On ℓ_p-norm Robustness of Ensemble Decision Stumps and Trees

Yihan Wang [1] **Huan Zhang** [2] **Hongge Chen** [3] **Duane Boning** [3] **Cho-Jui Hsieh** [2]

Abstract

Recent papers have demonstrated that ensemble stumps and trees could be vulnerable to small input perturbations, so robustness verification and defense for those models have become an important research problem. However, due to the structure of decision trees, where each node makes decision purely based on one feature value, all the previous works only consider the ℓ_∞ norm perturbation. To study robustness with respect to a general ℓ_p norm perturbation, one has to consider the correlation between perturbations on different features, which has not been handled by previous algorithms. In this paper, we study the problem of robustness verification and certified defense with respect to general ℓ_p norm perturbations for ensemble decision stumps and trees. For robustness verification of ensemble stumps, we prove that complete verification is NP-complete for $p \in (0, \infty)$ while polynomial time algorithms exist for $p = 0$ or ∞. For $p \in (0, \infty)$ we develop an efficient dynamic programming based algorithm for sound verification of ensemble stumps. For ensemble trees, we generalize the previous multi-level robustness verification algorithm to ℓ_p norm. We demonstrate the first certified defense method for training ensemble stumps and trees with respect to ℓ_p norm perturbations, and verify its effectiveness empirically on real datasets.

1. Introduction

It has been observed that small human-imperceptible perturbations can mislead a well-trained deep neural network (Goodfellow et al., 2015; Szegedy et al., 2013), which leads to extensive studies on robustness of deep neural network models. In addition to strong attack methods that can find adversarial perturbations in both white-box (Carlini & Wagner, 2017; Madry et al., 2018; Chen et al., 2018; Zhang et al., 2019a; Xu et al., 2019) and black-box settings (Chen et al., 2017; Ilyas et al., 2018; Brendel et al., 2018; Cheng et al., 2019a; 2020), various algorithms have been proposed for formal robustness verification (Katz et al., 2017; Gehr et al., 2018; Zhang et al., 2018; Weng et al., 2018; Zhang et al., 2019d; Wang et al., 2018b) and improving the robustness of neural networks (Madry et al., 2018; Wong & Kolter, 2018; Wong et al., 2018; Zhang et al., 2019c;b).

In this paper, we consider the robustness of ensemble decision trees and stumps. Although tree based model ensembles, including Gradient Boosting Trees (GBDT) (Friedman, 2001) and random forest, have been widely used in practice, their robustness properties have not been fully understood. Recently, Cheng et al. (2019a); Chen et al. (2019a); Kantchelian et al. (2016) showed that adversarial examples also exist in ensemble trees, and several recent works considered the problem of robustness verification (Chen et al., 2019b; Ranzato & Zanella, 2019; 2020; Törnblom & Nadjm-Tehrani, 2019) and adversarial defense (Chen et al., 2019a; Andriushchenko & Hein, 2019; Chen et al., 2019e; Calzavara et al., 2019; 2020; Chen et al., 2019d) for ensemble trees and stumps. However, most of these works focus on evaluating and enhancing the robustness for ℓ_∞ norm perturbations, while ℓ_p norm perturbations with $p < \infty$ were not considered. Since each node or each stump makes decision by looking at only a single feature, the perturbations are independent across features in ℓ_∞ robustness verification and defense for tree ensembles, which makes the problem intrinsically simpler than the other ℓ_p norm cases with $p < \infty$. In fact, we will show that in some cases verifying ℓ_p norm and ℓ_∞ norm belong to different complexity classes – verifying ℓ_p norm robustness of an ensemble decision stump is NP-complete for $p \in (0, \infty)$ while polynomial time algorithms exist for $p = 0, \infty$.

In practice, robustness on a single ℓ_∞ norm is not sufficient – it has been demonstrated that an ℓ_∞ robust model can still be vulnerable to invisible adversarial perturbations in other ℓ_p norms (Schott et al., 2018; Tramèr & Boneh, 2019). Additionally, there are cases where an ℓ_p norm threat model is more suitable than ℓ_∞ norm. For instance, when the perturbation can be made only to few features, it should be modeled as an ℓ_0 norm perturbation. Thus, it is crucial to have robustness verification and defense algorithms that can work for general ℓ_p norms. In this paper, We give a comprehensive study of this problem for tree based models.

[1] Tsinghua University, Beijing, China [2]UCLA, Los Angeles, USA [3]MIT, Cambridge, USA. Correspondence to: Yihan Wang <wangyihan617@gmail.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

Table 1. Summary of the algorithms and their complexity for robustness verification of ensemble trees and stumps. Blue cells are the contribution of this paper.

	Verification method	ℓ_∞	ℓ_0	$\ell_p, p \in (0, \infty)$
Single Tree	complete	Linear (Chen et al., 2019b)	Linear (Sec 3.1)	Linear (Sec 3.1)
Ensemble Stump	complete	Polynomial (Andriushchenko & Hein, 2019)	Linearithmic (Sec 3.2)	NP-complete (Sec 3.2)
	incomplete	Not needed	Not needed	Approximate Knapsack (Sec 3.2)
Ensemble Tree	complete	NP-complete (Kantchelian et al., 2016)		
	incomplete	Multi-level (Chen et al., 2019b)	Extended Multi-level (Sec 3.3)	

Our contribution can be summarized as follows:

- In the first part of paper, we consider the problem of verifying ℓ_p norm robustness of tree and stump ensembles. For a single decision tree, similar to the ℓ_∞ norm case, we show that the problem of complete robustness verification of ℓ_p norm robustness can be done in linear time. However, for ensemble decision stump, although complete ℓ_∞ norm verification can be done in polynomial time, it's NP-complete for verifying ℓ_p norm robustness when $p \in (0, \infty)$. We then provide an efficient algorithm to conduct sound but incomplete verification by dynamic programming. For tree ensembles, the ℓ_p case is NP-complete for any p and we propose an efficient algorithm for computing a reasonably tight lower bound. Table 1 the algorithms proposed in our paper and previous works, as well as their complexity.

- Based on the proposed robustness verification algorithms, we develop training algorithms for ensemble stumps and trees that can improve certified robust test errors with respect to general ℓ_p norm perturbations. Experiments on multiple datasets verify that the proposed methods can improve ℓ_p norm robustness where the previous ℓ_∞ norm certified defense (Andriushchenko & Hein, 2019) cannot.

The rest of the paper is organized as follows. In Section 2, we introduce the robustness verification and certified defense problems. In Section 3, we discuss complexity and algorithms for ℓ_p norm robustness verification for ensemble stumps and trees. In Section 4, we show how to use our proposed verification algorithms to train ensemble stumps and trees with certified ℓ_p norm robustness. Experiments on multiple datasets are conducted in Section 5.

2. Background and Related Work

Background Assume $F : \mathbb{R}^d \to \{1, \dots, C\}$ is a C-way classification model, given a correctly classified example $\boldsymbol{x}_0$ with $F(\boldsymbol{x}_0) = y_0$, an adversarial perturbation is defined as $\boldsymbol{\delta} \in \mathbb{R}^d$ such that $F(\boldsymbol{x}_0 + \boldsymbol{\delta}) \neq y_0$.

Definition 1 (Robustness Verification Problem)**.** *Given $F, \boldsymbol{x}_0$ and a perturbation radius ϵ, the robustness verification problem aims to determine whether there exists an adversarial example within ϵ ball around $\boldsymbol{x}_0$. Formally, we determine whether the following statement is true:*

$$F(\boldsymbol{x}_0 + \boldsymbol{\delta}) = y_0, \ \ \forall \|\boldsymbol{\delta}\|_p \leq \epsilon. \tag{1}$$

Giving the exact "yes/no" answer to (1) is NP-complete for neural networks (Katz et al., 2017) and tree ensembles (Kantchelian et al., 2016). **Adversarial attack** algorithms are developed to find an adverarial perturbation $\boldsymbol{\delta}$ that satisfies (1). For example, several widely used attacks have been developed for attacking neural networks (Carlini & Wagner, 2017; Madry et al., 2018; Goodfellow et al., 2015) and other general classifiers (Cheng et al., 2019b; Chen et al., 2019c). However, adversarial attacks can only find adversarial examples which do not provide a **sound** safety guarantee — even if an attack fails to find an adversarial example, it does not imply no adversarial example exists.

Robustness verification algorithms aim to find a **sound** solution to (1) — they output yes for a subset of yes instances of (1). However they may not be **complete**, in the sense that it may not be able to answer yes for all the yes instances of (1). Therefore we will refer solving (1) exactly as the "complete verification problem", while in general a verification algorithm can be incomplete[1] (providing a sound but incomplete solution to (1)). Below we will review existing works on verification and their connections to certified defense.

Robustness verification For neural network, it has been shown complete verification is NP-complete for ReLU networks, so many recent works have been focusing on developing efficient (but incomplete) robustness verification algorithms (Wong & Kolter, 2018; Zhang et al., 2018; Weng et al., 2018; Singh et al., 2018; Wang et al., 2018b; Singh et al., 2019; Dvijotham et al., 2018). Many of them follow the linear or convex relaxation based approach (Salman et al., 2019), where (1) is solved as an optimization problem with relaxed constraints. However, since ensemble trees are discrete step functions, none of these neural network verification algorithms can be effectively applied.

Specialized algorithms are required for verifying tree ensembles. Kantchelian et al. (2016) first showed that complete verification for ensemble tree is NP-complete when

[1] In some works, incomplete verification is referred to as "approximate" verification where the goal is to guarantee a lower bound for the norm of the minimum adversarial example, or "relaxed" verification emphasizing the relaxation techniques used to solving an optimization problem related to (1).

there are multiple trees with depth ≥ 2. An integer programming method was proposed for complete verification which requires exponential time. Later on, a single decision tree is verified for evaluating robustness of an RL policy in (Bastani et al., 2018). More recently, Chen et al. (2019b) gave a comprehensive study on the robustness of tree ensemble models; Ranzato & Zanella (2020) and Ranzato & Zanella (2019) proposed a tree ensemble robustness and stability verification method based on abstract interpretation; and Törnblom & Nadjm-Tehrani (2019) introduced an abstraction-refinement procedure which iteratively refines a partition of the input space. However, all these previous works only consider ℓ_∞ perturbation model (i.e., setting the norm to be $\|\boldsymbol{\delta}\|_\infty$ in (1)). The ℓ_∞ norm assumption makes verification much easier on decision trees and stumps as perturbations can be considered independently across features, aligning with the decision procedure of tree based models.

Certified Defense Many approaches have been proposed to improve the robustness of a classifier, however evaluating a defense method is often tricky. Many works evaluate model robustness based on **empirical robust accuracy**, defined as the percentage of correctly classified samples under a specific set of attacks within a predefined threat model (e.g., an ℓ_p ϵ-ball) (Madry et al., 2018; Chen et al., 2019a). However, using such measurement can lead to a false sense of robustness (Athalye et al., 2018), since robustness against a specific kind of attack doesn't give a sound solution to (1). In fact, many proposed empirical defense algorithms were broken under more sophisticated attacks (Athalye et al., 2018; Tramer et al., 2020). Instead, certified adversarial defense algorithms evaluate the classifier based on **certified robust accuracy**, defined as the percentage of correctly classified samples for which the robustness can be verified within the ϵ ball. Most of the certified defense algorithms are based on finding the weights to minimize the certified robust loss measured by some robustness verification algorithms (Wong & Kolter, 2018; Wong et al., 2018; Wang et al., 2018a; Mirman et al., 2018; Zhang et al., 2019b).

Several recent works studied robust tree based models. In (Chen et al., 2019a), an adversarial training approach is proposed to improve ℓ_∞ norm robustness of random forest and GBDT. Chen et al. (2019e) proposed another empirical defense also for ℓ_∞ norm robustness. The only certified defense that can provide provable robustness guarantees is given in (Andriushchenko & Hein, 2019), where they proposed a boosting algorithm to improve the **certified robust error** of ensemble trees and stumps with respect to ℓ_∞ norm perturbation. This method cannot be directly extended to ℓ_p norm perturbations since it relies on independence between features: when one feature is perturbed, the perturbations of other features are irrelevant.

3. ℓ_p-norm Robustness Verification of Stumps and Trees

The robustness verification problem for ensemble trees and stumps requires us to solve (1) given a model $F(\cdot)$. For some of the cases, we will show that computing (1) exactly (complete robustness verification) is NP-complete, so in those cases we will propose efficient polynomial time algorithms for computing a sound but incomplete solution to the robustness verification problem.

Summary of our results For a single decision tree, Chen et al. (2019b) shows that ℓ_∞ robustness can be evaluated in linear time. We show that their algorithm can be extended to the ℓ_p norm case for $p \in [0, \infty]$. Furthermore, we can also extend the multi-level ℓ_∞ verification framework (Chen et al., 2019b) for tree ensembles to general ℓ_p cases, allowing efficient and sound verification for general ℓ_p norm. For evaluating the robustness of an ensemble decision stump, Andriushchenko & Hein (2019) showed that the ℓ_∞ case can be solved in polynomial time, but their algorithm uses the fact that features are uncorrelated under ℓ_∞ norm perturbations so cannot be used for any $p < \infty$ case. We prove that the ℓ_0 norm robustness evaluation can be done in linear time, while for the ℓ_p norm case with $p \in (0, \infty)$, the robustness verification problem is NP-complete. We then propose an efficient dynamic programming algorithm to obtain a good lower bound for verification.

3.1. A single decision tree

We first consider the simple case of a single decision tree. Assume the decision tree has n leaf nodes and for a given example x with d features, starting from the root, x traverses the intermediate tree levels until reaching a leaf node. Each internal node i determines whether x will be passed to left or right child by checking $\mathbf{I}(x_{t_i} > \eta_i)$, where t_i is the feature to spilt at in node i and η_i is the threshold. Each leaf node v_i has a value v_i indicating the prediction value of the tree.

If we define B^i as the set of input x that can reach leaf node i, due to the decision tree structure, B^i can be represented as a d-dimensional box:

$$B^i = (l_1^i, r_1^i] \times \cdots \times (l_d^i, r_d^i]. \tag{2}$$

Some of the l, r can be $-\infty$ or $+\infty$. As discussed in Section 3.1 of (Chen et al., 2019b), the box can be computed efficiently in linear time by traversing the tree. To certify whether there exists any misclassified points under perturbation $\|\delta\|_p \leq \epsilon$, we can enumerate boxes for all n leaf nodes and check the minimum distance from $\boldsymbol{x}_0$ to each box. The following proposition shows that the ℓ_p norm distance between a point and a box can be computed in $O(d)$ time, and thus the complete robustness verification problem for a single tree can be solved in $O(dn)$ time.

Proposition 1. *Given a box $B = (l_1, r_1] \times \cdots \times (l_d, r_d]$ and a point $\boldsymbol{x} \in \mathbb{R}^d$. The minimum ℓ_p distance ($p \in [0, \infty]$) from x to B is $\|z - x\|_p$ where:*

$$z_i = \begin{cases} x_i, & l_i \le x_i \le u_i \\ l_i, & x_i < l_i \\ u_i, & x_i > u_i. \end{cases} \tag{3}$$

We define the operator $\text{dist}_p(B, x)$ to be the minimum ℓ_p distance between x to a box B. We define the ℓ_p norm ball $\text{Ball}_p(x, \epsilon) = \{x' | \|x' - x\|_p \le \epsilon\}$, and we use $\cap$ to denote the intersection between a ℓ_p ball and a box. $B \cap \text{Ball}_p(x, \epsilon) \neq \emptyset$ if and only if $\text{dist}_p(B, x) \le \epsilon$.

3.2. Ensemble decision stumps

A decision stump is a decision tree with only one root node and two leaf nodes. We assume there are T decision stumps and the i-th decision stump gives the prediction

$$f^i(x) = \begin{cases} w_l^i & \text{if } x_{t_i} < \eta^i \\ w_r^i & \text{if } x_{t_i} \ge \eta^i. \end{cases}$$

The prediction of a decision stump ensemble $F(x) = \sum_i f^i(x)$ can be decomposed into each feature in the following way. For each feature j, assume $j_1, \ldots, j_{T_j}$ are the decision stumps using feature j, we can collect all the thresholds $[\eta^{j_1}, \ldots, \eta^{j_{T_j}}]$. Without loss of generality, assume $\eta^{j_1} \le \cdots \le \eta^{j_{T_j}}$ then the prediction values assigned in each interval can be denoted as

$$g^j(x_j) = v^{j_t} \quad \text{if } \eta^{j_t} < x_j \le \eta^{j_{t+1}} \tag{4}$$

where

$$v^{j_t} = w_l^{j_1} + \cdots + w_l^{j_t} + w_r^{j_{t+1}} + \cdots + w_r^{j_{T_j}},$$

and x_j is the value of sample x on feature j. The overall prediction can be written as the summation over the predicted values of each feature:

$$F(x) = \sum_{j=1}^{d} g^j(x_j), \tag{5}$$

and the final prediction is given by $y = sgn(F(x))$.

ℓ_0 ensemble stump verification Assume $F(x)$ is originally positive and we want to make it as small as possible by perturbing δ features (in this case, δ should be a positive integer). For each feature j, we want to know *the maximum decrease of prediction value by changing this feature,* which can be computed as

$$c^j = \min_t v^{j_t} - g^j(x_j), \tag{6}$$

and we should choose δ features with smallest c^j values to perturb. Let S_δ denotes the set with δ smallest c^j values, we have

$$\min_{\|x - x'\|_0 \le K} F(x') = F(x) + \sum_{i \in S_K} c^j. \tag{7}$$

Therefore verification can be done exactly in $O(T + d\log(d))$ time, where $O(d\log(d))$ is the cost of sorting d values $\{c^1, ..., c^d\}$.

ℓ_p ensemble stump verification The difficulty of ℓ_p norm robustness verification is that the perturbations on each feature are correlated, so we can't separate all the features as in (Andriushchenko & Hein, 2019) for the ℓ_∞ norm case. In the following, we prove that the complete ℓ_p norm verification is NP-complete by showing a reduction from Knapsack to ℓ_p norm ensemble stump verification. This shows that ℓ_p norm verification can belong to a different complexity class compared to the ℓ_∞ norm case.

Theorem 1. *Solving ℓ_p norm robustness verification (with soundness and completeness) as in Eq. (1) for an ensemble decision stumps is NP-complete when $p \in (0, \infty)$.*

Proof. We show that a 0-1 Knapsack problem can be reduced to an ensemble stump verification problem. A 0-1 Knapsack problem can be defined as follows. Assume there are T items each with weight w_i and value v_i, the (decision version of) 0-1 Knapsack problem aims to determine whether there exists a subset of items S such that $\sum_{i \in S} w_i \le C$ and with value $\sum_{i \in S} v_i \ge D$.

Now we construct a decision stump verification problem with T features and T stumps from the 0-1 Knapsack problem, where each decision stump corresponds to one feature. Assume x is the original example, we define each decision stump to be

$$g^i(s) = -v_i I(s > \eta_i) + \frac{D}{T}, \quad \text{where } \eta_i = x_i + w_i^{(1/p)}, \tag{8}$$

where $I(\cdot)$ is the indicator function. The goal is to verify ℓ_p robustness with $\epsilon = C^{(1/p)}$. We need to show that this robustness verification problem outputs YES ($\min_{\|x - x'\|_p \le \epsilon} \sum_i g^i(x'_i) < 0$) if and only if the Knapsack solution is also YES. If the verification found $v^* = \min_{\|x - x'\|_p \le \epsilon} \sum_i g^i(x'_i) < 0$, let x' be the corresponding solution of verification, then we can choose the following S for 0-1 Knapsack:

$$S = \{i \mid x'_i > \eta_i\} \tag{9}$$

It is guaranteed that

$$\sum_{i \in S} w_i = \sum_{i \in S} |\eta_i - x_i|^p \le \sum_i |x'_i - x_i|^p \le \epsilon^p = C \tag{10}$$

and by the definition of g^i we have $\sum_i g^i(x'_i) = D - \sum_{i\in S} v_i \le 0$, so this subset S will also be feasible for the Knapsack problem. On the other hand, if the 0-1 Knapsack problem has a solution S, for robustness verification problem we can choose x' such that

$$x'_i = \begin{cases} \eta_i & \text{if } i \in S \\ x_i & \text{otherwise} \end{cases}$$

By definition we have $\sum_i g^i(x'_i) = D - \sum_{i\in S} v_i < 0$. Therefore the Knapsack problem, which is NP-complete, can be reduced to ℓ_p norm decision stump verification problem with any $p \in (0, \infty)$ in polynomial time. □

Incomplete Verification for ℓ_p robustness Although it's impossible to solve ℓ_p verification for decision stumps in polynomial time, we show sound verification can be done in polynomial time by dynamic programming, inspired by the pseudo-polynomial time algorithm for Knapsack.

Let $\eta^{j_1}, \dots, \eta^{j_{T_j}}$ be the thresholds for feature j and $v^{j_1}, \dots, v^{j_{T_j}}$ be the corresponding values, our dynamic programming maintains the following value for each ϵ: "given maximal ϵ perturbation to the first j features, what's the minimal prediction of the perturbed x". We denote this value as $D(\epsilon, j)$, then the following recursion holds:

$$D(\epsilon, j+1) = \min_{\delta\in[0,\epsilon]} D(\epsilon-\delta, j) + C(\delta, j+1),$$

where $C(\delta, j+1) := \min_{|x'_j - x_j|<\delta} g^j(x'_j)$ which can be pre-computed. Note that δ, ϵ can be real numbers so exactly running this DP requires exponential time. Our approximate algorithm allows ϵ, δ only up to certain precision. If we choose precision ν, then we only consider values $\nu, 2\nu, \dots, P\nu$ (the smallest P with $P\nu > \epsilon$). To ensure the verification algorithm is sound, the recursion will become

$$\tilde{D}(a\nu, j+1) = \min_{b\in\{1,\dots,a\}} \tilde{D}((a-b+1)\nu, j) + C(b\nu, j+1), \tag{11}$$

and the final solution should be $\tilde{D}(\lceil\epsilon\rceil, d)$ where $\lceil\epsilon\rceil := P\nu$ means rounding ϵ up to the closest grid. Note that the $+1$ term in the recursion is to ensure that the resulting value is a lower bound of the original solution. The verification algorithm can verify a sample in $O(Pd+T)$ time , in which d is dimension and P is the number of discretizations.

3.3. ℓ_p norm verification for ensemble decision trees

Kantchelian et al. (2016) showed that for general ensemble trees, complete ℓ_∞ robustness verification can formulated as a mixed integer linear programming problem, which is NP-Complete, and Chen et al. (2019b) proposed a fast polynomial time hierarchical verification framework to verify the model to a desired precision. For a tree ensemble with T trees and an input example x, Chen et al. (2019b) first check all the leaf nodes of each tree and only keep the leaf nodes that x can reach under the given perturbation. In the ℓ_∞ case, both the perturbation ball of x and the decision boundary of a leaf node can be represented as boxes (see Sec. 3.1), therefore it is easy to check whether the two boxes have an intersection. Then T trees are splited into $\frac{T}{K}$ groups, each with K trees. Trees from different groups are considered independently; the K trees within a group form a graph where each size-K clique in this graph represents a possible prediction value of all trees within this group given ℓ_∞ input perturbation. Enumerating all size-K cliques allows us to obtain the worst case prediction of the K trees within a group, and then we can combine the worst case predictions of all $\frac{T}{K}$ groups (e.g., directly adding all of them) to obtain an over-estimated worst case prediction of the entire ensemble. The results can be tightened by considering each group as a "virtual tree" and merge virtual trees into a new level of groups.

The most important procedure in (Chen et al., 2019b) is to check whether a set of leaf nodes from different trees within a group can form a valid size-K clique, which involves checking the intersections among the decision boundaries of leaf nodes from different trees and the intersection among the clique and the perturbation ball. We extend this procedure to ℓ_p setting in our work following two steps:

First, we check the intersection between input perturbation $\text{Ball}_p(x, \epsilon)$ and a box B^i using Proposition 1. Initially, we only consider the set of leaf node that has $\text{dist}_p(B^i, \boldsymbol{x}) \le \epsilon$ (B^i is the decision boundary of a leaf).

Second, in ℓ_∞ case, since the ℓ_∞ perturbation ball is also a box, it is possible to use the boxicity property to obtain intersections which are represented as size-K cliques in Chen et al. (2019b).This boxicity property is not hold anymore for general ℓ_p input perturbations. Chen et al. (2019b) showed that for a set of ℓ_∞ boxes $\{B^1, \dots, B^T\}$, if $B^i \cap B^j \neq \emptyset$ for all i, j $(i \neq j)$, and $B^i \cap \text{Ball}_\infty(x, \epsilon) \neq \emptyset$ for all i, then it guarantees that $B^1 \cap B^2 ... \cap B^T \cap \text{Ball}_\infty(x, \epsilon) \neq \emptyset$. However, for ℓ_p $(p \neq \infty)$ norm perturbation, under the same condition cannot guarantee that $B^1 \cap B^2 ... \cap B^T \cap \text{Ball}_p(x, \epsilon) \neq \emptyset$. In fact, even if $\text{Ball}_p(x, \epsilon) \cap B^t \neq \emptyset$ for any t, $B^1 \cap B^2 ... \cap B^T \cap \text{Ball}_\infty(x, \epsilon)$ can still be empty. A counter example with ℓ_1 is shown in Figure 1 and similar counter examples can be found for any $p < \infty$.

Therefore, we need to check whether $\bar{B} := B^1 \cap \cdots \cap B^T$, which is still a box, has nonempty intersection with input perturbation $\text{Ball}_p(x, \epsilon)$. This step can be computed using Proposition 1, which costs $O(d)$ time. After this additional procedure, we can safely generalize the ℓ_∞ framework to $\ell_p (p \ge 0)$ cases by simply replacing the procedure. We include the detail algorithm for enumerating the size-K cliques in Appendix 1.

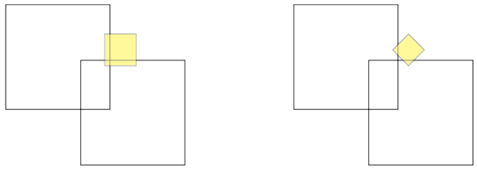

Figure 1. In the ℓ_p case, the perturbation ball is not a box and the general ℓ_p version of the Lemma 1 in (Chen et al., 2019b) is not true. Here we present a counter example in ℓ_1.

4. Training ℓ_p-robust Boosted Stumps and Trees

Based on the general ℓ_p verification algorithm for stump ensembles described in Section 3.2, we develop certified defense algorithms for training ensemble stumps and trees. The main challenge is that for $\ell_p(p > 0)$, different from the ℓ_∞ case, the correlation between features should be considered. Following the setting in (Andriushchenko & Hein, 2019), we use an exponential loss function L, where for a point $(x, y) \in \mathbb{R}^d \times \{-1, 1\}$, $L(yf(x)) = \exp(-yf(x))$. However, our algorithms can be generalized to other strictly monotonic and convex loss functions. We consider each training example $(x, y) \in \mathbb{S}$ is perturbed in $\text{Ball}_p(x, \epsilon)$, $\mathbb{S}$ is the training set.

4.1. ℓ_p robust boosted stumps

Given a decision stump ensemble $F(x) = \sum_{i=1}^{T} f_i(x)$ with T stumps, without loss of generality, we assume the first $T - 1$ stumps, defined as $F_{T-1}(x) = \sum_{i=1}^{T-1} f_i(x)$, are already trained and fixed, and our target is to update F with a new stump $f_T(x)$. Here we define a stump as $f(x) = w_l + \mathbf{1}_{x_j \geq b} w_r$ which splits the space at threshold b on feature j and predict w_l (left leaf prediction) or $w_l + w_r$ (right leaf prediction). Our goal is to select the 4 parameters (b, j, w_l, w_r) robustly by minimizing the minimax loss:

$$\min_{j,b,w_l,w_r} \sum_{(x,y)\in\mathbb{S}} \max_{\|\delta\|_p\leq\epsilon} L(yF(x+\delta)) \tag{12}$$

To solve this optimization, we first consider a sub-problem which finds the optimal w_r^* and w_l^* for a fixed split (j', b').

$$\begin{aligned} w_l^*, w_r^* = &\arg\min_{w_l,w_r} \sum_{(x,y)\in\mathbb{S}} \max_{\|\delta\|_p\leq\epsilon} L(yF(x+\delta)) \\ &\text{s.t. } j = j', b = b' \end{aligned} \tag{13}$$

For the inner maximization, we note that the loss function is monotonically decreasing, therefore we can replace the maximization as an minimization inside the loss function:

$$\begin{aligned} &\max_{\|\delta\|_p\leq\epsilon} L(yF(x+\delta)) \\ &= \max_{\|\delta\|_p\leq\epsilon} L\left(yF_{T-1}(x+\delta) + yf_T(x+\delta)\right) \\ &= L\left(\min_{\|\delta\|_p\leq\epsilon} (yF_{T-1}(x+\delta) + yf_T(x+\delta))\right) \\ &= L\left(\min_{\|\delta\|_p\leq\epsilon} (yF_{T-1}(x+\delta) + yw_l + yw_r\mathbf{1}_{x_{j'}+\delta_{j'}\geq b'})\right) \end{aligned}$$

The inner minimization can then be considered as a stump ensemble verification problem. According to Section 3.2, for each x, we can derive a lower bound of the inner minimization, denoted as $\tilde{D}(\lceil\epsilon\rceil, d)$:

$$\begin{aligned} &\min_{\|\delta\|_p\leq\epsilon} (yF_{T-1}(x+\delta) + yw_l + yw_r\mathbf{1}_{x_{j'}+\delta_{j'}\geq b'}) \\ &\geq \tilde{D}_{(x,y)}(\lceil\epsilon\rceil, d). \end{aligned}$$

For simplicity, we omit subscript (x, y) in the analysis below. Our goal is to give $\tilde{D}(\lceil\epsilon\rceil, d)$ as a function of w_l and w_r. This requires a small extension to the DP based verification algorithm. In (11), we can consider the d features in any order. We can solve the DP by first solving all other $d - 1$ features except j', and obtain $\tilde{D}_{\setminus j'}(a\nu, j)$ for all $a \in \{1, \cdots, P\}$ and $j \in \{1, \cdots .d - 1\}$ (we denote the DP table as $\tilde{D}_{\setminus j'}$ to emphasize that it does not include feature j'). $\tilde{D}_{\setminus j'}(a\nu, d - 1)$ is a lower bound of the minimum prediction value under perturbation $a\nu$ excluding all stumps involving feature j'. Then, the recursion for $\tilde{D}(\lceil\epsilon\rceil, d)$ needs to consider the minimum of two settings, representing the left or right leaf is selected for the last stump:

$$\begin{aligned} &\tilde{D}(\lceil\epsilon\rceil, d) = \min\left(\tilde{D}_L(\lceil\epsilon\rceil, d), \tilde{D}_R(\lceil\epsilon\rceil, d)\right) \\ &\tilde{D}_L(\lceil\epsilon\rceil, d) = \min_{a\in[P]} \left(\tilde{D}_{\setminus j'}((P-a+1)\nu, d-1) + C_L(a\nu, j')\right) \\ &\qquad + yw_l \\ &\tilde{D}_R(\lceil\epsilon\rceil, d) = \min_{a\in[P]} \left(\tilde{D}_{\setminus j'}((P-a+1)\nu, d-1) + C_R(a\nu, j')\right) \\ &\qquad + y(w_r + w_l) \\ &C_L(a\nu, j) = \min_{|x_j - x'_j|\leq a\nu, x'_j < b'} g^j(x') \\ &C_R(a\nu, j) = \min_{|x_j - x'_j|\leq a\nu, x'_j \geq b'} g^j(x') \end{aligned} \tag{14}$$

In (14), $\tilde{D}_L(\lceil\epsilon\rceil, d)$ and $\tilde{D}_R(\lceil\epsilon\rceil, d)$ denote the minimum prediction value of the sample (x, y) when perturbed into the left or right side of the split (j', b'). $C_L(a\nu, j), C_R(a\nu, j)$ denote the minimum prediction when x is perturbed into the left or right side of the split on feature j with perturbation $a\nu$, where $g^j(x)$ is defined as in (4) but with the last tree $f_T(x)$ excluded (i.e., computed on F_{T-1}).

After obtaining the lower bound of the inner minimization, instead of solving the original optimization (13), here we solve

$$w_l^*, w_r^* = \arg\min_{w_l,w_r} \sum_{(x,y)\in\mathcal{S}} L(\tilde{D}_{(x,y)}(\lceil\epsilon\rceil, d)). \tag{15}$$

Theorem 2. *$\sum_{(x,y)\in\mathcal{S}} L(\tilde{D}_{(x,y)}(\lceil\epsilon\rceil, d))$ defined in (15) is jointly convex in w_l, w_r.*

The proof can be found in the Appendix D. Based on this theorem, we can use coordinate descent to solve the minimization: fix w_r and minimize over w_l, then fix w_l and minimize over w_r (similar to Andriushchenko & Hein (2019)). For exponential loss, when w_r is fixed, we can use a closed form solution to update w_l (see Appendix B). When w_l is fixed, we use bisection to get the optimal w_r. For general loss functions, both w_l and w_r can be solved by bisection.

After estimating w_l^*, w_r^* in (13), we can iterate over all the possible split positions (j, b) and select the position with minimum robust loss. Our proposed general ℓ_p norm robust training algorithm for stump ensembles can train a new stump in $O(TN(Pd{+}T){+}dBT)$ time, where B is the number of candidate bs and N is the size of dataset. For fixed j, ϵ and precision, $\tilde{D}_{\setminus j'}(a\nu, d-1)$ is fixed for all $a \in [P]$ and can be pre-calculated, which costs $O(N(Pd{+}T))$ time. And in implementation, we only need to calculate $T+1$ different $\tilde{D}_{\setminus j'}(a\nu, d-1)$, which costs $O(TN(Pd+T))$ time. After obtaining $\tilde{D}_{\setminus j'}(a\nu, d-1)$, in each iteration, $\tilde{D}(\lceil\epsilon\rceil, d)$ can be derived in $O(T)$ time (despite having P discretizations, there are only T possible values in the minimization in Eq. (14), and an efficient implementation can exploit this fact). The bisection searching for w_l^* and w_r^* can also be finished in $O(1)$ time with fixed parameters. Thus the above algorithm can train a stump ensemble in $O(TN(Pd+T)+dBT)$ time.

4.2. ℓ_p robust boosted trees

Single decision tree Our goal is to solve (12) where F is a single tree. Different from the ℓ_∞ case, in ℓ_p cases, perturbation on one dimension can reduce the possible perturbation on other dimensions. Therefore, when updating a stump ensemble, perturbation bound ϵ will be consumed along the trajectory from the tree root to leaf nodes. Because the number of features is typically more than the depth of a decision tree, we use each feature only once along one trajectory on the decision tree. We define $S = \{(x,y)\in\mathbb{S} | \text{dist}_p(x, B^{N_k}) \le \epsilon\}$ as the set of samples that can fall into node N_k under ℓ_p norm ϵ bounded perturbation, and $(N_0, N_1, ..., N_{k-1})$ as the sequence of nodes on the trajectory from tree root N_0 to tree node N_k. Each node N_t $(0 \le t < k)$ contains a split (j_t, b_t) which splits the space on feature j_t at value b_t.

In the ℓ_p norm case, each example has an unique perturbation budget at node N_k, as some of the perturbation budget has been consumed in parent nodes splitting other features. For each sample (x, y), ℓ_p norm bounded perturbation in node N_k can be calculated along the trajectory by $\epsilon(x) = (\epsilon^p - \sum_{t\in E}(x_{j_t} - b_t)^p)^{\frac{1}{p}}$, where E is a subset of the node trajectory in which x and N_{t+1} are on the different sides of node N_t, $\forall t \in E$. Formally, we can define E as $\{t : t < k-1, \mathbf{1}(x_{j_t} \ge b_t) \ne \mathbf{1}(N_{t+1} \ge b_t)\}$, where $N_{t+1} \ge b_t$ denotes that $x'_{j_t} \ge b_t$, $\forall x' \in B^{N_{t+1}}$. This is different from previous works on ℓ_∞ perturbations. Now we consider training the node N_k and get the optimal parameters (j^*, b^*, w_l^*, w_r^*):

$$\begin{aligned} & j^*, b^*, w_l^*, w_r^* \\ & = \mathop{\arg\min}_{j,b,w_l,w_r} \sum_{(x,y)\in S} \max_{\|\delta\|_p \le \epsilon(x)} L(f(x+\delta)y), \end{aligned} \tag{16}$$

where $f(\cdot)$ is a new leaf node $f(x) = \mathbf{I}(x \ge b)w_r + w_l$, and when training node N_k, we only consider the training examples in S. The objective in (16) is similar to that in (12) except that there is only one stump to be trained. Therefore, we can use a similar procedure as in previous section to find the optimal parameters.

Boosted decision tree ensemble Given a tree ensemble with T trees $F(x) = \sum_{i=1}^{T} f_i(x)$, we fix the first $T-1$ trees and train a node N on the (T)-th decision tree $f_T(x)$. The optimization problem will be essentially the same as Eq. (16), but here for $(x, y) \in S$, we should also consider the first $T-1$ trees, along with prediction of node N:

$$\begin{aligned} & \max_{\|\delta\|_p \le \epsilon(x)} L(yF_T(x+\delta)) \\ & = \max_{\|\delta\|_p \le \epsilon(x)} L(yF_{T-1}(x+\delta) + y(w_l + \mathbf{1}_{x_j+\delta_j \ge b} w_r)) \\ & = L\left(\min_{\|\delta\|_p \le \epsilon(x)} (yF_{T-1}(x+\delta) + y(w_l + \mathbf{1}_{x_j+\delta_j \ge b} w_r))\right). \end{aligned}$$

Here $F_{T-1}(x)$ is the prediction from the ensemble of the first $T-1$ trees. We further lower bound the minimization:

$$\begin{aligned} & \min_{\|\delta\|_p \le \epsilon(x)} (yF_{T-1}(x+\delta) + y(w_l + \mathbf{1}_{x_j+\delta_j \ge b} w_r)) \\ & \ge \min_{\|\delta\|_p \le \epsilon(x)} yF_{T-1}(x+\delta) + \min_{\|\delta\|_p \le \epsilon(x)} y(w_l + \mathbf{1}_{x_j+\delta_j \ge b} w_r) \end{aligned}$$

The first part is the ℓ_p robustness verification for tree ensemble, which is challenging to solve efficiently during training time. Here we apply a relatively loose lower bound of $yF_{T-1}(x)$, where

$$\sum_{i=1}^{T-1} \min_{\|\delta\|_p \le \epsilon(x)} (yf(x+\delta)) \le \min_{\|\delta\|_p \le \epsilon(x)} yF_{T-1}(x)$$

We simply sum up the worst prediction on each previous tree, which can be easily maintained during training. By doing this relaxation, the problem is reduced to building a single tree to boost the ℓ_p norm robustness.

4.3. ϵ schedule

When features are correlated in ℓ_p cases, we find that it is important to have an ϵ schedule during the training process

Table 2. **General ℓ_p-norm ensemble stump verification.** This table reports verified test error (verified err.) and average per sample verification time (avg. time) of each method. For our proposed DP based verification, precision is also reported. For ℓ_0 verification, we report verified errors with $\epsilon_0 = 1$ (changing 1 pixels). For ℓ_0 norm, we also report r^*, which is the average the number features that can be perturbed at most while the prediction stays the same.

Dataset		ℓ_1 MILP (complete)		Ours ℓ_1 DP approx. (incomplete)			Ours vs. MILP		Ours ℓ_0 (complete) verification		
name	ϵ_∞	verified err.	avg. time	precision	verified err.	avg. time	MILP/ours	speedup	avg. robust r^*	verified err.	avg. time
breast-cancer	0.3	10.94%	.030s	0.01	10.94%	.00025s	1.00	120X	.04	95.62%	.0006s
diabetes	0.05	35.06%	.017s	0.0002	35.06%	.0004s	1.00	40X	.0	100.0%	.0005s
Fashion-MNIST shoes	0.1	10.45%	.105s	0.005	10.55%	.0013s	.99	80.8X	2.09	16.35%	.010s
MNIST 1 vs. 5	0.3	3.30%	0.11s	0.005	3.35%	0.0013s	1.00	71X	3.33	4.50%	.010s
MNIST 2 vs. 6	0.3	9.64%	0.099s	0.005	9.69%	.0012s	.98	82X	1.22	26.43%	.012s

Table 3. **General ℓ_p-norm tree ensemble verification.** We report verified test error (verified err.) and average per-example verification time (avg. time) of each method. K: size of cliques; L: number of levels in multi-level verification (defined similarly as in (Chen et al., 2019b)). Our ℓ_p incomplete verification can obtain results very close to complete verification (MILP), with huge speedups.

Dataset		ℓ_1 MILP		Ours ℓ_1 approx.				Ours vs. MILP	
name	ϵ	verified err.	avg. time	K	L	verified err.	avg. time	ratio of verified err.	speedup
breast-cancer	0.3	8.03%	.036s	3	2	8.03%	.012s	1.00	3X
diabetes	0.05	33.12%	.027s	3	2	33.12%	.012s	1.00	2.25X
Fashion-MNIST shoes	0.1	10%	.091s	3	2	10%	.011s	1.00	8.23X
MNIST 1 vs. 5	0.3	4.20%	0.088s	3	2	4.20%	.011s	1.00	8X
MNIST 2 vs. 6	0.3	8.60%	.098s	3	2	8.80%	.012s	.98	8.17X

– the ϵ increases gradually from small to large, instead of using a fixed large ϵ in the beginning. If one directly uses a large ϵ in the beginning, the first few stumps will allow too much perturbation and the later stumps tend to allow fewer perturbation, making it harder to explore the correlation between features. In ensemble stump training, we increase the ϵ when training a new stump, and in ensemble tree training, we increase the ϵ when height of the tree grows. We also include the choice of ϵ schedules in Appendix D.1.

5. Experimental Results

In this section we empirically test the proposed algorithms for ℓ_p robustness verification and training. The code is implemented in Python and all the experiments are conducted on a machine with 2.7 GHz Intel Core i5 CPU with 8G RAM. Our code is publicly available at https://github.com/YihanWang617/On-ell_p-Robustness-of-Ensemble-Stumps-and-Trees

5.1. ℓ_p stump and tree ensemble verification

ℓ_p stump ensemble verification We evaluate our incomplete ℓ_p verification method for stump ensembles on five real datasets. Ensembles are robustly trained using the ℓ_∞ training procedure proposed in (Andriushchenko & Hein, 2019), each of which contains 20 stumps.

For the ℓ_1 norm robustness verification problem, we have shown it's NP-complete to conduct complete verification. To demonstrate the tightness and efficiency of the proposed Dynamic Programming (DP) based verification, we also run the Mixed Integer Linear Programming (Kantchelian et al., 2016) to conduct complete verification, which can take exponential time. In Table 2, we can find that the proposed DP algorithm gives almost exactly the same bound as MILP, while being $50 - 100$ times faster. This speedup guarantees its further applications in certified robust training.

For the ℓ_0 norm robustness verification, we propose a linearithmic time algorithm for complete verification. The results for $\epsilon_0 = 1$ (changing only 1 feature) are also reported in Table 2. We can observe that the proposed method can conduct complete verification in less than 0.1 second. We find that some models are not robust to ℓ_0 perturbations with high verified errors. Since our verification method is complete, these models suffer from adversarial examples that change classification outcome by changing only 1 pixel.

ℓ_p tree ensemble verification We evaluate our incomplete ℓ_p verification method for tree ensembles on five real datasets. Ensemble models being verified are robustly trained with (Andriushchenko & Hein, 2019), each of which contains 20 trees.

Again, we compare our proposed algorithm with MILP-based complete verification (Kantchelian et al., 2016) which can take exponential time to get the exact bound. The results are presented in Table 3, and parameters of the proposed method (K and L) are also reported. We observe that the proposed verification method gets very tight verified errors while being much faster than the MILP solver.

Table 4. ℓ_1 **robust training for stump ensembles.** We report standard errors and ℓ_1 verified errors of our training methods (ℓ_1 training) versus the previous ℓ_∞ training algorithm. ϵ is the perturbation bound for each dataset. For $\epsilon = 1.0$ in mnist dataset, we train the models using $\epsilon_\infty = 0.3$. Our proposed ℓ_1 training can significantly reduce the ℓ_1 verified error, and the previous ℓ_∞ approach cannot as it was designed to reduce ℓ_∞ error only. We conduct a similar experiment for ℓ_2 norm in Appendix E.2.

Dataset				standard training		ℓ_∞ training(Andriushchenko & Hein, 2019)		ℓ_1 training (ours)	
name	ϵ_∞	ϵ_1	n. stumps	standard err.	verified err.	standard err.	verified err.	standard err.	verified err.
breast-cancer	0.3	1.0	20	0.73%	95.62%	4.37%	99.27%	1.46%	**35.77%**
diabetes	0.05	0.05	20	21.43%	37.66%	29.2%	35.06%	27.27%	**31.82%**
Fashion-MNIST shoes	0.1	0.1	20	6.60%	69.85%	7.50%	10.45%	7.10%	**10.35%**
	0.2	0.5	40	5.05%	87.5%	9.25%	57.05%	12.40%	**32.20%**
MNIST 1 vs. 5	0.3	0.3	20	1.23%	58.76%	1.68%	3.30%	1.28%	**2.81%**
	0.3	1.0	40	0.59%	66.01%	1.33%	17.46%	4.49%	**16.23%**
MNIST 2 vs. 6	0.3	0.3	20	3.17%	92.46%	4.52%	9.64%	3.71%	**8.24%**
	0.3	1.0	40	2.81%	99.49%	3.91%	44.22%	7.73%	**33.46%**

Table 5. ℓ_1 **robust training for tree ensembles.** We report standard and ℓ_1 robust test error for all the three methods. We also report ϵ for each dataset, and the number of trees in each ensemble. We also report the results of ℓ_2 robust training for tree ensembles in Appendix E.2.

Dataset					standard training		ℓ_∞ training (Andriushchenko & Hein, 2019)		ℓ_1 training (ours)	
name	ϵ_∞	ϵ_1	n. trees	depth	standard err.	verified err.	standard err.	verified err.	standard err.	verified err.
Fashion-MNIST shoes	0.2	0.5	5	5	4.65%	99.85%	7.85%	89.54%	18.71%	**65.18%**
breast-cancer	0.3	1.0	5	5	0.73%	99.26%	0.73%	99.63%	9.56%	**47.05%**
MNIST 1 vs. 5	0.3	0.8	5	5	0.64%	97.38%	0.64%	64.11%	4.59%	**36.23%**
MNIST 2 vs. 6	0.3	0.6	5	5	4.12%	100.0%	1.96%	52.33%	7.64%	**39.67%**

5.2. ℓ_p robust stump and tree ensemble training

ℓ_p **robust stump training** We evaluate our proposed certified training methods on two small size datasets and three medium-size datasets. All the models are trained with standard training, ℓ_∞ robust training (Andriushchenko & Hein, 2019) and our proposed general ℓ_p robust training algorithm (in experiments, we set $p = 1$. We also report the $p = 2$ results in Appendix E.2). Models of the same dataset are trained with the same set of hyperparameters (details can be found in the Appendix). We evaluate ℓ_1 verified test error using MILP. In our experiments, we choose different ϵ_∞ and ϵ_p such that the ℓ_∞ and ℓ_p perturbation balls do not contain each other. Standard error and verified robust test error of each model are reported in Table 4. We also report ℓ_∞ robustness of these models in Appendix E.1. We observe that the proposed training method can successfully get a more robust model against ℓ_1 perturbation compared to the previous ℓ_∞-norm only training method.

ℓ_p **robust tree training** We evaluate our ℓ_p robust training method for trees on subsets of three medium size datasets (dataset statistics can be found in the Appendix). We report the results of ℓ_1 robust training tree ensembles in Tables 5, and results of ℓ_2 robust training in Appendix E.2. It shows that our algorithm achieves better or at least comparable verified error in most cases. In addition, we also conduct an example to test the performance of certified training with respect to number of trees. In Figure 2, we compare ℓ_∞ and ℓ_1 robust training on fashion-mnist dataset and monitor

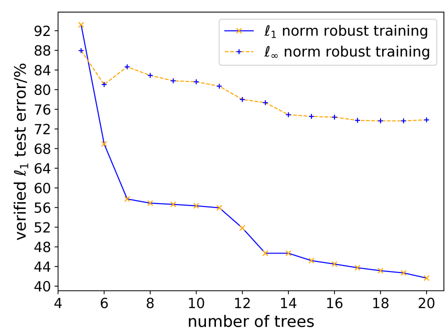

Figure 2. ℓ_1 **and** ℓ_∞ **robust training on fashion-mnist dataset** ($\epsilon_\infty = 0.2$ **and** $\epsilon_1 = 0.5$)**.** We compare verified errors during training when the number of stumps increases.

the performance over the first 20 stumps (the ϵ scheduling length is 5). We can observe that when number of stumps increases, the our ℓ_1 robust training can indeed gradually reduce ℓ_1 verified test error, where the ℓ_∞ robust training (as a reference) can only slightly improve ℓ_1 robustness.

6. Conclusion

In this paper, we first develop methods to efficiently verify the general ℓ_p norm robustness for tree-based ensemble models. Based on our proposed efficient verification algorithms proposed, we further derive the first ℓ_p norm certified robust training algorithms for ensemble stumps and trees.

Acknowledgement

We acknowledge Maksym Andriushchenko and Matthias Hein for providing their ℓ_∞ certified training code. This work is partially supported by NSF IIS-1719097, Intel, Google cloud and Facebook. Huan Zhang is supported by the IBM fellowship.

References

Andriushchenko, M. and Hein, M. Provably robust boosted decision stumps and trees against adversarial attacks. In *NeurIPS*, 2019.

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Bastani, O., Pu, Y., and Solar-Lezama, A. Verifiable reinforcement learning via policy extraction. In *Advances in Neural Information Processing Systems*, pp. 2494–2504, 2018.

Brendel, W., Rauber, J., and Bethge, M. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *ICLR*, 2018.

Calzavara, S., Lucchese, C., Tolomei, G., Abebe, S. A., and Orlando, S. Treant: Training evasion-aware decision trees. *arXiv preprint arXiv:1907.01197*, 2019.

Calzavara, S., Lucchese, C., Marcuzzi, F., and Orlando, S. Feature partitioning for robust tree ensembles and their certification in adversarial scenarios. *arXiv preprint arXiv:2004.03295*, 2020.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pp. 39–57. IEEE, 2017.

Chen, H., Zhang, H., Chen, P.-Y., Yi, J., and Hsieh, C.-J. Attacking visual language grounding with adversarial examples: A case study on neural image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2587–2597, 2018.

Chen, H., Zhang, H., Boning, D., and Hsieh, C.-J. Robust decision trees against adversarial examples. In *ICML*, 2019a.

Chen, H., Zhang, H., Si, S., Li, Y., Boning, D., and Hsieh, C.-J. Robustness verification of tree-based models. In *NeurIPS*, 2019b.

Chen, J., Jordan, M. I., and Wainwright, M. J. Hopskipjumpattack: A query-efficient decision-based adversarial attack. *arXiv preprint arXiv:1904.02144*, 2019c.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 15–26. ACM, 2017.

Chen, Y., Wang, S., Jiang, W., Cidon, A., and Jana, S. Training robust tree ensembles for security. *arXiv preprint arXiv:1912.01149*, 2019d.

Chen, Y., Wang, S., Jiang, W., Cidon, A., and Jana, S. Cost-aware robust tree ensembles for security applications. *arXiv preprint arXiv:1912.01149*, 2019e.

Cheng, M., Le, T., Chen, P.-Y., Yi, J., Zhang, H., and Hsieh, C.-J. Query-efficient hard-label black-box attack: An optimization-based approach. In *ICLR*, 2019a.

Cheng, M., Le, T., Chen, P.-Y., Zhang, H., Yi, J., and Hsieh, C.-J. Query-efficient hard-label black-box attack: An optimization-based approach. In *International Conference on Learning Representations*, 2019b. URL https://openreview.net/forum?id=rJlk6iRqKX.

Cheng, M., Singh, S., Chen, P., Chen, P.-Y., Liu, S., and Hsieh, C.-J. Sign-opt: A query-efficient hard-label adversarial attackh. In *ICLR*, 2020.

Dvijotham, K., Stanforth, R., Gowal, S., Mann, T., and Kohli, P. A dual approach to scalable verification of deep networks. *UAI*, 2018.

Friedman, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., and Vechev, M. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18. IEEE, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Ilyas, A., Engstrom, L., Athalye, A., and Lin, J. Query-efficient black-box adversarial examples. In *ICLR*, 2018.

Kantchelian, A., Tygar, J., and Joseph, A. Evasion and hardening of tree ensemble classifiers. In *ICML*, 2016.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pp. 97–117. Springer, 2017.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Mirman, M., Gehr, T., and Vechev, M. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pp. 3578–3586, 2018.

Ranzato, F. and Zanella, M. Robustness verification of decision tree ensembles. *OVERLAY@ AI* IA*, 2509:59–64, 2019.

Ranzato, F. and Zanella, M. Abstract interpretation of decision tree ensemble classifiers. In *AAAI*, pp. 5478–5486, 2020.

Salman, H., Yang, G., Zhang, H., Hsieh, C.-J., and Zhang, P. A convex relaxation barrier to tight robustness verification of neural networks. *arXiv preprint arXiv:1902.08722*, 2019.

Schott, L., Rauber, J., Bethge, M., and Brendel, W. Towards the first adversarially robust neural network model on mnist. *arXiv preprint arXiv:1805.09190*, 2018.

Singh, G., Gehr, T., Mirman, M., Püschel, M., and Vechev, M. Fast and effective robustness certification. In *NIPS*, 2018.

Singh, G., Gehr, T., Püschel, M., and Vechev, M. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):41, 2019.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *ICLR*, 2013.

Törnblom, J. and Nadjm-Tehrani, S. An abstraction-refinement approach to formal verification of tree ensembles. In *International Conference on Computer Safety, Reliability, and Security*, pp. 301–313. Springer, 2019.

Tramèr, F. and Boneh, D. Adversarial training and robustness for multiple perturbations. In *Advances in Neural Information Processing Systems*, pp. 5866–5876, 2019.

Tramer, F., Carlini, N., Brendel, W., and Madry, A. On adaptive attacks to adversarial example defenses. *arXiv preprint arXiv:2002.08347*, 2020.

Wang, S., Chen, Y., Abdou, A., and Jana, S. Mixtrain: Scalable training of formally robust neural networks. *arXiv preprint arXiv:1811.02625*, 2018a.

Wang, S., Pei, K., Whitehouse, J., Yang, J., and Jana, S. Efficient formal safety analysis of neural networks. In *NIPS*, 2018b.

Weng, T.-W., Zhang, H., Chen, H., Song, Z., Hsieh, C.-J., Boning, D., Dhillon, I. S., and Daniel, L. Towards fast computation of certified robustness for relu networks. In *ICML*, 2018.

Wong, E. and Kolter, J. Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*, 2018.

Wong, E., Schmidt, F., Metzen, J. H., and Kolter, J. Z. Scaling provable adversarial defenses. In *NIPS*, 2018.

Xu, K., Chen, H., Liu, S., Chen, P.-Y., Weng, T.-W., Hong, M., and Lin, X. Topology attack and defense for graph neural networks: an optimization perspective. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 3961–3967. AAAI Press, 2019.

Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. Efficient neural network robustness certification with general activation functions. In *NIPS*, 2018.

Zhang, H., Chen, H., Song, Z., Boning, D., Dhillon, I. S., and Hsieh, C.-J. The limitations of adversarial training and the blind-spot attack. In *ICLR*, 2019a.

Zhang, H., Chen, H., Xiao, C., Li, B., Boning, D., and Hsieh, C.-J. Towards stable and efficient training of verifiably robust neural networks. *arXiv preprint arXiv:1906.06316*, 2019b.

Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., and Jordan, M. I. Theoretically principled trade-off between robustness and accuracy. *arXiv preprint arXiv:1901.08573*, 2019c.

Zhang, H., Zhang, P., and Hsieh, C.-J. Recurjac: An efficient recursive algorithm for bounding jacobian matrix of neural networks and its applications. In *AAAI*, 2019d.