Information Sciences 633 (2023) 122–140

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Robust ML model ensembles via risk-driven anti-clustering of training data

Lara Mauri [a,*], Bruno Apolloni [a], Ernesto Damiani [a,b]

[a] Department of Computer Science, Università degli Studi di Milano, Milan, Italy
[b] Center for Secure Cyber-Physical Systems, Khalifa University of Science and Technology, Abu Dhabi, United Arab Emirates

ARTICLE INFO

Keywords:
Adversarial machine learning
Machine learning security
Robust ensemble models
Poisoning attack
Training set partitioning
Risk modeling

ABSTRACT

In this paper, we improve the robustness of Machine Learning (ML) classifiers against training-time attacks by linking the risk of training data being tampered with to the redundancy in the ML model's design needed to prevent it. Our defense mechanism is directly applicable to classifiers' training data, without any knowledge of the specific ML model to be hardened. First, we compute the training data proximity to class separation surfaces, identified via a reference linear model. Each data point is associated with a risk index, which is used to partition the training set by an unsupervised technique. Then, we train a learner for each partition and combine the learners' output in an ensemble. Our method treats the protected ML classifier as a black box and is inherently robust to transfer attacks. Experiments show that, for data poisoning rates between 6 and 25 percent of the training set, our method is more robust compared to benchmarks and to a monolithic version of the model trained on the whole training set. Our results make a convincing case for adopting training set partitioning and ensemble generation as a stage of ML models' development and deployment lifecycle.

1. Introduction

Machine Learning (ML) models have been successfully applied across a wide range of domains, especially after the advent of Deep Learning [1]. However, adversaries can attack ML models by interfering with the training phase or at the inference phase, after the model has been deployed. *Data poisoning* attacks occur when adversaries interfere with the learning process of a model by maliciously modifying a portion of the samples used to train it. A survey by Kumar et al. [2] found that industry practitioners consider training data poisoning as a major threat, claiming that they are not adequately equipped to defend, detect and react to data poisoning attacks.

Most defense techniques used against data poisoning attacks focus on protecting specific learning algorithms. They work in ideal scenarios, where defenders are well aware of which malicious actions will be taken against their training data and can react accordingly. In practice, defenders hardly ever know which type of attack will be launched, and even less the exact data points that will be targeted (excluding trivial cases such as outliers). In this paper, we help defenders to estimate which training data points should be considered at risk. To this end, we use the training set to generate a linear classifier and compute the hyper-planes it uses

* Corresponding author.
E-mail address: lara.mauri@unimi.it (L. Mauri).

https://doi.org/10.1016/j.ins.2023.03.085
Received 12 October 2022; Received in revised form 5 March 2023; Accepted 7 March 2023

Available online 13 March 2023

as decision boundaries between classes. Then, we assign a risk index to training data points by using an iterative procedure that takes into account their distance from such hyper-planes. The result is a *risk map* of the training set that defenders can use to adapt their defense strategy.

Based on this map, we describe a defense strategy against poisoning attacks based on *ensemble* methods. The purpose of classical ensemble learning is to improve *generalization* capabilities in non-adversarial settings [3]. We deploy an ensemble system composed of multiple learners to improve the *robustness* of ML models. Our approach is based on partitioning the training data set using an unsupervised technique (*anti-clustering*) that takes into account our risk map. The main advantage of our approach is that it is generic enough to be used in different settings. It does not require the defended model to be linear, and supports protection of all types of learners.

Our work is also relevant to attack modeling, where existing work mainly focuses on how to attack ML algorithms by injecting small fractions of well-crafted adversarial samples [4] into the training data set. These attack strategies rely on worst-case analysis where the modifications introduced into the training set are designed to maximize the damage on the targeted learning algorithm. However, they are strongly tied to the attacker being aware of the exact training algorithm used for the model under attack and able to exploit feedback from the attacked ML model to compute optimal attack instances [5].

We make a much more restrictive assumption on the adversary's knowledge: we assume the attacker is not aware of the model she will be attacking, although she has access to the training set that will be used to train it. Obviously, lacking the information to maximize a model-specific objective function, the attacker will have to devise an alternative strategy to decide which data points to attack. We assume that, as in the case of the defender, the attacker uses a *reference linear model* and preferentially attacks the points of the training set on the basis of their proximity to the separation surface. While this strategy may not be optimal with respect to a specific target model, it is optimal with respect to the information available to the attacker.

To the best of our knowledge, this is the first paper explicitly linking the risk of ML training data being tampered with to the redundancy in the ML model's design needed to prevent it.

The main advantage of our defense mechanism is that it is directly applicable to training data, without any knowledge of the underlying ML model. We rely on a linear reference model to represent the attacker's (and the defender's) knowledge about the ML-based system we intend to harden. Thus, our technique treats the protected model as a black box, which makes it inherently more robust to transfer attacks.

The rest of the paper is organized as follows. In Section 2, we survey related work in the area of adversarial attacks to ML. In Section 3, we present the first component of our defence framework, i.e. a method for assessing the risk associated with ML training data. The problem definition and the adversarial threat model are outlined in Sections 4 and 5, respectively. In Section 6, we detail our anti-clustering partition strategy for improving the robustness of ML models via ensemble composition. In Section 7, we show our experimental results. Finally, in Section 8, we draw our conclusions and discuss further research avenues.

2. Related work

Previous work has investigated strategies for mitigating data poisoning attacks at different stages of the ML pipeline. Many of the proposed techniques have limitations in terms of applicability, the type of attack they protect against, their effect on accuracy; also, they increase the training's complexity. In this section, we provide a synthetic overview of some approaches proposed in the literature. A complete survey can be found in [6].

A popular defence strategy against data poisoning attacks is *training data pre-filtering*, which identifies the directions along which poisoned data points deviate from their non-corrupted counterparts, and then sanitizes the training set by excluding the outliers along these directions [7]. Pre-filtering draws on *robust statistics*, which has been studying the fundamental problem of learning in the presence of outliers since the 1960s [8]. Recently, this problem has gathered renewed attention due to the pressing need to design robust ML models for high-dimensional data sets [9]. Paudice et al. [7] proposed a data sanitization mechanism to identify and re-label suspect training points. Their approach detects samples having a negative impact on the performance of ML classifiers and makes use of a *k-Nearest-Neighbors* (k-NN) model to assign to each of these samples the most common label among its k nearest neighbors. Unfortunately, outlier-based defences have been proven to be vulnerable to adaptive attacks that explicitly attempt to evade anomaly detection [10]. Another important family of defense techniques relies on *model enhancement* to augment the training set and/or harden the ML model to counter the effect of poisoning. Borgnia et al. [11] investigated the effects of data augmentation schemes on data poisoning and demonstrated that some data augmentation techniques such as mix-up and cutout can desensitize ML models to hostile perturbations of training data. These strategies can be viewed as the defense counterpart of *adversarial poisoning* [12], because they use *adversarial training* (which is commonly used as a defence against inference-time adversarial attacks) to defend ML models against training-time attacks. Albeit promising, this line of research is still in its infancy. At present, these techniques lack formal guarantees on their convergence and on the robustness properties they provide.

A line of work closer to ours tries to reduce the influence of poisoned samples via partitioning the training set and using each partition to train a different learner. The problem here is how to aggregate the learners' outputs to minimize the poisoning effects. Early studies focused on the standard *Bootstrap aggregation* (or bagging) framework [13], arguing that, in addition to accuracy, bagging can also improve robustness in adversarial settings. Biggio et al. [14] described an empirical defence based on such framework. They experimentally investigated whether bagging ensembles can be exploited to build robust classifiers against poisoning attacks, assessing the effectiveness of the approach on a spam filter and on a web-based intrusion detection system. The same authors investigated the use of bagging and random subspace methods [15] for constructing robust systems of multiple classifiers, extending the preliminary results presented in [16]. This early work provided neither a formal definition of robustness neither a proof that the application of bagging would achieve it.

More recently, research has focused on designing ad-hoc *intelligent data partitioning* schemes [17] leading to *provably robust defences* against data poisoning [18]. Some works focused on distributional robustness guarantees [19], while others focused on point-wise certified robustness [20].

Jia et al. [21] leveraged the intrinsic majority vote mechanism of k-NN and r-NN (*radius Nearest Neighbors*) and showed that they provide deterministic certified accuracy against both data poisoning and backdoor attacks.

Levine and Feizi [22] proposed a certifiable ensemble-based method where the partitioning of the training set into disjoint subsets is deterministically performed via *locality-sensitive hashing* (LSH), i.e. by computing a function that maps the n-dimensional training T into another n-dimensional set T', without preserving the distance between samples that were close to each other in T.

The diversity of the above defense techniques makes it difficult to compare them based on their original validation experiments, which use different data sets and parameters. Also, there is still no consensus in the literature on the robustness metrics to be used, although some quantitative metrics of model robustness in face of label-flipping attacks have been proposed [23]. In Section 7 we will rely on LSH to generate a benchmark and define a common metric based on *certified accuracy*.

3. Risk analysis of ML data assets

3.1. Risk estimation techniques

Assessing the risk of an attack to a given asset requires two estimations: the one of the attack's *severity*, based on the affected assets' value, and the one of the attack's *likelihood*, based on the known threats and vulnerabilities. The product of severity and likelihood is often used as the reference equation for risk quantification [24], $R = S \times L$, where L is a measure of likelihood and S is a severity value expressed in monetary units. Recent studies [25] introduced the notion of *risk index* as a joint quantification of severity and likelihood that can be quantized into discrete levels, e.g., *high risk*, *medium risk*, *low risk*, or computed as a continuous *risk score*. In terms of the data tampering risk, computing the risk score of individual data points can be modeled as learning a function on the data space, and is itself suitable for the application of ML models. A wide range of methods has been proposed to learn risk from examples, most of them assuming a linear regression model [25] where the risk score varies linearly with the distance in the data space. However, research has shown that for many types of data a linear approach to risk scoring is not appropriate [26], as close data points may have different severity or likelihood of attacks.

3.2. A risk score for ML training data

In our case, the ML data asset under attack is the training set.[1] We compute the $R = S \times L$ risk equation linking the data tampering attack's severity to the model's performance degradation after attack. This requires consensus on the performance degradation metrics to use. Research is ongoing on extracting "gold standard" data sets from input data spaces, providing held-out benchmarks suitable for measuring ML performance degradation [29]. However, such held-out data sets are forcibly problem-dependent. In terms of likelihood estimation, some work has been done on heuristics for estimating the likelihood of attacks to ML assets (including data tampering) on the basis of available information on the model deployment architecture [30]. This type of likelihood estimate applies to an entire data asset rather than to individual data items, and is not yet suitable for assessing the poisoning risk concerning individual data points.

In order to escape the well-known pitfalls of linear risk models [26], we compute our risk index as a non-linear function of the training set's data points. Low-degree polynomials with decreasing coefficients (in the form $a + bx + cx^2$) have been used since long in risk modeling, as they capture additional information through coefficients. The details of the computation of our risk index and of its role in our procedure are given in the next Section.

3.3. Computing a risk index on training data

A key notion of ML is that not all points in a training set have the same relevance. Our risk index computation is grounded in the concept of *sentry points* [31], the (usually few) points in a training set that are needed for learning a Boolean function with no tolerance on errors. For a linear model, the number of sentry points is less or equal to the number of dimensions of the data points plus 1.

In practice, we compute the risk index of each point in the training dataset based on its belonging or not to the set of the support vectors of a SVM trained on the entire dataset. We consider more "relevant" the points not far from the SVM separating hyperplanes, with a proper discrete graduation corresponding to the *colors* of the map shown in Fig. 1). We apply an iterative process to identify the support vectors of the hyperplanes separating classes according to a progressive pruning of the points supporting the separator in a previous iteration (see the pseudo-code in Algorithm 1). We define a distance scale separator through a succession of SVM classifications. In the first iteration, the support vectors are assigned the maximal suitability. Then, we obtain a second set of

[1] A complete *asset model* identifying ML data assets that can be subject to threats has been released by the European Network and Information Security Agency (ENISA) [27,28].

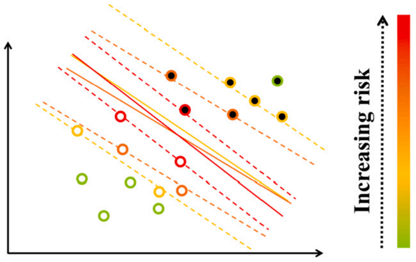


Fig. 1. A simplified 2D representation of risk-related color assignment according to Algorithm 1.

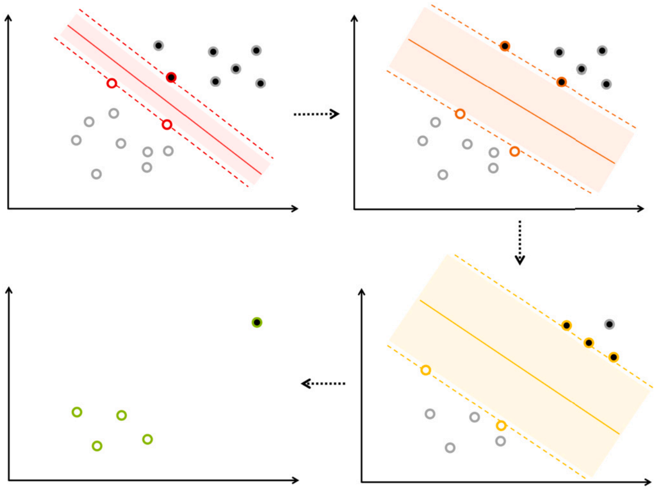


Fig. 2. Assigning different gradations of risk-related colors based on proximity to separator hyperplanes.

risky points (whose risk index has a lower value than the previous one) by iterating the SVM over the remaining set of points after removing the previously identified support vectors, and so on. Fig. 2 shows the operation of our algorithm in a two-dimensional space.

Algorithm 1 Generating color graduation.

```
Input: Original training data set $D = \{(x_i, y_i)\}_{i=1}^{n}$, linear separator $\mathcal{L}$, color set $\boldsymbol{c}$
Output: Color map $\boldsymbol{xc}$
 1: $D' \leftarrow D$
 2: $j \leftarrow 1$
 3: $\boldsymbol{xc} \leftarrow \emptyset$
 4: $t^{max} \leftarrow |\boldsymbol{c}| - 1$
 5: while $j < t^{max}$ do
 6:     Identify support vectors $sv_j$ of $\mathcal{L}_j$ separating data points in $D'$
 7:     Associate color $c_j$ to $sv_j$
 8:     Add $\{sv_j, c_j\}$ to $\boldsymbol{xc}$
 9:     Remove $sv_j$ from $D'$
10:     $j \leftarrow j + 1$
11: end while
12: $\forall x \in D'$:
13:         Associate color $c_0$ to $x$
14:         Add $\{x, c_0\}$ to $\boldsymbol{xc}$
15: return $\boldsymbol{xc}$
```

In this way, we characterize points which are close to the joint boundary of the classes. Mislabeling such points may induce severe errors in any learning procedure. Points more internal to the classes are less relevant *per se*; their addition has the role of increasing the number of points considered.

As described in Section 4, our risk index can be computed by both the defender and the attacker, although the use they will make of it is different. First of all, the attacker does not know the separator that will be exactly used by the defender and vice versa. Secondly, when two sets are not linearly separable, many more points than just the support vectors must be considered to determine the separation hyperplane.

The bordering condition depends on the function we adopt to separate the classes. For instance, a point that is far from a bordering hyperplane may result in being close to another bordering surface.

4. Problem formalization

We can now formalize poisoning as a binary linear classification problem. We assume the attacker to be able to manipulate the labels of some training data, i.e., she can perform label-flipping poisoning attacks. We can formalize the problem as a game between the defender looking for a classification setting that is robust against label flips and the attacker looking for a flipping strategy that degrades the performance of the classifier. The game can be represented as a bi-level optimization problem constrained by the attacker's budget

$$\max_{\boldsymbol{z}} \sum_{(\boldsymbol{x},y)\in T} V(y, f_{S'}(\boldsymbol{x}), \tag{1}$$

$$\text{s.t.} f_{S'} \in \arg\min_{f} \sum_{i=1}^{n} V(y, f_{S'}(\boldsymbol{x}_i)) + \gamma \|f\|^2, \tag{2}$$

$$\sum_{i=1}^{n} c_i z_i \le C, \quad z_i \in \{0,1\}, \tag{3}$$

where f is the classifier and V is the related loss function which depends on the difference between a target value y and the classifier output $f(\boldsymbol{x})$. f is trained on a training set $S = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$, which is corrupted by the attacker into $S = \{\boldsymbol{x}_i, y'_i\}_{i=1}^n$. The variable z denotes whether a label has been flipped ($\rightarrow z_i = 1$) or not ($\rightarrow z_i = 0$); each flip has a cost c_i, and the total flipping cost threshold is C. The attacker aims to maximize V with a proper setting of $\boldsymbol{z}$, as in (1), under the constraint (3); the defender aims to learn a function f_S which minimizes V, as in (2). However, this optimization problem proves very difficult to solve in general [32], depending on the complexity of f and on the discreteness of y. Indeed, the approaches we surveyed in Section 2 can be described as ways to address this issue, e.g. by reverting f to a linear function, identifying the points to be poisoned one at a time, performing label smoothing or by adjusting the probabilities with which y takes on its values. We propose an alternative attack-defense framework where the strategies of the attacker and the defender are the following:

- The attacker's strategy leverages point classification through a SVM classifier, using support vectors as candidate points to be flipped.

  Rationale: As support vectors determine the position of the hyperplane separating a pair of classes, either in the original space or in a kernel space, a simple live-out of one of them changes the position of the separator, resulting in a misclassification of the training set or, at least, in a decrease in the margin of one of its sides. Points close to the boundary may fly under the defender's radar. Thus, the attacker is interested in flipping the labels of these points.

- The defender's strategy takes advantage of ensemble composition, training different learners on disjoint subsets of the entire training set and merging their outputs according to a consensus algorithm [33]. The defender tries to tune partitioning to make the individual models diverse enough to be robust to poisoning while preserving accuracy.

  Rationale: If the attacker flips the label of a point with high risk index, this will degrade the performance of only one learner. Any point close to the high-risk one gets assigned to another partition, thus affecting a different learner. We can expect two benefits: (i) the classification of the latter point is not degraded by the flip of the former point's label, since it is fed to a different learner, (ii) the other learner will correctly classify the former point, contributing to a correct result in the majority voting mechanism.

5. Threat model

We can now state our assumptions about the attacker capabilities and her level of knowledge of the targeted ML model. Then, we provide a detailed description of the strategy and of the type of poisoning attack she can perform.

5.1. Attacker's power

The attacker's knowledge κ can be defined as a tuple $\theta = (D, X, f, w)$, where D is the training set, X is the feature set, f is the learning algorithm, and w are the parameters learned after training the ML model. We consider a realistic scenario where the attacker launches her attack on the training data points having limited information about the target ML model. In particular, we assume that the adversary knows the input feature representation X and the training data D, but not the learning algorithm f. The adversary builds her own *surrogate model* $\hat{f}$ (a linear SVM model) that she uses to estimate which points to attack. Therefore, the black-box attack with limited knowledge can be denoted with $\hat{\kappa} = (D, X, \hat{f}, \hat{w})$. We assume that the attacker has some control over a fraction of the training data used by the learning algorithm, and is restricted to changing the training labels, i.e., she can perform a label-flipping attack. Furthermore, the attacker aims to produce specific types of error, which means that she can decide the *direction* of the flip (e.g., causing only a certain label of her interest to flip). We denote the altered training set with $D' = \{(x_i, y_i)\}_{i=1}^n$.

5.2. Attack algorithm

Obviously, a rational attacker's intention would be to manipulate the most risky points. However, the attacker knows little about the target ML model. To select the samples that, if modified, would cause the maximum decrease in the accuracy of the target model, she uses a reference linear model to approximate the target discriminating function and to generate the probability distribution used for selecting the points to attack. In other words, before performing any flip, the attacker applies Algorithm 1 (see Section 3.3). A natural way of implementing this strategy is to employ a *probability weighting function* (p_{ℓ}). We model the attacker's rational behavior with a nonlinear function which modifies the weights different probabilities have according to the risk level associated to the data points and to the number of risky points belonging to each risk level:

$$p_{r\ell_i} = \frac{n_{r\ell_i} r\ell_i}{\sum_{j=1}^{|c|-1} n_{r\ell_j} r\ell_j} \tag{4}$$

where $n_{r\ell}$ is the number of data points having risk index $r\ell$ and $|c|$ is the desired number of risk levels identified after applying Algorithm 1. In Eq. (4) we assume that the attacker flips only the labels of the points corresponding to the identified support vectors, thus excluding the remaining points having risk index 0 (see line 12 of Algorithm 1). As a result, for each data point the probability of flipping depends on the level of risk associated with it (as defined by the SVM model) and on the total number of risky points having the same level of risk as it. The percentage of flipped points belonging to a given risk level grows as the level of risk increases; the higher the risk (and the corresponding number of risky points identified), the greater the probability a given data point gets flipped.

Algorithm 2 Risk-driven weighted probabilistic flipping attack.

```
Input: Original training set $D = \{(x_i, y_i)\}_{i=1}^{n}$, flip direction $d$, flipping budget $\epsilon$
Output: Contaminated training set $D'$
 1: $D' \leftarrow D$
 2: $j \leftarrow 0$
 3: $flag_i \leftarrow 0$
 4: while $j < \epsilon$ do
 5:     Extract a data point $(x'_i, y'_i)$ from $D'$ with probability $p_{r\ell_i}$
 6:     if $flag_i = 0$ then
 7:         if $y'_i = d$ then
 8:             $y'_i \leftarrow -y'_i$
 9:             $flag_i \leftarrow 1$
10:             Add $(x'_i, y'_i)$ to $D'$
11:         end if
12:     else Repeat from line 4
13:     end if
14:     $j \leftarrow j + 1$
15: end while
16: return $D'$
```

Algorithm 2 describes the flipping attack strategy we used. This algorithm is fully operational if the percentage of flipped points is relatively small. If the attacker can flip many labels, the result will be that all the labels of the risky points will be flipped, and the labels of other points not identified as support vectors in any of the iterations of the risk index computation algorithm will also be flipped randomly.

A second aspect concerns the direction of the label-flipping, which can be either mono- or bi-directional. A check must be in place to avoid multiple flips of the same label that could bring back the original value of the label. The first option (bi-directional flip) is less effective from the attacker's point of view, since the flips can offset their effect, leaving the separating hyperplane almost unchanged. The second option circumvents this drawback (in Algorithm 2, the check at line 7 indicates the use of mono-directional flips).

6. A closer view of our defence framework

Our defence strategy consists of three main components: (i) computation and polynomial transform of the risk indices associated with training data, (ii) risk-aware training set partitioning via anti-clustering, and (iii) ensemble composition for increasing robustness. The flowchart in Fig. 3 shows the different components of the proposed framework. The rest of this Section defines the characteristics of each component and how they relate to each other.

6.1. Feature space extension

The first component of our framework is the risk index defined in Section 3 as an explicit polynomial function of the *color*, which in turn is a quantization of point's distance from a separation hyperplane of our reference SVM model. We use the risk index to assign the data points to the training sets of different learners, in order to minimize the learners' exposure to poisoned data.[2]

[2] Should attack data be available in the form of reports on previously attacked data points, some alternative risk index could be learnt from them. In case of *rational attackers* the learnt risk index landscape will be close to the graph of our function over the color map.

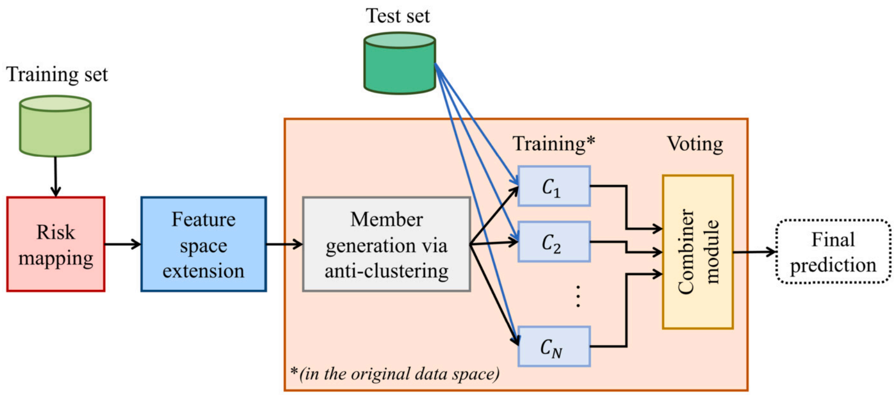


Fig. 3. Flowchart of the proposed defence framework.

We achieve this by performing a (temporary) feature space extension, adding the risk index to the data points' features, and then using an unsupervised technique (Section 6.3) on the extended data space to group data points into partitions that are highly diverse in terms of both risk and position. Each partition, stripped of the risk feature, is then used for training a separate learner, minimizing its exposure to potentially poisoned data. For the sake of flexibility, in addition to computing the risk index as a polynomial function of the color, we use this additional feature for rescaling the distance in the expanded feature space at (anti-)clustering time.

6.2. *Ensemble structuring and composition*

The second component of our defence framework is *ensemble learning*. Ensemble learning is a well-established method that has been proven to yield improved generalization capabilities. The central idea is to employ multiple learners (usually called *base learners*) and combine their predictions based on a consensus algorithm.

When individual predictions from different base learners are combined appropriately, the collective decision has on average better accuracy than that of any individual learner. We argue that, by properly partitioning data and combining output predictions, weak learners can become strong learners not only in terms of accuracy, but also in terms of robustness to poisoning attacks. Our approach relies on two key assumptions:

1. Individual models are sufficiently diverse to maintain high accuracy and be resistant to training-time attacks;
2. Enough intact models always remain in the ensemble so that the consensus output is not corrupted.

Assumption (1) hinges on the way base learners are trained, and comes as a direct result of our approach. Assumption (2) depends on several factors. The most significant ones are the adversary's budget and the strategy used for partitioning the training data.

6.2.1. *Ensemble member generation and combination rule*

When designing an ensemble-based system, three aspects need to be considered: (i) the partitioning method used to select the training data for each base learner, (ii) the specific procedure used for generating the ensemble members, and (iii) the combination rule for obtaining the ensemble decision. Our technique for training data partitioning is described in Section 6.3, while details on our choices regarding the other two aspects are provided below.

Ensemble-based systems can be seen as *algorithm-free-algorithms*, independent of the type of base learner used to create the ensemble. Model independence allows complete discretion in using the type of learner most suitable for a given application. The pool of base learners in the ensemble can be trained either from the same family or from different families of learning models. Here, we use the same base learner multiple times to generate the ensemble. We do not introduce diversity at model level, but at the level of the training data used by each model. Once the base models have been trained, it is necessary to aggregate their individual outputs to obtain a single final outcome. There are many approaches to model combination, including *linear combiners*, *product combiners*, and *voting combiners*. We adopt the third type of combination as it is amenable when models output class labels. Voting-based combination is known to be the optimal approach for combining independent classifiers [34].[3] The most popular variant of voting combination (known as *hard voting* or *majority voting*) requires every individual learner to vote for a single class. The class that received the most votes is chosen as the ensemble's output. Denoting by N the number of learners in the ensemble and L the number of classes, the decision of the t^{th} learner (C_t) is denoted by $d_{C_t,j} \in \{0,1\}$, where $j = 1,..,L$. If C_t decides for class ω_j, then $d_{C_t,j} = 1$, and 0 otherwise. The ensemble output for the majority voting combiner is computed as

[3] In fact, under minor assumptions 1) N is odd, and 2) each classifier has probability p of correctly classifying a given instance, the ensemble predicts the correct (not corrupted) label if at least $\lfloor N/2 \rfloor + 1$ base classifiers output the correct label.

$$max_{1\leq j\leq L}\sum_{t=1}^{N} d_{C_t,j} \tag{5}$$

6.2.2. Accuracy-diversity breakdown

The term *accuracy-diversity breakdown* of an ensemble of learners is due to two distinct elements, namely the accuracy of the individual learners and their combination. The ensemble succeeds in achieving better accuracy than any of its members if and only if its individual learners are accurate and diverse. However, the exact relationship between diversity and accuracy is difficult to pinpoint. On the one hand, if each base learner makes the same errors, there is no utility in combining their outputs. On the other hand, if base learners are maximally accurate, they provide the same correct predictions, but there is no distinction between them.

The breakdown of the ensemble error depends on both the type of error function and the combination rule. The case of a classification problem with a majority vote combiner is a challenging one, and no accepted theoretical framework capturing the accuracy-diversity breakdown currently exists for it [35]. Regarding robustness, Levine and Feizi [22] introduced a deterministic lower bound on the efficacy of majority voting which is stated as follows. Let us denote by $f_i(x)$ the x classification by the i-th base learner, and by $n_c(x)$ the number of base learners classifying x as c, where the final result of their ensemble $g(x)$ is the index of the maximum n_c. Then, let:

$$t(x) = \lfloor \frac{n_{g(x)} - \max_{c\neq(x)}(c_c(x))}{2} \rfloor \tag{6}$$

Any flipping of less than $t(x)$ labels does not change $g(x)$.

The threshold $t(x)$ represents a lower bound on the number of poisoning flips a defender may withstand with no damage. As a matter of fact, this number is definitely higher in all the protocols we experimented with. Equation (6) highlights a crucial trade-off between robustness and accuracy rooted on the number k of partitions. The gap $n_{g(x)} - \max_{c\neq(x)}(c_c(x))$ grows with the number of partitions k, whereas the accuracy of the single learner decreases with k increasing, because the size of the local training set decreases as well. Levine and Feizi used very large values of k, on the order of thousands of partitions for the MNIST benchmark dataset [36], to exhibit the robustness of their strategy. We will employ a dozen partitions for the same dataset to provide a proper balance between robustness and accuracy (Section 7).

6.3. Anti-clustering for training set partitioning

Diversity plays an important role in ML. Specifically, diversity of the training data ensures they can provide more discriminatory information, resulting in improved performance of the learned model [37]. As we will see, data diversification can also be used for defensive purposes.

6.3.1. Data diversification

By simply reversing the logic behind the popular *k-means* clustering method, Späth [38] and Valev [39] independently coined the term *anti-clustering* to denote a type of data partitioning that ensures similarity between partitions by enforcing dissimilarity within each partition. In other words, the objective is to provide high intra-group dissimilarity and low inter-group dissimilarity. Formally, given the set of elements $T = \{t_1, ..., t_n\}$ and the number of subsets k into which T has to be partitioned, the anti-clustering partitioning defines a set of disjoint partitions (*anti-clusters*) $P_1, ..., P_k$ satisfying the following conditions:

$$\bigcup_{i=1}^{k} P_i = T \tag{7}$$

$$P_i \cap P_k = \emptyset, \forall i, k \in \{1, ..., k\}, i \neq k \tag{8}$$

Condition (7) ensures that every element in T is assigned to at least one of the anti-clusters. On the other hand, condition (8) requires pairwise disjoint anti-clusters. This means that no pair of two anti-clusters can contain the same element. Having partitions of equal size is not mandatory for anti-clustering methods [40]. However, obtaining comparable sized partitions is a desirable property that avoids trivial micro-partitions.

The performance and statistical validity of ML models are affected by both the size of the input data partitions and the distribution of samples across the different partitions [41]. For these reasons, we impose the following additional restriction:

$$\mid P_i \mid = \mid P_k \mid, \forall i, k \in \{1, ...k\} \tag{9}$$

From condition (9) it follows that if the number of elements in T, denoted by N, is a multiple of the number of desired partitions k, then each anti-cluster consists of $\frac{N}{k}$ elements. By contrast, in the case where N is not divisible by k, the anti-clusters will differ by one in their size.

6.3.2. Dissimilarity measure and objective function

Anti-clustering partitioning ensures that points close to each other end up in different partitions. To take advantage of this characteristic for our purposes, we choose a specific criterion for anti-clustering based on information in a *dissimilarity matrix* that accounts for the relevance of the points.

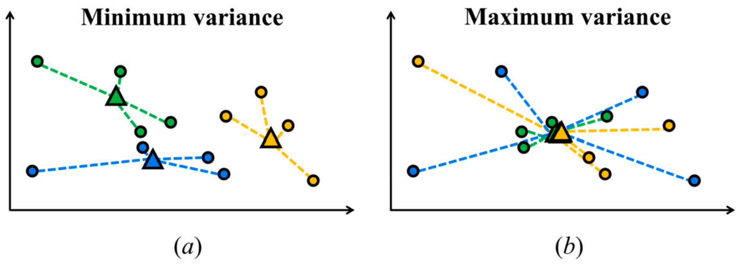


Fig. 4. k-Means objectives. In (*a*) we see that k-means clustering minimizes the variance within partitions. The logic is reversed in (*b*), where anti-clustering k-means maximizes the variance within partitions.

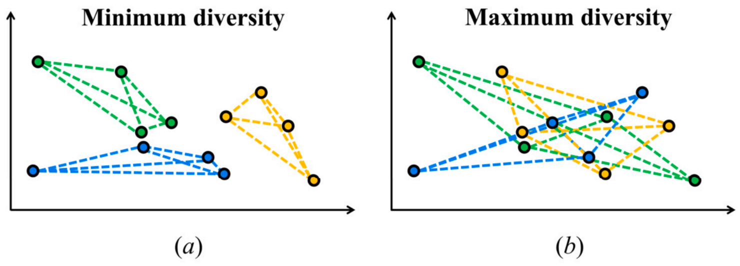


Fig. 5. Cluster editing objectives. (*a*) Cluster editing minimizes the diversity objective (sum of pairwise distances within each cluster), whereas (*b*) anti-cluster editing maximizes it.

Assuming $\pi = \{P_1, ..., P_k\}$ is a partitioning of the set T, where the i-th partition P_i contains a subset of all given elements in T, and assuming Π is the set of all possible partitions, the objective function $f : \Pi \rightarrow \mathbb{R}^+$ associates a positive real number $f(\{P_1, ..., P_k\})$ with each partition. The generic anti-clustering problem may be formulated as follows: *find a feasible partitioning* π^* *such that*

$$f(\pi^*) = max\{f(\pi) \mid \forall \pi \in \Pi\} \tag{10}$$

As for classical clustering methods, there is a wide range of criteria for anti-clustering as well [42]. The most intuitive criterion is the direct reversal of the k-means clustering logic. The goal of *k-means anti-clustering* is to maximize the within-group variance, where within-cluster heterogeneity is measured as the sum of the squared Euclidean distances between individual data points and cluster centers. Fig. 4 shows in a two-dimensional space how the point assignments to three different equal-sized partitions may vary depending on whether the objective function minimizes (k-means clustering) or maximizes (k-means anti-clustering) the variance.

For our scheme, we adopt another type of criterion: *anti-cluster editing*, which is the reverse of the cluster editing clustering paradigm. This criterion measures within-cluster heterogeneity as the sum of pairwise dissimilarity between elements within the same group. The corresponding objective function, which the anti-clustering should maximize, is:

$$D_{diversity} = \Sigma_{1 \le i \le j \le n} d_{ij} x_{ij} \tag{11}$$

where the variable x_{ij} is used to identify whether two elements belong to the same anti-cluster or not:

$$x_{ij} = \begin{cases} 1, & \text{if } x_i \in P_k \wedge x_j \in P_k \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

Fig. 5 shows different assignment of points according to whether the objective function minimizes (cluster editing) or maximizes (anti-cluster editing) diversity. Differently from Fig. 4, where the dashed lines (i.e., the input values of the objective function) are drawn to connect each point with the centroid of the cluster to which the point is assigned. In Fig. 5 the lines are drawn between pairs of points within the same cluster. In Eq. (11) the dissimilarity measure d_{ij} usually corresponds to the Euclidean distance (or squared Euclidean distance), though, in principle, any distance metrics can be used. We employ Euclidean distance in our approach as well, but introducing a distortion in its calculation that takes into account the risk index (Section 6.1) associated with each point. We scale the Euclidean distance to take into account the closeness of the points to the separator hyperplane. The general criterion is that two data points will be closer the higher is their risk index. Thus, a simple distortion is induced via Eq. (13). We use $N \times N$ non-negative symmetric matrix $D = [d_{ij}]$ to represent pairwise dissimilarity measurements, whose entries are computed as follows:

$$d_{ij} = \frac{\| i - j \|}{max(r_i, r_j)} \tag{13}$$

$\| i - j \|$ is the Euclidean distance between two points i and j in the M-dimensional space, and the distortion defined at the fraction denominator is given by the maximum between the risks r_i and r_j of the points under consideration.

7. Experimental evaluation

We evaluated experimentally the impact of our technique on model robustness [43], and compared it to a benchmark derived from the literature. Our experimentation is carried out on two data sets: MNIST, the well-known general benchmark on handwritten digits classification, and MRT, a security dataset of images representing executable program behavior. The MRT dataset has been proposed for competitive training [44] of binary malware classifiers that classify pictorial representations of the activity of ordinary and hostile software (malware). While the MNIST data set is suitable to model a *general* disruption scenario where the attacker aims to decrease the binary classifier's overall performance, the MRT data-set supports a *targeted* disruption scenario where label flipping is aimed at fooling a malware detector, injecting in its training set images that represent a specific malware's behavior but are mislabeled as the behavior of ordinary software.

In terms of the evaluation metrics, we focus on *certified accuracy* as a measurement of certified robustness against any poisoning attacks under certain conditions. There is no single definition of certified accuracy, and different certifiably robust approaches have been proposed in the literature [45]. We rely on the definition we provided in Section 7.2.2.

7.1. Heuristic

Finding a partitioning that maximizes diversity is computationally challenging for $k \geq 2$ and an arbitrary M-dimensional Euclidean space problem. Especially when the number of elements is large, obtaining the optimal anti-clustering partitioning in an acceptable running time is extremely difficult with an exact algorithm. To alleviate these issues, in our implementation we opted for a heuristic algorithm. In particular, we used the *exchange method* proposed by Papenberg and Klau [40].

The procedure is based on exchanging elements between different anti-clusters such that each swap improves the objective value by the largest possible margin. The following steps are repeated for each element. First, samples are randomly assigned to different anti-clusters. Then, the algorithm simulates each possible exchange with elements in other anti-clusters on the basis of the initial assignment – for a total of $(N - \frac{N}{k})$ swaps. The swap that maximally improves the objective function is performed.

7.2. Experimental design

We are now ready to present in detail the design of our experimentation.

7.2.1. Target model

As mentioned earlier, our defense technique is not tied to a specific ML model. Still, to assess its benefit, we need a target to apply it to. We opted for a simple Convolutional Neural Network (CNN) both as the target model and as the base learner for the ensemble. We used our reference SVM model for identifying the risk levels (playing the part of the defender) and the data points to attack (playing the part of the adversary). The SVM has been implemented via the CRANE library of Python. The training of all learners has been carried out in TensorFlow-Keras.

7.2.2. Metrics

Our evaluation metric has three components. We use *clean accuracy* computed on test data to evaluate how models perform on pristine data. The second metric, *after poisoning accuracy*, describes how ML models perform on label-flipped data with varying poisoning rate. This allows us to investigate how our algorithm performs against different poisoning ratios. In addition, we use third metrics, *certified accuracy*, again evaluated on the test set and defined as follows. For a given flipping percentage it is the ratio between: (i) the number of items that have a different label when classified by the function learned from the original training set and the one learned from the poisoned version of the training set, and (ii) the test set size. Additionally, we consider the *ensemble gap*, a measure that applies to the ensemble methods. On each element of the test set, it reckons the difference between the number of base learners whose output coincides with the majority vote result and the number referring to the second highest voted. A large gap denotes high robustness of the result to further perturbations of the training set.

7.2.3. Risk levels

We use 10 colors i.e., ten different values for the risk index calculated using Algorithm 1 (see Section 3.3), plus a 0-level collecting images which do not correspond to any support vector identified during the 10 iterations of the algorithm.

7.2.4. Attacker's budget

In the literature on poisoning attacks the budget of the attacker, in terms of the number of points she is able to modify, almost never exceeds $20-25\%$ of the size of the training set [46]. We use the label-flipping strategy described in Algorithm 2 (see Section 5.2), with a mono-directional flipping. We assess the efficacy of our strategy for multiple flipping rates ρ ranging from 0.08% to 25%. With $\rho \in (0.08, 8)$ Algorithm 2 fully exploits the riskiness difference, where the risk level $r\ell$ translates to the drawing probability via the formula (4). Outside of this range, the sole difference that matters is the one between support vectors and non support vectors (see Section 5.2).

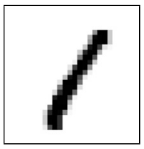


Fig. 6. A sample image from MNIST dataset.

Table 1
Comparison of accuracy and ensemble gap (which applies only to *anticl* and *classic* methods) when no attack is performed, and $k = 15$ partitions.

Method	mon_clean	$classic\_clean$	$anticl\_clean$
Accuracy	0.99594 ± 0.00130	0.99371 ± 0.00075	0.98917 ± 0.00065
Mean gap	-	14.8714	14.907

7.3. Benchmark

To understand the rationale underlying our benchmark design, we consider a generic training set D, two data points $p, q \in D$, and a mapping $h : D \rightarrow U$, where $U \subseteq 2^D$ is a set of partitions over D. We denote the distance between p and q as $|p$ - $q|$ and the probability that $h(p) = h(q)$ as $Pr(h(p) = h(q))$. Given a suitable probability threshold P_1 and a predefined radius R, to behave as a Locality Sensitive Hashing (LSH) h must satisfy the following equation [47]:

$$| p - q | < R \rightarrow Pr(h(p) = h(q)) \geq P_1 \tag{14}$$

Hash techniques [22] partition the training set into disjoint subsets via a hash function, shuffling data among partitions based on their hash.

Rather than choosing a specific hash function h as a benchmark, we derive our benchmarks partitioning the training set randomly without repetition [48], regardless of the risk levels identified by means of Algorithm 1. The partition is then verified to have hash-like data shuffling property for suitable R and $P - 1$ via Eq. (14) before using it as a benchmark. The details of this verification are reported in Section 7.5.1. In our experiments, we compare this *classic* benchmark to the performance of our partitioning, performed according to the anti-clustering technique (*anticl*). We also report (denoting it by *mon*, short for “monolithic”) on the performance of the target ML model trained on the whole training set with the traditional ensemble-less architecture.

7.4. MNIST experiment

Our first experiment was carried out on the classic MNIST benchmark, which contains a collection of 70000 images of hand-written digits from 0 to 9, where the training set and the test set include 60000 and 10000 samples, respectively. This dataset is a typical benchmark for classification algorithms because of the high variability of representations for the same digit. Each sample is represented as a feature vector consisting of a 28×28 grid of 256-valued gray shades. We normalized the pixel values by scaling them to the range $[0, 1]$. In line with the literature, we defined a binary classification task on a portion of the MNIST dataset, i.e. distinguishing between digits 1 and 7. A sample data item is shown in Fig. 6. The result is a total of 15170 images of dimension 784, with 13007 images in the training set and 2163 in the test set. For this experiment, flipping changes the 1 labels to 7.

7.4.1. MNIST: experimental results and analysis

In a preliminary experiment, we randomly selected a subset of 3000 points from the training set and performed the label-flipping attack with different poisoning rates to decide how many partitions k to use for our ensemble trained on all of the data (13007 points). We tested three different values for k, namely 3, 15, and 30. Obviously, the number of partitions k is a parameter that controls the trade-off between robustness and accuracy. By analyzing the curves shown in Fig. 7, we decided to focus on $k = 15$ partitions as a good compromise between robustness and accuracy. The smaller slope of the curve as the percentage of poisoning increases is a property that denotes robustness, albeit at the cost of poor accuracy when few points are poisoned. For each of the experiments we have performed 10 repetitions, thus Table 2, which summarizes our results in terms of accuracy for the three methods, presents average values derived from 10 different runs performed on the whole training set considered.

No attack scenario. First of all, Table 1 shows that in the absence of attacks (denoted by the suffix $\_clean$), the order of accuracy is as follows: $mon\_clean > classic\_clean > anticl\_clean$. The first inequality seems trivial, since, in the absence of local minima, learning based on the whole dataset is more efficient than composing multiple learners on partial training sets. The second inequality confirms that on clean data the partitioning underlying $classic\_clean$, defeats our “smarter” partitioning strategy. However, as Table 1 shows, the difference in accuracy is quite shallow. Although the *classic* benchmark exhibits slightly higher accuracy than the *anticl* one, and thus closer to that of *mon*, our method resulted in only a slight degradation in accuracy when no attack is performed.

Under attack. To evaluate our method’s behavior under attack, we started the experiment by checking which would be the most efficient strategy on the part of the attacker.

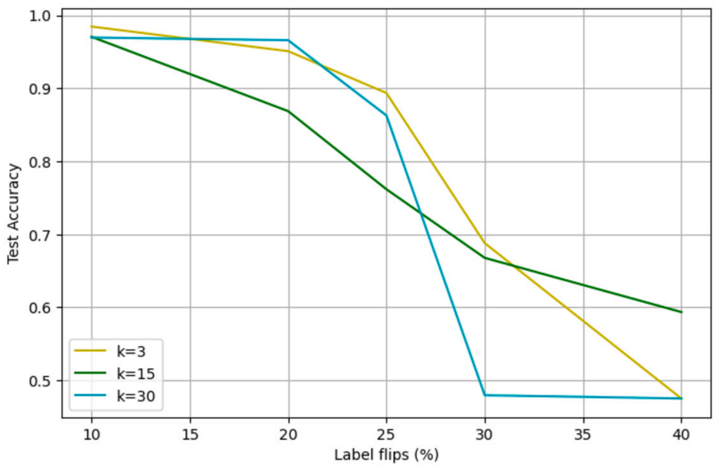


Fig. 7. Accuracy curves for percentage of label-flipping ranging from 10% to 40% and $k \in \{3, 15, 30\}$ partitions.

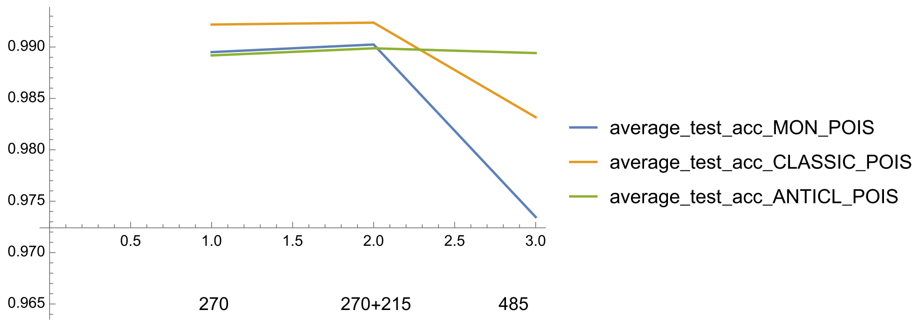


Fig. 8. Accuracy trend of the MNIST dataset with increased number of flips for both bordering (risk levels $1-10$) and inner data points (risk level 0).

Fig. 8 shows a natural order for the experiments where:

a) We flip the labels of 270 images belonging to risk levels $1-10$, with probabilities calculated according to (4);
b) Then, we flip the labels of 215 images belonging to the risk level 0 (lying far from the linear separator);
c) Again, we perform a mono-directional label flip on 485 images, but this time all belonging to risk levels $1-10$, with probabilities calculated according to (4).

We observe that, with respect to effectiveness, we obtain: $\mathbf{a} \simeq \mathbf{b} < \mathbf{c}$. Regarding the ordering of methods in case of attack (denoted by the suffix *_pois*), for low flipping rates we have $classic\_pois > mon\_pois > anticl\_pois$ whereas for higher rates of poisoning, *anticl* turns out to be far superior to the others: $anticl\_pois > classic\_pois > mon\_pois$.

Adding 215 non-support vector images leaves the accuracy almost unchanged, with a very small increase due to the *random shaking* effect of the non-support vector (inner) images. Conversely, choosing additional images close to the boundaries (i.e., from the support vectors) changes performance in a way that will be discussed later. This confirms our strategy for high percentages of poisoning, which consists of first flipping all support vector images and then saturating the desired percentage of poisoned points with inner images.

To confirm the above observations, we carried out a run of the experiment where all flip percentages are less than 3.5%, and all poisoned points are selected from risk levels $1-10$. In particular, percentages range from 0.8% to 3.3% of the training set. This involves from 2% to 81% of the points having risk levels $1-10$, with an absolute number of flipped points ranging from 11 to 431. In a second run of the experiment, we considered higher percentages of poisoning, i.e., between 10% and 25%. Clearly, in this second case, we adopted mono-directional flipping of all the points of levels $1-10$ having label 1 (for a total of 527 points) plus other points taken from level 0 until saturating the desired percentage. Thus, the total number of flipped points (both levels $1-10$ and level 0) varies from 1301 to 5203. As we can observe from Fig. 9, the results confirm that when the poisoning rate is consistent (say from 6% to 25%), our method has a higher accuracy compared to both *classic* and *mon* methods. In particular, the percentages between 10 and 20 are the ones for which *anticl* is more robust.

In a final run of our experiment, we assessed the degree of degradation reached by the different methods with flipping percentages of more than 30% of the training set. The results reported in Table 2 (last two lines) show that regardless of the method, accuracy decreases dramatically when the rate is so high as to bring the classification to accuracy values not much different from a mere

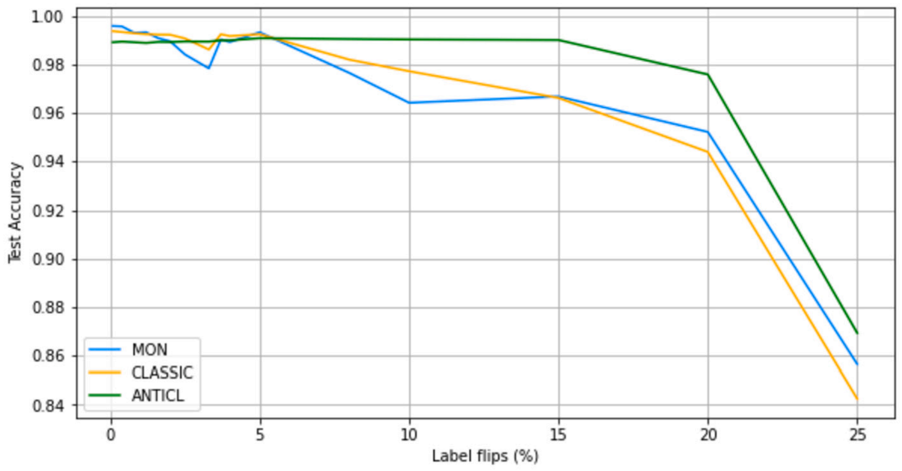


Fig. 9. Accuracy comparison with MNIST dataset for $k = 15$ partitions and poisoning rate ranging from 0.08% to 25% of the training set size.

Table 2
Results of the experiment on the MNIST dataset using different label-flipping attack strategies. Averaged test accuracy plus/minus the standard deviation as a function of the poisoning points. $\%_{tr}$ and $\%_{rl_{1\_10}}$ are the percentage of flipped points with respect to the total number of training data and the percentage of flipped points with respect to the total number of points with risk level $1-10$ with label 1, respectively. $\#_{rl_{1\_10}}$ is the numerical equivalent of $\%_{rl_{1\_10}}$, $\#_{rl_0}$ is the number of flipped points with risk level 0, and $\#_{tot}$ is the total number of flipped points (levels $0-10$).

Label flips					Acc^*		
$\%_{tr}$	$\%_{rl_{1\_10}}$	$\#_{rl_{1\_10}}$	$\#_{rl_0}$	$\#_{tot}$	MON	$CLASSIC$	$ANTICL$
0.08	2	11	0	11	0.99579±0.00153	0.99366±0.00065	0.98908±0.00049
0.4	10	54	0	54	0.99556±0.00117	0.99329±0.00058	0.98936±0.00068
0.8	20	108	0	108	0.99301±0.00215	0.99278±0.00049	0.98908±0.00082
1.2	31	162	0	162	0.99320±0.00254	0.99241±0.00032	0.98881±0.00091
1.6	41	216	0	216	0.99079±0.00316	0.99223±0.00052	0.98922±0.00061
2	51	270	0	270	0.98950±0.00223	0.99218±0.00059	0.98918±0.00054
2.5	61	323	0	323	0.98404±0.00421	0.99066±0.00153	0.98941±0.00066
3.3	81	431	0	431	0.97831±0.00787	0.98608±0.00273	0.98936±0.00065
3.7	51	270	215	485	0.99024±0.00197	0.99237±0.00049	0.98987±0.00079
4	51	270	270	540	0.98918±0.00579	0.99163±0.00120	0.98987±0.00070
5	39	207	443	650	0.99320±0.00396	0.99241±0.00054	0.98881±0.00078
8	98	520	520	1040	0.97646±0.00465	0.98192±0.00428	0.99042±0.00084
10	100	527	774	1301	0.96417±0.01027	0.97716±0.00454	0.99024±0.00107
15	100	527	1424	1951	0.96675±0.00677	0.96615±0.00947	0.99001±0.00183
20	100	527	2074	2601	0.95214±0.01629	0.94392±0.01272	0.97582±0.00442
25	100	527	2725	3252	0.85654±0.12899	0.84248±0.04135	0.86934±0.05094
30	100	527	3375	3902	0.57318±0.09154	0.59052±0.04075	0.53550±0.04942
40	100	527	4676	5203	0.47531±0.00014	0.47526±1.17027	0.47526±1.17027

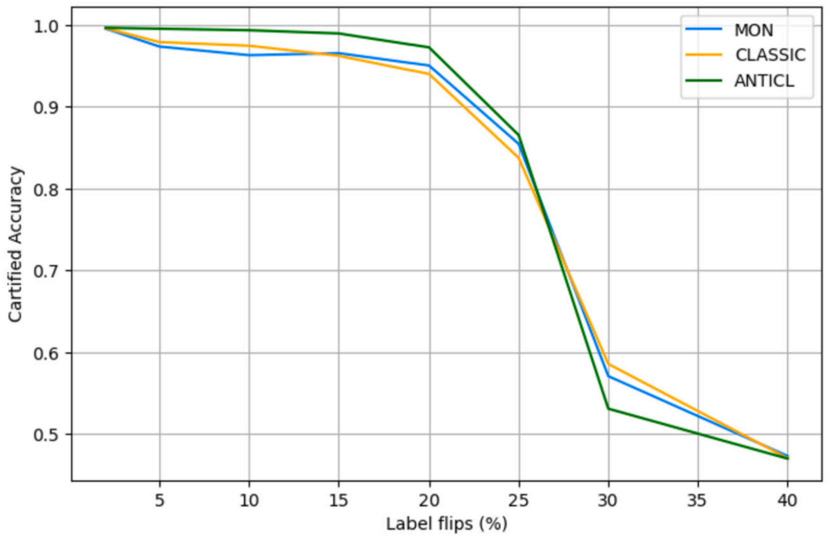
coin toss (accuracy close to 50%). The accompanying trend in standard deviations confirms the general behavior: we have moderate values for no or low poisoning rates (say less than 25%), which strongly increase beyond this threshold, until reaching equivalent values in the case of the two ensemble methods that are definitely lower than the corresponding value for *mon*. It is worth noting that, although in principle an attacker can arbitrarily increase the number of corrupted points in the training set, in real-world applications, significantly increasing the poisoning rate inevitably leads to making the attack obvious.

Results on certified accuracy. In Table 3 we report the results up to the poisoning percentage of 40% in terms of the actual number of points in the test set that did not change their labels after the label-flipping attack. The corresponding graphs are given in Fig. 10. We note a benefit similar to that observed for the accuracy metric: up to 25% *anticl* prevails over the other two methods, where up to 20% the number of certified points is very high. Beyond the 25% threshold, the trend is reversed, and training corruption leads to essentially random classifications.

Ensemble gap. The gap between the number of voters of the highest and second highest voted result of an ensemble is a measure of the robustness of an ensemble classifier against poisoning attacks, as highlighted by (6). The second row of Table 1 shows a small superiority of *anticl* over *classic* in case of a clean training set, which apparently reverses in the case of a poisoned training set.

Table 3
Certified accuracy against the MNIST dataset, with $k = 15$ partitions for *anticl* and *classic* methods. $cert_{acc}$ is the percentage quantifying the output changes in case of label contamination of the training set; $cert_{points}$ is the corresponding number of certified points.

Label flips	$cert_{points}$			$cert_{acc}$		
$\%_{tr}$	MON	CLASSIC	ANTICL	MON	CLASSIC	ANTICL
2	2153	2154	2156	0.99565	0.99625	0.99676
5	2106	2118	2153	0.97383	0.97928	0.99565
10	2083	2108	2149	0.96329	0.97484	0.99380
15	2088	2081	2141	0.96574	0.96250	0.98987
20	2056	2034	2104	0.95067	0.94036	0.97286
25	1848	1811	1872	0.85478	0.83767	0.86555
30	1234	1266	1148	0.57068	0.58562	0.53088
40	1023	1017	1016	0.47304	0.47031	0.46985



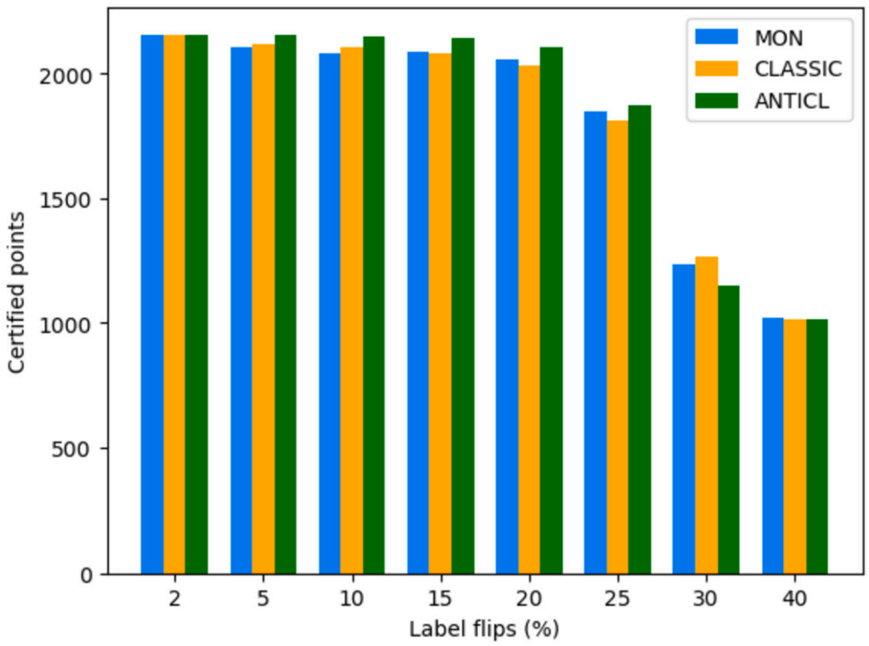


Fig. 10. Certified accuracy comparison against the MNIST dataset, with $k = 15$ partitions for *anticl* and *classic* methods. The figure on the top shows the certified accuracy to label-flipping poisoning attacks. The figure on the bottom shows how the corresponding number of certified points changes as the flip percentage increases.

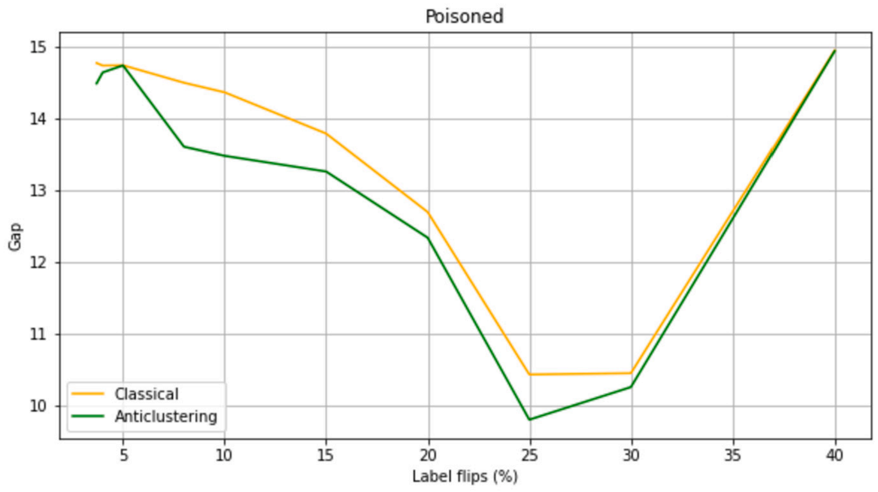


Fig. 11. Gap comparison between the *anticl* and *classic* ensemble methods with the MNIST dataset.

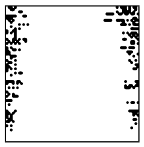


Fig. 12. A MRT dataset's data point showing the pictorial representation of the events generated by malware in a 500 msec time window.

A smaller gap with the poisoned training set again denotes a superiority of our method, which raises more contested voters when the majority result is wrong. The gap difference of the two methods is so small that it invests exactly the faulty classified points, the number of which in turn increases with the poisoning rate. Thus, within the previously mentioned range $(6\%, 25\%)$ of the poisoning rate, we observe this virtuous phenomenon that tends to vanish for higher rates (see Fig. 11).

7.5. MRT experiment

The accuracy of behavioral malware detection largely depends on the availability of manually labeled training data. Manipulating malware data by targeted label flipping can cause dangerous malware mis-classifications in the field [49]. The MRT dataset is a manually labeled dataset composed of 4465 samples of malware behavior, taken from sand-boxed execution of a set of over 170 malware programs and 200 benign applications. Each data point represents the number of events (out of a vocabulary of 128 event types) generated by a software program in a sampling interval of 500 msec. The data set is highly unbalanced (3000 benign, 1465 malign samples). The pictorial representation of each data point is computed via direct decimal to binary conversion, obtaining 4465 black and white images composed of 64 * 64 pixels. A sample MRT data item is shown in Fig. 12. This experiment provides an example of the process to be followed for applying our technique in practice. Malware classifiers are deployed on devices as frozen, read-only code. To fool the classifiers, attackers can only target training data, by performing mono-directional flipping and changing the label "malign" into "benign" for some malware.

The malware detector developer who does not fully trust the training data applies our methodology by (i) partitioning according to our method the training set received from the labeler, (ii) training a ML model for each partition, and (iii) combining the ML models into an ensemble, which is deployed on the devices. The choice of certified accuracy as the effectiveness metrics is a natural one on the part of the developer, as it expresses the success rate of the attacker in modifying the malware detector's performance.

7.5.1. Practical implementation details

We refer to our CNN target model, this time trained on the MRT data set. The *classic* benchmark for MRT has been validated as follows. We extracted a sample of 50 elements from the training set. For each pair of points in the training set, we computed their Euclidean distance, obtaining a total of 1225 values, and computed the average distance R. We then selected the pairs with distance falling within the range $[R-\sigma,\ R+\sigma]$, where σ is the standard deviation. For each of the corresponding data points, we ascertained to which partitions they were assigned and computed the ratio between the pairs of points for which the target partition was divergent and the total number of pairs. This verifies the hash-like data shuffling behavior of this partitioning. The benchmark we used has $R = 2.641484$ and $P_1 = 56\%$ (see Section 7.3), in line with the literature.

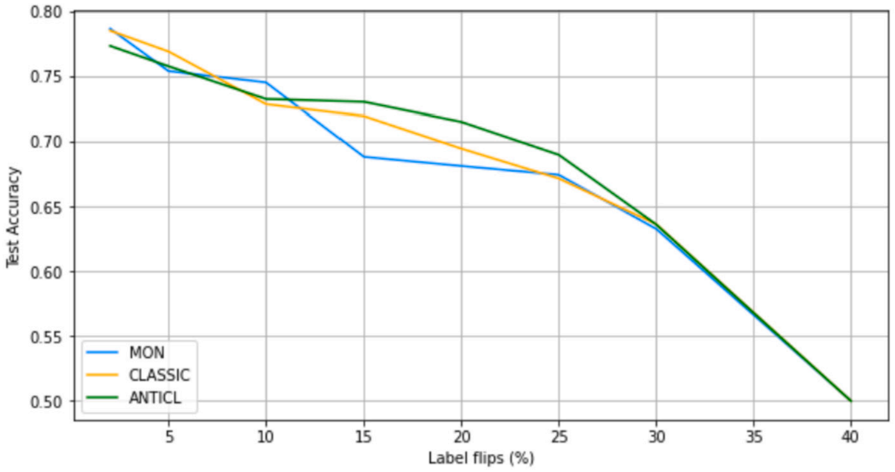


Fig. 13. Accuracy comparison for $k = 5$ partitions and poisoning rate up to 40% of the MRT training set size.

Table 4
Certified accuracy against the MRT dataset, with $k = 5$ partitions for *anticl* and *classic* methods. $cert_{acc}$ is the percentage quantifying the output changes in case of label contamination of the training set; $cert_{points}$ is the corresponding number of certified points.

Label flips	$cert_{points}$			$cert_{acc}$		
$\%_{tr}$	MON	CLASSIC	ANTICL	MON	CLASSIC	ANTICL
2	1056	1100	1115	0.88058	0.917	0.92991
5	886	1063	1138	0.73875	0.88614	0.94875
10	1034	1043	1089	0.86178	0.86916	0.90797
15	926	970	1119	0.77238	0.80880	0.93285
20	921	985	1075	0.76797	0.82107	0.89654
25	968	970	1006	0.807	0.80875	0.83891
30	913	936	926	0.76125	0.78025	0.77241
40	660	714	756	0.55075	0.59533	0.63033

7.5.2. MRT: experimental results and analysis

We report the results of the MRT experiments. Taking into account the results of the first experiment and the smaller size of the MRT dataset compared to MNIST, we decided to focus on $k = 5$ partitions as a good compromise between robustness and accuracy. The next subsections contain the comparative analysis of the performance for this number of partitions.

No attack scenario. The best accuracy of the ML model on MRT achieved by the literature [50] is around 0.95 for a CNN monolithic model, while the accuracy we obtained with our implementation is around 0.82. This difference may be due to hyper-parameters tuning, to architectural differences or to other factors like a higher number of training epochs. In the absence of attacks, the order of accuracy is $mon\_clean \geq classic\_clean > anticl\_clean$ as in the first experiment. However, all partitioning techniques deliever similar results in terms of accuracy on clean data.

Under attack. This experiment results show that the monolithic model is quite vulnerable to attack, with accuracy falling below 0.7 already with a 15% of flipped samples. This is quite critical for a malware detector. The $classic$ benchmark handles small attack budgets quite well but performs badly for budgets above 10%. Our $anticl$ technique performs steadily better for budgets larger than 10% (Fig. 13). We also assessed the degree of degradation reached by the different methods with percentages between 30% and 40% of the MRT training set. The results, reported in Fig. 13 (right-hand side) show that accuracy decreases somewhat less than in the previous experiment (especially for training corruption of 30%), although the trend is pretty similar.

Results on certified accuracy. In terms of certified accuracy, in this experiment *anticl* clearly prevails over the other two methods for all attackers' budgets, keeping the certified accuracy over 90% even for a flip rate of 20% (Fig. 14). Results are summarized in Table 4.

7.6. Discussion

Both our experiments support two notions: (i) deploying an ensemble with a risk-driven partitioning of the training set instead of a monolithic ML model can alleviate the impact of targeted label flipping even employing a small number of partitions, and (ii)

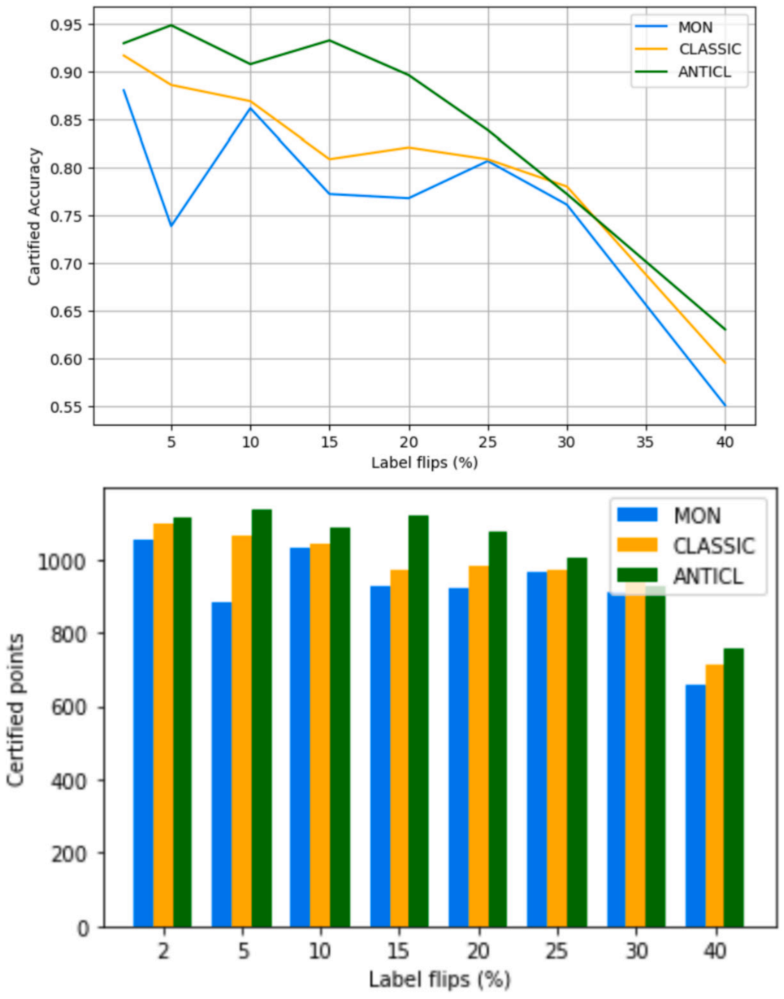


Fig. 14. Certified accuracy comparison against the MRT dataset, with $k = 5$ partitions for *anticl* and *classic* methods. The figure on the top shows the certified accuracy to label-flipping poisoning attacks. The figure on the bottom shows how the corresponding number of certified points changes as the flip percentage increases.

risk-driven anti-clustering provides better certified accuracy than hash-style shuffling, at the price of a slightly lower performance on clean data. We argue the first result is of high practical importance, as the number of learners that can be actually deployed may depend on in-production resource constraints and should be kept as low as possible. The second result confirms the expectation that anti-clustering provides a sounder dispersion control than hashing.

8. Conclusions

Developing appropriate defense mechanisms specific for ML assets is of paramount importance. In this paper, we focused on training data, which is a key asset of any ML system, and investigated how to make ML models more robust w.r.t. training data poisoning attacks.

A major gap we identified in the literature is that defense techniques are strictly tied to the target ML model's hyper-parameters. We argue that from the security standpoint these defense techniques provide an understanding of the vulnerabilities of specific ML models, but cannot be used as a foundation of a general methodology for securing ML assets [30].

Another important gap we found in the literature is the lack of general approaches to estimate the risk associated with ML data assets.

In an effort toward filling these two gaps, we described a defense mechanism applicable to a variety of ML models, combining ensemble composition and security risk analysis. Our technique relies on a SVM as the reference model to represent the attacker's (and the defender's) knowledge about the targets.

In addition to linear classification, SVMs can efficiently perform non-linear classification by mapping their inputs into a higher-dimensional feature space. In our future work, we plan to investigate how the structure of this mapping can be used by either contender to take advantage of available context information, especially the one regarding the model to be attacked or defended. We also plan to introduce *generic risk landscapes*, where the color and the corresponding risk index value of the data points depend on context information, such as non-uniform cost of the attack for different regions in the input data space.

We believe our work lays the foundation for standardizing automatic support of training set partitioning and ensemble generation within the ML models' development and deployment lifecycle.

CRediT authorship contribution statement

Lara Mauri: Conceptualization, Data curation, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Bruno Apolloni:** Formal analysis, Supervision. **Ernesto Damiani:** Conceptualization, Methodology, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgement

This work has been partly funded by the Università degli Studi di Milano with the project SOV-EDGE-HUB (n. PSRL621EDAMI_01), within the *Grandi Sfide d'Ateneo* program.

References

[1] D.A. Neu, J. Lahann, P. Fettke, A systematic literature review on state-of-the-art deep learning methods for process prediction, Artif. Intell. Rev. 55 (2) (2022) 801–827.

[2] R.S.S. Kumar, M. Nyström, J. Lambert, A. Marshall, M. Goertzel, A. Comissoneru, M. Swann, S. Xia, Adversarial machine learning-industry perspectives, in: 2020 IEEE Security and Privacy Workshops (SPW), IEEE, 2020, pp. 69–75.

[3] O. Sagi, L. Rokach, Ensemble learning: a survey, WIREs Data Min. Knowl. Discov. 8 (4) (2018), https://doi.org/10.1002/widm.1249.

[4] R. Schuster, C. Song, E. Tromer, V. Shmatikov, You autocomplete me: poisoning vulnerabilities in neural code completion, in: 30th USENIX Security Symposium (USENIX Security 21), USENIX Association, 2021, pp. 1559–1575.

[5] S. Mei, X. Zhu, Using machine teaching to identify optimal training-set attacks on machine learners, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI'15, AAAI Press, 2015, pp. 2871–2877.

[6] M.A. Ramirez, S.-K. Kim, H.A. Hamadi, E. Damiani, Y.-J. Byon, T.-Y. Kim, C.-S. Cho, C.Y. Yeun, Poisoning attacks and defenses on artificial intelligence: a survey, arXiv preprint, arXiv:2202.10276, 2022.

[7] A. Paudice, L. Muñoz-González, E.C. Lupu, Label sanitization against label flipping poisoning attacks, in: ECML PKDD 2018 Workshops, Springer International Publishing, Cham, 2019, pp. 5–15.

[8] F.R. Hampel, Contributions to the Theory of Robust Estimation, University of California, Berkeley, 1968.

[9] A. Prasad, A.S. Suggala, S. Balakrishnan, P. Ravikumar, Robust estimation via robust gradient estimation, J. R. Stat. Soc., Ser. B, Stat. Methodol. 82 (3) (2020) 601–627.

[10] P.W. Koh, J. Steinhardt, P. Liang, Stronger data poisoning attacks break data sanitization defenses, arXiv:1811.00741, 2021.

[11] E. Borgnia, J. Geiping, V. Cherepanova, L. Fowl, A. Gupta, A. Ghiasi, F. Huang, M. Goldblum, T. Goldstein, Dp-instahide: provably defusing poisoning and backdoor attacks with differentially private data augmentations, arXiv:2103.02079, 2021.

[12] J. Geiping, L. Fowl, G. Somepalli, M. Goldblum, M. Moeller, T. Goldstein, What doesn't kill you makes you robust(er): adversarial training against poisons and backdoors, arXiv:2102.13624, 2021.

[13] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.

[14] B. Biggio, I. Corona, G. Fumera, G. Giacinto, F. Roli, Bagging classifiers for fighting poisoning attacks in adversarial classification tasks, in: Proceedings of the 10th International Conference on Multiple Classifier Systems, MCS'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 350–359.

[15] T.K. Ho, The random subspace method for constructing decision forests, IEEE Trans. Pattern Anal. Mach. Intell. 20 (8) (1998) 832–844, https://doi.org/10.1109/34.709601.

[16] B. Biggio, G. Fumera, F. Roli, Multiple classifier systems under attack, in: N. El Gayar, J. Kittler, F. Roli (Eds.), Multiple Classifier Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 74–83.

[17] E. Rosenfeld, E. Winston, P. Ravikumar, Z. Kolter, Certified robustness to label-flipping attacks via randomized smoothing, in: H.D. III, A. Singh (Eds.), Proceedings of the 37th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 119, PMLR, 2020, pp. 8230–8241.

[18] M. Weber, X. Xu, B. Karlaš, C. Zhang, B. Li Rab, Provable robustness against backdoor attacks, arXiv:2003.08904, 2021.

[19] J. Gao, A. Karbasi, M. Mahmoody, Learning and certification under instance-targeted poisoning, arXiv:2105.08709, 2021.

[20] J. Jia, X. Cao, N.Z. Gong, Intrinsic certified robustness of bagging against data poisoning attacks, in: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, the Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, AAAI Press, 2021, pp. 7961–7969.

[21] J. Jia, X. Cao, N.Z. Gong, Certified robustness of nearest neighbors against data poisoning attacks, CoRR, arXiv:2012.03765 [abs], 2020.

[22] A. Levine, S. Feizi, Deep partition aggregation: provable defense against general poisoning attacks, arXiv:2006.14768, 2021.

[23] A. Subbaswamy, R. Adams, S. Saria, Evaluating model robustness and stability to dataset shift, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2021, pp. 2611–2619.

[24] V. Bellandi, S. Cimato, E. Damiani, G. Gianini, A. Zilli, Toward economic-aware risk assessment on the cloud, IEEE Secur. Priv. 13 (6) (2015) 30–37.

[25] Z. Li, W. Xu, H. Shi, Y. Zhang, Y. Yan, Security and privacy risk assessment of energy big data in cloud environment, Comput. Intell. Neurosci. (2021) 2021.

[26] D.M. Johnson, C. Xiong, J.J. Corso, Semi-supervised nonlinear distance metric learning via forests of max-margin cluster hierarchies, IEEE Trans. Knowl. Data Eng. 28 (4) (2016) 1035–1046, https://doi.org/10.1109/TKDE.2015.2507130.
[27] ENISA, AI cybersecurity challenges – threat landscape for artificial intelligence, December 2020.
[28] B. Caroline, B. Christian, B. Stephan, B. Luis, D. Giuseppe, E. Damiani, H. Sven, L. Caroline, M. Jochen, D.C. Nguyen, et al., Securing machine learning algorithms, 2021.
[29] L. Mauri, E. Damiani, Estimating degradation of machine learning data assets, ACM J. Data Inf. Qual. (JDIQ) 14 (2) (2021) 1–15.
[30] L. Mauri, E. Damiani, Modeling threats to AI-ML systems using STRIDE, Sensors 22 (17) (2022), https://doi.org/10.3390/s22176662, https://www.mdpi.com/1424-8220/22/17/6662.
[31] B. Apolloni, S. Bassis, D. Malchiodi, P. Witold, The Puzzle of Granular Computing, Studies in Computational Intelligence, vol. 138, Springer Verlag, 2008.
[32] A.E. Cinà, S. Vascon, A. Demontis, B. Biggio, F. Roli, M. Pelillo, The hammer and the nut: is bilevel optimization really needed to poison linear classifiers?, CoRR, arXiv:2103.12399 [abs], 2021.
[33] Z. Yang, L. Li, X. Xu, B. Kailkhura, T. Xie, B. Li, On the certified robustness for ensemble models and beyond, arXiv:2107.10873, 2021.
[34] C. Zhang, Y. Ma, Ensemble Machine Learning: Methods and Applications, Springer, Boston, MA, 2012.
[35] G. Brown, Ensemble learning, in: C. Sammut, G.I. Webb (Eds.), Encyclopedia of Machine Learning, Springer, 2010, pp. 312–320.
[36] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (1998) 2278–2324, http://yann.lecun.com/exdb/mnist/.
[37] Z. Gong, P. Zhong, W. Hu, Diversity in machine learning, IEEE Access 7 (2019) 64323–64350, https://doi.org/10.1109/ACCESS.2019.2917620.
[38] H. Späth, Anticlustering: maximizing the variance criterion, Control Cybern. 15 (1986) 213–218.
[39] V. Valev, Set partition principles, in: J. Kozesnik (Ed.), Transactions of the Ninth Prague Conference on Information Theory, Statistical Decision Functions, and Random Processes, Prague, 1982, Springer Netherlands, T. Prague, 1983, pp. 251–256.
[40] M. Papenberg, G.W. Klau, Using anticlustering to partition data sets into equivalent parts, Psychol. Methods 26 (2) (2021) 161–174, https://doi.org/10.1037/met0000301.
[41] A. Dagli, N. McCarroll, D. Vasilenko, Data partitioning for ensemble model building, Int. J. Cloud Comput., Serv. Archit. (IJCCSA) 7 (3/4) (2017).
[42] M.J. Brusco, J.D. Cradit, D. Steinley, Combining diversity and dispersion criteria for anticlustering: a bicriterion approach, Br. J. Math. Stat. Psychol. 73 (3) (2020).
[43] S. Scher, A. Trügler, Robustness of machine learning models beyond adversarial attacks, CoRR, arXiv:2204.10046 [abs], 2022, https://doi.org/10.48550/arXiv.2204.10046.
[44] H. Al-Hamadi, M. Morcos, Gan-based training for binary classifier, https://kaggle.com/competitions/gan-based-training-for-binary-classifier, 2022.
[45] L. Li, X. Qi, T. Xie, B. Li, Sok: certified robustness for deep neural networks, CoRR, arXiv:2009.04131 [abs], 2020.
[46] B. Biggio, F. Roli, Wild patterns: ten years after the rise of adversarial machine learning, Pattern Recognit. 84 (2018) 317–331, https://doi.org/10.1016/j.patcog.2018.07.023.
[47] D. Li, W. Zhang, S. Shen, Y. Zhang, Ses-lsh: shuffle-efficient locality sensitive hashing for distributed similarity search, in: 2017 IEEE International Conference on Web Services (ICWS), 2017, pp. 822–827.
[48] J. Karasek, R. Burget, O. Morskỳ, Towards an automatic design of non-cryptographic hash function, in: 2011 34th International Conference on Telecommunications and Signal Processing (TSP), IEEE, 2011, pp. 19–23.
[49] F. Maasmi, M. Morcos, H. al Hamadi, E. Damiani, Identifying applications' state via system calls activity: a pipeline approach, in: 2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS), 2021, pp. 1–6.
[50] F. Wang, H. AlHammadi, E. Damiani, A visualized malware detection framework with CNN and conditional GAN, in: IEEE Bigdata Cup 2022, 2022, pp. 801–817.