Proceedings of the
1st Workshop on Artificial Intelligence and Formal Verification, Logics, Automata and Synthesis (OVERLAY),
Rende, Italy, November 19–20, 2019

Robustness Verification of Decision Tree Ensembles

Francesco Ranzato[1] and Marco Zanella[2]

[1,2]Dipartimento di Matematica, University of Padova, Italy
[1]ranzato@math.unipd.it
[2]mzanella@math.unipd.it

Abstract

We study the problem of formally and automatically verifying robustness properties of decision tree ensemble classifiers such as random forests and gradient boosted decision tree models. A recent stream of works showed how abstract interpretation can be successfully deployed to formally verify neural networks. In this work we push forward this line of research by designing a general abstract interpretation-based framework for the formal verification of robustness and stability properties of decision tree ensemble models. Our method may induce complete robustness checks of standard adversarial perturbations or output concrete adversarial attacks. We implemented our abstract verification technique in a tool called silva, which leverages an abstract domain of not necessarily closed real hyperrectangles and is instantiated to verify random forests and gradient boosted decision trees. Our experimental evaluation on the MNIST dataset shows that silva provides a precise and efficient tool which advances the current state-of-the-art in tree ensembles verification.

1 Introduction

Adversarial machine learning [2, 9, 12] is a hot topic studying vulnerabilities of machine learning (ML) in adversarial scenarios. Adversarial examples have been found in diverse application fields of ML such as image classification, spam filtering, malware detection, and the current defense techniques include adversarial model training, input validation, testing and automatic verification of learning algorithms [9]. Formal verification of ML classifiers started to be an active field of investigation, in particular for robustness properties of (deep) neural networks. A classifier is stable for some (typically very small) perturbation of its input objects which represents an adversarial attack when it assigns the same class to all the samples within that perturbation, so that impercetible malicious alterations of input objects should not deceive a stable classifier. Formal verification methods for neural networks may rely on a number of different techniques: linear approximation of functions [25, 26], semidefinite relaxations [14], logical SMT solvers [10, 11], symbolic interval propagation, [23] abstract interpretation [8, 17, 18] or hybrid synergistic approaches [1, 19, 24]. Abstract interpretation [6] is a *de facto* standard technique used since forty years for designing static analysers and verifiers of programming languages. Recently, abstract interpretation has been successfully applied for designing precise and scalable robustness verification tools of (deep) neural network models [8, 17, 18]. While all these verification techniques consider neural networks as ML model,

in this work we focus on decision tree ensemble methods, such as random forests and gradient boosted decision tree models, which are widely applied in different fields having sensible adversarial scenarios, notably image classification, malware detection, intrusion detection and spam filtering. It is only very recently that adversarial attacks and robustness issues of decision tree ensembles started to be a subject of investigation [4, 5, 7, 20, 22, 21].

Contributions: Following the aforementioned stream of works applying abstract interpretation for certifying ML models, we design a general abstract interpretation-based framework for the formal verification of stability properties of decision tree ensemble models. Our verification algorithm of ensembles of decision trees: (1) is domain agnostic, since it can be instantiated to any abstract domain which represents properties of real vectors, such as simple hyperrectangles of real intervals or involved linear relations; (2) is firmly based on the basic soundness principle of abstract interpretation and correspondingly rely on sound approximations of functions used in decision tree classifiers; (3) under certain assumptions may induce complete robustness checks against adversarial perturbations; (4) is able to output concrete adversarial samples. Our formal verification methodology has been implemented in C in a tool called *silva* (Latin for *forest*) which leverages an abstract domain of possibly open real hyperrectangles and has been applied to random forests and gradient boosted decision trees. Our experimental evaluation on the standard MNIST dataset shows that *silva* provides a precise and efficient tool both for deriving the robustness metrics of forest classifiers and for performing a complete check of the robustness to adversarial perturbations of input samples. Our evaluation also compared the performance of *silva* against a recent robustness verification tool of tree ensembles [22], and this showed that *silva* improves on the current state-of-the-art. As an example, the following three images:

show: on the left, an original image O from the test set of MNIST, correctly classified as 7 by a random forest using 100 decision trees with maximum depth 100; in the middle, an image A which is automatically generated by *silva* as adversarial attack of O and obtained as output of the robustness verification of O for a perturbation ± 1 of the brightness values of its pixels; on the right, an image showing which pixels were changed from O to A (gray means unchanged, black $+1$, white -1). The image A is an adversarial example because it is classified as either 2 or 7 (i.e., there is a tie between the scores of 2 and 7) by the same 100×100 random forest which classifies O as 7.

2 Experimental Evaluation

We implemented a tool named *silva* (Latin word for *forest* which stands for *Silvarum Interpretation Valens Lator Analysis*, efficient analyzer of forests by (abstract) interpretation) whose source code in C (about 5K LOC) is publicly available on GitHub [15].

The underlying algorithm, whose details are omitted here, takes a shape representing a perturbation applied to a sample, then iteratively decomposes it into finer abstract partitions until it manages to prove (or disprove) stability. This procedure is guaranteed to converge due to monotonicity of the refinement step, hence *silva* is *complete* and returns a definitive answer for every sample.

We used *silva* for inferring stability of random forest classifiers on MNIST, which have been trained with different combinations of number of trees, maximum tree depth, splitting criteria (Gini and entropy impurity measures) and voting schemes (average and max). We consider a classifier *stable* on a sample with respect to a perturbation whenever every point in the perturbed region is labeled as the original sample. We also consider the four possible interaction between accuracy and stability, hence we say classifier on a sample is *robust* when it is correct and stable, *vulnerable* when it is wrong but stable, *fragile* when a it is correct but unstable and *broken* when it is wrong and unstable. We compared the performance of *silva* against a recent robustness verification tool of tree ensembles called VoTE [22], both for random forests and gradient boosted decision trees. In our experiments RFs have been trained by scikit-learn while CatBoost has been used for GBDTs. Since all these forest classifiers rely on univariate hard splits of

type $\mathbf{x}_i \leq k$, *silva* has been instantiated to the hyperrectangle abstract domain $\mathcal{H}$. Our experiments were run on a AMD Ryzen 7 1700X 3.0GHz CPU.

Setup: An automatic verifier Ver of robustness properties of a ML classifier C could be used for two main purposes:

P_1: to assess the robustness properties of C by inferring some stability metrics on a large test set T.

P_2: to check the robustness of C on an input sample $\mathbf{x} \in X$ against some adversarial perturbation $Prt(\mathbf{x})$; when Ver infers that C is not robust on $\mathbf{x}$ then Ver should also output a (set of) counterexample(s) in $Prt(\mathbf{x})$;

A complete verifier such as *silva* always outputs a TRUE/FALSE answer for each input sample, provided that computational time and memory constraints are met. When a complete verifier Ver is used for reaching a purpose of type P_1, it makes sense to add a timeout option to Ver to set a time limit for verifying the stability of an input sample $\mathbf{x}$, so that if the timeout applies then the stability check on $\mathbf{x}$ is considered to be inconclusive. In our experiments using *silva* for evaluating the robustness properties of tree ensemble classifiers, we utilized a timeout per sample, so that the stability metrics may be given by a percentage ranging within an interval, thus taking into account inconclusive stability checks.

Our experimental evaluation makes use of the standard MNIST dataset. Let us recall that MNIST consists of 70000 gray scale pictures of hand-written digits (with $\mathcal{L} = \{0, ..., 9\}$), where each image of 28×28 pixels is encoded as a vector of 784 integer values in $[0, 255]$ representing the brightness of a pixel (0 black, 255 white). The standard training set of MNIST consists of 60000 samples, while its test set T includes the remaining 10000 samples. We considered the standard perturbation $Prt_{\infty,\epsilon}$ [3] with $\epsilon = 1$, meaning that adversarial attacks may brighten/darken each pixel up to a 0.5% magnitude. Hence, for each input image $\mathbf{x}$, its perturbation $Prt_{\infty,1}(\mathbf{x})$ includes 3^{784} possible attacks.

Stability of RFs: In a preliminary study we compare accuracy, stability and execution time for random forests trained with the Gini and entropy criteria and using different voting schemes (average and max). We observed that forests trained with the entropy criterion tend to be more stable without negative effects on the accuracy, while difference between the two voting schemes becomes negligible for forests of realistic size. Based on this assessment, for the successive experiments we considered random forests trained with the entropy criterion and using the average score, which appears to be an optimal configuration for the stability metric. Fig.1 depicts the relationship between stability and accuracy, where darker points represent forests with lower depth d.

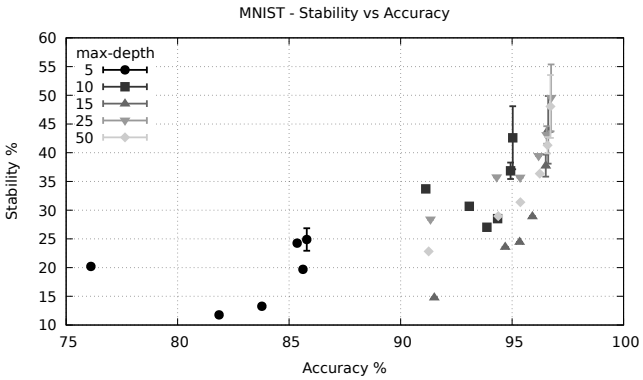

Figure 1: Comparison of different Random Forests (MNIST)

The number B of trees of these RFs is not depicted in the above chart and ranges in $\{5, 10, 15, 25, 50, 75\}$. The vertical bars represent intervals of stability due to inconclusive checks within the 1 second per sample timeout. We observe that RFs having the same maximum depth d tend to cluster together and that for RFs with $d \leq 10$ no clear interpretation of their stability can be derived, thus suggesting that RFs with $d \leq 10$ should not be considered for assessing the stability on MNIST. On the other hand, deeper forests with $d \geq 15$ tend to a vertical alignment, thus revealing a growing stability with comparable accuracy.

Fixed a depth d, we may observe that both accuracy and stability grow with B. It is worth remarking that for a given accuracy, the highest stability is achieved by RFs with depth $d = 25$ rather than $d = 50$, thus hinting that to increase d beyond some threshold may not positively affect the accuracy while reducing the stability. This finding shows that by taking into account both accuracy and stability the overall effectiveness of a random forest model does not necessarily increases with its size. In particular, the 75×25 random forest turns out to be the most accurate and stable. Tab.1 also displays the stability-based metrics defined earlier.

RF							
B	d	acc. %	stab. %	rob. %	frag. %	vuln. %	break. %
5	5	76.1	20.2	18.0	58.1	2.2	21.7
5	25	91.3	28.4	27.9	63.5	0.5	8.1
5	50	91.3	22.8	22.4	68.9	0.4	8.3
25	5	85.6	19.7	19.2	66.4	0.5	13.9
25	25	96.2	39.4	39.3	56.8	0.1	3.7
25	50	96.2	36.4	36.2	60.0	0.1	3.6
50	5	85.4	24.2	23.6	61.7	0.6	14.0
50	25	96.5	43.1 ±1.5	43.0 ±1.5	53.5 ±1.5	0.1	3.4
50	50	96.6	41.3 ±1.4	41.2 ±1.4	55.4 ±1.4	0.1	3.3

Table 1: Results for Random Forests (MNIST)

RF					GBDT				
B	d	TpS(s)	TpS10(s)	MTpS	B	d	TpS(s)	TpS10(s)	MTpS
25	5	0.00	0.00	0.04	50	10	0.00	0.00	0.01
25	10	0.00	0.01	0.60	75	5	0.00	0.00	0.02
50	5	0.07	0.66	34.69	75	10	0.00	0.01	0.32
50	10	0.44	4.36	42.25	100	10	0.01	0.15	32.99

Table 2: Analysis time measures.

Here, it is worth remarking that: (i) stability and robustness are closely related; (ii) vulnerability decreases with the overall size $B \times d$ of the RF, and for RFs having size $\leq 50 \times 25$ we found a significant percentage of misclassified input samples ($\approx 0.4\%$) with adversarial attacks which are consistently classified with the same wrong label; (iii) breakage appears to converge towards $\approx 3.5\%$ for larger forests.

Verification Time per Sample: In a context of using *silva* as a complete verifier for checking whether a given input sample $\mathbf{x} \in X$ is robust against the adversarial perturbation $Prt_{\infty,1}(\mathbf{x})$, Tab.2 displays the average verification Time per Sample (TpS), the average verification Time for the Samples whose verification time is above the 90th percentile of the distribution (TpS10), the Maximum verification Time per Sample (MTpS). It is worth observing that the worst case verification time of *silva* never exceeds 1 minute and that the average verification time on the hardest input samples is always less than 5 seconds.

Comparison with VoTE: Tab.3 shows the results of the comparison of *silva* with VoTE, a recent robustness verifier for tree ensemble classifiers based on an abstraction-refinement approach [22]. We replicated the experiments on the MNIST dataset as described in the paper [22] and compared the results in terms of robustness and verification time (on our machine). Each experiment is run on RFs and GBDTs trained with the same parameters: RFs are trained using scikit-learn with Gini/average parameters, while GBDTs are trained by CatBoost with default learning rate and softmax voting scheme. We followed [22] by setting an overall timeout of 7 hours for the verification time of the full test set of MNIST.

RF		*silva*			VoTE			GBDT		*silva*			VoTE		
B	d	acc.%	rob.%	time(s)	acc.%	rob.%	time(s)	B	d	acc.%	rob.%	time(s)	acc.%	rob.%	time(s)
25	5	85.7	16.3	4	84.5	13.6	8	50	10	95.6	94.0	1	95.3	92.4	5
25	10	94.1	24.2	20	94.1	25.7	11	75	5	94.8	91.3	1	94.7	89.6	4
50	5	85.8	17.8	576	86.1	14.2	833	75	10	96.0	93.4	6	96.0	91.9	116
50	10	94.4	30.2	4353	94.6	31.4	7704	100	10	96.2	93.0	147	96.4	91.9	974
75	5	86.2	19.8±0.4	8289	86.0	–	timeout	150	10	96.7	92.2±0.4	9459	96.7	–	timeout

Table 3: Comparison with VoTE

The difference between *silva* and VoTE on the accuracies of the input RFs and GBDTs is negligible and due to the randomness of the learning algorithm. The difference on the robustness ratios inferred by *silva* and VoTE may reach 3.6% (for the RF 50×5): the reasons why this happens are not clear and would require an in-depth inspection of the VoTE source code. Overall, it turns out that *silva* is faster than VoTE: on the larger forests verified by VoTE within 7 hours, *silva* achieves a speedup factor of ≈ 1.7 on the RF of size 50×10 and of ≈ 6.6 on the GBDT of size 100×10. The table also shows that by applying a timeout of 60 seconds per sample on the RF 75×5 and the GBDT 150×10, *silva* is able to output in at most 2.5 hours a rather precise estimate ($\pm 0.4\%$) of the robustness metric for these large RF and GBDT where VoTE exceeds the 7 hours limit.

3 Future Work

We believe that this work represents a step forward in applying formal verification methods to machine learning models, in particular a very well known program analysis technique such as abstract interpretation. As a benefit of this principled approach, we singled out the role of abstract interpretation for designing a complete verifier of robustness properties of decision tree classifiers. We think that more advanced techniques could be used for abstracting combinations of paths in different decision trees, as those successfully applied in static program analysis (e.g. trace partitioning [16]). We also plan to resort to abstract interpretation in order to train decision tree classifiers which are provably robust, namely, to apply abstract interpretation to training algorithms rather than to trained classification algorithms, similarly to the approach of [13] for training provably robust neural networks.

References

[1] G. Anderson, S. Pailoor, I. Dillig, and S. Chaudhuri. Optimization and abstraction: A synergistic approach for analyzing neural network robustness. In *Proc. 40th ACM Conf. on Programming Language Design and Implementation (PLDI 2019)*, pages 731–744. ACM, 2019.

[2] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. J. Goodfellow, A. Madry, and A. Kurakin. On evaluating adversarial robustness. *CoRR*, abs/1902.06705, 2019.

[3] N. Carlini and D. A. Wagner. Towards evaluating the robustness of neural networks. In *Proc. of 2017 IEEE Symposium on Security and Privacy (SP2017)*, pages 39–57, 2017.

[4] H. Chen, H. Zhang, D. S. Boning, and C. Hsieh. Robust decision trees against adversarial examples. In *Proc. 36th Int. Conf. on Machine Learning, ICML 2019*, volume 97, pages 1122–1131, 2019.

[5] H. Chen, H. Zhang, S. Si, Y. Li, D. S. Boning, and C. Hsieh. Robustness verification of tree-based models. *CoRR*, abs/1906.03849, 2019.

[6] P. Cousot and R. Cousot. 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)*, pages 238–252. ACM, 1977.

[7] G. Einziger, M. Goldstein, Y. Sa'ar, and I. Segall. Verifying robustness of gradient boosted models. In *Proc. 33rd AAAI Conf. on Artificial Intelligence*, pages 2446–2453, 2019.

[8] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. T. Vechev. AI2: safety and robustness certification of neural networks with abstract interpretation. In *Proc. 2018 IEEE Symposium on Security and Privacy (SP2018)*, pages 3–18, 2018.

[9] I. Goodfellow, P. McDaniel, and N. Papernot. Making machine learning robust against adversarial inputs. *Commun. ACM*, 61(7):56–66, 2018.

[10] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu. Safety verification of deep neural networks. In *Proc. Intern. Conf. on Computer Aided Verification (CAV2017)*, pages 3–29. Springer, 2017.

[11] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In R. Majumdar and V. Kunčak, editors, *Proc. Intern. Conf. on Computer Aided Verification (CAV2017)*, pages 97–117. Springer, 2017.

[12] A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial machine learning at scale. In *Proc. 5th Int. Conf. on Learning Representations (ICLR 2017)*, 2017.

[13] M. Mirman, T. Gehr, and M. Vechev. Differentiable abstract interpretation for provably robust neural networks. In *Proc. Int. Conf. on Machine Learning (ICML 2018)*, pages 3575–3583, 2018.

[14] A. Raghunathan, J. Steinhardt, and P. Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Proc. Annual Conf. on Neural Information Processing Systems (NeurIPS 2018)*, pages 10900–10910, 2018.

[15] F. Ranzato and M. Zanella. *Silva*, a tool for robustness verification of decision tree ensembles. https://github.com/abstract-machine-learning/silva, 2019.

[16] X. Rival and L. Mauborgne. The trace partitioning abstract domain. *ACM Trans. Program. Lang. Syst.*, 29(5), Aug. 2007.

[17] G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. T. Vechev. Fast and effective robustness certification. In *Proc. Annual Conf. on Neural Information Processing Systems 2018, (NeurIPS 2018)*, pages 10825–10836, 2018.

[18] G. Singh, T. Gehr, M. Püschel, and M. Vechev. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.*, 3(POPL 2019):41:1–41:30, Jan. 2019.

[19] G. Singh, T. Gehr, M. Püschel, and M. T. Vechev. Boosting robustness certification of neural networks. In *Proc. 7th Int. Conf. on Learning Representations (ICLR 2019)*. OpenReview.net, 2019.

[20] J. Törnblom and S. Nadjm-Tehrani. Formal verification of random forests in safety-critical applications. In *Proc. 6th Int. Workshop on Formal Techniques for Safety-Critical Systems (FTSCS 2018)*, pages 55–71. Springer, 2018.

[21] J. Törnblom and S. Nadjm-Tehrani. An abstraction-refinement approach to formal verification of tree ensembles. In *Proc. 2nd Int. Workshop on Artificial Intelligence Safety Engineering, held with SAFECOMP*. Springer, 2019.

[22] J. Törnblom and S. Nadjm-Tehrani. Formal verification of input-output mappings of tree ensembles. *arXiv preprint arXiv:1905.04194*, 2019.

[23] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana. Efficient formal safety analysis of neural networks. In *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS 2018)*, pages 6369–6379, 2018.

[24] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana. Formal security analysis of neural networks using symbolic intervals. In *Proc. 27th USENIX Security Symposium*, pages 1599–1614. USENIX Association, 2018.

[25] T. Weng, H. Zhang, H. Chen, Z. Song, C. Hsieh, L. Daniel, D. S. Boning, and I. S. Dhillon. Towards fast computation of certified robustness for ReLU networks. In *Proc. 35th Int. Conf. on Machine Learning, (ICML 2018)*, pages 5273–5282, 2018.

[26] H. Zhang, T. Weng, P. Chen, C. Hsieh, and L. Daniel. Efficient neural network robustness certification with general activation functions. In *Proc. Annual Conference on Neural Information Processing Systems (NeurIPS 2018)*, pages 4944–4953, 2018.