ScienceDirect

Computers & Security

Volume 155, August 2025, 104468

Protecting machine learning from poisoning attacks: A risk-based approach ☆

Nicola Bena [a b 1], Marco Anisetti [a], Ernesto Damiani [a b], Chan Yeob Yeun [b], Claudio A. Ardagna [a b 1]

Show more

Outline | Share | Cite

https://doi.org/10.1016/j.cose.2025.104468 ↗

Abstract

The ever-increasing interest in and widespread diffusion of Machine Learning (ML)-based applications has driven a substantial amount of research into offensive and defensive ML. ML models can be attacked from different angles: poisoning attacks, the focus of this paper, inject maliciously crafted data points in the training set to modify the model behavior; adversarial attacks maliciously manipulate inference-time data points to fool the ML model and drive the prediction of the ML model according to the attacker's objective. Ensemble-based techniques are among the most relevant defenses against poisoning attacks and replace the monolithic ML model with an ensemble of ML models trained on different (disjoint) subsets of the training set. They assign data points to the training sets of the models in the ensemble (routing) randomly or using a hash function, assuming that evenly distributing poisoned data points positively influences ML robustness. Our paper departs from this assumption and implements a risk-based ensemble technique where a risk management process is used to perform a smart routing of data points to the training sets. An extensive experimental evaluation demonstrates the effectiveness of the proposed approach in terms of its soundness, robustness, and performance.

Previous Next

Keywords

Ensemble; Machine learning; Poisoning; Risk; Robustness

1. Introduction

From the release of chatbots based on large-language models such as ChatGPT, artificial intelligence (AI), in general, and machine learning (ML), in particular, has become one of the most hyped technologies in the world. ML solutions have already entered the market with an expected economic impact of $19.9 trillion by 2030[2]; companies embracing AI reported productivity gains up to 50%.[3] Use cases are countless, from agriculture (Khanh et al., 2024) to malware detection (Bena et al., 2024, Piskozub et al., 2021), from emotion recognition (D'Errico et al., 2024, Zhang et al., 2023) to social network security (Caruccio et al., 2023), from engineering (La Ferlita et al., 2023) to password strength assessment (Atzori et al., 2024) and healthcare (Bellandi et al., 2024, Maghool et al., 2024), to name but a few.

This exponential and ever-increasing growth has sparked an intense debate on the importance of secure and trustworthy AI, involving researchers (e.g., Anisetti et al., 2023b and Cinà et al., 2023), practitioners (e.g., Nicolae et al., 2019), and policymakers (e.g., the European Union Artificial Intelligence Act[4]), and opening a whole new body of research on offensive and defensive ML.

This paper focuses on ML robustness with particular reference to *poisoning attacks*, which are carried out at training time by injecting malicious samples in the training set with the objective of degrading the robustness of the ML model in general (e.g., Aryal et al., 2022, Biggio et al., 2012 and Calzavara et al., 2025), or with respect to a particular class (e.g., Chen et al., 2019) or (set of) data point (e.g., Chen et al., 2017, Gu et al., 2017 and Shafahi et al., 2018). Poisoning defenses are commonly divided in *dataset* and *model hardening*. The former attempts to identify (e.g., De Gaspari et al., 2024), remove (e.g., Peri et al., 2020), or heal (e.g., Rosenfeld et al., 2020) poisoned data points. The latter attempts to modify the ML model to increase its robustness. In this context, ensemble techniques replace the monolithic ML model with an ensemble of ML models (e.g., Anisetti et al., 2023a, Jia et al., 2021, Levine and Feizi, 2021 and Wang et al., 2022). Ensemble techniques have achieved remarkable robustness in numerous scenarios (Anisetti et al., 2023a, Levine and Feizi, 2021, Mauri et al., 2023, Wang et al., 2022) by replacing the monolithic ML model with an ensemble of ML models trained on (disjoint) subsets of the training set. They are based on a *routing approach* that assigns data points to the training sets of the models in the ensemble randomly (Jia et al., 2021) or using a hash function (Anisetti et al., 2023a, Levine and Feizi, 2021, Wang et al., 2022), assuming that *evenly distributing poisoned data points positively influences ML robustness*. Our paper aims to relax this assumption, which might not hold in a general case, and focuses on the following research question:

Can we enhance ML robustness by implementing an ensemble technique that builds on a routing approach that considers the specificity of each data point in the training set?

Our paper aims to answer this question in a scenario that focuses on binary classification, where the attacker controls the training set and maliciously flips the labels (*label flipping*) to decrease the recall of the resulting ML model. Specifically, we propose a risk-based ensemble technique that substitutes random or hash-based routing in literature with a *risk-based* routing. Our ensemble technique implements a risk management process that *(i)* assesses the *poisoning risk* of each data point according to a set of indicators called *Indicators of Poisoning* (IoPs) (*risk assessment*) and *(ii)* generates different disjoint subsets of the training set, each used to train a ML model in the ensemble, according to two alternative routing algorithms driven by the risk computed at point *(i)* (*risk treatment*). The prediction at inference time is finally retrieved according to majority voting. An extensive experimental evaluation, varying datasets, poisoning attacks, percentage of poisoning, and number of models in the ensemble, shows that our technique provides higher robustness compared to existing ensemble techniques, even with a moderate number of models, provided that the training set is sufficiently robust.

Our contribution is threefold. We first define and implement a set of IoPs that analyze the risk of data points in the training set from different perspectives. We then implement a new ensemble technique based on a risk management process. To the best of our knowledge, our ensemble technique is the first model-hardening approach that implements a *smart* routing based on risk analysis. We finally extensively evaluate our technique in terms of its soundness, robustness, and performance.

The remainder of this paper is organized as follows. Section 2 provides an overview of poisoning attacks and defenses. Section 3 introduces our threat model and our approach at a glance. Section 5 describes our risk management process on the basis of the IoPs in Section 4. Section 6 presents the evaluation process and experimental settings, while Section 7 details the experimental results. Section 8 discusses our main findings. Section 9 presents the related work and Section 10 draws our conclusions.

2. Background

Attacks against ML models occur at two main stages: training and inference. Training-time attacks aim to tamper the model to change its predictions at inference time (*poisoning attacks*), while inference-time attacks aim to fool the model into wrongly classifying specific data points (*adversarial attacks*). These attacks virtually affect every ML model in numerous application scenarios (Cinà et al., 2023).

Poisoning attacks, the focus of this paper, apply malicious modifications to the dataset by inserting specially crafted data points, or removing/perturbing existing data points. A poisoning attack can be *(i) untargeted*, aiming to reduce generic quality metrics such as accuracy (e.g., Aryal et al., 2022,

Biggio et al., 2012 and Calzavara et al., 2025); *(ii) semi-targeted*, aiming to reduce quality metrics related to a specific class or subset of data points (e.g., avoid the identification of malware samples Chen et al., 2019); or *(iii) targeted*, aiming to misclassify specific data points at inference time, without significantly decreasing the overall ML model performance (e.g., attacks based on *feature collision* Shafahi et al., 2018 or *backdoor attacks* Chen et al., 2017, Gu et al., 2017).[5]

A poisoning attack employs a strategy to *select* the data points to be poisoned. When the objective is untargeted, the strategy can correspond to random choice (e.g., Anisetti et al., 2023a, Aryal et al., 2022 and Nowroozi et al., 2024). The strategy frequently depends on the knowledge of the attacker on the ML model: *(i) black box*, when the attacker has (complete) knowledge of the dataset only; *(ii) white box*, when the attacker has complete knowledge of the dataset and ML model, including any defenses; and *(iii) gray box*, when the attacker has partial knowledge of either the dataset or ML model.

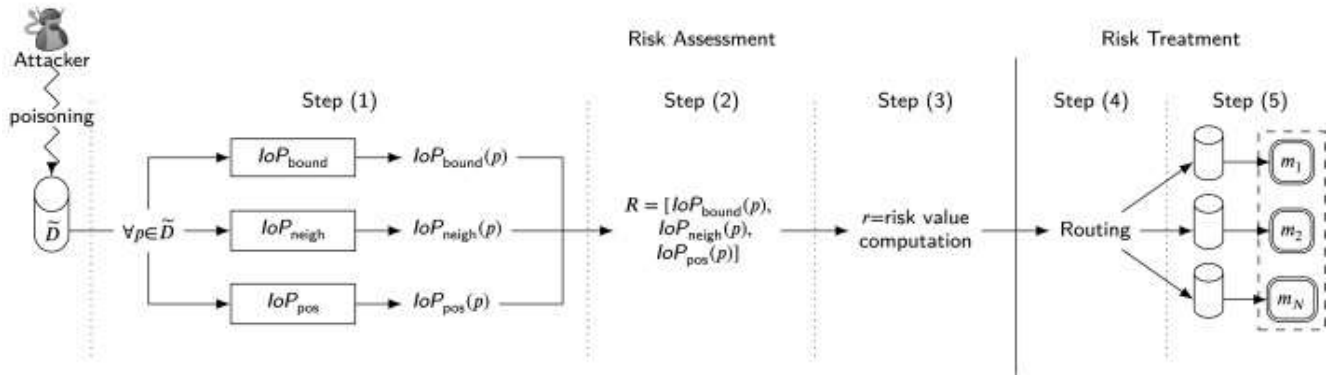
A poisoning attack then perturbs *(i)* the sample/features of the data point (e.g., De Gaspari et al., 2024); *(ii)* the label of the data point (called *label flipping*, e.g., Anisetti et al., 2023a, Aryal et al., 2022 and Calzavara et al., 2025); and *(iii)* the sample/features and labels of the data point. *Label flipping* is one of the simplest and most powerful attacks, and thus widely investigated. Its efficacy has been demonstrated in several domains and ML models, from k-Nearest Neighbors (e.g., Aryal et al., 2023, Shahid et al., 2022, Yerlikaya and Bahtiyar, 2022 and Zhang et al., 2023) to Support Vector Machine (e.g., Aryal et al., 2023, Biggio et al., 2012, Yerlikaya and Bahtiyar, 2022 and Zhang et al., 2021), from decision tree (e.g., Calzavara et al., 2025, Shahid et al., 2022 and Zhang et al., 2021) to random forest (e.g., Anisetti et al., 2023a, Aryal et al., 2023, Dunn et al., 2020, Nowroozi et al., 2024, Shahid et al., 2022, Yerlikaya and Bahtiyar, 2022, Zhang et al., 2021 and Zhang et al., 2023), from neural network (e.g., Aryal et al., 2023, Dunn et al., 2020, Shahid et al., 2022 and Zhang et al., 2023) to deep neural network (e.g., Taheri et al., 2020 and Zhang et al., 2021), to name but a few.

Poisoning defenses aim to harden/heal the poisoned dataset or, alternatively, the resulting model. They can provide *empirical* (i.e., experimentally evaluated) or *certified* (i.e., formally proven) robustness.

Dataset-hardening techniques modify the attacked dataset to reduce the impact or presence of poisoned data points. The most recent advances point to the identification of suspicious data points followed by their removal or healing. These techniques operate in the *(i) input space*, where poisoned data points are identified using, for instance, additional classifiers and outlier detectors (e.g., Frederickson et al., 2018 and Nowroozi et al., 2024); *(ii) model space*, where a model-dependent representation of the data points simplifies the identification of poisoned data points according to, for instance, a feature extractor (De Gaspari et al., 2024), their gradient (e.g., Hong et al., 2020), the confidence of their prediction (e.g., Du et al., 2020), and their representation encoded in batch normalization layers (e.g., De Gaspari et al., 2024) or in the penultimate layer of neural network models (e.g., Peri et al., 2020). Model space is preferred in high-dimensional settings suffering from the *curse of dimensionality* (i.e., distance metrics becoming useless, data becoming

sparse, sensitivity to outliers) (Blum et al., 2020, Diakonikolas et al., 2019, Kumar et al., 2020, Yang et al., 2020). Dataset hardening based on *certified* robustness provide a formal bound on the correctness of the prediction with respect to the number of poisoned data points. These techniques include *(i) smoothing*, where the label of each data point is replaced according to the label of its neighbors (Rosenfeld et al., 2020); *(ii)* *differential privacy*, where noise is added during training, such that the predictions of a differentially private model are indistinguishable from those of an undefended model, up to a certain bound (Abadi et al., 2016, Hong et al., 2020, Ma et al., 2019).

Model-hardening techniques modify the ML model to reduce its vulnerability to poisoning and increase its robustness. The most recent advances point to (certified) robustness based on an ensemble of ML models. At training time, data points are assigned to the training sets of the models in the ensemble according to *bagging* (i.e., sampling with replacement) (e.g., Jia et al., 2021), hashing with (e.g., Wang et al., 2022) and without overlapping (e.g., Levine and Feizi, 2021). At inference time, the prediction is retrieved according to majority voting. Certified robustness is achieved using a very large number of models in the ensemble (larger than 1000 in some cases).



Download: Download high-res image (198KB)

Download: Download full-size image

Fig. 1. Overview of our risk-based ensemble technique.

3. Our approach

We present our threat model (Section 3.1), an overview of our risk-based ensemble technique (Section 3.2), and a conceptual modeling of our approach using the Information System *Security Risk* (ISSRM) domain model (Matulevičius, 2017) (Section 3.3).

3.1. Threat model

Our threat model considers *semi-targeted poisoning attacks to binary classifier*. It is defined in terms of the attacker and defender models.

Attacker model consists of the following assumptions.

- Scenario: binary classification, tabular datasets, and shallow ML. Tabular datasets still represent a significant portion of ML tasks; shallow ML models are the most suited models for these datasets (Grinsztajn and Edouard Oyallon, 2022) and are frequently preferred over deep ML models due to their advantages in terms of, for instance, explainability and sustainability.

- **Objective: semi-targeted**. The attacker aims to decrease the ability of the ML model to correctly classify the *target* class (i.e., the one with label "1"). It therefore primarily focuses on decreasing the recall of the ML model. For instance, in a scenario of ML-based malware detection, the attacker aims to decrease the ability to detect malware samples.

- **Knowledge: black box**. The attacker has full access to the dataset but does not know which ML model the victim (*defender*) is using, nor optimizes the attack according to a specific defense.

- **Poisoning technique: label flipping**. The attacker poisons the training set by flipping the labels of some data points, selected according to a given strategy. The attacker has a *budget* that limits the number of data points that can be poisoned.

Defender model consists of the following assumptions.

- **Knowledge: black box**. The defender does not know if the training set has been poisoned, nor how many data points have been attacked. She also does not have access to the original (non-poisoned) dataset. The defender however assumes that the training set has been poisoned and applies the defense according to a set of possible attacks.

- **Defense: risk-based ensemble.** The defender adopts the risk-based ensemble technique proposed in this paper (Section 3.2), which applies risk management to increase ML robustness against poisoning.

Our threat model follows the general trend of cybersecurity attacks, which are mostly unsophisticated but impactful (Eurpean Union Agency for Cybersecurity, 2022). For instance, label flipping is a simple but dangerous perturbation even in untargeted threat models (e.g., Anisetti et al. (2023a)); black-box knowledge is reasonable in most poisoning scenarios, where the downstream ML model is unknown to the attacker and the defender only knows a set of possible attacks targeting ML training.

3.2. Our approach in a Nutshell

Fig. 1 shows our risk-based ensemble technique. It consists of two main phases:

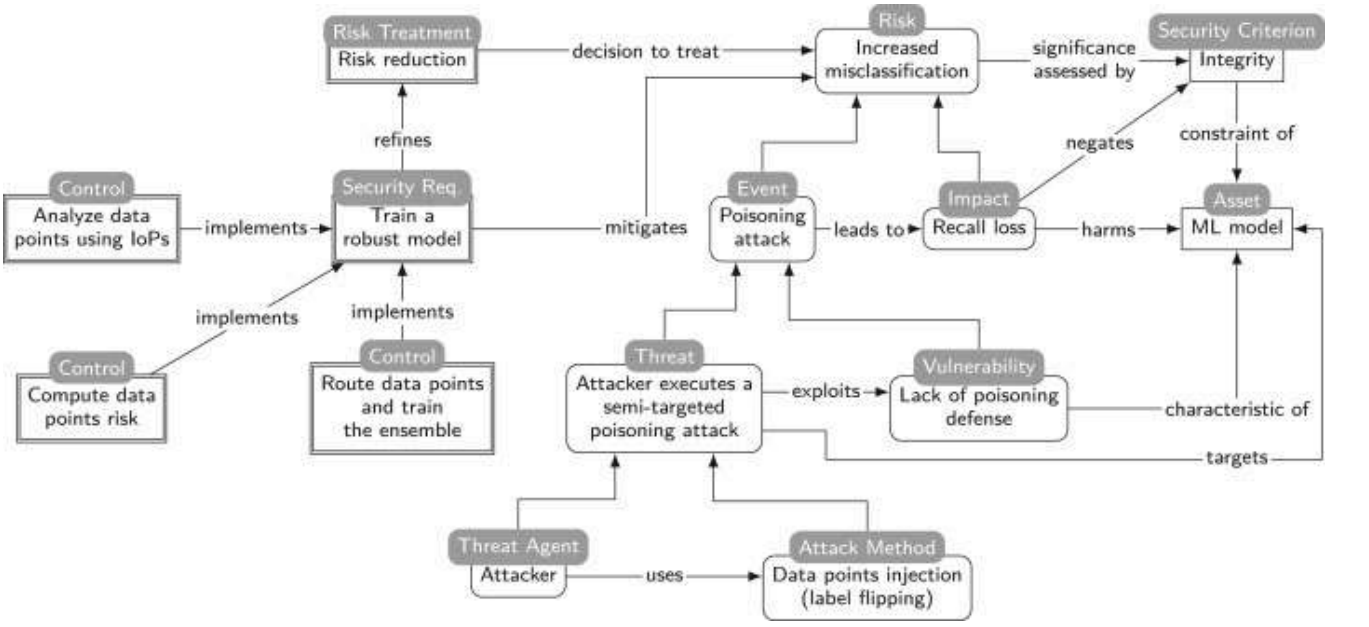
- *risk assessment* analyzes each data point in the training set according to a set of *Indicators of Poisoning – IoPs* (Step (1) in Fig. 1) and aggregates their output (Step (2) in Fig. 1) to compute the *poisoning risk* entailed by each data point (Step (3) in Fig. 1);
- *risk treatment* creates N disjoint subsets of the training set according to the identified risk (Step (4) in Fig. 1) and trains N ML models independently forming the ensemble (Step (5) in Fig. 1).

We note that our technique does not perform healing or filtering, which may have adverse effects caused by false positives. We also note that the prediction at inference time is retrieved through majority voting among all ML models in the ensemble.

3.3. ISSRM domain model

Fig. 2 presents the conceptual modeling of our approach using the *Information System Security Risk* (ISSRM) domain model (Matulevičius, 2017). The ISSRM domain model shows the relations and dependencies between all components of our approach including threats, assets, risk assessment, and risk treatment, as discussed in the following. Our approach aims to guarantee the integrity (*security criterion*) of an ML model and its predictions (*asset*), which are exposed to the *risk* of increased misclassification by poisoning attacks. Following Section 3.1, the attacker (*threat agent*) poisons the training set (*threat*) flipping the labels of a set of data points (*attack method*). The attacker's goal is to decrease the recall of the undefended (*vulnerability*) ML model at inference time (*impact*). This risk is counteracted by the ensemble technique in Fig. 1 (*risk treatment*).

In the remainder of this paper, we describe the Indicators of Poisoning (Section 4) that drive risk assessment (Section 5.1) and treatment (Section 5.2) in our risk-based ensemble technique.



Download: Download high-res image (434KB)

Download: Download full-size image

Fig. 2. Conceptual modeling of our risk-based approach according to the Information System *Security Risk* (ISSRM) domain model (Matulevičius, 2017). Straight-corners rectangles depict *asset concepts*, rounded-corners rectangles *risk assessment concepts*, while double-line rectangles *risk treatment concepts*. We report the asset "ML model" without distinguishing between *IS* and *Business* assets.

4. Indicators of poisoning

Indicators of poisoning (IoPs) contribute to the evaluation of the poisoning risk associated with each data point in a training set. Formally, an IoP takes as input a data point $p_i \in \widetilde{D}$ and the corresponding training set $\widetilde{D}$, and returns as output a value $\in \mathbb{R}^n$ modeling the IoP contribution to the poisoning risk. We consider three IoPs, namely, IoP_{bound}, IoP_{neigh}, IoP_{pos}, that analyze data points from different perspectives.

4.1. IoP boundary IoP_{bound}

IoP_{bound} evaluates each data point p_i according to its distance from the classification boundary. The lower the distance of p_i from the classification boundary, the higher the risk. IoP_{bound} approximates such distance using a helper SVM model with linear kernel. Given $p_i \in \widetilde{D}$, IoP_{bound} returns a value proportional to the (approximated) distance of p_i from the classification boundary, as follows.

$$IoP_{\text{bound}}\left(p_i, \widetilde{D}\right) = \left[abs\left(dot\left(p_i, HelperModel.coefficients^{\text{T}}\right) + HelperModel.intercept\right)\right], \quad (1)$$

where *abs* is the modulus operator, *dot* is the dot operator, $HelperModel.coefficients^{\text{T}}$ are the transposed *coefficients* of the helper model, and $HelperModel.intercept$ is the *intercept*. We note that $\widetilde{D}$ (and, consequently, p_i) is first normalized following the best practices for SVM models.

4.2. IoP neighborhood IoP_{neigh}

IoP_{neigh} evaluates each data point p_i with respect to the distribution of the neighbor data points. The higher the distance of p_i from the neighbors of the same class and the lower from the neighbors of the opposite class, the higher the risk. IoP_{neigh} implements a four-step process. It first splits $\widetilde{D}$ into disjoint subsets $\{\widetilde{D}_l\}$ containing similar data points using a HDBSCAN model. Second, given $p_i \in \widetilde{D}_l$, it retrieves the k-nearest data points of p_i in $\widetilde{D}_l$ using a helper k-Nearest Neighbor model. Third, it splits the k data points into two groups according to their label in $\widetilde{D}_l$. Finally, it returns as output the average distance of p_i from the data points of the two groups, as follows.

$$IoP_{\text{neigh}}\left(p_i, \widetilde{D}_l\right) = \left[avg\left(d\left(p_i, p_j\right) \quad \forall p_j \in Neigh\left(p_i, \widetilde{D}_l, k\right) \quad \text{s.t.} \quad label\left(p_j\right) = label\left(p_i\right)\right)\right], \quad (2)$$

$$avg\big(d\left(p_i, p_j\right) \quad \forall p_j \in Neigh\left(p_i, \widetilde{D}_l, k\right) \quad \text{s.t.}$$
$$label\left(p_j\right) \neq label\left(p_i\right)\big],$$

where avg is the average, d is the distance (e.g., Euclidean), $Neigh$ returns the k-nearest neighbor data points of p_i in $\widetilde{D}$, and $label$ returns the label of the given input data point.

4.3. IoP position IoP_{pos}

IoP_{pos} evaluates each data point p_i according to its position with respect to the data points belonging to the same and opposite classes. The higher the distance of p_i from its class and the lower the distance of p_i from the opposite one, the higher the risk. IoP_{pos} approximates such distances on the basis of the *class centroids* computed using a helper kMeans model. Given $p_i \in \widetilde{D}$, IoP_{pos} returns the distance of p_i from such centroids, as follows.

$$IoP_{\text{pos}}\left(p_i, \widetilde{D}\right) = \big[d\left(p_i, HelperModel.centroid\left(label\left(p_i\right)\right)\right), \quad (3)$$
$$d\left(p_i, HelperModel.centroid\left(\neg\, label\left(p_i\right)\right)\right)\big]$$

where $HelperModel.centroid$ returns the centroid of the class corresponding to the given input label $label\left(p_i\right)$ and the opposite label $\neg\, label\left(p_i\right)$.

5. Risk management process

We present the risk management process built on risk assessment (Section 5.1) and treatment (Section 5.2) at the basis of our ensemble technique.

5.1. Risk assessment

Risk assessment evaluates the *poisoning risk* carried by each data point in the training set. It can be defined as follows.

Definition 1

Risk assessment takes as input the training set $\widetilde{D} = \{p_1, \ldots, p_n\}$ and returns as output a set of pairs $\{(p_1, r_1), \ldots, (p_n, r_n)\}$, where

- p_i is a data point $\in \widetilde{D}$, and
- $r_i \in \mathbb{N}$ is the corresponding *risk value*.

Risk assessment is implemented as a three-step process (Steps (1)–(3) in Fig. 1).

Step (1) analyzes each p_i on the basis of the IoPs in Section 4. Each IoP is applied on $p_i \in \widetilde{D}$ and the corresponding training set $\widetilde{D}$, normalizing the retrieved output in $[0, 1]$.[6] The IoP output is *discretized* according to the different levels of risks using an arbitrary discretization function. With no lack of generality, in our approach, we use a binary discretization with 1 meaning *risky* point; 0,

otherwise. In the following, we denote with $IoP(p_i)$ the execution of a specific IoP on p_i, including discretization.

Step (2) concatenates the binarized output of each IoP into a vector R_i (Step (2) in Fig. 1):

$$R_i = [IoP_{\text{bound}}(p_i), IoP_{\text{neigh}}(p_i), IoP_{\text{pos}}(p_i)]$$

Step (3) computes the risk value r_i of p_i according to two alternatives: *Count* and *Max* (Step (3) in Fig. 1).

- **Count** counts the number of IoPs whose output is higher than 1.

  $$r_i = \sum_{j=1}^{|R_i|} R_i[j]$$

  Count creates different risk *levels*, and resembles a finer-grained *logical and*.

- **Max** returns 1 if at least one IoP returned 1; 0, otherwise.

  $$r_i = \begin{cases} 1 & \exists R_i[j] = 1 \qquad j = 1, \ldots, |R_i| \\ 0 & \text{otherwise} \end{cases}$$

  Max resembles a *logical or*.

5.2. Risk treatment

Risk treatment *acts* on the computed risk to reduce its impact, by building an ensemble of N ML models (Step (5) in Fig. 1). It can be defined as follows.

Definition 2
Risk treatment takes as input the set of pairs $\{(p_1, r_1), \ldots, (p_n, r_n)\}$ computed during risk assessment and returns as output an ensemble of N models $\{m_1, \ldots, m_N\}$, where m_i is a ML model trained on disjoint training set $\{p_j, \ldots, p_k\} \subset \{p_1, \ldots, p_n\}$ identified according to the corresponding risk values $\{r_j, \ldots, r_k\}$.

Risk treatment first defines the training sets of models $\{m_i\}$ in the ensemble (*routing*).[7] We consider two routing approaches that aim to minimize the poisoning impact on the ensemble: *(i) Round-robin*, routing a small fraction of risky data points to each model; *(ii) Sink*, routing all risky data points to one model.

- **Round-robin.** Given $\widetilde{D}$, it evenly routes data points in $\widetilde{D}$ with the same risk value to each model in the ensemble, starting from the highest risk value. *Round-robin* ensures that each training set has roughly the same size and number of data points with a specific risk value.

- **Sink.** Given $\widetilde{D}$, it routes all data points with the highest risk value to the first model m_1, and routes the remaining ones according to *Round-robin*. *Sink* ensures that all training sets but the first one have roughly the same size.

Upon routing is completed, risk treatment independently trains the N models in the ensemble, according to the chosen ML algorithm (Step (5) in Fig. 1).

At inference time, each input data point p_i is routed to all models in the ensemble. The predicted label is retrieved using majority voting among the labels predicted by all models in the ensemble.

6. Evaluation process

We present the evaluation process, the target datasets, and poisoning attacks at the basis of the experimental results in Section 7.

Table 1. Datasets.

Name	N° data points (N° per class)	N° features	Sparsity (%)	N° data points (Preproc.)	Training set size (N° per class)	Test set size (N° per class)
Musk2 (M2)	6598 (5581/1017)	166	0.28	2034	1525 (760/765)	509 (257/252)
Spambase (SB)	4061 (1813/2788)	57	77.44	3626	2719 (1363/1356)	907 (450/457)

Table 2. Evaluated risk pipelines. Each pipeline is instantiated on different values of N.

Type	Pipelines
Complete	• IoP_{all} + *Count*+ *Round-robin* • IoP_{all} + *Count*+ *Sink* • IoP_{all} + *Max*+ *Round-robin* • IoP_{all} + *Max*+ *Sink*
Oracle	• *Oracle*+ *Round-robin*, *Oracle*+ *Sink*
Baseline	• hash (Anisetti et al., 2023a)

6.1. Evaluation process in a Nutshell

An instance of our evaluation process takes as input *(i)* a dataset split in training and test sets using a ratio 0.75–0.25 and used across all evaluation processes; *(ii)* a poisoning attack and a set of poisoning budgets ϵ; *(iii)* an ML algorithm; *(iv)* an instance of our risk management process in Section 5; *(v)* the number of *repetitions*. The process consists of 4 steps as follows.

- **Step (1): Poisoning.** It applies the poisoning attack on the training set, creating one poisoned training set $\widetilde{D}$ for each value of ϵ.
- **Step (2): Training and evaluation of the monolithic models.** It trains and evaluates two types of monolithic models: *vanilla* and *filtered*. *Vanilla monolithic models* are trained on both non-poisoned and poisoned training sets according to the given ML algorithm. *Filtered monolithic models* are trained on each poisoned training set at Step (1) with poisoned data points removed. Step (2) then computes monolithic models' quality metrics (i.e., accuracy, precision, and recall) using the test set. It repeats training and testing for the given number of repetitions, and averages the results.
- **Step (3): Training of the risk-based ensemble.** It applies the risk management process in Section 5 on each generated training set and computes quality metrics (i.e., accuracy, precision, and recall) using the test set, averaging the results over the given number of repetitions.
- **Step (4): Evaluation.** It evaluates the robustness of a target model $\boldsymbol{ref}$ by comparing its quality metrics $QM(\boldsymbol{ref})$ (i.e., accuracy, precision, and recall) against the corresponding quality metrics $QM(base)$ of a baseline model $\boldsymbol{base}$ trained on the same training set, as follows.

$$\Delta(\boldsymbol{ref}, \boldsymbol{base}) = QM(\boldsymbol{ref}) - QM(\boldsymbol{base}) \quad (4)$$

A positive value of $\Delta(\boldsymbol{ref}, \boldsymbol{base})$ means that $\boldsymbol{ref}$ is better than $\boldsymbol{base}$ according to quality metric QM.

Our evaluation process instantiates different configurations of the risk management process (Section 7.1) for each pair of dataset and poisoning attack. We thus repeat Steps (3)–(4) for each configuration.

6.2. Target datasets

We used two well-known tabular datasets in literature (Table 1). We applied pre-processing and retrieved two balanced datasets (*N° data points (Preproc.)*) that substantially differ in terms of number of data points (*N° data points (N° per class)*), number of features (*N° features*), and sparsity (*Sparsity (%)*). The choice of diverse datasets allowed us to evaluate our approach in two opposite configurations. A more extensive evaluation on additional datasets are left for our future work.

Musk2 (M2) is a dataset for the identification of musk molecules (Dua and Graff, 2017). It consists of two classes: *musk* and *non-musk*. The dataset includes all the low-energy conformations (shapes) of molecules, which have been generated and manually annotated starting from 141 musk and non-musk molecules. The initial dataset consists of 6598 data points: 1017 labeled *musk* (i.e., class "1") and 5581 labeled *non-musk* (i.e., class "0"). We subsampled data points labeled *non-musk* and retrieved a balanced dataset of 2034 data points.

Spambase (SB) is a dataset for the identification of spam emails (Dimitrakakis and Bengio, 2004, Hush et al., 2007, Wang and Witten, 2002). The dataset includes features that model the content and orthography of emails, such as the occurrence of particular words and the number of consecutive capital letters. The initial dataset consists of 4061 data points: 2788 labeled *spam* (i.e., class "1") and 1813 labeled *non-spam* (i.e., class "0"). We subsampled data points labeled *spam* and retrieved a balanced dataset of 3626 data points.

6.3. Poisoning attacks

We implemented two semi-targeted label flipping poisoning attacks, *Boundary* and *Clustering*, adapting attacks (Peri et al., 2020, Taheri et al., 2020) in literature to our threat model in Section 3.1. Both attacks flip the label of a given number of data points labeled "1" (bounded by ϵ), which are selected as follows.

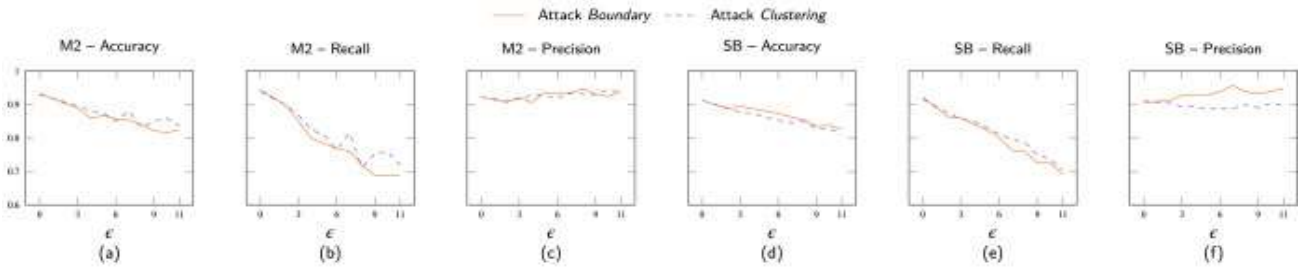
- **Boundary** selects data points closer to the classification boundary. The attack trains an SVM model with linear kernel on the (scaled) training set. It then sorts data points labeled "1" proportionally to their distance from the classification boundary, and poisons the first ϵ data points.
- **Clustering** selects data points closer to the opposite class. The attack trains a model using kMeans with $k = 2$ (binary classification) on the training set. It then sorts data points labeled "1" proportionally to their distance from the centroid representing the opposite class. It finally attacks the first ϵ data points.

7. Experimental results

We present the results of our experimental evaluation following the evaluation process in Section 6 and the experimental settings in Section 7.1. We first validate our threat model (Section 7.2). We then evaluate the soundness of our risk-based ensemble technique against poisoning attacks (Section 7.3) and its robustness in comparison with the state of the art (Section 7.4). We finally measure the performance (execution time) of our technique (Section 7.5).

7.1. Experimental settings

We evaluated several instances of our risk management process in Section 5 varying: *(i)* the poisoning budget ϵ (%) in $\{2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$; *(ii)* the poisoning attack in *Boundary* and *Clustering*; *(iii)* the dataset in M2 and SB; *(iv)* the number N of models in the ensemble in $\{3, 9, 15, 21, 27, 33\}$.



Download: Download high-res image (147KB)

Download: Download full-size image

Fig. 3. Impact of poisoning attacks against the vanilla monolithic model, using datasets M2 and SB and varying the percentage of poisoned data points ϵ.

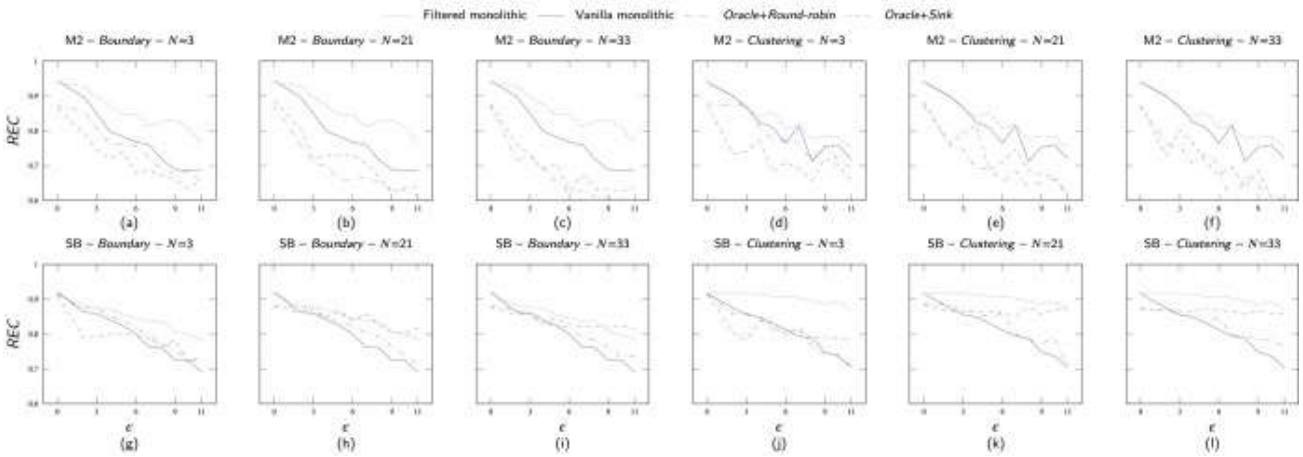
Each process instance is called *risk pipeline* and is defined in terms of *(i)* IoPs, *(ii)* risk value computation (*Count* or *Max*), *(iii)* routing approach (*Round-robin* or *Sink*), and *(iv)* number N of models in the ensemble. For instance, IoP_{all} + *Count* + *Round-robin*(N =3) denotes a pipeline including all IoPs (IoP_{bound}, IoP_{neigh}, IoP_{pos}) aggregated using *Count*, and routing data points to an ensemble of $N = 3$ models according to routing approach *Round-robin*.

Our baseline is the hash-based ensemble technique in Anisetti et al. (2023a). It builds an ensemble of N models, each model trained on a disjoint subset of the training set forming N disjoint training sets. To this aim, it computes the hash of each data point according to *md5*, applies the modulo operator (modulo N), and evenly routes data points according to their modulo to the N disjoint training sets. At inference time, data points are routed to each model in the ensemble and the final prediction is retrieved using majority voting. This approach is highly effective against untargeted (Anisetti et al., 2023a) and targeted poisoning attacks (albeit with small variations (Jia et al., 2021, Levine and Feizi, 2021)). This technique, referred to as *hash pipeline*, is the most similar to our approach in terms of threat model and ensemble technique (see the comparison in Section 9). They both rely on an ensemble of models trained on disjoint training sets, differing in how training sets are built.

Table 2 shows the pipelines used in our experiments: *(i) risk* pipelines including the IoPs in Section 4; *(ii) risk oracle* pipelines, where risk assessment assigns 1 to r_i if p_i is poisoned, 0 otherwise; *(iii) hash* pipelines implementing the ensemble technique in Anisetti et al. (2023a).

In total, we executed 5368 different experiments, each of which repeated 5 times. The execution platform is an Apple MacBook Pro with 10 CPUs Apple M1 Pro, 32 GBs of RAM, operating system macOS Sequoia 15, using Python 3.10.12 with libraries *scikit-learn* v1.3.1 (Pedregosa et al., 2011),

numpy v1.24.4 (Harris et al., 2020), *pandas* v2.1.1 (Wes McKinney, 2010, The pandas development team, 2020), *plotly* v5.24.0,[8] and *xarray* v2023.9.0 (Hoyer and Hamman, 2017).



Download: Download high-res image (340KB)

Download: Download full-size image

Fig. 4. Impact of poisoning attacks against vanilla and filtered monolithic models and risk oracle pipelines, varying datasets M2 and SB and percentage of poisoned data points ϵ.

7.2. Threat model validity

We validated our threat model in Section 3.1 by measuring the impact of the poisoning attacks (*Boundary* and *Clustering*) and datasets (M2 and SB) on the quality metrics accuracy, precision, and recall of the vanilla monolithic model varying the poisoning budget ϵ.

Fig. 3 shows our results. We can first observe that the monolithic model achieves similar quality on the non-poisoned datasets: accuracy = 0.931, precision = 0.922, and recall = 0.941 on dataset M2; accuracy = 0.912, precision = 0.908, and recall = 0.918 on dataset SB. We can then observe that both attacks have the highest impact on recall (*REC*), which decreases to 0.785 on average across all attacks and datasets. The impact on accuracy is lower (0.859), while we observed a positive impact on precision (0.921). In particular, attack *Boundary* causes a 16.9% worsening on *REC* on average across datasets compared to 14.1% of attack *Clustering*. Comparing the two datasets, *REC* decreases of 17.3% on average with M2, compared to 13.7% of SB. Considering ϵ, we can observe that it strongly affects *REC*, which shows the steepest trend among quality metrics, varying between 0.883 ($\epsilon = 2$) and 0.7 ($\epsilon = 11$) on average across datasets and attacks.

The above results support our assumptions in the threat model in Section 3.1 and, therefore, our experimental evaluation entirely focused on recall *REC*.

7.3. Soundness

We evaluated the soundness of our approach, measuring whether the risk management process in Section 5 is effective to increase the robustness of ML models. We thus compared the recall *REC* retrieved by *(i)* risk oracle pipelines (*Oracle + Sink* and *Oracle + Round-robin*), that is, pipelines where poisoned data points are known to the defender. The defender optimally routes data points to the training sets used to train the models in the ensemble according to routing strategies *Sink* and *Round-robin*, respectively; *(ii)* filtered monolithic model, that is, an ensemble with $N = 1$, where poisoned data points are known to the defender and filtered out from the training set; and *(iii)* vanilla monolithic model.

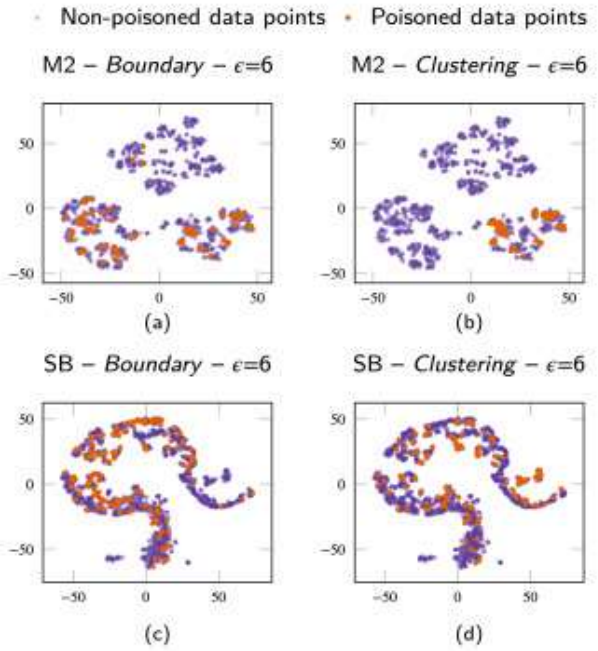
We consider the vanilla monolithic as our bottom line, the oracle pipelines as our real optimum, and the filtered monolithic as our theoretical optimum.

Fig. 4 shows our results, which are dataset-depending. Focusing on SB, on average, the filtered monolithic model is the best one ($REC = 0.874$), followed by *Oracle + Sink* ($REC = 0.845$) and *Oracle + Round-robin* ($REC = 0.816$), each of which is better than the vanilla monolithic model ($REC = 0.804$). In addition, the gap between the filtered monolithic model and *Oracle + Sink* becomes smaller when $N > 3$ ($REC = 0.854$ on average).

Focusing on dataset M2, we can observe that risk oracle pipelines are always worse than the monolithic models, meaning that our risk-based routing is ineffective, even when the poisoned data points are known to the defender. On average, the filtered monolithic model exhibits the highest recall ($REC = 0.841$), followed by the vanilla monolithic model ($REC = 0.793$), *Oracle + Sink* ($REC = 0.722$), and *Oracle + Round-robin* ($REC = 0.722$).

The reason of these results lies in the structure and cardinality of the datasets. To demonstrate this observation, we inspected the datasets using *T-distributed Stochastic Neighbor Embedding* (t-SNE) to reduce them to a two-dimensional space. Fig. 5 shows that M2 can be roughly divided in three clusters, with attack *Clustering* targeting one cluster and attack *Boundary* targeting two clusters only (see poisoned data points in orange in Fig. 5). Filtering or routing these data points, using the filtered monolithic model or a risk pipeline, respectively, creates a strongly imbalanced training set, where an entire region of the space can be removed or substantially reduced in cardinality across the models in the ensemble. On the contrary, Fig. 5 shows that poisoned data points in SB are fairly distributed under both attacks, meaning that their removal or routing does not create imbalance. In addition, we note that the ensemble experiences a higher loss with M2 due to its small cardinality.

To conclude, M2 is intrinsically weak, meaning that poisoning has a larger impact as discussed in Section 7.2, while decreasing the utility of our risk-based ensemble technique. This finding is also confirmed in literature on simpler attacks (Anisetti et al., 2023a). SB is instead more robust and our risk-based ensemble techniques can contribute to increase the overall ML robustness.



Download: Download high-res image (326KB)

Download: Download full-size image

Fig. 5. Dimension-reduced datasets M2 and SB. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

According to the above results, in the remainder of this section, we focus on SB, using vanilla monolithic model as our bottom line and oracle pipelines as our optimum.

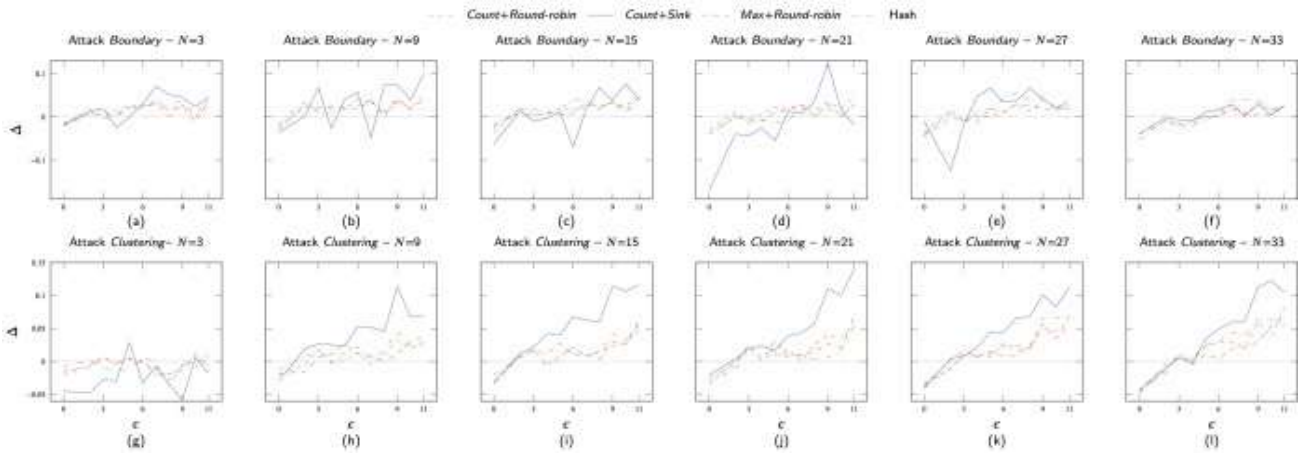
7.4. Robustness

We evaluated the robustness of our approach, comparing our risk-based and state-of-the-art hash-based ensemble techniques (target models $\boldsymbol{ref}$) against the vanilla monolithic model (baseline model $\boldsymbol{base}$) using metric $\boldsymbol{\Delta}$ in Eq. (4).

Table 3. Configurations of the IoPs and risk assessment, following an empirical analysis.

IoP	Hyperp.	*Boundary*		*Clustering*	
		Norm.	**Bin.**	**Norm.**	**Bin.**
IoP_{bound}	–	*max*	**output** $\geq$ **0.81**	*max*	**output** $\geq$ **0.82**

IoP	Hyperp.	*Boundary*		*Clustering*	
		Norm.	Bin.	Norm.	Bin.
IoP_{neigh}	$k = 50$	*div*	$(\text{output}[0] \leq 0.32 \wedge \text{output}[1] \leq 0.11) \vee (0.047 \leq \text{output}[0] \leq 0.17 \wedge \text{output}[1] \leq 0.32)$	*div*	$(0.18 \leq \text{output}[0] \leq 0.47 \wedge \text{output}[1] \leq 0.01) \vee (\text{output}[0] \leq\approx 0.064 \wedge \text{output}[1] = 0) \vee (\text{output}[1] = 0 \wedge \text{output} = 0)$
IoP_{pos}	–	*max*	$\text{output}[0] \leq 0.175 \wedge \text{output}[1] \geq 0.82$	*div*	$\text{output} \geq 0.18 \wedge \text{output} \leq 0.40$



Download: Download high-res image (377KB)

Download: Download full-size image

Fig. 6. Δ of risk pipelines varying percentage of poisoned data points ϵ.

Table 3 shows our configurations and Fig. 6 our results. We note that values above the horizontal dotted line at $\Delta = 0$ model scenarios where risk-based or hash-based ensemble model is better than the monolithic model. We note that Fig. 6 excludes risk pipelines based on *Max* + *Sink* because the corresponding Δ is $\approx 30\times$ worse than other risk pipelines ($\Delta = -0.468$ compared to $\Delta = 0.015$ on average).

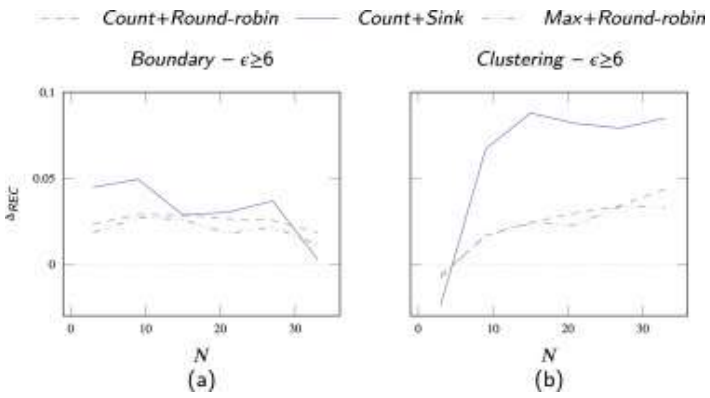
On average across all attacks and N, we can first observe that most pipelines exhibit $\Delta > 0$, meaning that they are more robust than the vanilla monolithic model. Pipelines based on *Count* +

Sink show highest robustness with $\Delta = 0.02$, compared to $\Delta = 0.014$ of hash, $\Delta = 0.013$ of *Count* + *Round-robin*, and $\Delta = 0.011$ of *Max* + *Round-robin*. Pipelines based on *Count* + *Sink* captures 48% of the optimum (*Oracle* + *Sink*), compared to 33% of hash, 31% of *Count* + *Round-robin*, and 26% of *Max* + *Round-robin*.

We can then observe that Δ increases as ϵ increases, meaning that our ensemble technique becomes more robust as the amount of poisoned data points increases. On average across datasets and attacks, risk pipelines exhibit $\Delta = -0.005$ when $\epsilon \leq 5$, and $\Delta = 0.031$ when $\epsilon \geq 6$. For instance, pipeline *Count* + *Sink* captures 59% of the global optimum when $\epsilon \geq 6$, 8% otherwise. The robustness of vanilla monolithic model is then strongly affected by an increasing ϵ, as opposed to our ensemble technique.

Focusing on attack *Boundary* and averaging over N, we can observe that *Count* + *Sink* improves as ϵ increases, following the general trend. With $\epsilon \geq 6$ and on average across N, *Count* + *Sink* is the best approach capturing 46% of the optimum. Focusing on attack *Clustering* and averaging over N, we can observe that *Count* + *Sink* is the best model regardless of ϵ, reaching 75% of the optimum. The remaining pipelines are similar, with hash reaching the 27% and *Max* + *Round-robin* the 23% of the optimum.

Fig. 7 shows the behavior of Δ against attacks *Boundary* and *Clustering* averaged over N and $\epsilon \geq 6$. We can first observe that the best pipeline *Count* + *Sink* exhibits a growth pattern from $N = 3$ to $N = 9$ followed by further growth and stabilization (*Clustering*) or slight decrease, stabilization, and decrease (*Boundary* – the most impactful attack). The initial growth is larger (**379%**) under attack *Clustering* than under attack *Boundary* (**10%**). We can then observe that pipelines *Count* + *Round-robin* and *Max* + *Round-robin* follow a similar pattern under the two attacks. Their behavior is decreasing (**−47%**) under the most impactful attack *Boundary* and increasing (**129%**) under *Clustering* from $N = 9$ to $N = 33$.



Download: Download high-res image (123KB)

Download: Download full-size image

Fig. 7. Average Δ on dataset SB under attacks *Boundary* and *Clustering* with $\epsilon \geq 6$, varying N.

We can finally conclude that our risk-based ensemble technique exhibits a higher *REC* than the vanilla monolithic model and hash-based ensemble technique on the poisoned datasets, meaning that our technique increases the robustness compared to an undefended model and the state of the art across attacks.

7.5. Performance

We measured the performance of our ensemble technique in terms of the time needed to execute the entire risk pipeline, from risk assessment to risk treatment. We also measured the total execution time of training a monolithic model as our baseline.

Table 4 summarizes our settings. We considered a worst cases scenario where SB (the largest dataset) was poisoned using attack *Boundary* (the most impactful attack) and $\epsilon = 6$ (the rounded down, median of the considered values of ϵ). Then, we generated several training sets using random oversampling on poisoned SB. We then measured the execution time of the monolithic model and of the given risk pipelines following configurations in Table 3. We repeated each execution until results converged.[9] We evaluated risk pipelines including *(i)* all IoPs and *(ii)* one IoP at time.

Fig. 8 shows an excerpt of our results, focusing on the worst performance results retrieved by *Sink*. We can observe that execution time linearly increases as the dataset cardinality $|\widetilde{D}|$ increases; the impact of the number N of models is negligible (6.109 s on average on IoP_{all} + *Count* + *Sink* with $N = 3$, 6.195 s with $N = 19$, 6.212 s with $N = 33$) thanks to our implementation supporting vectorized and parallelized executions. IoP_{neigh} has the largest impact on performance, because it additionally performs clustering (a performance of 6.056 s on average when used as the only IoP, 6.172 s when used in conjunction with IoP_{bound} and IoP_{pos}); IoP_{bound} and IoP_{pos} show negligible execution times when used alone (0.213 s on average), but higher than the monolithic model (0.087 s on average).

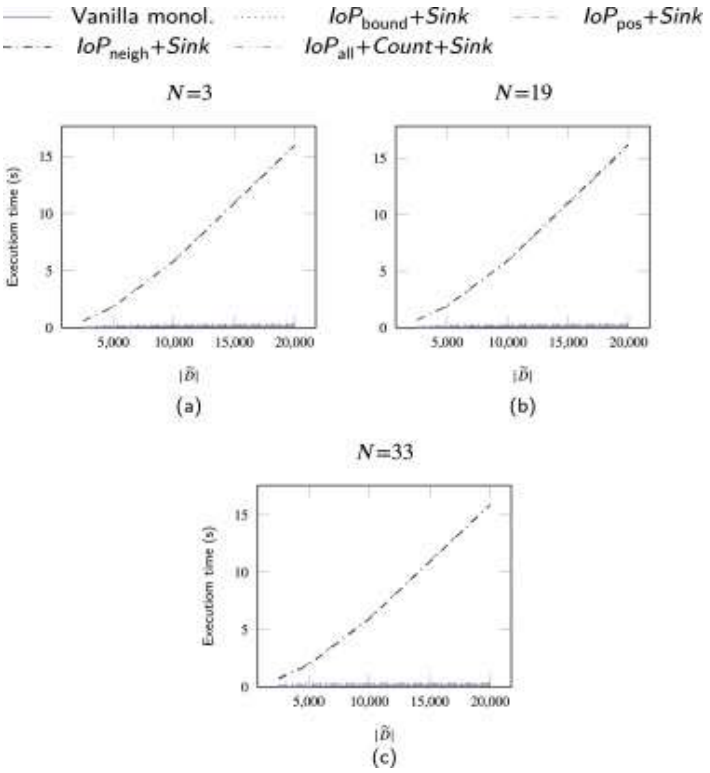
8. Discussion

Our research question sought to understand whether an ensemble technique built on a risk-based routing can improve the robustness of ML models under the threat model in Section 3.1. We evaluated our risk-based routing varying datasets, attacks, percentage of poisoned data points, and risk pipelines. Four findings emerge as follows.

Table 4. Settings for performance evaluation.

Parameter	Values	
Pipelines	• IoP_{bound} + *Round-robin*	• IoP_{bound} + *Sink*
	• IoP_{neigh} + *Round-robin*	• IoP_{neigh} + *Sink*
	• IoP_{pos} + *Round-robin*	• IoP_{pos} + *Sink*

Parameter	Values
	• IoP_{all} + *Count*+ *Round-robin* • IoP_{all} + *Max*+ *Round-robin* • Vanilla monolithic model • IoP_{all} + *Count*+ *Sink* • IoP_{all} + *Max*+ *Sink*
N	3, 19, 33
$\lvert\widetilde{D}\rvert$	2,500, 5,000, 10,000, 20,000
Base model	Decision tree



Download: Download high-res image (186KB)

Download: Download full-size image

Fig. 8. Performance of our technique varying dataset cardinality $\lvert\widetilde{D}\rvert$ (oversampled from dataset SB under *Boundary* with $\epsilon = 6$).

F1

The effectiveness of an ensemble model strongly depends on the dataset. Finding F1 can be observed in Fig. 4, where the effectiveness of our ensemble technique differs according to the dataset. Fig. 4(a)–(f) shows that the oracle pipelines (our optimum) are always worse than the monolithic models when applied on dataset M2; Fig. 4(g)–(l), instead, shows that pipeline *Oracle* + *Sink* approaches the filtered monolithic model (our theoretical optimum) when applied on dataset SB: $REC = 0.845$ and $REC = 0.874$, resp., on average. A similar

behavior also emerges with the filtered monolithic model. On the non-poisoned datasets, recall *REC* is 0.94 on M2 and 0.914 on SB. On the poisoned datasets, the filtered monolithic model *worsens faster* on dataset M2 ($REC = 0.841$) than on SB ($REC = 0.874$). As discussed in Section 7.3 and Fig. 5, the reason lies in the structure and cardinality of the datasets. For instance, dataset M2 is inherently weak due to its small cardinality and localized distribution of poisoned data points in specific regions of the space. In this case, the poisoning impact cannot be counteracted even when the poisoned data points are known to the defender.

F2

Our risk-based ensemble technique provides higher robustness than state-of-the-art ensemble techniques. Finding F2 can be observed in Fig. 6, where the recall of pipeline *Count* + *Sink* outperforms the other pipelines, and in particular the state of the art (Anisetti et al., 2023a, Levine and Feizi, 2021) represented by the hash pipeline. On average with $\epsilon \geq 6$, *Count* + *Sink* captures 58.5% of the optimum *Oracle* + *Sink*, compared to only 34% of the hash pipeline. Our results show that a risk-based routing of data points has a positive impact on the robustness of ML against poisoning attacks.

F3

The number N of models in the ensemble impacts the result. Finding F3 can be observed in Fig. 7. All pipelines show a similar behavior across attacks. First, a growing trend is observed as N increases until the optimum is reached; then, a decreasing trend is observed due to the decreased cardinality of the training sets used to train the models in the ensemble that negatively impacts on the overall recall. This behavior is stronger in *Count* + *Sink*, where the first model in the ensemble receives all risky data points further impacting on the cardinality of the other models. In particular, *Count* + *Sink* reaches the optimum at $N = 9$ under *Boundary* ($\Delta = 0.031$) and $N = 15$ under *Clustering* ($\Delta = 0.055$). These results show that ML robustness does not require large values of N.

F4

Routing all risky data points to one model provides the highest robustness. Finding F4 can be first observed in Fig. 4(g)–(l) (for dataset SB), where *Oracle* + *Sink* is the best ensemble technique overall ($REC = 0.845$ on average compared to $REC = 0.713$ of *Oracle* + *Round-robin*). F4 is further supported by our results in Fig. 6, Fig. 7, where *Count* + *Sink* exhibits the highest robustness: $\Delta = 0.02$ ($\Delta = 0.047$ when $\epsilon \geq 6$). Following F3, our results show that routing all risky data points to the same model increases the robustness more than evenly spreading, but may create an unbalanced ensemble if N is too high.

9. Comparison with related work

Recalling Sections 2 Background, 3 Our approach, our risk-based ensemble technique can be classified as a *model-hardening* defense. We therefore discuss related work in this area (Section 9.1) and compare it with our approach (Section 9.2).

Table 5. Comparison with related work.

Ref.	Attacker model				Defender model		
	Scenario	**Objective**	**Knowledge**	**Technique**	**Model applicability**	**Data point analysis**	**Description**
Jia et al. (2021)	SA, TB	T	–	Any	✓	✗	Ensemble with majority voting; routing using *bagging*; models trained independently
Levine and Feizi (2021) (A)	SA, TB	T	–	Any	✓	✗	Ensemble with majority voting; routing to disjoint subsets using hash and modulo; models trained independently
Levine and Feizi (2021) (B)	SA, TB	T	–	LF	✓	✗	Ensemble with majority voting; routing to disjoint subsets using hash and modulo; each model trained on unlabeled training set ∪ labeled subset
Chen et al. (2022)	SA, TB	UT, T	–	Any	✓	✗	Ensemble with majority voting; routing to subsets using *bagging with hash*; models trained independently
Geiping et al. (2022)	SA, TB	T	W	Any, BD	✗	✗	Modified adversarial training
Wang et al. (2022)	SA, TB	T	–	Any	✓	✗	Ensemble with majority voting; routing to non-disjoint subsets using two hash functions and modulo; models trained independently
Anisetti et al.	TB	UT	B	LF, O	✗	✗	Ensemble with majority voting; routing to disjoint

Ref.	Attacker model				Defender model		
	Scenario	Objective	Knowledge	Technique	Model applicability	Data point analysis	Description
(2023a)							subsets using hash, modulo, and round-robin; models trained independently
Bansal et al. (2023)	SA	T	–	BD	✗	✗	Custom fine-tuning in multi-modal ML, breaking malicious mappings introduced by poisoned data points
Mauri et al. (2023)	SA, TB	–	B	LF	✓	✓	Ensemble with majority voting; routing to disjoint subsets according to the (distance-based) risk maximizing diversity; models trained independently
Rezaei et al. (2023)	SA, TB	–	–	Any	✗	✗	Ensemble with custom two-stage voting trained as in Levine and Feizi (2021) (A) and Wang et al. (2022)
Xing et al. (2024)	SA	T	–	BD	✗	✗	Ensemble with majority voting; routing to disjoint subsets using *bagging*; each model trained varying architecture and using *self-ensembling*
This paper	TB	ST	B	LF	✓	✓	Ensemble with majority voting; routing to disjoint subsets using a risk management process; models trained independently

Comparison with related work in terms of *(i)* Scenario (Scenario): raw samples (SA), or tabular dataset (TB); *(ii)* Objective (Objective): untargeted (UT), semi-targeted (ST), targeted (T); *(iii)* Knowledge (Knowledge):

black- (B), or white-box (W); *(iv)* Poisoning technique (Technique): any (Any), label flipping (LF), backdoor (BD), or other (O); *(v)* Model applicability (Model applicability): generalizable (✓) or not (✗); *(vi)* Data point analysis (Data point analysis): analysis of each data point (✓) or not (✗). "–" means *unspecified*.

9.1. Related work

The line of research closest to the work in this paper focuses on ensemble techniques that modify the training process and the ML model to increase empirical ML robustness. Anisetti et al. (2023a) designed an ensemble technique that counteracts untargeted poisoning attacks perturbing features or labels, and focuses on random forest models. The proposed approach creates N disjoint subsets of the training set by evenly distributing its data points using hash and modulo. These subsets are then used to train N models in the ensemble. The prediction at inference time is retrieved using majority voting among the N models. Mauri et al. (2023) proposed an ensemble technique based on a two-stage approach. First, it computes the *risk* of each data point according to its distance from the separating hyperplanes of an SVM model. Second, it routes data points to disjoint subsets of the training set maximizing their diversity according to their risk. Xing et al. (2024) designed an ensemble technique protecting against *backdoor poisoning* attacks, where the attacker adds imperceptible noise to specific data points at both training *and* inference time to cause misclassification of perturbed inference-time data points. In particular, this defense randomly routes data points to N disjoint subsets of the training set and trains N models with different architectures using *self-ensembling*.

Another important line of research focuses on ensemble techniques providing certified robustness. Jia et al. (2021) designed an ensemble based on *bagging*. It routes data points to N subsets of the training set uniformly with replacement, trains one model for each subset, and retrieves the prediction at inference time using majority voting. The defense counteracts a generalized attacker model altering the training set with addition, deletion, and modification. The robustness of this defense is certified, bounding the correctness of an inference-time prediction with the maximum number of poisoned data points. Levine and Feizi (2021) proposed two similar techniques also providing certified robustness. The first technique protects against *any* poisoning techniques (denoted as "(A)" in Table 5). It routes data points to the N disjoint subsets of the training set according to their hash modulo N, and then proceeds as in Jia et al. (2021). The second technique protects against label flipping (denoted as "(B)" in Table 5). It starts with hash-based routing and then trains N semi-supervised models, each of which takes as input the entire unlabeled training set and the corresponding labeled subset. Wang et al. (2022) improved the first technique in Levine and Feizi (2021) with a different routing algorithm. It splits the training set into several subsets smaller than in Levine and Feizi (2021) using hash and modulo, and then uses a second hash function and modulo to assign each subset to different buckets. The resulting N buckets are used to train N models independently, and the inference-time prediction is retrieved using majority voting. Rezaei et al. (2023) defined a voting technique for ensemble-based defenses. First, it records the predicted label of each model. Second, it records each model confidence probability (*logits*) of the

top-two labels of the first round: the label with the highest number of votes is the predicted one. This voting is applied to the first technique in Levine and Feizi (2021) and Wang et al. (2022), improving robustness. Chen et al. (2022) focused on a different definition of certified robustness, called *collective robustness*, which certifies the prediction correctness of a model *overall*, opposed to the correctness of individual predictions. The definition is applied to a modified version of bagging-based ensemble, replacing random routing with hash and modulo.

Finally, another line of research focuses on techniques that adapt the training process of a monolithic model to increase ML robustness. Bansal et al. (2023) focused on backdoor attacks in multi-modal (image and text) ML. The proposed approach fine-tunes a pre-trained, backdoored model forcing it to learn independent representations of each modality, breaking the malicious mappings introduced by poisoned data points. Geiping et al. (2022) proposed a different approach to model hardening that adapts *adversarial training*, a technique originally defined to counteract inference-time attacks. Specifically, it injects correctly labeled poisoned data points in the training set. This technique protects neural network-based models against targeted poisoning attacks including backdoor.

9.2. Comparison

Table 5 compares our approach and existing model-hardening defenses according to different aspects. Column *Attacker Model* specifies the attacker model in Section 3.1 in terms of the scenario, objective, knowledge, and poisoning technique. *Scenario* evaluates the type of dataset: *(i)* tabular datasets (TB); *(ii)* raw samples (SA). *Objective* evaluates the objective of the attacker: *(i)* untargeted (UT); *(ii)* semi-targeted (ST); *(iii)* targeted (T). *Knowledge* evaluates the knowledge of the attacker: *(i)* black-box (B); *(ii)* white-box (W). *Poisoning technique* evaluates how data points are crafted: *(i)* any perturbation (Any); *(ii)* label flipping (LF); *(iii)* specific backdoor attacks (BD); *(iv)* other attacks (O). Column *Defender Model* specifies the defender model in terms of the applicability to different ML algorithms (Model applicability) and analysis of data point peculiarities (An.). *Model applicability* specifies whether the defense is applicable to any ML algorithm (✓) or not (✗). *Data point analysis* specifies whether the defense analyzes the peculiarities of the data points (✓) or not (✗). *Description* describes the defense. Finally, "–" denotes an unspecified value.

According to Table 5, we can first observe that our approach is the only model-hardening defense targeting semi-targeted poisoning attacks. Moreover, our approach is one of the few defenses with a carefully defined threat model, which clarifies the boundaries of the analysis. Finally, our approach and the approach in Mauri et al. (2023) are the sole implementing a risk-based technique. However, differently from the work in Mauri et al. (2023), our approach *(i)* analyzes each data point from multiple angles using IoPs, *(ii)* implements two opposite routing functions, and *(iii)* is generalizable over each step of the risk management process, from the specific IoPs to their aggregation and routing.

10. Conclusions

The ever-increasing spread of machine learning has raised several concerns about the security and trustworthiness of ML models. In this paper, we proposed a risk-based ensemble technique that increases the robustness of ML models against poisoning attacks targeting their recall. Our technique performs a *smart routing* of data points during the training of the models in the ensemble following a risk management process. Throughout fine-grained experiments, we demonstrated that our technique increases the robustness with respect to *(i)* an undefended, vanilla monolithic model and *(ii)* state-of-the-art ensemble techniques based on hash functions, provided that the dataset is sufficiently robust. In conclusion, we observed a strong dependence of the technique on the structure of the dataset to be protected and the need of tuning the parameters of our approach to the specific scenario, aspects that will be further analyzed in our future work.

CRediT authorship contribution statement

Nicola Bena: Writing – original draft, Software, Methodology, Investigation, Formal analysis, Data curation. **Marco Anisetti:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Ernesto Damiani:** Supervision, Methodology, Conceptualization. **Chan Yeob Yeun:** Supervision, Methodology, Conceptualization. **Claudio A. Ardagna:** Writing – review & editing, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: All authors reports financial support was provided by Technology Innovation Institute. Marco Anisetti, Claudio Ardagna, Nicola Bena, Ernesto Damiani reports financial support was provided by European Commission. If there are other authors, they 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

The data are available for download at https://doi.org/10.13130/RD_UNIMI/JMKSAW ↗.

References

Abadi et al., 2016 Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L., 2016. Deep Learning with Differential Privacy. In: Proc. of ACM CCS 2016. Vienna, Austria.

Google Scholar ↗

Anisetti et al., 2023a M. Anisetti, C.A. Ardagna, A. Balestrucci, N. Bena, E. Damiani, C.Y. Yeun
On the robustness of random forest against untargeted data poisoning: An ensemble-based approach
IEEE Trans. Sustain. Comput., 8 (4) (2023)
Google Scholar ↗

Anisetti et al., 2023b M. Anisetti, C.A. Ardagna, N. Bena, E. Damiani
Rethinking certification for trustworthy machine-learning-based applications
IEEE Internet Computing, 27 (6) (2023)
Google Scholar ↗

Aryal et al., 2022 Aryal, K., Gupta, M., Abdelsalam, M., 2022. Analysis of Label-Flip Poisoning Attack on Machine Learning Based Malware Detector. In: Proc. of IEEE Big Data 2022. Osaka, Japan.
Google Scholar ↗

Aryal et al., 2023 K. Aryal, M. Gupta, M. Abdelsalam
Analysis of label-flip poisoning attack on machine learning based malware detector
(2023)
ArXiv Prepprint arXiv:2301.01044 ↗
Google Scholar ↗

Atzori et al., 2024 M. Atzori, E. Calò, L. Caruccio, S. Cirillo, G. Polese, G. Solimando
Evaluating password strength based on information spread on social networks: A combined approach relying on data reconstruction and generative models
Online Soc. Netw. Media, 42 (2024), Article 100278
View PDF View article View in Scopus ↗ Google Scholar ↗

Bansal et al., 2023 Bansal, H., Singhi, N., Yang, Y., Yin, F., Grover, A., Chang, K.-W., 2023. CleanCLIP: Mitigating Data Poisoning Attacks in Multimodal Contrastive Learning. In: Proc. of IEEE/CVF ICCV 2023. Paris, France.
Google Scholar ↗

Bellandi et al., 2024 V. Bellandi, P. Ceravolo, J. Maggesi, S. Maghool
Data management for continuous learning in EHR systems
ACM Transactions on Internet Technology (2024)
Google Scholar ↗

Bena et al., 2024 N. Bena, M. Anisetti, G. Gianini, C.A. Ardagna
Certifying accuracy, privacy, and robustness of ML-based malware detection
SN Comput. Sci., 5 (2024)
Google Scholar ↗

Biggio et al., 2012 Biggio, B., Nelson, B., Laskov, P., 2012. Poisoning Attacks against Support Vector Machines. In: Proc. of ICML 2012. Edinburgh, UK.

Google Scholar ↗

Blum et al., 2020 A. Blum, T. Dick, N. Manoj, H. Zhang

Random smoothing might be unable to certify l_∞ robustness for high-dimensional images

J. Mach. Learn. Res., 21 (2020)

Google Scholar ↗

Calzavara et al., 2025 Calzavara, S., Cazzaro, L., Vettori, M., 2025. Timber! Poisoning Decision Trees. In: Proc. of IEEE SaTML 2025. Copenhagen, Denmark.

Google Scholar ↗

Caruccio et al., 2023 L. Caruccio, G. Cimino, S. Cirillo, D. Desiato, G. Polese, G. Tortora

Malicious account identification in social network platforms

ACM Transactions on Internet Technology, 23 (4) (2023)

Google Scholar ↗

Chen et al., 2022 Chen, R., Li, Z., Li, J., Yan, J., Wu, C., 2022. On Collective Robustness of Bagging Against Data Poisoning. In: Proc. of ICML 2022. Baltimore, MD, USA.

Google Scholar ↗

Chen et al., 2017 X. Chen, C. Liu, B. Li, K. Lu, D. Song

Targeted backdoor attacks on deep learning systems using data poisoning

(2017)

arXiv preprint arXiv:1712.05526 ↗

Google Scholar ↗

Chen et al., 2019 Chen, S., Xue, M., Fan, L., Ma, L., Liu, Y., Xu, L., 2019. How Can We Craft Large-Scale Android Malware? An Automated Poisoning Attack. In: Proc. of IEEE AI4Mobile 2019. Hangzhou, China.

Google Scholar ↗

Cinà et al., 2023 A.E. Cinà, K. Grosse, A. Demontis, S. Vascon, W. Zellinger, B.A. Moser, A. Oprea, B. Biggio, M. Pelillo, F. Roli

Wild patterns reloaded: A survey of machine learning security against training data poisoning

ACM CSUR, 55 (13s) (2023)

Google Scholar ↗

De Gaspari et al., 2024 De Gaspari, F., Hitaj, D., Mancini, L.V., 2024. Have You Poisoned My Data? Defending Neural Networks Against Data Poisoning. In: Proc. of ESORICS 2024. Bydgoszcz,

Poland.

Google Scholar ↗

D'Errico et al., 2024 D'Errico, L., di Nardo, E., Ciaramella, A., Staffa, M., 2024. A 3D-CNNs Approach to Classify Users' Emotion through EEG-based Topographical Maps in HRI. In: Proc. of ACM/IEEE HRI 2024. Boulder, CO, USA.

Google Scholar ↗

Diakonikolas et al., 2019 Diakonikolas, I., Kamath, G., Kane, D., Li, J., Steinhardt, J., Stewart, A., 2019. Sever: A Robust Meta-Algorithm for Stochastic Optimization. In: Proc. of ICML 2019. Long Beach, CA, USA.

Google Scholar ↗

Dimitrakakis and Bengio, 2004 Dimitrakakis, C., Bengio, S., 2004. Online policy adaptation for ensemble classifiers. In: Proc. of ESANN 2004. Bruges, Belgium.

Google Scholar ↗

Du et al., 2020 Du, M., Jia, R., Song, D., 2020. Robust anomaly detection and backdoor attack detection via differential privacy. In: Proc. of ICLR 2020. Addis Ababa, Ethiopia, Poster – VIRTUAL.

Google Scholar ↗

Dua and Graff, 2017 D. Dua, C. Graff

UCI machine learning repository

(2017)

URL: http://archive.ics.uci.edu/ml ↗

Google Scholar ↗

Dunn et al., 2020 C. Dunn, N. Moustafa, B. Turnbull

Robustness evaluations of sustainable machine learning models against data poisoning attacks in the internet of things

Sustain., 12 (16) (2020)

Google Scholar ↗

Eurpean Union Agency for Cybersecurity, 2022 Eurpean Union Agency for Cybersecurity

ENISA Threat Landscape 2022: Technical Report

European Union Agency for Cybersecurity (2022)

Google Scholar ↗

Frederickson et al., 2018 Frederickson, C., Moore, M., Dawson, G., Polikar, R., 2018. Attack Strength vs. Detectability Dilemma in Adversarial Machine Learning. In: Proc. of IJCNN 2018. Rio de Janeiro, Brazil.

Google Scholar ↗

Geiping et al., 2022 J. Geiping, L. Fowl, G. Somepalli, M. Goldblum, M. Moeller, T. Goldstein
What doesn't kill you makes you robust(er): How to adversarially train against data poisoning
(2022)
arXiv preprint arXiv:2102.13624 ↗
Google Scholar ↗

Grinsztajn and Edouard Oyallon, 2022 Grinsztajn, L., Edouard Oyallon, G.V., 2022. Why do tree-based models still outperform deep learning on typical tabular data?. In: Proc. of NeurIPS 2022. New Orleans, LA, USA.
Google Scholar ↗

Gu et al., 2017 Gu, T., Dolan-Gavitt, B., Garg, S., 2017. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. In: Proc. of Machine Learning and Computer Security. Long Beach, CA, USA, Workshop at NeurIPS 2017.
Google Scholar ↗

Harris et al., 2020 C.R. Harris, K.J. Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith, R. Kern, M. Picus, S. Hoyer, M.H. van Kerkwijk, M. Brett, A. Haldane, J.F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T.E. Oliphant
Array programming with NumPy
Nat., 585 (7825) (2020)
Google Scholar ↗

Hong et al., 2020 S. Hong, V. Chandrasekaran, Y. Kaya, T. Dumitraş, N. Papernot
On the effectiveness of mitigating data poisoning attacks with gradient shaping
(2020)
arXiv preprint arXiv:2002.11497 ↗
Google Scholar ↗

Hoyer and Hamman, 2017 S. Hoyer, J. Hamman
Xarray: N-D labeled arrays and datasets in Python
JORS, 5 (1) (2017)
Google Scholar ↗

Hush et al., 2007 D.R. Hush, C. Scovel, I. Steinwart
Stability of unstable learning algorithms
Mach. Learn., 67 (3) (2007)
Google Scholar ↗

Jia et al., 2021 Jia, J., Cao, X., Gong, N.Z., 2021. Intrinsic Certified Robustness of Bagging against Data Poisoning Attacks. In: Proc. of AAAI 2021. Virtual.

Google Scholar ↗

Khanh et al., 2024 P.T. Khanh, T.T.H. Ngoc, S. Pramanik

Evaluation of machine learning models for mapping soil salinity in Ben Tre province, Vietnam

Multimedia Tools Appl. (2024)

Google Scholar ↗

Kumar et al., 2020 Kumar, A., Levine, A., Goldstein, T., Feizi, S., 2020. Curse of Dimensionality on Randomized Smoothing for Certifiable Robustness. In: Proc. of ICML 2020. Virtual.

Google Scholar ↗

La Ferlita et al., 2023 A. La Ferlita, Y. Qi, E. Di Nardo, O. el Moctar, T.E. Schellin, A. Ciaramella

A comparative study to estimate fuel consumption: A simplified physical approach against a data-driven model

J. Mar. Sci. Eng., 11 (4) (2023)

Google Scholar ↗

Levine and Feizi, 2021 Levine, A., Feizi, S., 2021. Deep Partition Aggregation: Provable Defenses against General Poisoning Attacks. In: Proc. of ICLR 2021. Vienna, Austria.

Google Scholar ↗

Ma et al., 2019 Ma, Y., Zhu, X., Hsu, J., 2019. Data Poisoning against Differentially-Private Learners: Attacks and Defenses. In: Proc. of IJCAI 2019. Macao, China.

Google Scholar ↗

Maghool et al., 2024 S. Maghool, V. Bellandi, P. Ceravolo

Technologies and strategies for continuous learning through electronic health records data

Advances in Intelligent Healthcare Delivery and Management: Research Papers in Honour of Professor Maria Virvou for Invaluable Contributions (2024)

Google Scholar ↗

Matulevičius, 2017 R. Matulevičius

Domain model for information systems security risk management

Fundamentals of Secure System Modelling, Springer International Publishing (2017)

Google Scholar ↗

Mauri et al., 2023 L. Mauri, B. Apolloni, E. Damiani

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

Inform. Sci., 633 (2023)

Google Scholar ↗

Nicolae et al., 2019 M.-I. Nicolae, M. Sinn, M.N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I.M. Molloy, B. Edwards
Adversarial robustness toolbox v1.0.0
(2019)
arXiv preprint arXiv:1807.01069 ↗
Google Scholar ↗

Nowroozi et al., 2024 E. Nowroozi, N. Jadalla, S. Ghelichkhani, A. Jolfaei
Mitigating label flipping attacks in malicious URL detectors using ensemble trees
(2024)
Google Scholar ↗

Pedregosa et al., 2011 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay
Scikit-learn: Machine learning in python
J. Mach. Learn. Res., 12 (2011)
Google Scholar ↗

Peri et al., 2020 Peri, N., Gupta, N., Huang, W.R., Fowl, L., Zhu, C., Feizi, S., Goldstein, T., Dickerson, J.P., 2020. Deep k-NN Defense Against Clean-Label Data Poisoning Attacks. In: Proc. of ECCV 2020.
Google Scholar ↗

Piskozub et al., 2021 Piskozub, M., De Gaspari, F., Barr-Smith, F., Mancini, L., Martinovic, I., 2021. MalPhase: Fine-Grained Malware Detection Using Network Flow Data. In: Proc. of ACM ASIA CCS 2021. Hong Kong, China.
Google Scholar ↗

Rezaei et al., 2023 Rezaei, K., Banihashem, K., Chegini, A., Feizi, S., 2023. Run-off Election: Improved Provable Defense against Data Poisoning Attacks. In: Proc. of ICML 2023. Honolulu, HI, USA.
Google Scholar ↗

Rosenfeld et al., 2020 Rosenfeld, E., Winston, E., Ravikumar, P., Kolter, Z., 2020. Certified Robustness to Label-Flipping Attacks via Randomized Smoothing. In: Proc. of ICML 2020. Virtual.
Google Scholar ↗

Shafahi et al., 2018 Shafahi, A., Huang, W.R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., Goldstein, T., 2018. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In: Proc. of NeurIPS 2018. Montréal, QC, Canada.
Google Scholar ↗

Shahid et al., 2022 A.R. Shahid, A. Imteaj, P.Y. Wu, D.A. Igoche, T. Alam

Label flipping data poisoning attack against wearable human activity recognition system
(2022)
arXiv preprint arXiv:2208.08433 ↗
Google Scholar ↗

Taheri et al., 2020 R. Taheri, R. Javidan, M. Shojafar, Z. Pooranian, A. Miri, M. Conti
On defending against label flipping attacks on malware detection systems
Neural Comput. Appl., 32 (18) (2020)
Google Scholar ↗

The pandas development team, 2020 The pandas development team
Pandas-dev/pandas: Pandas
(2020), 10.5281/zenodo.3509134 ↗
Google Scholar ↗

Wang et al., 2022 Wang, W., Levine, A., Feizi, S., 2022. Improved Certified Defenses against Data Poisoning with (Deterministic) Finite Aggregation. In: Proc. of ICML 2022. Baltimore, MD, USA.
Google Scholar ↗

Wang and Witten, 2002 Wang, Y., Witten, I.H., 2002. Modeling for Optimal Probability Prediction. In: Proc. of ICML 2002. Sidney, Australia.
Google Scholar ↗

Wes McKinney, 2010 Wes McKinney, 2010. Data Structures for Statistical Computing in Python. In: Proc. of SciPy 2010. Austin, TX, USA.
Google Scholar ↗

Xing et al., 2024 Xing, Z., Lan, Y., Yu, Y., Cao, Y., Yang, X., Yu, Y., Yu, D., 2024. BDEL: A Backdoor Attack Defense Method Based on Ensemble Learning. In: Proc. of PRICAI 2024. Kyoto, Japan.
Google Scholar ↗

Yang et al., 2020 Yang, G., Duan, T., Hu, J.E., Salman, H., Razenshteyn, I., Li, J., 2020. Randomized Smoothing of All Shapes and Sizes. In: Proc. of ICML 2020. Virtual.
Google Scholar ↗

Yerlikaya and Bahtiyar, 2022 F.A. Yerlikaya, Ş. Bahtiyar
Data poisoning attacks against machine learning algorithms
Expert Syst. Appl., 208 (2022)
Google Scholar ↗

Zhang et al., 2021 H. Zhang, N. Cheng, Y. Zhang, Z. Li
Label flipping attacks against Naive Bayes on spam filtering systems

Appl. Intell., 51 (7) (2021)

Google Scholar ↗

Zhang et al., 2023 Z. Zhang, S. Umar, A.Y.A. Hammadi, S. Yoon, E. Damiani, C.A. Ardagna, N. Bena, C.Y. Yeun

Explainable data poison attacks on human emotion evaluation systems based on EEG signals

IEEE Access, 11 (2023)

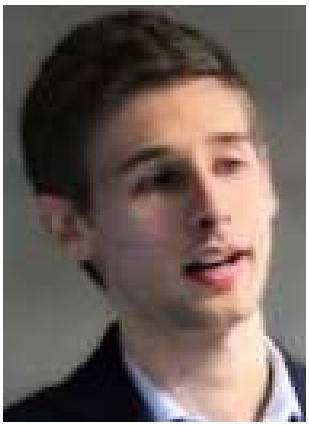
Google Scholar ↗

Cited by (1)

Adversarial artificial intelligence in radiology: Attacks, defenses, and future considerations

2025, Diagnostic and Interventional Imaging

Show abstract



Nicola Bena is a postdoc at the Department of Computer Science, Università degli Studi di Milano. His research interests are in the area of security of modern distributed systems with particular reference to certification, assurance, and risk management techniques. He has been visiting scholar at Khalifa University and at INSA Lyon.

Marco Anisetti is Full Professor at the Università degli Studi di Milano. His research interests are in the area of computational intelligence, and its application to the design and evaluation of complex systems. He has been investigating innovative solutions in the area of assurance evaluation of cloud security and AI. In this area he defined a new scheme for continuous and incremental cloud security certification, based on distributed assurance evaluation architecture.

Ernesto Damiani is Full Professor at the Department of Computer Science, Università degli Studi di Milano, where he leads the Secure Service-oriented Architectures Research (SESAR) Laboratory. He is also the Founding Director of the Center for Cyber–Physical Systems, Khalifa University, United Arab Emirates. He received an Honorary Doctorate from Institute National des Sciences Appliquees de Lyon, France, in 2017, for his contributions to research and teaching on big data analytics. He serves as Editor in Chief for IEEE Transactions on Services Computing. His research interests include cybersecurity, big data, and cloud/edge processing, and he has published over 680 peer-reviewed articles and books. He is a Distinguished Scientist of ACM and was a recipient of the 2017 Stephen Yau Award.

Chan Yeob Yeun (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in information security from Royal Holloway, University of London, in 1996 and 2000, respectively. After his Ph.D. degree, he joined Toshiba TRL, Bristol, U.K., and later, he became the Vice President at the Mobile Handset Research and Development Center, LG Electronics, Seoul, South Korea, in 2005. He was responsible for developing mobile TV technologies and related security. He left LG Electronics, in 2007, and joined ICU (merged with KAIST), South Korea, until August 2008, and then the Khalifa University of Science and Technology, in September 2008. He is currently a Researcher in cybersecurity, including the IoT/USN security, cyber–physical system security, cloud/fog security, and cryptographic techniques, an Associate Professor with the Department of Electrical Engineering and Computer Science, and the Cybersecurity Leader of the Center for Cyber–Physical Systems (C2PS). He also enjoys lecturing for M.Sc. degree in cyber security and Ph.D. degree in engineering courses at Khalifa University. He has published more than 140 journal articles and conference papers, nine book chapters, and ten international patent applications. He also works on the editorial board of multiple international journals and on the steering committee of international conferences.

Claudio A. Ardagna is Full Professor at the Università degli Studi di Milano, the Director of the CINI National Lab on Data Science, and co-founder of Moon Cloud srl. His research interests are in the area of cloud–edge and AI security and assurance, and data science. He has published more than 170 contributions in international journals, conference/workshop proceedings, and chapters in international books. He has been visiting professor at Université Jean Moulin Lyon 3 and visiting researcher at Beijing University of Posts and Telecommunications, Khalifa University, George Mason University. He is member of the Steering Committee of IEEE TCC, member of the editorial board of the IEEE TCC and IEEE TSC, and secretary of the IEEE Technical Committee on Services Computing.

☆ Research supported, in parts, by *(i)* TII, United Arab Emirates under Grant 8434000394; *(ii)* project BA-PHERD, funded by the European Union – NextGenerationEU, under the National Recovery and Resilience Plan (NRRP) Mission 4

Component 2 Investment Line 1.1: “Fondo Bando PRIN 2022” (CUP G53D23002910006); *(iii)* MUSA – Multilayered Urban Sustainability Action – project, funded by the European Union – NextGenerationEU, under the National Recovery and Resilience Plan (NRRP) Mission 4 Component 2 Investment Line 1.5: Strengthening of research structures and creation of R&D “innovation ecosystems”, set up of “territorial leaders in R&D” (CUP G43C22001370007, Code ECS00000037); *(iv)* project SERICS (PE00000014) under the NRRP MUR program funded by the EU – NextGenerationEU; *(v)* 1H-HUB and SOV-EDGE-HUB funded by Università degli Studi di Milano, Italy – PSR 2021/2022 – GSA – Linea 6; and *(vi)* Università degli Studi di Milano, Italy under the program “Piano di Sostegno alla Ricerca”. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the Italian MUR. Neither the European Union nor the Italian MUR can be held responsible for them.

1 Work performed as visiting scholar to C2PS, Computer Science Department, Khalifa University, P.O. Box: 12778, Abu Dhabi, United Arab Emirates.

2 https://www.idc.com/getdoc.jsp?containerId=prUS52600524 ↗.

3 https://www.forrester.com/blogs/genai-will-shatter-and-reconstruct-every-company-and-industry/ ↗.

4 https://eur-lex.europa.eu/eli/reg/2024/1689/oj ↗.

5 We note that this distinction is not always sharp.

6 We note that normalization is performed once, after the analysis of each data point.

7 We use “routing to the i-th model” as a shortcut to “routing to the training set of the i-th model”.

8 https://plotly.com/python/ ↗.

9 Using *pytest* v7.4.4 (https://docs.pytest.org/en/stable/ ↗) and *pytest-benchmark* v4.0.0 (https://github.com/ionelmc/pytest-benchmark ↗).