On the Robustness of Ensemble-Based Machine Learning Against Data Poisoning

Marco Anisetti, Claudio A. Ardagna, Alessandro Balestrucci, Nicola Bena, Ernesto Damiani, Chan Yeob Yeun

Abstract—Machine learning is becoming ubiquitous. From financial to medicine, machine learning models are boosting decision-making processes and even outperforming humans in some tasks. This huge progress in terms of prediction quality does not however find a counterpart in the security of such models and corresponding predictions, where perturbations of fractions of the training set (poisoning) can seriously undermine the model accuracy. Research on poisoning attacks and defenses even predates the introduction of deep neural networks, leading to several promising solutions. Among them, ensemble-based defenses, where different models are trained on portions of the training set and their predictions are then aggregated, are getting significant attention, due to their relative simplicity and theoretical and practical guarantees. The work in this paper designs and implements a hash-based ensemble approach for ML robustness and evaluates its applicability and performance on random forests, a machine learning model proved to be more resistant to poisoning attempts on tabular datasets. An extensive experimental evaluation is carried out to evaluate the robustness of our approach against a variety of attacks, and compare it with a traditional monolithic model based on random forests.

Index Terms—Machine Learning; Poisoning; Ensemble; Random forest

1 Introduction

With the introduction of deep neural networks in the last decade, machine learning (ML) is now leaving academia and powering an increasing the number of applications, from finance [1], to smart grid [2], weather forecast [3], and medicine [4]. This diffusion is also due to the fact that machine learning models are reportedly performing better than humans in some specific tasks [5].

Given their increasingly-widespread diffusion, even in critical application scenarios, it is of paramount importance to properly evaluate the security of ML models. As such, one of the most relevant threat vectors are data, being ML models trained on (very) large datasets. In particular, *poisoning attacks* include attacks carried out during training time by maliciously altering the dataset used for training, with the aim of decreasing the overall classification accuracy, or misclassifying some specific inputs when the model is deployed. Poisoning attacks have been reported in several application scenarios, from malware detection [6], to biometrics [7], healthcare [8], and source code completion [9] and against several types of machine learning models, from support vector machines [10], to decision trees [8], random forests [6], and neural networks [11], to name but a few. Solutions counteracting poisoning are vary, and range from improving the poisoned dataset by removing or repairing (suspicious) data points [12]–[16], to strengthening the ML model itself, to make it less subject to poisoning [17]–[19]. Among the latter, the usage of ensemble is becoming one of the preferred and most studied approaches [17]–[19]. It consists of training several ML models on different (possibly partially overlapped) partitions of the training set, and then aggregating their predictions. These solutions can also provide a theoretical bound on the correctness of the prediction according to the extent of poisoning.

However, ensembles are mostly studied in image recognition scenarios with deep neural networks [17]–[19], leaving open questions on their effectiveness in different scenarios such as their application to tabular datasets and ensemble-based ML algorithms (e.g., random forest).

Our paper aims to fill in these research gaps, by evaluating the robustness of a hash-based ensemble approach against data poisoning. A hash-based ensemble is based on hash functions to route data points in the original training set in different partitions used to train different models in the ensemble. We consider a random forest algorithm as our worst case, being an ensemble algorithm by itself, and a variety of *untargeted* poisoning attacks. We implement the hash-based ensemble-based approach in [18], where *i)* each model of the ensemble is trained on a disjoint partition of the training set to which data points are assigned according to hashing, and *ii)* the final prediction is retrieved according to majority voting.

Contrary to state of the art, our paper evaluates ensembles of small to moderate size (i.e., up to 21 random forests). Throughout finer-grained experiments, we show that the usage of even the smallest ensemble does protect from poisoning. In addition, we show that plain random forests are highly-sensitive to perturbations based on label flipping, while almost insensitive to others, a result consistent with

Research supported, in parts, by Technology Innovation Institute (TII) under Grant 8434000394, and Università degli Studi di Milano under the program "Piano sostegno alla ricerca".

- *Marco Anisetti, Claudio A. Ardagna, Nicola Bena, Ernesto Damiani are with the Department of Computer Science, Università degli Studi di Milano, Milan, Italy. Alessandro Balestrucci is with Consorzio Interuniversitario per l'Informatica. Ernesto Damiani, Chan Yeob Yeun are with Khalifa University of Science and Technology, Abu Dhabi, UAE. E-mail: {marco.anisetti, claudio.ardagna, nicola.bena, ernesto.damiani}@unimi.it, alessandro.balestrucci@consorzio-cini.it, {ernesto.damiani, chan.yeun}@ku.ac.ae*

other work in literature [11].

Our contribution is twofold. We first evaluate the robustness of a hash-based ensemble approach [18] to increase the robustness of random forests against poisoning attacks in a tabular dataset. We then evaluate such robustness by varying the poisoning perturbation and comparing the performance of our approach and a plain vanilla random forest.

The remainder of this paper is organized as follows. Section 2 discusses the state of the art in the context of poisoning attacks and defenses. Section 3 presents the poisoning protection based on ensemble of random forests, whose robustness evaluation against the perturbations in Section 4 is the goal of this paper. Section 5 describes the evaluation process, while Section 6 details the result of such a process. We conclude our paper with a discussion on the obtained insights in Section 7, and we finally draw our conclusions in Section 8.

2 Background and Related Work

The research community has worked hard to strengthen the security of machine learning (ML) models. Sophisticated attacks are continuously proposed and novel defenses defined to counteract them.

Attacks can be classified according to the stage where the attack occurs. On one side, *adversarial attacks* occur at inference time and consist of specially-crafted data points that are routed to the ML model to cause a faulty or wrong inference. Their goal is in fact the misclassification of such data points. On the other side, *poisoning attacks*, the focus of this paper, occur at training time and inject poisoned data points in the dataset. They aim to reduce the accuracy of the model or cause the misclassification of specific data points at inference time.

Poisoning attacks alter the dataset with malicious data points. They are created by perturbing existing data points in terms of *i)* samples or values of the features [20]; *ii)* labels, having the advantage of not creating strangely-looking data points [10], [21], [22]. Perturbations can be crafted according to *i)* a specific goal such as misclassification of *positive* data points, for instance in spam detection (*targeted poisoning*), requiring sophisticated perturbations such as *feature collision* [23]; *ii)* , or accuracy reduction only (*untargeted poisoning*). The latter often corresponds to random perturbations [10], and is the focus of this paper.

Defenses against poisoning attacks can be performed in two main ways: *dataset strengthening* or *model strengthening*.[1] Dataset strengthening aims to increase the quality of the dataset by removing or repairing poisoned data points, detected with some heuristics. Detection heuristics include clustering [15], [16], [22], influence of data points based on their gradients [12], [24], and outlier detection [15], to name but a few. Healing techniques include *randomized smoothing* and *differential privacy*. In the former, each data point is *smoothed* (i.e., its label is replaced) according to its neighbour data points. Smoothing has been initially proposed to counteract inference-time attacks [25], and then adapted to poisoning [14]. Similarly, in the latter, noise is added during training such that predictions of a model trained on the original dataset are indistinguishable from those of a model trained on the corresponding poisoned dataset, up to a certain ϵ [13], [26].

Model strengthening aims to increase the robustness of the ML model by altering the model itself, such that the effect of poisoning is reduced. The main technique for healing is ensemble, when the ML model is replaced with an ensemble of the same ML model [17]–[19]. This technique splits the training set in different partitions according to some strategies, and each partition is the training set of a model of the ensemble. Intuitively, this reduces the influence of poisoned data points, since each model is trained on a smaller fraction of poisoned data points. Some of the above techniques, including ensemble [17]–[19], randomized smoothing [14], and differential privacy [13], can provide a certifiable guarantee such that the model prediction is correct up to a certain amount of poisoning on the dataset.

3 Our Approach

Our hash-based ensemble approach focused on increasing the robustness of ML models is built on the following pillars.

- *Strong base models (no neural networks)*: traditional ensemble models rely on *weak base models* such as decision trees, whose predictions are strengthened by aggregation [27]. On the other hand, ensemble-based poisoning protections are based on neural networks (e.g., [17], [18]). We rather consider random forests as base models. Being random forest an algorithm natively built on an ensemble of models, we consider a worst case scenario of an ensemble of ensembles. As we will discuss in Section 6, this poses our evaluation in a scenario that is already resilient to a range of poisoning attacks and generally outperforms other classes of machine learning algorithms [28], [29].
- *Small number of base models*: ensemble-based protections (e.g., [18]) utilize a large number of partitions and base models (>1000 in some cases). We instead consider smaller numbers (up to 21) to increase usability and evaluate the degree of protection.
- *Tabular datasets for binary classification*: most of attacks and defenses are benchmarked in scenarios where images are the inputs of the models (i.e., feature extraction is performed directly within the model) [18]. We instead consider tabular datasets for binary classification, which are still a significant portion of ML.
- *Untargeted poisoning*: most of defenses are benchmarked against targeted poisoning (e.g., [15]), where few specially-crafted data points are injected in the training set. Our evaluation departs from the assumption of a generic attacker model where the attacker is free to alter the dataset with targeted manipulations, and focuses instead on untargeted poisoning, where the attacker alters existing data points according to default perturbations with the aim of reducing the accuracy of the model.

Figure 1 shows an overview of our approach based on ensemble of random forests. Given a tabular dataset, it

[1] We note that this distinction is not always sharp.

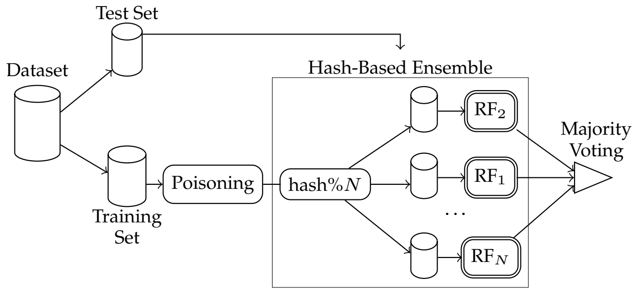

Figure 1: Overview of our hash-based ensemble approach.

extracts a portion forming the training set (e.g., 80%). The training set can be then poisoned according to our generic attack model and using the perturbations in Section 4. Following the work in [18], our approach retrieves the hash value of each data point of the training set according to a given hash function. It then partitions such set in N subsets, with N being the number of random forests of the ensemble. Each data point is assigned to a partition according to the modulo operator (modulo N) applied on the corresponding hash value (e.g., data points whose hash value modulo N is 0 are assigned to partition 0, and so on). Each n-th partition corresponds to the training set of the n-th random forest of the ensemble. In other words, each random forest is trained on a disjoint partition of the training set. The final prediction is then retrieved according to majority voting.

Following the implementation of the ensemble approach in Figure 1, we comparatively evaluate the robustness of plain random forests and ensemble of random forests against different poisoning attacks in Section 4. This is, to the best of our knowledge, a first tentative in this direction.

4 Poisoning Attacks

The poisoning attack strategy adopted in our ensemble in Section 3 is built on different perturbations (*zeroing, noising, out-of-ranging, label flipping*) each implementing a specific poisoning attack that is tested independently. Each perturbation takes as input a training set, denoted as D, and returns as output the poisoned training set, denoted as $\widetilde{D}$. Other inputs include the percentage of data points and features to alter (denoted as ϵ_{p} and ϵ_{f}, respectively), according to the specific perturbation. Poisoned training sets are then partitioned in disjoint training sets, each used to train a model of the ensemble, according to the evaluation process in Section 5.1. We note that each perturbation selects the data points and the corresponding features to poison at random according to ϵ_{p} and ϵ_{f}. In particular, the selected features are the same for every perturbation, to ensure proper comparison.

Throughout the description of the perturbations, we consider as example a binary classification task (classes A and B), and a 5-feature data point p with value $\langle 0, 10, 15, 0, 1\rangle$ belonging to class A. We consider the second feature to be selected for poisoning.

Perturbation *zeroing* produces a poisoned training set $\widetilde{D}$, where the selected data points are perturbed by changing the values of the selected features to 0. For instance, the corresponding poisoned data point of p has value $\langle 0, \mathbf{0}, 15, 0, 1\rangle$.

Perturbation *noising* produces a poisoned training set $\widetilde{D}$, where the selected data points are perturbed by replacing the values of the selected features with a value within the distribution of the same feature in the opposite class. For instance, let us consider the second feature of data point p. It takes value in $[0, 10]$ for class A, and $[20, 40]$ for class B. The corresponding poisoned data point has value $\langle 0, \mathbf{37}, 15, 0, 1\rangle$.

Perturbation *out-of-ranging* produces a poisoned training set $\widetilde{D}$, where the selected data points are perturbed by changing the value of the selected features with values outside their valid range. For instance, let us consider the second feature of data point p. The corresponding poisoned data point has value $\langle 0, -\mathbf{1}, 15, 0, 1\rangle$.

Finally, perturbation *label flipping* produces a poisoned training set $\widetilde{D}$, where the selected data points are perturbed by flipping their labels. For instance, the corresponding poisoned data point of p (belonging to class A) has the same values of p with label changed to B.

We note that the effectiveness of these perturbations strongly depend on the number and value of data points *actually* perturbed. For instance, let us consider perturbation *zeroing*. Assuming that the first feature of data point p (with value 0) is selected for poisoning, the corresponding poisoned data point is not altered.

5 Evaluation Process

We present the evaluation process and experimental settings at the basis of the experimental results in Section 6.

5.1 Evaluation Process in a Nutshell

We implemented the evaluation process in Figure 2 to validate the robustness of our ensemble approach in Section 3 against poisoning attacks in Section 4. To this aim, for each attack, we calculate the accuracy loss between the plain (monolithic) model and our ensemble approach with majority voting trained on both original and poisoned dataset. Figure 2 shows two different paths with and without poisoning, generating poisoned data or forwarding the original data respectively. It also shows the path used to generate the monolithic model.

Our process takes as input: *i)* the original dataset; *ii)* the number of random forests N composing the ensemble; *iii)* the perturbation type; *iv)* the percentage of data points ϵ_{p} and *v)* features ϵ_{f} to poison.

The evaluation process consists of four steps as follows.

Step 1: Preparation. It splits the dataset into training (denoted as D) and test sets, that is, *held out*. Test set is left untouched for the rest of process.

Step 2: Poisoning. It applies the selected perturbation to the training set D producing a poisoned training set $\widetilde{D}$, according to the percentages of poisoning received as input.

Step 3: Creation of training sets. It builds the training sets for the monolithic and ensemble models. It first creates N empty sets (*partitions*). Second, given a (original or poisoned) training set, each data point is converted

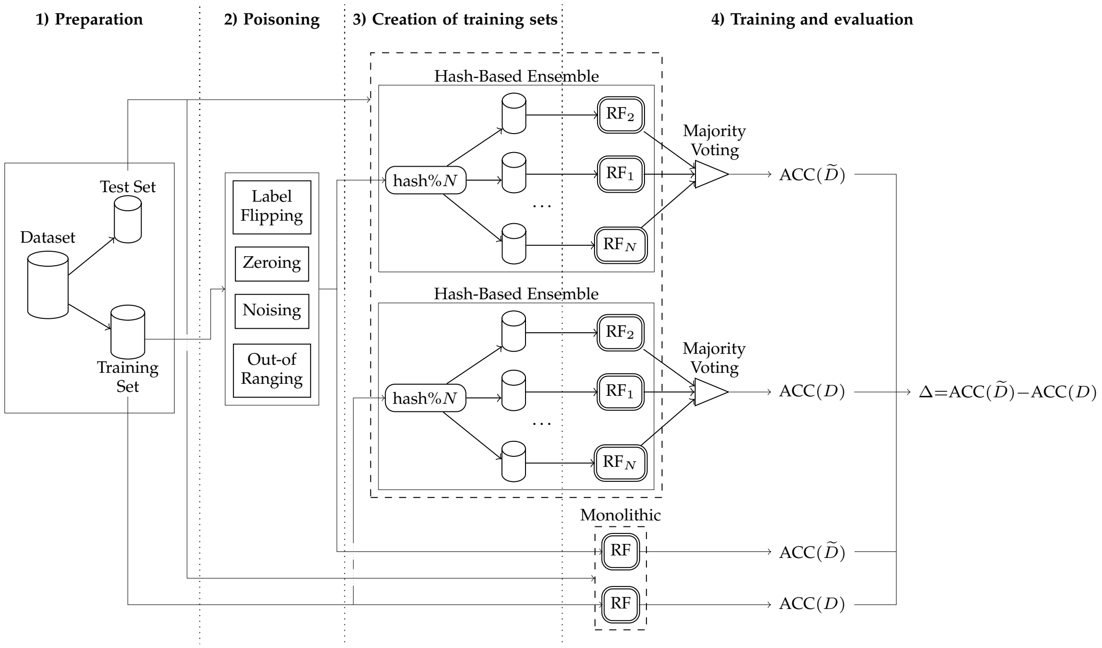

Figure 2: Evaluation process based on the ensemble approach in Figure 1.

to a string by concatenating the value of each feature. For instance, data point p with value $\langle 0, 10, 15, 0, 1\rangle$ becomes `0101501`. Third, the hash of such string is retrieved according to a specific hash algorithm. For instance, the hash value of `0101501` according to MD5 is `adf5c364bc3a61133eb2360f7dd0b8f2` (in hexadecimal). Fourth, the modulo operator (modulo N) is applied to the hash value converted back to a number. The result of this operation indicates to which partition the data point belongs, that is, the result n of modulo corresponds to the n+1 partition. This step produces the N partitions of the input training set: each partition is then used to train one of the N random forests of the ensemble. For instance, assuming N=17, the modulo of the hash of p is 1, hence, it is assigned to partition 1. Finally, the training set of the monolithic model is the input training set. We note that this step is repeated both on the original and poisoned dataset.

Step 4: Training and evaluation. It first trains both the ensemble and monolithic models separately on the original and poisoned training sets created at Step 3. Once the monolithic and ensemble models are created, their accuracy is calculated using the test set isolated at Step 1 and compared using evaluation metric *delta*, denoted as Δ, as follows.

$$\Delta = \text{ACC}(\widetilde{D}) - \text{ACC}(D), \tag{1}$$

where ACC(D) is the accuracy retrieved on the original training set and ACC($\widetilde{D}$) is the accuracy retrieved on the poisoned training set.

We note that Δ measures the accuracy variation in a model trained on a poisoned training set with regards to the same model trained on the original training set. A negative value of Δ indicates that the model trained on a poisoned dataset decreases in accuracy, a positive value indicates that the model trained on a poisoned dataset increases in accuracy, a value equals to 0 indicates same result.

Finally, we note that we executed the entire process three times averaging accuracy and Δ.

5.2 Experimental Settings

We experimentally evaluated the approach in this paper using two datasets that significantly differ in cardinality, number of features, and sparsity.

Musk2 is an open dataset for the identification of musk molecules [30]. It consists of 2,034 data points organized in 166 features and divided in two classes: 1,017 data points of class *musk* and 1,017 data points of class *non-musk*. The dataset is collected by including different conformations (shapes) of musk and non-musk molecules. In particular, all the low-energy conformations of 141 initial molecules have been generated and manually annotated. The dataset has a low degree of sparsity ($\approx$0.273%), and, together with the number of data points and features, puts our evaluation in a best case of a dataset perfectly-suited for random forest classification.

Android malware is a closed dataset for the detection of malware on Android devices. It consists of 14,508 data points organized in 25,802 features and divided in two classes: 7,254 data points of class *malware* and 7,254 data points of class *non-malware*. The dataset is collected on Android devices with benign and malign apps installed, by capturing the system calls performed by the apps. Any sequence of three consecutive system calls is a feature, whose value is the number of times such sequence has been called. The dataset has a large number of features with regards to the number of data points, making it suboptimal for training. To improve it, we reduced the number of features according to *InfoGain* [31], a feature ranking method selecting those features that reduce the *entropy* in the dataset (i.e., the most informative features with regards to the dataset). With this method, we reduced the number of features to

1,000. This dataset has a high degree of sparsity ($\approx$92.37%), and, together with the number of data points and features, puts our evaluation in a challenging case of a dataset not particularly suited for random forest classification.

The settings of our experiments varied *i)* the number of random forests N of our ensemble approach in $\{3, 5, 7, 9, 11, 13, 15, 17, 19, 21\}$; *ii)* the perturbations in *zeroing*, *noising*, *out-of-ranging* and *label flipping* (Section 4); *iii)* the percentage of poisoned data points $\epsilon_{\rm p}$ in $[10\%, 35\%]$, step 5%; *iv)* the percentage of poisoned features on each data point $\epsilon_{\rm f}$ in $[10\%, 35\%]$, step 5%. We note that $\epsilon_{\rm f}$ is not applicable to perturbation *label flipping*. Each combination of these parameters represented an instance of the evaluation process in Section 5.1. In addition, we used hash algorithm MD5, putting ourselves in a worst-case scenario of an outdated hash function (supposedly) poorly assigning data points to partitions, to determine whether our ensemble approach can still provide some protection against poisoning. Finally, we configured random forests according to the well-known practices in the state of the art[2].

To conclude, our experiments have been built on the ML library *Weka* [32] version 3.8 running on Java version 8. We executed our process on a VM equipped with 16 vCPUs Intel Core Processor (Broadwell, no TSX) 2.00 GHz and 48 GBs of RAM.

6 Experimental Results

We present the results of our evaluation first discussing *label flipping* (Section 6.1) and then *zeroing*, *noising*, *out-of-ranging* (Section 6.2) for datasets Musk2 and Android malware. Label flipping is the most effective perturbation substantially affecting the behavior of monolithic model, while *zeroing*, *noising*, *out-of-ranging* do not produce substantial accuracy degradation on it. We note that we omit the percentage symbol when discussing values of Δ, accuracy, as well as percentage of data points $\epsilon_{\rm p}$ and features $\epsilon_{\rm f}$ to poison. We also note that, according to our evaluation process in Section 5.1, all the results are averaged on three executions.

6.1 Label Flipping

Tables 1(a)–(b) show the average results retrieved by executing perturbation *label flipping* against datasets Musk2 (Table 1(a)) and Android malware (Table 1(b)), varying the percentage of poisoned data points $\epsilon_{\rm p}$ and the number of random forests N of the ensemble. The column with N=1 indicates the monolithic model, while the row with $\epsilon_{\rm p}$= 0 (rows with gray background in Tables 1(a)–(b)) indicates the accuracy ACC(D) retrieved from the model trained on the original dataset, that is, the dataset with 0% of poisoned data points. Each cell is divided in two parts. The top-most part reports the Δ in Equation (1), retrieved according to the accuracy of the model trained on the poisoned training set and the one on the original training set. The bottom-most part reports the accuracy ACC($\widetilde{D}$) retrieved by the model trained on the poisoned training set.

Our results first show that the accuracy retrieved on the two datasets vary significantly. The monolithic model shows ACC(D)=91.872 on dataset Musk2 and ACC(D)=98.828 on dataset Android malware. This difference of $\approx$7 points is repeated also for all configurations, reaching the peak with $\epsilon_{\rm p}$=35. On the contrary, variations in Δ are comparable especially for low values of $\epsilon_{\rm p}$ and N.

Our results additionally show two clear trends. First, as the percentage of poisoned data points increases, the corresponding Δ worsens (i.e., decreases), that is, the more label flips, the higher the accuracy decrease. This trend can be observed downward column by column. For instance, consider the smallest ensemble N=3. Δ worsens from -1.560 with $\epsilon_{\rm p}$=10 to -15.435 with $\epsilon_{\rm p}$=35 on dataset Musk2, and from -0.402 to -14.536 on dataset Android malware. Second, as the number of random forests in the ensemble increases, the corresponding Δ improves (i.e., increases), that is, the larger the ensemble, the lesser the accuracy decrease. This trend can be observed rightward row by row. For instance, consider the worst perturbation $\epsilon_{\rm p}$=35. Δ improves from -22.414 of the monolithic model to -6.814 with N=21 on dataset Musk2 (improvement of $\approx$70%). This improvement is even better on dataset Android malware, moving from -22.578 to -0.942 (improvement of $\approx$96%).

Figure 3 shows variation of Δ for the monolithic (N=1) and the smallest (N=3) and largest ensembles (N=21) for the two datasets Musk2 (denoted as M2) and Android malware (denoted as AM), varying the number of $\epsilon_{\rm p}$. We note that the monolithic model experiences the largest accuracy decrease as expected, ranging from -3.448 with $\epsilon_{\rm p}$=10 to -22.414 with $\epsilon_{\rm p}$=35 on dataset Musk2, and from -2.344 to -22.578 on dataset Android malware, that is, accuracy drops to 69.458 and 76.520, respectively, in the worst case. Instead, our ensemble approach shows higher robustness and keeps the accuracy drop well under control. This can be noticed even with the smallest ensemble N=3, with Δ at least doubling with regards to the monolithic model in most of the cases (from -10.887 to -5.513 on average on the two datasets). When considering dataset Musk2, it increases from -3.448 to -1.560 with $\epsilon_{\rm p}$=10, and from -22.414 to -15.435 with $\epsilon_{\rm p}$=35 (-6.158 on average with N=3). When considering dataset Android malware, it increases from -2.344 to -0.402 with $\epsilon_{\rm p}$=10 and from -22.578 to -14.535 with $\epsilon_{\rm p}$=35 (-4.868 on average with N=3). In other words, when the number of random forests N increases, Δ increases too, as Figure 3 shows.

Finally, when the number of random forests N is greater then 9, we note that Δ improves significantly on both the datasets. This is more evident with dataset Android malware, where Δ is 77.5% of the cases higher than -1, for $N\geq$ 9, compared to only 46.667% of the cases for N<9. The same trend can be observed in dataset Musk2, but the impact of the $\epsilon_{\rm p}$ is still noticeable. In fact, its worse value is -3.202 with $\epsilon_{\rm p}\leq$20 and $N\geq$9, compared to -11.330 with $\epsilon_{\rm p}$>20 and $N\geq$9, that is, Δ worsens with larger values of $\epsilon_{\rm p}$. This means that ACC($\widetilde{D}$)$\leq$ACC(D)$\pm$2.492 on average globally with our ensemble approach, compared to ACC($\widetilde{D}$)$\leq$ACC(D)$\pm$10.887 with the monolithic model.

Overall, our results show that plain random forests are sensitive to *label flipping*, but its effect can be easily counteracted using our ensemble approach.

[2] https://weka.sourceforge.io/doc.dev/weka/classifiers/trees/RandomForest.html

Table 1: Results of *label flipping* varying number of random forests N and percentage of poisoned data points ϵ_p.

		Number of random forests N of the ensemble										
		1	3	5	7	9	11	13	15	17	19	21
Poison. data points ϵ_p (%)	0	0.000 91.872	0.000 91.133	0.000 90.805	0.000 89.901	0.000 90.230	0.000 90.066	0.000 89.984	0.000 89.737	0.000 88.752	0.000 88.423	0.000 88.177
	10	-3.449 88.423	-1.560 89.573	-0.083 90.722	-0.328 89.573	0.493 90.723	0.000 90.066	-0.493 89.491	-0.410 89.327	0.246 88.998	-0.246 88.177	0.821 88.998
	15	-6.650 85.222	-2.874 88.259	-1.314 89.491	-1.149 88.752	-0.247 89.983	-1.232 88.834	-1.560 88.424	0.327 90.066	-3.202 85.550	-0.575 87.849	0.411 88.588
	20	-7.553 84.319	-1.970 89.163	-0.657 90.148	-1.149 88.752	-0.985 89.245	-1.478 88.588	-1.971 88.013	-2.052 87.685	-1.806 86.946	-0.657 87.767	-0.657 87.520
	25	-11.330 80.542	-4.187 86.946	-4.269 86.535	-3.448 86.453	-2.463 87.767	-2.135 87.931	-3.449 86.535	-3.612 86.125	-2.052 86.700	-0.903 87.521	-2.709 85.468
	30	-16.092 75.780	-10.920 80.213	-6.158 84.647	-4.597 85.304	-5.665 84.565	-3.941 86.125	-5.173 84.811	-5.254 84.483	-3.695 85.057	-5.255 83.169	-3.694 84.483
	35	-22.414 69.458	-15.435 75.698	-12.562 78.243	-8.949 80.952	-9.688 80.542	-11.330 78.736	-9.442 80.542	-9.113 80.624	-7.636 81.116	-10.181 78.243	-6.814 81.363

(a) Dataset Musk2

		Number of random forests N of the ensemble										
		1	3	5	7	9	11	13	15	17	19	21
Poison. data points ϵ_p (%)	0	0.000 98.828	0.000 98.541	0.000 98.449	0.000 98.047	0.000 98.070	0.000 97.886	0.000 97.828	0.000 97.679	0.000 97.483	0.000 97.311	0.000 97.104
	10	-2.344 96.484	-0.402 98.139	-0.184 98.265	-0.023 98.024	0.023 98.093	-0.046 97.840	-0.068 97.760	0.000 97.679	-0.034 97.449	-0.069 97.242	0.081 97.185
	15	-4.447 94.381	-0.943 97.598	-0.333 98.116	-0.150 97.897	-0.161 97.909	-0.069 97.817	-0.321 97.507	-0.230 97.449	-0.367 97.116	0.046 97.357	0.092 97.196
	20	-6.641 92.187	-1.862 96.679	-1.092 97.357	-0.356 97.691	-0.219 97.851	-0.265 97.621	-0.310 97.518	-0.138 97.541	-0.137 97.346	-0.241 97.070	0.161 97.265
	25	-11.100 87.728	-4.332 94.209	-1.999 96.450	-1.494 96.553	-1.230 96.840	-0.644 97.242	-0.505 97.323	-0.138 97.541	-0.471 97.012	-0.195 97.116	-0.252 96.852
	30	-16.052 82.776	-7.136 91.405	-4.148 94.301	-2.620 95.427	-1.873 96.197	-1.218 96.668	-1.597 96.231	-0.655 97.024	-0.723 96.760	-0.655 96.656	-0.574 96.530
	35	-22.578 76.250	-14.536 84.005	-8.595 89.854	-6.067 91.980	-4.355 93.715	-3.562 94.324	-3.630 94.198	-2.298 95.381	-1.964 95.519	-1.884 95.427	-0.942 96.162

(b) Dataset Android malware

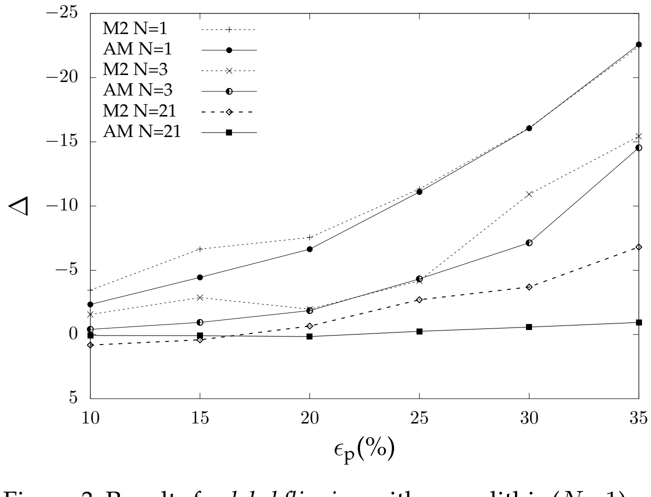

Figure 3: Results for *label flipping* with monolithic (N=1) and the smallest (N=3) and largest (N=21) ensemble models for datasets Musk (denoted as M2) and Android Malware (denoted as AM).

6.2 Other Attacks

Figure 4 shows the results for perturbations *zeroing*, *noising*, and *out-of-ranging* in Section 4 in terms of accuracy on datasets Musk2 and Android malware. Δ is not presented in Figure 4 since it does not show any major trends, being -0.316 on average, oscillating between -3.531 and 0.985.

Our results with these perturbations show two clear trends opposed to label flipping. First, they marginally affect the monolithic model, with Δ=0.023 on average (Δ=0.013 in *zeroing*, Δ=0.074 in *noising*, and Δ=-0.129 in *out-of-ranging*). Only in the case of dataset Android malware, Δ is always <0 in *noising* and *out-of-ranging*. As a consequence, there are no major improvements in Δ when using our ensemble approach, with Δ=-0.346 on average, (Δ=-0.157 in *zeroing*, Δ=-0.257 in *noising*, and Δ=-0.623 in *out-of-ranging*).

Second, as depicted in Figure 4, the accuracy decreases as the number of random forests N increases, but with a relatively small average difference of 2.660 between N=3 and N=21. In all the perturbations, this decrease is more pronounced in dataset Musk2 (decrease of 3.818, from 90.941 with N=3 to 87.124 with N=21, on average) than in dataset Android malware (decrease of 1.501, from 98.467 with N=3 to 96.968 with N=21, on average).

In summary, these results show that monolithic models are significantly less sensitive to these perturbations compared to label flipping, with Δ always larger than -0.492, meaning that $\text{ACC}(\widetilde{D}) \leq \text{ACC}(D) \pm -0.316$ on average.

7 Discussion

Our main research question focused on investigating the behavior of random forests against poisoning attacks. We

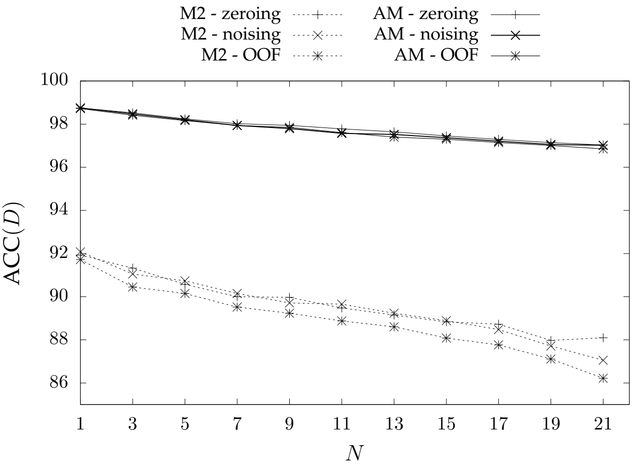

Figure 4: Results for other attacks averaged over ϵ_p. Perturbation *out-of-ranging* is abbreviated as *OOF*.

evaluated the monolithic model (i.e., ensemble of decision trees) and our ensemble approach (i.e., ensemble of random forests) varying the type (i.e., perturbations) and impact (i.e., ϵ_p and ϵ_f) of poisoning. Our main findings are as follows.

F1: Monolithic model is highly sensitive to label flipping only. This result is clear in Figure 3, where the monolithic models are significantly worse (i.e., above) the ensemble models. We also note that Δ of the monolithic models retrieved under flipping is -10.887 on average, while it is -0.316 on average under *zeroing*, *noising* and *out-of-ranging*, with a difference of 97.10%. In addition, the accuracy decrease Δ caused by label flipping is proportional with the percentage ϵ_p of poisoned data points. This can be noted by comparing $\mathrm{ACC}(\widetilde{D})$ and Δ on the datasets in Tables 1(a)–(b), where $\mathrm{ACC}(\widetilde{D})$ and Δ progressively worsens as ϵ_p increases.

F2: The effectiveness of perturbations depends also on the characteristics of the dataset and of the ensemble. In the case of label flipping, this is noticeable by comparing downward the right-hand side of Table 1(a)–(b). Being Musk2 a smaller dataset, Δ worsens more rapidly as the percentage of poisoned data points increase. Δ also worsens as N increases, because the cardinality of the individual partitions and training sets is increasingly reduced. In the case of the remaining perturbations, this is noticeable by comparing the lines of datasets Musk2 and Android malware in Figure 4. Also in this case, accuracy worsens at a higher rate in Musk2 as N increases, with *out-of-ranging* being the most effective perturbation.

F3: Ensemble of random forests is an adequate protection from untargeted label flipping. This result emerges by comparing the increases of Δ when considering our ensemble approach with regards to the monolithic model in Tables 1(a)–(b). For instance, considering dataset Android malware, accuracy becomes >95 with at least $N{=}9$ random forests in our ensemble approach for $\epsilon_\mathrm{p}{\leq}30$ of poisoned data points, and keeps increasing slightly with N. Instead, for $\epsilon_\mathrm{p}{>}30$ of poisoned data points, the ensemble approach starts suffering of not-negligible accuracy decreases, being <95 in most of the cases. We note that, for at least $N{=}15$, this decrease can be still considered negligible, being accuracy ≥ 95. In general, the improvement provided by our ensemble approach is significant, as summarized in Figure 3, where the highest lines corresponding to the monolithic model are always significantly worse than those of the ensemble models.

F4: Ensemble of decision trees is an adequate protection from untargeted perturbations other than label flipping. This is noticeable by comparing the accuracy of monolithic and ensemble models in Figures 4, and is a direct consequence of F1. Perturbations *zeroing*, *noising*, and *out-of-ranging* obtain an accuracy decrease, but this decrease largely fails to make the monolithic model unusable in practice, as its accuracy is always ± 0.9 with regards to the original accuracy. This implies that our ensemble approach is redundant in this scenario, and explains the accuracy decrease we observed as N increases. In practice, our approach only reduces the cardinality of the training set of each base model from $|D|$ to $|D|/N$, hence affecting classification accuracy. This tradeoff is advantageous in label flipping, when the accuracy decrease caused by the smaller training set is balanced by containing the accuracy decrease caused by poisoning. It is detrimental for other perturbations where the accuracy decrease caused by poisoning is negligible.

From the above findings, we can finally conclude that **random forest (a native ensemble algorithm) or an ensemble of random forests trained on disjoint partitions, provide an empirically-strong robustness against untargeted poisoning attacks of reasonable degree**. When the perturbation is trivial (*zeroing*, *noising*, *out-of-ranging*) a random forest is sufficient, while when it involves label flipping, an ensemble of random forests is needed. The size of the ensemble must however be carefully balanced to avoid accuracy decrease due to an oversized ensemble approach.

Finally, we note that the results in this paper are in line with other work on ML robustness, for instance [11], [17], claiming that *i)* if the amount of poisoning is reasonable, an ensemble strategy can reduce the influence of poisoned data points to the resulting model, and *ii)* random forests are, in some cases, more robust than other models (e.g., naive bayes, neural networks) [11], [28], [29].

8 Conclusions

Machine learning models play an increasingly vital role in the digital services we interact with. As a consequence, the need for properly securing such models from attacks is a key issue being investigated by the research community. The evaluation in this paper aims to shed new light on the usage of ensembles as a means of protection from poisoning attacks. While ensembles have been already proposed in the context of certified protection in the domain of image recognition, little has been done in the context of random forests (an ensemble algorithm) and ensemble of random forests (our approach). Throughout finer-grained experiments, our results show that untargeted label flipping, even if performed randomly as in our case, is a very dangerous type of perturbation, significantly degrading the performance of plain models. A simple yet effective countermeasure consists in training models on disjoint training sets, then aggregating their predictions with majority voting. Other perturbations are less effective, and the aggregation of decision trees is an effective countermeasure. The paper leaves space for future work. First, we plan to enrich our set of perturbations with targeted attacks. Second, we plan to develop a complete benchmark considering the new perturbations, several datasets with diverse peculiarities (e.g., sparsity levels), and different hash function.

References

[1] F. Rundo, F. Trenta, A. L. di Stallo, and S. Battiato, "Machine Learning for Quantitative Finance Applications: A Survey," *Applied Sciences*, vol. 9, no. 24, 2019.

[2] E. Hossain, I. Khan, F. Un-Noor, S. S. Sikander, and M. S. H. Sunny, "Application of Big Data and Machine Learning in Smart Grid, and Associated Security Concerns: A Review," *IEEE Access*, vol. 7, 2019.

[3] R. Chen, W. Zhang, and X. Wang, "Machine Learning in Tropical Cyclone Forecast Modeling: A Review," *Atmosphere*, vol. 11, no. 7, 2020.

[4] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, "Machine learning applications in cancer prognosis and prediction," *Computational and Structural Biotechnology Journal*, vol. 13, 2015.

[5] J. G. Richens, C. M. Lee, and S. Johri, "Improving the accuracy of medical diagnosis with causal machine learning," *Nature Communications*, vol. 11, no. 1, Aug 2020.

[6] J. Y. Chang and E. G. Im, "Data Poisoning Attack on Random Forest Classification Model," in *Proc. of SMA 2020*, Ramada Plaza Jeju, Jeju, Republic of Korea, September 2020.

[7] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[8] M. Mozaffari-Kermani, S. Sur-Kolay, A. Raghunathan, and N. K. Jha, "Systematic Poisoning Attacks on and Defenses for Machine Learning in Healthcare," *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 6, 2015.

[9] R. Schuster, C. Song, E. Tromer, and V. Shmatikov, "You Autocomplete Me: Poisoning Vulnerabilities in Neural Code Completion," in *Proc. of USENIX 2021*, Virtual, Aug. 2021.

[10] B. Biggio, B. Nelson, and P. Laskov, "Support Vector Machines Under Adversarial Label Noise," in *Proc. of ACML 2011*, Taoyuan, Taiwan, November 2011.

[11] C. Dunn, N. Moustafa, and B. Turnbull, "Robustness Evaluations of Sustainable Machine Learning Models against Data Poisoning Attacks in the Internet of Things," *Sustainability*, vol. 12, no. 16, 2020. [Online]. Available: https://www.mdpi.com/2071-1050/12/16/6434

[12] I. Diakonikolas, G. Kamath, D. Kane, J. Li, J. Steinhardt, and A. Stewart, "Sever: A Robust Meta-Algorithm for Stochastic Optimization," in *Proc. of ICML 2019*, Long Beach, CA, USA, June 2019.

[13] Y. Ma, X. Zhu, and J. Hsu, "Data Poisoning against Differentially-Private Learners: Attacks and Defenses," in *Proc. of IJCAI 2019*, Macao, China, August 2019.

[14] E. Rosenfeld, E. Winston, P. Ravikumar, and Z. Kolter, "Certified Robustness to Label-Flipping Attacks via Randomized Smoothing," in *Proc. of ICML 2020*, Virtual, June 2020.

[15] N. Peri, N. Gupta, W. R. Huang, L. Fowl, C. Zhu, S. Feizi, T. Goldstein, and J. P. Dickerson, "Deep k-NN Defense Against Clean-Label Data Poisoning Attacks," in *Proc. of ECCV 2020*, August 2020.

[16] P. W. Koh, J. Steinhardt, and P. Liang, "Stronger data poisoning attacks break data sanitization defenses," *Machine Learning*, Nov 2021.

[17] J. Jia, X. Cao, and N. Z. Gong, "Intrinsic Certified Robustness of Bagging against Data Poisoning Attacks," in *Proc. of AAAI 2021*, Virtual, February 2021.

[18] A. Levine and S. Feizi, "Deep Partition Aggregation: Provable Defenses against General Poisoning Attacks," in *Proc. of ICLR 2021*, Vienna, Austria, May 2021.

[19] W. Wang, A. Levine, and S. Feizi, "Improved Certified Defenses against Data Poisoning with (Deterministic) Finite Aggregation," *arXiv preprint arXiv:2202.02628*, 2022.

[20] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proc. of ICLR 2014*, Banff, Canada, April 2014.

[21] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding Deep Learning Requires Rethinking Generalization," in *Proc. of ICLR 2017*, Toulon, France, April 2017.

[22] A. Paudice, L. Muñoz-González, and E. C. Lupu, "Label Sanitization Against Label Flipping Poisoning Attacks," in *Proc. of ECML PKDD 2018 Workshops*, Dublin, Ireland, September 2018.

[23] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks," in *Proc. of NeurIPS 2018*, Montréal, QC, Canada, December 2018.

[24] A. Prasad, A. S. Suggala, S. Balakrishnan, and P. Ravikumar, "Robust estimation via robust gradient estimation," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 82, no. 3, 2020.

[25] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion Attacks Against Machine Learning at Test Time," in *Proc. of ECML PKDD 2013*, Prague, Czech Republic, September 2013.

[26] S. Hong, V. Chandrasekaran, Y. Kaya, T. Dumitraş, and N. Papernot, "On the Effectiveness of Mitigating Data Poisoning Attacks with Gradient Shaping," *arXiv preprint arXiv:2002.11497*, 2020.

[27] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "A comparison of decision tree ensemble creation techniques," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 173–180, 2007.

[28] H. Zhang, N. Cheng, Y. Zhang, and Z. Li, "Label flipping attacks against naive bayes on spam filtering systems," *Applied Intelligence*, vol. 51, no. 7, Jul 2021.

[29] F. A. Yerlikaya and Şerif Bahtiyar, "Data poisoning attacks against machine learning algorithms," *Expert Systems with Applications*, vol. 208, 2022.

[30] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[31] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, no. 1, 1986.

[32] M. A. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explor.*, vol. 11, no. 1, pp. 10–18, 2009.