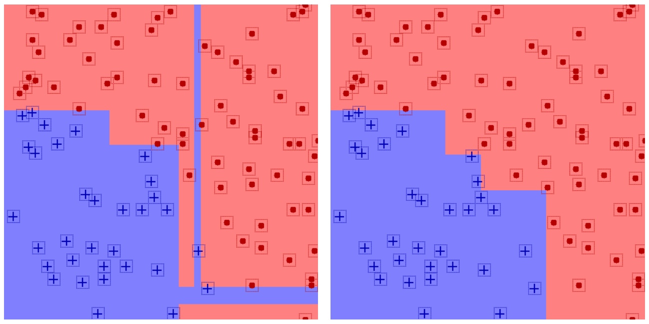


Figure 1: Classifiers trained with scikit-learn (left) and MetaSilvae (right) on the same dataset.

being much more compact and therefore interpretable (as advocated by [38] for all ML models) and efficient (as advocated by [1, Section 3.1] for ensemble models) tree models.

Illustrative Example. Fig. 1 depicts an example of an artificial dataset consisting of 100 2-dimensional uniformly distributed samples $(x_1, x_2) \in [0,1]^2$ labeled as a blue cross when $x_1^2 + x_2^2 < 0.5$ holds and as a red bullet otherwise. The left diagram represents a decision tree classifier which has been trained by scikit-learn, while the right diagram has been trained by MetaSilvae using as objective performance function $\varphi(\text{accuracy}, \text{robustness}) = 90\%$ accuracy + 10% robustness. The two diagrams in Fig. 1 show that both classifiers achieve 100% accuracy. The scikit-learn tree introduces two blue regions cutting the red area in order to achieve an accurate classification of two blue cross samples, thus making the classification on samples too close to the borders not robust for a 2% square perturbation surrounding each input sample. The MetaSilvae training algorithm is able to avoid this lack of robustness since it searches for cut hyperplanes which preserve robustness whenever possible. As displayed by the two diagrams, robustness with respect to the 2% square perturbation turns out be, resp., 87% and 94% for the left and right decision trees, thus showing a significant increase while still preserving the same 100% accuracy.

Related Work. While adversarial training of neural networks has been widely studied, few works addressed how to train decision trees which are robust to adversarial attacks [2, 11, 12, 14]. In particular, we compared our experimental results with the robust gradient boosted decision trees of [2, 14]. Abstract interpretation [19, 46] techniques have been fruitfully applied for designing precise and scalable robustness verification algorithms and adversarial training techniques for a range of ML models [10, 26, 36, 37, 43, 44, 47–49]. In particular, to our knowledge, [36] is the only work using an abstract interpretation technique for adversarial training of ML models, notably deep neural networks.

2 BACKGROUND

Classifiers and Metrics. Given an input space $X \subseteq \mathbb{R}^d$ of numerical vectors and a finite set of labels/classes $\mathcal{L} = \{y_1, \dots, y_m\}$, a classifier is a function $C : X \to \wp_+(\mathcal{L})$, where $\wp_+(\mathcal{L})$ denotes the set of nonempty subsets of $\mathcal{L}$. C associates at least one label to every input in X and multiple output labels can be used to model ties in output classification (e.g. ties in voting schemes). Training algorithms take a ground truth dataset $D \subseteq X \times \mathcal{L}$ as input and output a classifier $C : X \to \wp_+(\mathcal{L})$ which minimizes/maximizes criteria such as a loss function for neural networks or information gain for decision trees.

Classifiers can be evaluated and compared according to several performance metrics. Accuracy on a test set is a standard metric for assessing a classification model: given a test set $T \subseteq X \times \mathcal{L}$ of correctly labeled samples, the accuracy of a classifier $C : X \to \wp_+(\mathcal{L})$ on T is $acc_T(C) \triangleq |\{(\boldsymbol{x}, y) \in T \mid C(\boldsymbol{x}) = \{y\}\}|/|T|$. One typically aims at training classifiers having a nearly perfect accuracy on suitably crafted test sets. However, according to a growing belief [27], accuracy is not enough in ML, because the robustness properties of a classifier may affect its safety and generalization. Given a perturbation function $P : X \to \wp(X)$ modeling a notion of closeness for input samples, the robustness of C w.r.t. P for a test set T is defined by $rob_{T,P}(C) \triangleq |\{(\boldsymbol{x}, y) \in T \mid C(\boldsymbol{x}) = \{y\},\ \forall \boldsymbol{x'} \in P(\boldsymbol{x})\colon C(\boldsymbol{x'}) = C(\boldsymbol{x})\}|/|T|$. Perturbation regions $P(\boldsymbol{x}) \subseteq X$ are used to model adversarial attacks to input samples $\boldsymbol{x}$, i.e., negligible alterations of input vectors aimed at deceiving a classifier. Widely studied perturbations are those induced by ℓ_p norms, in particular the ℓ_∞ maximum norm [13], which, given an alteration threshold $\epsilon > 0$, defines a perturbation as follows: $P_{\infty,\epsilon}(\boldsymbol{x}) \triangleq \{\boldsymbol{x'} \in X \mid \|\boldsymbol{x} - \boldsymbol{x'}\|_\infty \leq \epsilon\}$, where $\|(x_1, \dots, x_d)\|_\infty = \max\{x_1, \dots, x_d\}$. We also consider a more general notion of robustness called stability which encodes the ability of a classifier C of producing the same output on every sample within a perturbation region, therefore including the cases where C is inaccurate on some input: this is defined as $st_{T,P}(C) \triangleq |\{(\boldsymbol{x}, y) \in T \mid \forall \boldsymbol{x'} \in P(\boldsymbol{x})\colon C(\boldsymbol{x'}) = C(\boldsymbol{x})\}|/|T|$. We will mostly use stability rather than robustness since we deem stability to be a more comprehensive metric to use in adversarial training.

Decision Trees and Tree Ensembles. Decision trees are well established ML models used for both classification and regression tasks. In this work we consider standard classification trees commonly known as CART (Classification And Regression Trees) [9]. A classification decision tree $t : X \to \wp_+(\mathcal{L})$ is inductively defined as follows. (1) A base tree t is a leaf λ storing a frequency distribution of labels for the samples of the training set which some algorithmic rule (canonically the maximum frequency) converts to one or more predicted labels; thus, we simply consider $\lambda \in \wp_+(\mathcal{L})$ and therefore, for all $\boldsymbol{x} \in X$, $t(\boldsymbol{x}) \triangleq \lambda$. (2) A composite tree t is $\gamma(split, t_l, t_r)$ where $split : X \to \{\mathbf{t}, \mathbf{f}\}$ is a Boolean split criterion for the internal parent node of its left and right subtrees t_l and t_r; thus, for all $\boldsymbol{x} \in X$, $t(\boldsymbol{x}) \triangleq$ **if** $split(\boldsymbol{x})$ **then** $t_l(\boldsymbol{x})$ **else** $t_r(\boldsymbol{x})$. A tree stump is of the form $\gamma(split, \lambda_l, \lambda_r)$, i.e., it has a single internal node and two leaves. The training of a decision tree t guarantees that every leaf of t is reachable from at least one sample in the training set. Although split rules in general could be of any type, the most common decision trees employ univariate hard splits of the form $split(\boldsymbol{x}) \triangleq x_i \leq k$ for some feature $i \in [1, d]$ and threshold $k \in \mathbb{R}$.

Tree ensembles, also known as forests, are sets of decision trees which together contribute to formulate a unique classification output. Training algorithms as well as methods for computing the final output class(es) vary among different tree ensemble models. Random forests (RFs) [8] are a major instance of tree ensemble where each tree of the ensemble is trained independently from the other trees on a random subset of the features. Gradient boosted decision trees (GBDTs) [24] represent a different training algorithm where an ensemble of trees is incrementally build by training each new tree on the basis of the data samples which are mis-classified by the previous trees. For RFs, the final classification output is typically obtained through a voting mechanism (e.g., majority voting), while GBDTs are usually trained for binary classification problems and use a binary reduction scheme, such as one-vs-all or one-vs-one, for multi-class classification.

Our MetaSilvae trees are standard CARTs and we will use the random forest model for training an ensemble of MetaSilvae trees. In our experimental evaluation, on the one hand we will compare MetaSilvae with natural RFs in order to show that MetaSilvae training is able to make RFs robust, and, on the other hand, MetaSilvae models will be also compared with the state-of-the-art of robust GBDTs, that is, the adversarially trained models by [2] and [14].

3 TRAINING AS COMBINATORIAL OPTIMIZATION

Given a training dataset $D = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\} \subseteq X \times \mathcal{L}$, a training algorithm explores a hypothesis space $\mathcal{H} \subseteq X \to \wp_+(\mathcal{L})$ searching for a classification model in $\mathcal{H}$ which maximizes some performance function $f : (X \to \wp_+(\mathcal{L})) \to \mathbb{R}$. Training set, performance function and search strategy will determine how the output classification model is computed. In the case of CART decision trees, the training process first builds a tree stump consisting of a single internal node labeled by a univariate hard split $S_{i,k} \triangleq x_i \leq k$, where the attribute i and the threshold k are selected among all possible $i \in [1, d]$ and $k \in \mathbb{R}$. Each split candidate $S_{i,k}$ yields two new leaves λ_l and λ_r which store a distribution frequency of training samples $(\#y_1, \ldots, \#y_m) \in \mathbb{N}^m$ grouped by labels $y_i \in \mathcal{L}$, i.e., equivalently, a probability distribution $(p_1, \ldots, p_m) \in \mathbb{R}^m$ for labels which is used to compute either the entropy h or Gini g indexes which encode the information gain for a leaf $\lambda = (p_1, \ldots, p_m)$ as follows:

$$h(\lambda) \triangleq -\textstyle\sum_{i=1}^{m} p_i \cdot log_m(p_i) \qquad g(\lambda) \triangleq 1 - \textstyle\sum_{i=1}^{m} p_i^2$$

These indexes are used to estimate the purity of a split $S_{i,k}$ by averaging their values on its leaves λ_l and λ_r as follows:

$$H(S_{i,k}) \triangleq (|\lambda_l| h(\lambda_l) + |\lambda_r| h(\lambda_r))/(|\lambda_l| + |\lambda_r|)$$
$$G(S_{i,k}) \triangleq (|\lambda_l| g(\lambda_l) + |\lambda_r| g(\lambda_r))(|\lambda_l| + |\lambda_r|)$$

where $|\lambda|$ denotes the number of samples reaching the leaf λ for the split $S_{i,k}$. The training algorithm selects a split candidate $S_{i,k}$ which minimizes H or G, and the process will be repeated until no more splits are possible, i.e. every leaf contains only samples with a same label, or other custom criteria are met such as maximum tree depth, maximum number of leaves or minimum number of samples per leaf (these are tunable parameters, e.g., with scikit-learn).

This training process therefore corresponds to a greedy search algorithm which tries to find a globally optimal tree by looking at local split information only. This approach exhibits two major drawbacks:

(A) it does not take robustness (or stability) into account;
(B) it is limited by information from local splits only.

These limitations may often lead to overfitting, which is a well-known phenomenon with decision trees [7, 29]. Pruning techniques, e.g. [6, 32, 33], can be used to counteract overfitting by reducing the size of the tree in order to simplify the model, although this may often lead to a loss of accuracy and may be applied just as a post-training step.

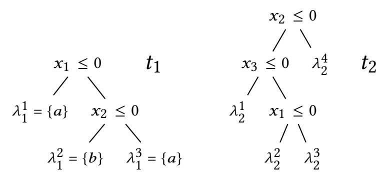


Figure 2: Two decision trees t_1 (left) and t_2 (right).

Of course, local properties of decision trees, in general, cannot be lifted globally to the whole tree, as it is the case of stability in this simple example.

Example 3.1. Consider the tree t_1 in Fig. 2 and its right subtree t_r rooted at the split node $x_2 \leq 0$. Consider an input region $Y = \{(x_1, x_2) \in \mathbb{R}^2 \mid -1 \leq x_1 \leq 1, -1 \leq x_2 \leq 0\}$. Then, the set of reachable leaves from Y in t_r is labeled as $\{b\}$, hence ensuring stability of t_r in Y, while the set of reachable leaves from Y in t_1 is labeled as $\{a, b\}$, thus making the classification of the tree t_1 not stable in Y. □

We will design a global tree training algorithm which takes stability into account. Given a (finite) set M of performance metrics m of type $\mathcal{H} \to \mathbb{R}$, such as accuracy on a given test set and stability on a given test set w.r.t. a perturbation, we combine them through a comprehensive performance function $\varphi_M : \mathcal{H} \to \mathbb{R}$ that will be maximized during the training process. For example, this performance function can be a simple linear combination $\varphi_M(C) = \sum_{i=1}^{|M|} w_i m_i(C)$. If φ_M is differentiable and, for some metric $m_k \in M$, $\frac{\partial \varphi_M}{\partial m_k} \geq 0$, then a classifier C_{opt} which is computed by an ideal trainer by maximizing φ_M will be optimal for the performance metric m_k, meaning that for any classifier $C \in \mathcal{H}$, if $m_j(C_{\text{opt}}) = m_j(C)$ for all $j \neq k$, then $m_k(C_{\text{opt}}) \geq m_k(C)$.

Of course, metrics such as accuracy and stability can only be estimated on a finite test subset of the input space X, meaning that global optima cannot be computed precisely, so that training algorithms effectively will find classifiers which are locally optimal for a finite test subset T of X. Finding a locally optimal classifier for T is not necessary in practice, as local optimality is unlikely to be an appropriate measure of global optimality and could even lead to overfitting. We argue that heuristic search strategies such as genetic algorithms may provide a viable and effective training procedure of tree classifiers which closely approximate the ideal optimal solutions and may improve the state-of-the-art in practice.

4 METASILVAE

A significant number of training procedures for decision trees based on evolutionary algorithms have been investigated, as surveyed by [4]. The underlying basic idea is that evolutionary algorithms perform a robust global, as opposed to local, search in the hypothesis space and tend to cope better with relationships between different features than greedy methods [23]. To the best of our knowledge, we put forward the first adversarial training procedure for decision tree (or tree ensemble) classifiers based on a genetic algorithm, called MetaSilvae, which targets to maximize both accuracy and stability metrics. MetaSilvae crucially relies on a complete formal verification method of the stability (or robustness) of ensembles of decision trees.

Complete Verification of Stability. We design and implement a decision tree adversarial training method based on a genetic algorithm, that we call MetaSilvae (MS). This genetic learning algorithm internally relies on Silva [44], an open source tool based on abstract interpretation [19, 46] for the formal verification of stability properties of decision tree ensembles. Silva performs a static analysis of ensembles of decision trees in an *abstract domain* $A \subseteq \wp(\mathbb{R}^d)$ which represents properties of interests of real vectors, such as hyperrectangles of intervals providing lower and upper bounds to vector components or a domain of linear relations between vector components. Silva first approximates in the abstract domain A an adversarial region $P(\boldsymbol{x}) \in \wp(\mathbb{R}^d)$ of an input vector $\boldsymbol{x} \in \mathbb{R}^d$ and then abstractly computes a sound *overapproximation* of the set of leaves of a decision tree (or an ensemble of trees) which are reachable from the adversarial samples ranging in $P(\boldsymbol{x})$. This static analysis is based on the *soundness* principle of abstract interpretation meaning that no leaf reachable from some sample in $P(\boldsymbol{x})$ can be missed. When adversarial attacks are modeled by the maximum norm perturbation $P_{\infty,\epsilon}(\boldsymbol{x})$ and the analysis is performed on the abstract domain of hyperrectangles the output returned by Silva turns out to be *complete*, meaning that each leaf computed by Silva is actually reached by some adversarial input ranging in $P_{\infty,\epsilon}(\boldsymbol{x})$, namely, no false positive (i.e., a false reachable leaf) may happen. Silva therefore provides a complete certification algorithm for the stability (or robustness) of an input sample $\boldsymbol{x}$ under adversarial attacks in $P_{\infty,\epsilon}(\boldsymbol{x})$ and, in turn, this verification tool allows us to derive *precisely* (thanks to completeness) the stability $st_{T,P_{\infty,\epsilon}}$ and robustness $rob_{T,P_{\infty,\epsilon}}$ metrics, as defined in Section 2, for a tree ensemble classifier on a test set T.

To conclude this brief outline of Silva, let us recall that the abstract domain of hyperrectangles HR_d consists of d-dimensional vectors of intervals $[l, u]$, where the bounds $l, u \in \mathbb{R} \cup \{-\infty, +\infty\}$ are such that $l \leq u$, for example $([0.1, 0.9], [-\infty, 0], [0.5, +\infty]) \in \mathrm{HR}_3$. Thus, a hyperrectangle $[\boldsymbol{l}, \boldsymbol{u}] = ([\boldsymbol{l}_1, \boldsymbol{u}_1], ..., [\boldsymbol{l}_d, \boldsymbol{u}_d]) \in \mathrm{HR}_d$ represents all the real vectors $\boldsymbol{x} \in \mathbb{R}^d$ such that, for all $j \in [1, d]$, $\boldsymbol{l}_j \leq \boldsymbol{x}_j \leq \boldsymbol{u}_j$, i.e., such that lower/upper bounds of their components are correctly approximated by $[\boldsymbol{l}, \boldsymbol{u}]$.

MetaSilvae. By exploiting the stability information provided by Silva, for training on a dataset T a decision tree in a hypothesis space $\mathcal{H}$, we consider as objective performance function $\varphi_T : \mathcal{H} \to \mathbb{R}$ a linear combination of accuracy and stability, i.e., $\varphi_T(t) \triangleq w_a acc_T(t) + w_s st_{T,P}(t)$ for a given perturbation P and for weights $w_a, w_s \in \mathbb{R}$. Since accuracy and stability are independent of each other, for nonnegative weights $w_a, w_s \geq 0$, φ_T is differentiable and such that $\frac{\partial \varphi_T}{\partial acc_T} = w_a \geq 0$, $\frac{\partial \varphi_T}{\partial st_{T,P}} = w_s \geq 0$, so that an ideal training algorithm which maximizes φ_T will output optimal decision trees. For instance, in the training example depicted in Fig. 1 we used as objective function $\varphi(t) \triangleq 0.9acc(t) + 0.1st_{\pm 2\%}(t)$.

Our suboptimal solution is computed by the genetic Algorithm 1 called MetaSilvae (MS). At line 1, a set of trees is generated and

Algorithm 1: MetaSilvae

```
1  population = generateInitialTrees();
2  nextPopulation = ∅;
3  while ¬ stopCriterion() do
       // Elitist Selection
4      nextPopulation.push(population.selectBest($\varphi_T$));
5      while ¬ nextPopulation.isFull() do
           // Roulette Wheel Selection
6          parentA = population.select($\varphi_T$);
7          parentB = population.select($\varphi_T$);
           // Single-Point Crossover
8          offspring = crossover(parentA, parentB);
           // Mutation With Probability p
9          if rand() < p then
               // Grow or Grow-Or-Prune
10             offspring.mutate();
11         nextPopulation.push(offspring);
12     population = nextPopulation;
13     nextPopulation = ∅
14 return population.selectBest($\varphi_T$)
```

stored as initial *population*, where MS can either start with a set of base trees consisting of a single leaf only (which is our choice in the experimental evaluation) or by any set of pre-trained decision trees. Then, the while-loop at lines 3-13 generates the *nextPopulation* and iterates until a stop criterion, such as a timeout or a bound on the *population* size, is met. At each iteration, the best tree w.r.t. our performance function φ_T will be first selected at line 4 from the current population to carry over to the next one unchanged. This therefore implements an elitist selection strategy for GAs [3] which ensures that the performance function will not decrease from the current population to the next. The new individuals will be generated by the while-loop at lines 5-11, where two trees *parentA* and *parentB* are first selected from the current population, then combined in an offspring by a crossover operation, which can finally be mutated for enhancing genetic diversity.

Selection. A fitness proportional selection scheme [39] is employed at lines 6-7, also known as roulette wheel, which selects an individual t_i from a current population Y with a probability p_i which is proportional to its fitness as determined by the objective function φ_T as follows: $p_i \triangleq \frac{\varphi_T(t_i)}{\sum_{t \in Y} \varphi_T(t)}$. It should be remarked that in this selection MetaSilvae exploits crucially the stability value $st_{T,P}(t)$ given by the verification tool Silva which is used by $\varphi_T(t)$.

Crossover. The information of the two selected trees *parentA* and *parentB* is combined at line 8 by the following variation of the standard single-point crossover procedure [41]:

1. a subtree t_A is randomly selected and pruned from *parentA*;
2. a subtree t_B is randomly selected from *parentB*;
3. t_B is inserted as subtree of *parentA* in place of the pruned subtree t_A;

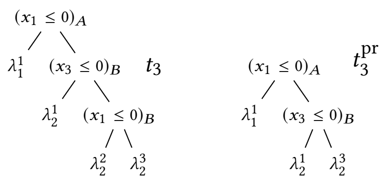


Figure 3: Decision tree before (left) and after pruning (right).

4. consistency of the new tree is enforced by pruning those nodes which do not contain information and then by reshaping the tree as needed.

Let us illustrate this crossover function by a simple example.

Example 4.1. Consider as parent trees t_1 and t_2 in Fig. 2, and assume that the nodes labeled with splits $x_2 \leq 0$ and $x_3 \leq 0$ are selected, resp., as subtrees t_A of t_1 and t_B of t_2. The tree t_B is inserted in t_1 in place of t_A, as depicted by the left tree t_3 in Fig. 3. However, after this insertion, the path $(x_1 \leq 0)_B \rightarrow \lambda_2^2$ becomes unfeasible due to the parent constraint $(x_1 > 0)_A$ at the root, so that the subtree of t_3 rooted at $(x_1 \leq 0)_B$ must be pruned of the left path $(x_1 \leq 0)_B \rightarrow \lambda_2^2$, thus yielding the output tree t_3^{pr} depicted on the right of Fig. 3. □

Mutation. After crossover, mutation of the offspring happens at lines 9-10 with probability p and can either prune a randomly selected subtree or transform a leaf into a single split. In the latter case, consistency of splits must be preserved so that logically inconsistent trees are never generated. The mutation probability p is specified as a tunable parameter of MetaSilvae, as well as the adopted mutation strategy which can be chosen between *grow* only and *grow-or-prune*. In both cases the selection is made on a stochastic basis by considering the entropy h of each node (or, alternatively, the Gini index g). We describe this mutation procedure more in detail through an example.

Example 4.2. Consider the decision tree t_4 on the left of Fig. 4 whose nodes are decorated as superscripts with their entropy values. A grow mutation starts from the root node $x_1 \leq 0$, computes the entropy h of its left and right children, and move toward one of them with a probability which is proportional to their respective entropies, in our example the node $(x_3 \leq 0)^{0.9}$ is chosen. Once a leaf is reached, in our example $(\ldots)^{0.8}$, the samples associated to that leaf are used to generate a set of features i and thresholds k which defines a set of split candidates $x_i \leq k$. The split candidate maximizing the objective function φ_T is then selected by applying the split and evaluating accuracy and stability of the corresponding tree. For example, the split $x_2 \leq 2.5$ is selected for growing the leaf $(\ldots)^{0.8}$ and this yields the tree t_4^{gr} depicted on the right of Fig. 4. □

It should be remarked that while the standard learning method for CART trees is a greedy algorithm which incrementally builds a decision tree by locally computing new split nodes or new leaves [9], in MetaSilvae the selection of a candidate split relies on a stability test (performed by Silva) on the whole corresponding candidate tree, thus meaning that the MS learning process is inherently not incremental and consequently could be computationally burdensome. We therefore introduced in MS an optimization parameter,

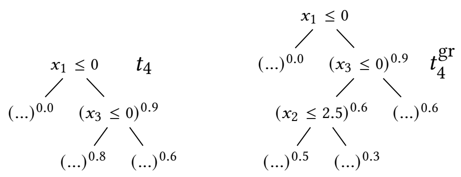
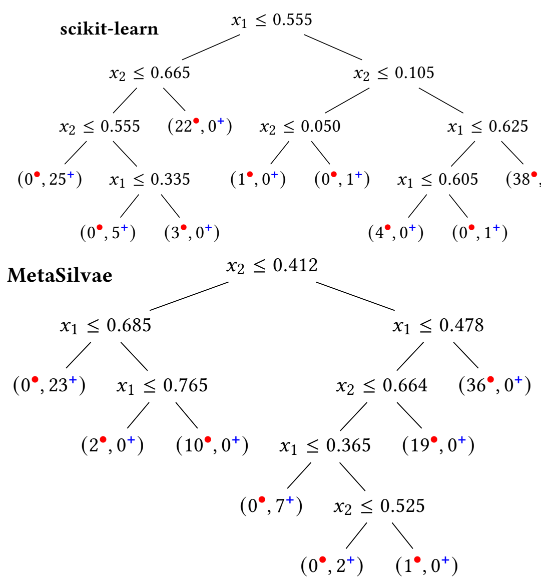


Figure 4: An example of tree mutation.

called "aggressiveness" in the implementation of MS, which allows us to set up a threshold on the number of split candidates which are immediately evaluated during the mutation phase, while those exceeding this threshold are delayed. It turns out that this optimization is effective in reducing the computational burden without sacrificing the overall generalization, because the evaluation of split candidates which are ruled out by this threshold is delayed to later iterations of MetaSilvae.

For the *grow-or-prune* mutation, a similar strategy is applied. Starting from the root node, the mutation algorithm iteratively moves either to the right of left children with a probability which is proportional to their entropy h and, at any point, the current node has a probability $1 - h$ of being pruned. By doing so, splits with low entropies are likely to be pruned, thus enhancing the tree compactness and stability at the bearable cost of losing some accuracy. If no pruning happens then the mutation proceeds as in the *grow* case.

Output. At the end of the whole evolutionary process, the best tree classifier of the final population, which is also the best tree encountered during the whole process due to elitism, is returned as output decision tree. As an example, the diagrams in Fig. 1 have been generated by the following two decision trees, where the first one has been trained by the standard algorithm of scikit-learn and the second one by MetaSilvae. In both trees, a leaf $\lambda = (n^\bullet, m^+)$ denotes that λ stores n red bullet samples and m blue cross samples.



Tree Ensembles. Multiple instances of MetaSilvae can be run (also in parallel: this chance was exploited in our experimental evaluation on a cluster) in order to generate a tree ensemble, which in our experiments is a random forest. When training a random forest, we need to achieve a significant random diversification among its trees, typically obtained by a random sampling of the set of features which each tree can explore when searching for a new split. MS allows to specify the number N of features which can be inspected for training a tree. A subset of features of size $\leq N \in [1, d]$ is randomly extracted, where each feature has the same probability of being selected. This feature selection happens only once before the training, and it is therefore applied tree-wise by MS. The resulting set of trees can be used as a single classifier by applying standard voting mechanisms (such as a simple majority vote) used for natural random forests.

5 EXPERIMENTAL EVALUATION

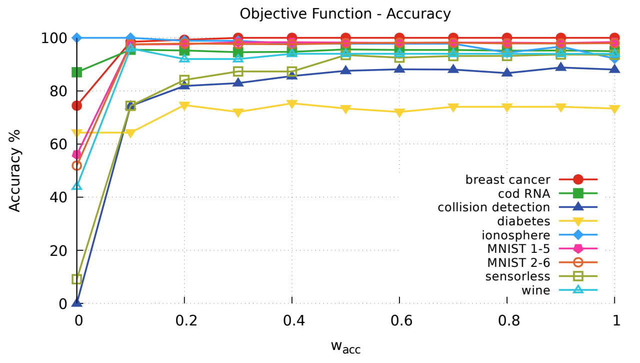
We implemented MetaSilvae as a self-contained C program whose source code (about 3K LOC) together with datasets, classification models and scripts is publicly available on GitHub [45]. Our experiments were run on a cluster of 15 computing nodes, each equipped with two Intel Xeon CPU E5520 at 2.27GHz with 8 cores and 32GB RAM. In our experiments we considered the collection of standard datasets with numerical features summarized in Tab.1: these datasets are used in [2, 14, 52], while wine is a UCI dataset [20].

Dataset	#classes	#features	values	#train	#test
breast-cancer	2	10	[1,10]	546	137
cod-rna	2	8	[0,1]	59535	271617
collision-detection	2	6	[0,1]	30000	3000
diabetes	2	8	[0,1]	614	154
fashion-mnist	10	784	[0,255]	60000	10000
ionosphere	2	34	[-1,1]	260	90
mnist	10	784	[0,255]	60000	10000
mnist-1-5	2	784	[0,255]	12162	2026
mnist-2-6	2	784	[0,255]	11875	1989
sensorless	11	48	[0,1]	48509	10000
wine	2	13	[0,1]	128	50

Table 1: Summary of datasets

Setup. MS does not require a specific tuning of hyperparameters for training decision trees, although the choice of the objective function φ plays a fundamental role. As described in Section 4, we adopted a weighted linear combination of accuracy and stability given by $\varphi_T(t) = w_{\text{acc}} acc_T(t) + w_{\text{stab}} st_{T,P}(t)$, for a given training dataset T, perturbation P and weights $w_{\text{acc}}, w_{\text{stab}} \in [0, 1]$ such that $w_{\text{acc}} + w_{\text{stab}} = 1$. Fig. 5 shows the impact of selecting different weights ($w = 0.1k$ for $k \in \{0, ..., 10\}$) on the final tree classifier in terms of accuracy and stability on the whole training set T.

As expected, higher values of w_{acc} tend to yield more accurate models, where we observed a converging behavior when accuracy exceeds 70%. On the other hand, stability increases with higher w_{stab} weights, it is above 45% and often above 60% already with $w_{\text{stab}} = 0.1$, although stability generally may display a more varying behavior than accuracy. These experiments hint that an effective and well balanced choice of weights can be obtained with $w_{\text{acc}} = 0.9, w_{\text{stab}} = 0.1$. It is worth remarking that by setting $w_{\text{stab}} = 1$ (and therefore $w_{\text{acc}} = 0$), as expected, MS will always produce perfectly stable models corresponding to a constant function, which is generally (but not always) inaccurate.



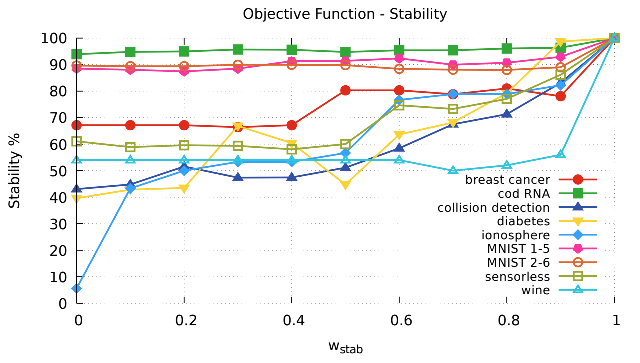


Figure 5: Impact of Different Weights.

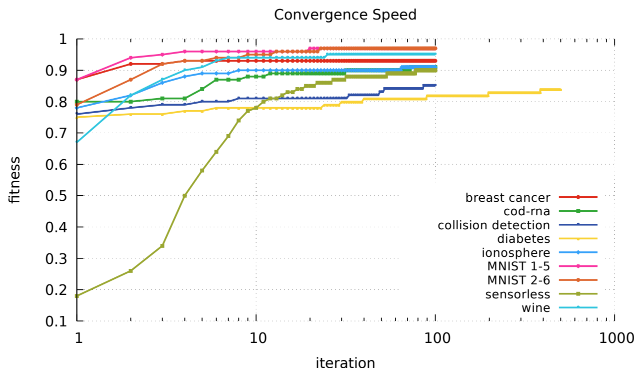


Figure 6: Convergence Speed.

We analysed the rate of convergence of the fitness function φ_T along iterations of MS training. Fig. 6 displays the fitness of the best individual of *population* after each iteration of MS, where the number of iterations is shown on a logarithmic scale. We observe that the best gains in fitness are achieved during the first iterations and MS tends to converge within 50-70 iterations, with the exception of the dataset diabetes which needs about 500 iterations to converge. Hence, the maximum number of iterations has been set to 500 for diabetes and to 100 for the other datasets. This same method is used to decide between *grow* and *grow-or-prune* mutations as well as their "aggressiveness" threshold, as described in Section 4.

Results. Tab. 2 summarizes the main data of our MetaSilvae models together with their accuracy and stability metrics. MS models have been trained for the objective performance function $\varphi_T(t) = 0.9acc_T(t) + 0.1st_{T,P_{\infty,\epsilon}}(t)$ where stability is w.r.t. a maximum norm perturbation $P_{\infty,\epsilon}$ whose magnitudes ϵ are displayed as absolute and percentage values. These magnitudes ϵ are the same used by [2, 14] for their adversarial training on 6 common datasets: breast-cancer, cod-rna, diabetes, fashion-mnist, mnist-1-5, mnist-2-6. It is worth remarking that some of these perturbations are quite challenging, since they peak to ±30.2% for mnist-1-5/2-6 and ±33.3% for breast-cancer. For fashion-mnist and mnist we trained a random forest of 50 MS-robust trees adopting a standard majority voting for output classification. For the remaining datasets it was enough to train a single robust decision tree since in these cases random forests do not bring practical benefits in accuracy or stability, thus making these single tree models efficient and easily interpretable. It is worth observing that MS training times are modest and acceptable even for the more demanding datasets fashion-mnist and mnist. The stability metrics w.r.t. $P_{\infty,\epsilon}$ have all been computed with no imprecision by Silva on the whole test sets.

Dataset	**MS training**				**MS metrics**	
	# of trees	max depth	$P_{\infty,\epsilon}$ (%)	time(s) per tree	*acc%*	*st%*
breast-cancer	1	6	3 (±33.3%)	0.2	100.0	89.1
cod-rna	1	12	0.025 (±2.5%)	20.0	95.6	89.9
collision-det.	1	12	0.1 (±10%)	28.8	87.5	45.4
diabetes	1	15	0.05 (±5%)	2.1	76.0	68.8
fashion-mnist	50	22	25 (±10%)	3593.1	86.4	46.9
ionosphere	1	10	0.2 (±10%)	0.2	97.8	84.4
mnist	50	20	30 (±11.8%)	3376.4	95.6	81.7
mnist-1-5	1	11	77 (±30.2%)	14.4	97.9	94.0
mnist-2-6	1	10	77 (±30.2%)	11.7	98.1	88.7
sensorless	1	15	0.01 (±1%)	133.0	94.7	57.4
wine	1	5	0.1 (±10%)	0.1	92.0	68.0

Table 2: Performance of MetaSilvae

Since MetaSilvae utilizes a seed provided by a random number generator, in order to analyse the impact of randomness on the output models, the MS training has been repeated 1000 times for each dataset by selecting distinct values of the seed. It turns out that accuracy of most models is within ±5% (and often ±2.5%) of the median, with the sole exception of wine. The results for stability are similar, where the stability of the models turns out to be within ±5% of the median, with the exception of ionosphere, where unstable (meaning stability < 30%) models can be produced, although 75% of these 1000 trees has a stability ≥ 30%, and ≥ 65% for half of them.

***MetaSilvae vs RF.*.** We compared our robust models trained by MetaSilvae with natural random forests trained by scikit-learn [42]. Hyperparameters for training random forests have been selected by a randomized grid search on the number of trees ranging in the interval [5, 100], maximum depth in [5, 100], and either Gini G or entropy H for split purity. By relying on the stability verification by Silva [44], we selected the hyperparameters which maximize the same objective function φ_T used in MS training. Tab. 3 shows the accuracy and stability metrics on the whole test sets for RF as compared to MS models, where, for the sake of fair comparison, in the average of the relative stability gains of MS models we excluded the RFs with 0% stability.

Dataset	**RF**		**MetaSilvae**		**Stability Gain**	
	acc%	*st%*	*acc%*	*st%*	abs.	rel.
breast-cancer	**100.0**	10.2	**100.0**	**89.1**	+78.9%	8.7×
cod-rna	**97.9**	62.3	95.6	**89.9**	+27.6%	1.4×
collision-det.	**94.8**	21.0	87.5	**45.4**	+24.4%	2.2×
diabetes	**78.6**	20.8	76.0	**68.8**	+48.1%	3.3×
fashion-mnist	**86.4**	0.0	**86.4**	**46.9**	+46.9%	∞
ionosphere	96.7	0.0	**97.8**	**84.4**	+84.4%	∞
mnist	94.9	0.0	**95.6**	**81.7**	+81.7%	∞
mnist-1-5	**99.8**	19.0	97.9	**94.0**	+75.0%	4.9×
mnist-2-6	**99.2**	0.0	98.1	**88.7**	+88.7%	∞
sensorless	**99.9**	22.2	94.7	**57.4**	+35.2%	2.6×
wine	**94.0**	62.0	92.0	**68.0**	+6.0%	1.1×
Average	**94.7**	19.8	92.9	**74.0**	+54.3%	3.4×

Table 3: Comparison Random Forests vs MetaSilvae

It turns out that all the MS models are significantly more stable than RFs (+54.3% on average). On the other hand, the average accuracy slightly decreased (−1.8%) w.r.t. natural RFs. We also emphasize that every single MS tree outperforms the corresponding Random Forest in terms of stability on every dataset but wine whose RF model already features a significant stability > 60%. With a very significant rise in stability at least a slight drop in accuracy is generally expected, although it is worth observing that in some notable cases the accuracy increased (mnist) or remained the same (fashion-mnist).

We considered the distribution of accuracy and stability among trees in RF and MS forests for the mnist dataset under a $P_{\infty,1}$ perturbation, where every forest consists in 50 trees. RF trees exhibit a median accuracy of ≈ 79.5%, with a minimum of 78.0% and a maximum of 81.5%, while for MS trees we observed a higher median accuracy of ≈ 86.5%, ranging from 86.0% to 87.5%. We observed a similar but stronger effect for stability, where RF trees show a median of ≈ 25%, ranging from 5% to 55%, whereas MS trees exhibit a stable median of 95% and range between 90% and 99%, thus revealing that each MS tree is significantly more accurate and stable than each RF tree.

We also compare an *efficiency metric* for RFs and MetaSilvae, where the efficiency of a classifier C w.r.t. a metric m measures which size of C is required to achieve a given value of m. Here, we define an efficiency metric for a tree ensemble C which takes into account how many leaves of C are needed to reach a given performance of accuracy and stability: if $m(C) \in [0, 1]$ is the performance of C for a metric m then the corresponding efficiency of C is defined by $\textsc{Eff}_m(C) \triangleq m(C)/leaves(C)$, where $leaves(C) \triangleq \sum_{t \in C} leaves(t)$ is the total number of leaves of trees in C. Thus, the higher $\textsc{Eff}_m(C)$ the better is the efficiency of C for m. Tab. 4 compares the number of leaves and efficiency for accuracy and stability (w.r.t. $P_{\infty,\epsilon}$) for RFs and MetaSilvae. It turns out that MetaSilvae is much more efficient than RFs: the average of the ratios $\frac{leaves(MS)}{leaves(\mathrm{RF})}$ is 51.4% while the average relative efficiency gains of MS w.r.t. RFs (i.e., the ratio

$\textsc{Eff}_m(MS)/\textsc{Eff}_m(\text{RF}))$ are, resp., 174× for accuracy, and 1825× for stability (by excluding in this average 4 datasets whose $\textsc{Eff}_{st}$ for RFs is close to 0).

Dataset	*leaves*(C)		$\textsc{Eff}_{acc}(C)$		$\textsc{Eff}_{st}(C)$	
	RF	**MS**	**RF**	**MS**	**RF**	**MS**
breast-cancer	641	4	$1.56{\cdot}10^{-3}$	$2.50{\cdot}10^{-1}$	$1.59{\cdot}10^{-4}$	$2.21{\cdot}10^{-1}$
cod-rna	89757	85	$1.09{\cdot}10^{-5}$	$1.13{\cdot}10^{-2}$	$6.94{\cdot}10^{-6}$	$1.06{\cdot}10^{-2}$
collision-det.	39678	96	$2.39{\cdot}10^{-5}$	$9.11{\cdot}10^{-3}$	$5.29{\cdot}10^{-6}$	$4.73{\cdot}10^{-3}$
diabetes	2583	83	$3.04{\cdot}10^{-4}$	$9.15{\cdot}10^{-3}$	$8.04{\cdot}10^{-5}$	$8.29{\cdot}10^{-3}$
fashion-mnist	47549	119986	$1.81{\cdot}10^{-3}$	$7.20{\cdot}10^{-4}$	$0.00{\cdot}10^{+0}$	$3.90{\cdot}10^{-4}$
ionosphere	493	17	$1.96{\cdot}10^{-3}$	$5.75{\cdot}10^{-2}$	$0.00{\cdot}10^{+0}$	$4.97{\cdot}10^{-2}$
mnist	45268	133652	$2.09{\cdot}10^{-3}$	$7.15{\cdot}10^{-4}$	$0.00{\cdot}10^{+0}$	$6.11{\cdot}10^{-4}$
mnist-1-5	3231	30	$3.08{\cdot}10^{-2}$	$3.26{\cdot}10^{+0}$	$5.88{\cdot}10^{-3}$	$3.13{\cdot}10^{+0}$
mnist-2-6	4881	76	$2.03{\cdot}10^{-2}$	$1.29{\cdot}10^{+0}$	$0.00{\cdot}10^{+0}$	$1.17{\cdot}10^{+0}$
sensorless	15704	150	$6.36{\cdot}10^{-5}$	$6.31{\cdot}10^{-3}$	$1.41{\cdot}10^{-5}$	$3.83{\cdot}10^{-3}$
wine	220	15	$4.27{\cdot}10^{-3}$	$6.13{\cdot}10^{-2}$	$2.82{\cdot}10^{-3}$	$4.53{\cdot}10^{-2}$

Table 4: Efficiency of Random Forests and MetaSilvae

MetaSilvae vs Robust Gradient Boosted Trees. Finally, in Tab. 5 we compare our MS robust models with the adversarially trained tree models of [2], denoted by AH19, and [14], denoted by CZBH19. These are gradient boosted decision trees of the same type of XGBoost trees [16], which, to the best of our knowledge, represent the state-of-the-art of adversarially trained GBDTs. Although our MetaSilvae robust training generates tree ensembles which are random forests, and RFs and GBDTs are tree ensemble models with unrelated training principles, we nevertheless compare these different models since these robust GBDTs were the only adversarially trained decision trees found in literature. We considered 6 common datasets and, as already recalled, the perturbation $P_{\infty,\epsilon}$ is exactly the same used in [2] and [14]. Let us remark that the accuracy and robustness metrics of AH19 and CZBH19 models are taken from Table 7 of the supplemental of [2], because Silva cannot be used to compute the robustness of GBDTs. We also compare the objective performance function φ_T used for training MetaSilvae, where accuracy and robustness weigh, resp., 90% and 10%.

	CZBH19			**MetaSilvae**		
Dataset	*acc%*	*rob%*	obj. φ_T	*acc%*	*rob%*	obj. φ_T
breast-cancer	99.3	86.9	0.98	**100.0**	**89.1**	**0.99**
cod-rna	89.8	75.8	0.88	**95.6**	**88.3**	**0.95**
diabetes	**77.9**	**59.7**	**0.76**	76.0	59.1	0.74
fashion-mnist	85.6	34.9	0.80	**86.4**	**44.8**	**0.82**
mnist-1-5	**99.7**	**97.1**	**0.99**	97.9	93.1	0.97
mnist-2-6	**99.5**	**93.1**	**0.99**	98.1	88.1	0.97
Average	91.9	74.6	0.90	**92.3**	**77.1**	**0.91**

Dataset	**AH19**			**MetaSilvae**		
	acc%	*rob%*	obj. φ_T	*acc%*	*rob%*	obj. φ_T
breast-cancer	99.3	**93.4**	**0.99**	**100.0**	89.1	**0.99**
cod-rna	93.1	78.7	0.92	**95.6**	**88.3**	**0.95**
diabetes	72.7	**64.3**	0.72	**76.0**	59.1	**0.74**
fashion-mnist	85.8	**76.8**	**0.85**	**86.4**	44.8	0.82
mnist-1-5	**99.8**	**98.7**	**0.99**	97.9	93.1	0.97
mnist-2-6	**99.3**	**96.2**	**0.99**	98.1	88.1	0.97
Average	91.7	**84.7**	**0.91**	**92.3**	77.1	**0.91**

Table 5: Comparison with CZBH19 (top) and AH19 (bottom)

For AH19 models, we also compare their size, i.e. total number of leaves, and their efficiency with MS models. Tab. 6 displays the relative efficiency gains of MS w.r.t. AH19 models, which are, resp., the ratios $\frac{leaves(MS)}{leaves(\text{AH19})}$, $\frac{\textsc{Eff}_{acc}(MS)}{\textsc{Eff}_{acc}(\text{AH19})}$ and $\frac{\textsc{Eff}_{rob}(MS)}{\textsc{Eff}_{rob}(\text{AH19})}$.

Dataset	**AH19**	**MS**	**Efficiency Gain MS/AH19**		
	# *leaves*	# *leaves*	*leaves*	*acc*	*rob*
breast-cancer	80	4	**5.0**%	**20.1**×	**18.9**×
cod-rna	913	85	**9.3**%	**11.0**×	**12.1**×
diabetes	79	83	105.0%	1.0×	0.9×
fashion-mnist	97279	119986	123.3%	0.8×	0.5×
mnist-1-5	3258	30	**0.9**%	**106.5**×	**102.4**×
mnist-2-6	3777	76	**2.0**%	**49.1**×	**45.5**×

Table 6: Efficiency of AH19 and MetaSilvae

On average, it turns out that: (i) AH19 models are moderately more robust (+7.6%) than MS; (ii) MS are slightly more robust (+2.5%) than CZBH19 models; (iii) MS models are slightly more accurate of both AH19 (+0.6%) and CZBH19 (+0.4%) models; (iv) all three models exhibit the same average performance according to the objective function φ_T; (v) MS models are significantly more efficient than AH19 models both for size (average 40.9%), accuracy (average 31.4×) and robustness (average 30.1×). Our MS models compete with and often improve on the robust GBDTs of [2, 14] while being more compact and therefore interpretable and efficient tree models.

6 CONCLUSION

We believe that this work contributes to push forward the use of formal verification methods in machine learning, in particular a very well known program analysis technique such as abstract interpretation has been proved successful for training decision tree classifiers which are both accurate and robust and compete with and often improve on the state-of-the-art of adversarial training of gradient boosted decision trees while being much more compact and therefore interpretable and efficient tree models. As future work, we plan to investigate the problem of fairness verification [18, 35] for ensembles of decision trees by leveraging abstract interpretation techniques along the lines of this paper. The final goal will be to design a fairness-aware learning algorithm for decision trees, for both notions of individual and group fairness [5].

ACKNOWLEDGMENTS

The doctoral fellowship of Marco Zanella was funded by *Fondazione Bruno Kessler* (FBK), Trento, Italy. This work has been partially funded by the *University of Padova*, under the SID2018 project "Analysis of STatic Analyses (ASTA)", by the *Italian Ministry of Research MIUR*, under the PRIN2017 project no. 201784YSZ5 "AnalysiS of PRogram Analyses (ASPRA)", and by *Facebook Research*, under a "Probability and Programming Research Award".

REFERENCES

[1] Omar Y. Al-Jarrah, Paul D. Yoo, Sami Muhaidat, George K. Karagiannidis, and Kamal Taha. 2015. Efficient Machine Learning for Big Data: A Review. *Big Data Research* 2, 3 (2015), 87 – 93. https://doi.org/10.1016/j.bdr.2015.04.001 Big Data, Analytics, and High-Performance Computing.

[2] Maksym Andriushchenko and Matthias Hein. 2019. Provably robust boosted decision stumps and trees against adversarial attacks. In *Proc. 33rd Annual Conf. on Neural Information Processing Systems (NeurIPS 2019)*. 12997–13008.

[3] Shumeet Baluja and Rich Caruana. 1995. Removing the Genetics from the Standard Genetic Algorithm. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML 1995)*. Morgan Kaufmann, 38–46. https://doi.org/10.1016/b978-1-55860-377-6.50014-1

[4] Rodrigo Coelho Barros, Márcio Porto Basgalupp, André Carlos Ponce de Leon Ferreira de Carvalho, and Alex Alves Freitas. 2012. A Survey of Evolutionary Algorithms for Decision-Tree Induction. *IEEE Trans. Syst. Man Cybern. Part C* 42, 3 (2012), 291–312. https://doi.org/10.1109/TSMCC.2011.2157494

[5] Reuben Binns. 2020. On the Apparent Conflict between Individual and Group Fairness. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (Barcelona, Spain) *(FAT* 2020)*. ACM, New York, NY, USA, 514–524. https://doi.org/10.1145/3351095.3372864

[6] Jeffrey P Bradford, Clayton Kunz, Ron Kohavi, Cliff Brunk, and Carla E Brodley. 1998. Pruning decision trees with misclassification costs. In *European Conference on Machine Learning*. Springer, 131–136.

[7] Max Bramer. 2013. *Avoiding Overfitting of Decision Trees*. Springer, 121–136. https://doi.org/10.1007/978-1-4471-4884-5_9

[8] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32. https://doi.org/10.1023/A:1010933404324

[9] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. Wadsworth.

[10] Stefano Calzavara, Pietro Ferrara, and Claudio Lucchese. 2020. Certifying Decision Trees Against Evasion Attacks by Program Analysis. In *Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security (LNCS)*. Springer.

[11] Stefano Calzavara, Claudio Lucchese, and Gabriele Tolomei. 2019. Adversarial Training of Gradient-Boosted Decision Trees. In *Proc. 28th ACM Int. Conf. on Information and Knowledge Management (CIKM 2019)* (Beijing, China). 2429–2432. https://doi.org/10.1145/3357384.3358149

[12] Stefano Calzavara, Claudio Lucchese, Gabriele Tolomei, Seyum Assefa Abebe, and Salvatore Orlando. 2020. TREANT: training evasion-aware decision trees. *Data Mining and Knowledge Discovery* (2020). https://doi.org/10.1007/s10618-020-00694-9

[13] Nicholas Carlini and David A. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *Proc. of 2017 IEEE Symposium on Security and Privacy*. 39–57. https://doi.org/10.1109/SP.2017.49

[14] Hongge Chen, Huan Zhang, Duane S. Boning, and Cho-Jui Hsieh. 2019. Robust Decision Trees Against Adversarial Examples. In *Proc. 36th Int. Conf. on Machine Learning, (ICML 2019)*. 1122–1131. http://proceedings.mlr.press/v97/chen19m.html

[15] Hongge Chen, Huan Zhang, Si Si, Yang Li, Duane S. Boning, and Cho-Jui Hsieh. 2019. Robustness Verification of Tree-based Models. In *Proc. 33rd Annual Conf. on Neural Information Processing Systems (NeurIPS 2019)*. 12317–12328. http://papers.nips.cc/paper/9399-robustness-verification-of-tree-based-models

[16] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016)*. ACM, 785–794. https://doi.org/10.1145/2939672.2939785

[17] Hwi-Yeon Cho and Yong-Hyuk Kim. 2019. Stabilized training of generative adversarial networks by a genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion, (GECCO 2019)*. ACM, 51–52. https://doi.org/10.1145/3319619.3326774

[18] Alexandra Chouldechova and Aaron Roth. 2020. A Snapshot of the Frontiers of Fairness in Machine Learning. *Commun. ACM* 63, 5 (April 2020), 82–89. https://doi.org/10.1145/3376898

[19] Patrick Cousot and Radhia Cousot. 1977. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. 4th ACM Symp. on Principles of Programming Languages (POPL 1977)*. ACM, 238–252. https://doi.org/10.1145/512950.512973

[20] Dheeru Dua and Casey Graff. 2019. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[21] Gil Einziger, Maayan Goldstein, Yaniv Sa'ar, and Itai Segall. 2019. Verifying Robustness of Gradient Boosted Models. In *Proc. 33rd AAAI Conf. on Artificial Intelligence*. 2446–2453. https://aaai.org/ojs/index.php/AAAI/article/view/4089

[22] Elif Ersoy, Erinç Albey, and Enis Kayis. 2020. A CART-based Genetic Algorithm for Constructing Higher Accuracy Decision Trees. In *Proceedings of the 9th International Conference on Data Science, Technology and Applications (DATA 2020)*. SciTePress, 328–338. https://doi.org/10.5220/0009893903280338

[23] Alex A. Freitas. 2002. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer.

[24] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of Statistics* (2001), 1189–1232.

[25] Zhiwei Fu, Bruce L. Golden, Shreevardhan Lele, S. Raghavan, and Edward A. Wasil. 2003. A Genetic Algorithm-Based Approach for Building Accurate Decision Trees. *INFORMS J. Comput.* 15, 1 (2003), 3–22. https://doi.org/10.1287/ijoc.15.1.3.15152

[26] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. 2018. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *Proc. 2018 IEEE Symposium on Security and Privacy*. 3–18. https://doi.org/10.1109/SP.2018.00058

[27] Ian Goodfellow, Patrick McDaniel, and Nicolas Papernot. 2018. Making Machine Learning Robust Against Adversarial Inputs. *Commun. ACM* 61, 7 (2018), 56–66. https://doi.org/10.1145/3134599

[28] John H Holland. 1984. Genetic algorithms and adaptation. In *Adaptive Control of Ill-Defined Systems*. Springer, 317–333.

[29] Ned Horning. 2013. Introduction to decision trees and random forests. *Am. Mus. Nat. Hist* 2 (2013), 1–27.

[30] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning: with Applications in R*. Springer.

[31] Dariusz Jankowski and Konrad Jackowski. 2014. Evolutionary Algorithm for Decision Tree Induction. In *Computer Information Systems and Industrial Management*. Springer, 23–32.

[32] Michael J Kearns and Yishay Mansour. 1998. A Fast, Bottom-Up Decision Tree Pruning Algorithm with Near-Optimal Generalization. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*. 269–277.

[33] Boonserm Kijsirikul and Kongsak Chongkasemwongse. 2001. Decision tree pruning using backpropagation neural networks. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'01)*, Vol. 3. IEEE, 1876–1880.

[34] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2017. Adversarial Machine Learning at Scale. In *Proc. 5th Int. Conf. on Learning Representations (ICLR 2017)*. https://openreview.net/forum?id=BJm4T4Kgx

[35] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2019. A Survey on Bias and Fairness in Machine Learning. *CoRR* abs/1908.09635 (2019). http://arxiv.org/abs/1908.09635

[36] Matthew Mirman, Timon Gehr, and Martin T. Vechev. 2018. Differentiable Abstract Interpretation for Provably Robust Neural Networks. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*. 3575–3583. http://proceedings.mlr.press/v80/mirman18b.html

[37] Christoph Müller, Gagandeep Singh, Markus Püschel, and Martin T. Vechev. 2020. Neural Network Robustness Verification on GPUs. *CoRR* abs/2007.10868 (2020). arXiv:2007.10868 https://arxiv.org/abs/2007.10868

[38] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. 2019. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences* 116, 44 (2019), 22071–22080. https://doi.org/10.1073/pnas.1900654116

[39] Frank Neumann, Pietro Simone Oliveto, and Carsten Witt. 2009. Theoretical Analysis of Fitness-Proportional Selection: Landscapes and Efficiency. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO 2009)* (Montreal, Québec, Canada). ACM, 835–842. https://doi.org/10.1145/1569901.1570016

[40] A. Papagelis and D. Kalles. 2000. GA Tree: genetically evolved decision trees. In *Proceedings 12th IEEE Internationals Conference on Tools with Artificial Intelligence (ICTAI 2000)*. 203–206.

[41] G. Pavai and T. V. Geetha. 2016. A Survey on Crossover Operators. *ACM Comput. Surv.* 49, 4, Article 72 (Dec. 2016), 43 pages. https://doi.org/10.1145/3009966

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[43] Francesco Ranzato and Marco Zanella. 2019. Robustness Verification of Support Vector Machines. In *Proceedings of the 26th International Static Analysis Symposium (SAS 2019) (LNCS vol. 11822)*. Springer, 271–295. https://doi.org/10.1007/978-3-030-32304-2_14

[44] Francesco Ranzato and Marco Zanella. 2020. Abstract Interpretation of Decision Tree Ensemble Classifiers. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2020)*. 5478–5486. https://aaai.org/ojs/index.php/AAAI/article/view/5998

[45] Francesco Ranzato and Marco Zanella. 2021. *MetaSilvae* GitHub Repository. https://github.com/abstract-machine-learning/meta-silvae.

[46] Xavier Rival and Kwangkeun Yi. 2020. *Introduction to Static Analysis: An Abstract Interpretation Perspective*. The MIT Press.

[47] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T. Vechev. 2018. Fast and Effective Robustness Certification. In *Proc. Annual Conf. on Neural Information Processing Systems 2018 (NeurIPS 2018)*. 10825–10836. http://papers.nips.cc/paper/8278-fast-and-effective-robustness-certification

[48] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. 2019. An Abstract Domain for Certifying Neural Networks. *Proc. ACM Program. Lang.* 3, POPL 2019, Article 41 (Jan. 2019), 30 pages. https://doi.org/10.1145/3290354

[49] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev. 2019. Boosting Robustness Certification of Neural Networks. In *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*. OpenReview.net. https://openreview.net/forum?id=HJgeEh09KQ

[50] M. Srinivas and L. M. Patnaik. 1994. Genetic algorithms: a survey. *Computer* 27, 6 (1994), 17–26.

[51] John Törnblom and Simin Nadjm-Tehrani. 2019. An Abstraction-Refinement Approach to Formal Verification of Tree Ensembles. In *Proc. 2nd Int. Workshop on Artificial Intelligence Safety Engineering, held with SAFECOMP*. Springer.

[52] John Törnblom and Simin Nadjm-Tehrani. 2020. Formal verification of input-output mappings of tree ensembles. *Sci. Comput. Program.* 194 (2020), 102450. https://doi.org/10.1016/j.scico.2020.102450

[53] Peter D. Turney. 1995. Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm. *J. Artif. Int. Res.* 2, 1 (April 1995), 369–409.

[54] Petra Vidnerová and Roman Neruda. 2020. Vulnerability of classifiers to evolutionary generated adversarial examples. *Neural Networks* 127 (2020), 168–181. https://doi.org/10.1016/j.neunet.2020.04.015