UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE

Computer Science Department
Ph.D. Program in Computer Science
XXXIV Cycle

Data Partitioning and Compensation Techniques for Secure Training of Machine Learning Models

INF/01 INFORMATICA

Doctoral Dissertation of
Lara MAURI

Supervisor
Prof. Ernesto DAMIANI

Co-Supervisor
Prof. Bruno APOLLONI

PhD Coordinator
Prof. Paolo BOLDI

Academic Year 2020/2021

This dissertation was reviewed and approved by the following three referees:

Prof. Kim-Kwang Raymond Choo
Department of Information Systems and Cyber Security
University of Texas at San Antonio, Texas, USA

Prof. Mauro Conti
Department of Mathematics
University of Padua, Italy

Dr. Yaniv Harel
Blavatnik Interdisciplinary Cyber Research Center
Tel Aviv University, Israel

A mia madre

Abstract

Advances in Machine Learning (ML), coupled with increased availability of huge amounts of data collected from diverse sources and improvements in computing power, have led to a widespread adoption of ML-based solutions in critical application scenarios. However, ML models intrinsically introduce new security vulnerabilities within the systems into which they are integrated, thereby expanding their attack surface. The security of ML-based systems hinges on the robustness of the ML model employed. By interfering with any of the phases of the learning process, an adversary can manipulate data and prevent the model from learning the correct correlations or mislead it into taking potentially harmful actions. Adversarial ML is a recent research field that addresses two specific research topics. One of them concerns the identification of security issues related to the use of ML models, and the other concerns the design of defense mechanisms to prevent or mitigate the detrimental effects of attacks.

In this dissertation, we investigate how to improve the resilience of ML models against training-time attacks under black-box knowledge assumption on both the attacker and the defender. The main contribution of this work is a novel defense mechanism which combines ensemble models (an approach traditionally used only for increasing the generalization capabilities of the model) and security risk analysis. Specifically, the results from the risk analysis in the input data space are used to guide the partitioning of the training data via an unsupervised technique. Then, we employ an ensemble of models, each trained on a different partition, and combine their output based on a majority voting mechanism to obtain the final prediction. Experiments are carried out on a publicly available dataset to assess the effectiveness of the proposed method. This novel defence technique is complemented by two other contributions, which respectively support using a Distributed Ledger to make training data tampering less convenient for attackers, and using a quantitative index to compute ML models' performance degradation before and after the deployment of the defense. Taken together, this set of techniques provides a framework to improve the robustness of the ML lifecycle.

In questioni di scienza,
l'autorità di un migliaio di persone
non vale tanto quanto la scintilla di ragione
di un singolo individuo.

– Galileo Galilei

Acknowledgements

Un ringraziamento particolare al mio supervisor Ernesto Damiani, la cui passione, dinamicità e correttezza sono state un importante supporto e stimolo durante tutto il percorso di dottorato.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

MACHINE Learning (ML) and data-driven technologies have proven to be extremely effective in understanding and predicting the behavior of real-world systems, especially after the advent of Deep Learning. Although ML approaches have shown and still do demonstrate remarkable performance and successful applications across a wide range of domains, vulnerabilities underlying learning algorithms are considered an intrinsic limitation that could result in catastrophic consequences when ML models are exposed to adversarial behavior. If multiple ML models rely on each other for decision making, compromising a single model might automatically trigger a cascading effect that leads all other models to make wrong decisions as well. Moreover, if ML models receive input from the physical world, e.g. from sensors, they can suffer from adversarial manipulation of physical objects. Therefore, security and integrity of ML models pose great concern, particularly when ML-based components are integrated into sensitive areas such as safety-critical infrastructures.

In the ML context, adversaries have the chance to attack the system at various levels either by interfering with the training phase of the model or the inference phase, after the model has been deployed. The focus of this dissertation is on the so-called data poisoning attacks, which are considered very harmful as they target what is unique to ML: training data. This type of attack occurs when adversaries are able to directly influence the learning process by maliciously modifying a portion of the samples used to train the models themselves. A recent work by

Kumar et al. [2] found that industry practitioners rank data poisoning as one of the most troubling threat against ML-based systems, claiming that they are not adequately equipped to defend, detect and react to such ML attacks. This suggests the need for new defensive mechanisms for ML-based systems that are generic and applicable to different types of models while making minimal assumptions about attackers.

Most existing defense techniques against training-time attacks focus on protecting learning algorithms from specific types of attack. This corresponds to scenarios where the defender is assumed to be aware of which malicious actions will be taken against the training data set so as to react accordingly to defend it. However, in practice the defender can hardly know in advance the kind of attack that will be launched or the exact data points that will be targeted (excluding trivial cases such as outliers). In light of the above considerations, in this dissertation we consider a defender who has no knowledge about the attacker's strategy. Therefore, before implementing any protection mechanism, estimating which points can be considered most at risk becomes essential. To this end, we propose the use of linear classifiers to approximate the separating decision boundaries between classes and assign a different risk index to points by using an iterated procedure that takes into account their distance from the identified hyperplanes. The result is a risk map the defender uses to adapt the defense mechanism based on his believes about what the attacker may do.

We propose a novel defense strategy against poisoning attacks which is based on ensemble methods. The purpose of classical ensemble learning is to improve generalization capabilities in non-adversarial settings [3, 4, 5]. In this work, we focus on using multiple learners in an ensemble system to improve the robustness of ML models while achieving accurate performance. The intuitive explanation for employing an ensemble-based methodology stems from the fact that an adversary has to corrupt more than one sub-model in order to make the whole ensemble useless. Our approach is based on partitioning the training data set into a number of partitions, whereby the partition assignment for each data point is driven by an

unsupervised technique (anti-clustering) which takes into account the aforementioned risk index. The main advantage of the approach is that our technique is generic enough to be potentially used in different settings. It does not require the model to be necessarily linear, thereby allowing deep networks to be used.

On the attack front, existing work mainly focuses on how to successively attack popular ML algorithms by injecting small fractions of well-crafted adversarial samples [6, 7, 8] into the training data set. These attack strategies usually rely on worse-case analysis where the modifications introduced into the training set are purposely designed to maximise the damage on the targeted learning algorithm. However, they are strongly tied to the attacker being aware of the exact model to attack or being able to exploit feedback from the learner to find optimal attack instances [9, 10]. In this dissertation, we make a more restrictive assumption on the adversary's knowledge: we assume the attacker is not aware of the model he is attacking, rather he only has information about (and can access to) the training set used by the target model. Obviously, lacking useful information to use to maximize a certain 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 then preferentially attacks (in a randomized fashion) the points of the training set on the basis of their proximity to the separation surface. Therefore, the attack is not optimal with respect to the target model, though it may be optimal with respect to the attacker's (lack of) knowledge.

1.1 Motivation and Research Questions

As Machine Learning is applied to increasingly sensitive tasks, it has become crucial to be able to deal with noisy or potentially untrustworthy training data. Our work is motivated by the emerging requirement that ML models need to be robust to worst-case tampering with the data assets used in their training. Clearly, significantly improving the ability of ML techniques to cope with attacks requires defense mechanisms that do not merely rely on assumptions about attackers and are not

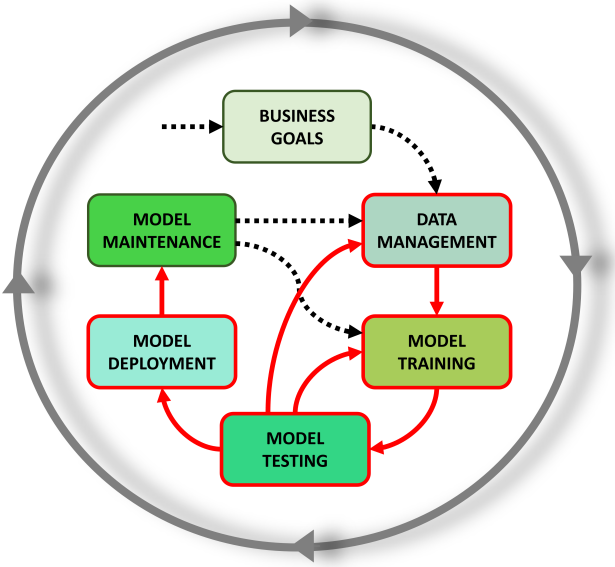


Figure 1: ML lifecycle stages covered by the techniques proposed in the thesis work (red frame).

strictly coupled to specific learning algorithms. This fosters their practicality and has inherent advantages in dealing with adversarial transferability.

We wish to provide an actionable contribution to the study of robust ML, from both a theoretical and empirical perspective. Our work spans a number of topics, both theoretical and applied, dealing with alleviating the consequences of learning in the presence of attacks. Our ambitious goal (though we will, of course, fall short of this task) is to provide a co-ordinated panoply of techniques that can be used within the ML models' lifecycle to assess data tampering risks and minimize tampering likelihood and severity. Our main thrust, however, will be in devising theoretically sound techniques for alleviating training set data tampering, showing that they transfer well to practice. In order to be applicable to a variety of ML models, our techniques rely on linear reference models (namely *Support Vector Machine* (SVM) and *Nearest Convex-Hull* (NCH) classifiers) to represent the attacker's (and the defender's) knowledge about the ML-based systems we intend to harden. Taking into consideration a typical ML lifecycle [11], the stages we cover with the techniques proposed in this work are marked in red (Figure 1).

Based on the general motivation given above, we came up with the following three main research questions.

[**RQ1**]: *How to model the attack (and defence) choices in a way compatible to different levels of knowledge about the ML model to be attacked?*

[**RQ2**]: *How to devise a model-independent defence technique (targeting the model or its data assets) to make attacks less convenient and effective?*

[**RQ3**]: *Which metric should be used to assess the level of degradation of ML models before and after the application of defence techniques?*

1.2 Main Contributions

Our main contribution toward the research questions [**RQ1**] and [**RQ2**] concerns the model training phase. We introduce a data partitioning technique which aims to mitigate the detrimental effect of corrupted data in the learning process by creating diverse subsets of the training data to be fed to the base learners of an ensemble model. The proposed defense technique also acts at the model deployment phase, as the model can be adapted according to the specific environment into which it operates. With the additional components (representing possible extensions of the technique proposed in the main part of the dissertation) we intervene before and after training, i.e. in the data management and model testing stages. In particular, we act at the level of data collection to reach a consensus on which data should be considered trustworthy and can therefore be used during training. In this sense, our contribution is the proposal of a data compensation-based scheme leveraging the security properties of Distributed Ledger-based systems to support ML models' training set selection. The approach requires those who want

to contribute data to propose it as a ledger update in such a way that it is less convenient for attackers to insert corrupted data. Regarding the post training phase, our contribution lies in the definition of an index measuring the severity of ML models' degradation against a golden standard held-out dataset. The degradation index is expressed as a function of the deformation of the convex hulls of the classes calculated on the held-out dataset. Its calculation can be helpful in deciding whether re-training is needed [**RQ3**]. The two auxiliary techniques provide *input consensus* and *a posteriori control*, respectively. The overall objective of the three proposed solutions, either used individually or in combination, is to deliver a comprehensive methodology to strengthen the security of practical ML-based systems against adversarial attacks.

1.3 Experimental Design

The thesis contains two distinct, though related, experiments. The first experiment deals with the effectiveness of the proposed ML architecture in delivering accuracy (and certified accuracy) in the presence of attacks. The second experiment concerns the capability of our proposed ML index to be used as a predictor of accuracy changes. We have selected two reference benchmarks widely used in the literature, namely the Modified National Institute of Standards and Technology (MNIST) [12] and the Belgium Traffic Sign Classification (BTSC) [13] datasets:

- ◇ MNIST is a large dataset of labeled handwritten digits that is commonly used for training various image processing systems. It contains black and white images, normalized to fit into a 28x28 pixel bounding box and anti-aliased to introduce grayscale levels. MNIST was created by extending and "re-mixing" the samples from the original NIST's training dataset.

- ◇ The BTSC dataset is extracted from the larger BelgiumTS and contains cropped images around annotations for 62 different classes of traffic signs. BTSC is split into a training part with 4591 images and a testing part with 2534 images. The split follows that of the original BelgiumTS recorded in Belgium, in urban environments from Flanders region, by GeoAutomation.

Both datasets are suitable for validating common tasks within the scope of computer vision, a field that has also recently been extensively investigated to assess the adversarial robustness of various deep neural network (DNN) architectures and approaches [14, 15]. Notably, MNIST is one of the most commonly used dataset for which finding adversarially robust models is still considered an open problem [16]. Adversarial attacks against DNN draw significant attention due to the widespread use of such architectures in critical tasks. For instance, quick and accurate identification of traffic signs is a primary concern of autonomous driving systems and partially automated vehicles [17, 18].

The rationale behind our benchmark selection lies in the different purposes of the experimentations. The first experiment needed to reveal changes in accuracy when training data items are tampered with. Therefore, data should contain samples sourced from different individuals (as MNIST handwritten samples do). The second experiment needed to identify the points in the training set that can be used to compute an held-out data set for evaluating and predicting accuracy changes. For this reason, the data set should contain multiple takes of the same image, as BTSC does (on average there are 3 images/annotations for each physically distinct traffic sign).

1.4 Dissertation Outline

This dissertation is divided into two main parts. Part I contributes to answering [**RQ1**] and [**RQ2**] by introducing the defence framework we propose to counter poisoning attacks in scenarios where the attacker has no knowledge of the exact model he is attacking, and, at the same time, the defender does not know the attacker's exact attack strategy. Part II discusses two research approaches aimed at answering [**RQ2**] and [**RQ3**]. These approaches are intended as auxiliary, integrative components of the overall approach proposed in Part I, and require additional validation. In detail, the structure of the chapters is organized as follows.

Part I: Risk-driven Ensemble Techniques for Secure Training

In *Chapter 2*, after brief introductory information on ML phases and paradigms, we present an overview of the domain of adversarial ML, including the most significant types of defence approaches proposed in the literature against data poisoning attacks along with the limitations of existing schemes.

In *Chapter 3*, we present the first component of our defence framework, i.e., a method for assessing the risk associated with ML training data which is based on assigning a risk index to points in relation to their proximity to the separation surfaces identified with a linear model (specifically, a SVM classifier). [**RQ 1-2**]

In *Chapter 4*, we describe the partitioning strategy we use to partition the training data set via an unsupervised technique (anti-clustering) accounting for the above risk index, and the ensemble construction that aggregates the sub-model outputs using majority voting. [**RQ 1-2**]

In *Chapter 5*, we evaluate the effectiveness of our approach founded on the symmetrical (lack of) information between the attacker and the defender; we assume the attacker can corrupt a portion of the training set labels. [**RQ 1**]

Part II: Auxiliary Techniques

In *Chapter 6*, we introduce an index estimating the severity of ML model's data assets degradation which serves to monitor the performance of deployed models (a posteriori), and which can be used to instantiate the parameters of the ensemble-based technique (Part I) during re-training. [**RQ 3**]

In *Chapter 7*, we discuss the use of Distributed Ledger Technologies as a means to foster the contribution of trustworthy data for use when training ML models, proposing a Proof-of-Useful-Work based on the notion of reciprocity. [**RQ 2**]

Lastly, *Chapter 8* summarizes the contributions of this dissertation and presents concluding remarks with discussions on potential directions for future work.

The following table shows a mapping between the research questions provided in Section 1.1 and the thesis chapters.

RQ	*Chapter*
1	3,4,5
2	3,4,7
3	6

1.5 Publications

Part of the research findings of this dissertation have been published or under review for publication in the following journals and conferences.

◇ *Journals:*

- (**under review**) Lara Mauri, Bruno Apolloni, and Ernesto Damiani. An Anti-clustering Partition Strategy for Improving Robustness of Ensemble ML Classifiers. *Submitted to Computers & Electrical Engineering*, 2022 [**Chapters 3-4-5**]
- Lara Mauri and Ernesto Damiani. Estimating Degradation of Machine Learning Data Assets. *ACM Journal of Data and Information Quality (JDIQ)*, 14(2):1–15, 2021 [**Chapter 6**]

◇ *Conferences:*

- Lara Mauri, Ernesto Damiani, and Stelvio Cimato. Be your Neighbor's Miner: Building Trust in Ledger Content via Reciprocally Useful Work. In *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*, pages 53–62, 2020 [**Chapter 7**]

Part I

Risk-driven Ensemble Techniques for Secure Training

Chapter 2

Background and Literature Review

This chapter presents the background of our research, reviews relevant work in the area of secure learning, and identify the limitations and gaps in the literature. First, we briefly introduce the ML lifecycle and some basic learning paradigms. Then, we present the security vulnerabilities of ML-based systems at different stages of learning. Lastly, we review prior work in the area of adversarial learning, describing existing training-time attack and defense strategies as well as the main shortcomings we identified in such approaches.

2.1 Preliminary Concepts

2.1.1 The Machine Learning Lifecycle

The notion of *ML lifecycle* relates to the steps organizations follow to develop an ML model and integrate it into a fully-fledged system. As Figure 2 shows, the process employed to produce and use ML models typically consists of a series of independent stages that are performed in an iterative fashion. Below we describe the range of activities encountered within each stage.

The preliminary step in an ML project is to reach a clear understanding of the business context, defining the business goals to be achieved along with the data

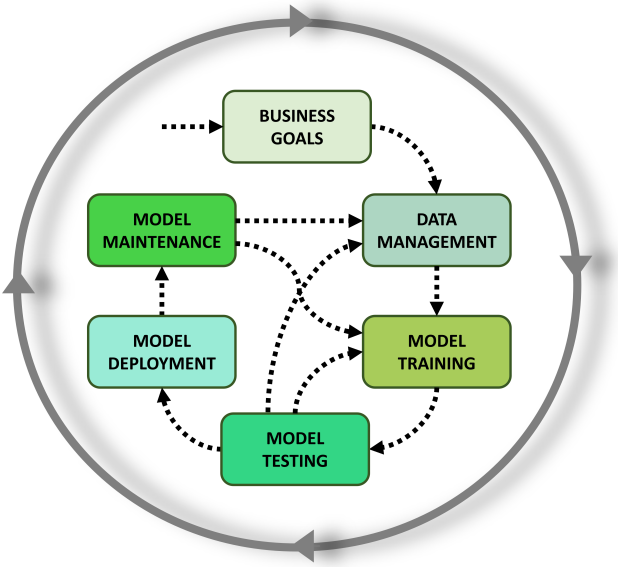


Figure 2: A typical machine learning lifecycle.

needed to achieve them. Obviously, the entire process is backed by system-level requirements from which derive the operating constraints for the ML model.

Data Management is the first actual phase of the process, and includes the following set of data curation operations. The *data ingestion* activity is responsible for the collection of all the data needed to achieve the business goal. Ingested data are usually arranged as multidimensional data points (also called data vectors). The second data management activity, *data exploration*, inspects and displays data through plots or charts. *Pre-processing* is concerned with creating a consistent dataset suitable for training, testing and evaluation. Several techniques are employed to clean, wrangle, and curate the data so as to convert it into the right format, remove noise, and anonymize it as needed. *Feature selection* reduces the number of dimensions composing each data vector in order to obtain a reduced dataset that better represents the underlying problem.

The core phase in the ML model development process is *Model Training*, which deals with selecting the ML model structure and learning the internal model parameters from the data. Depending on the nature of the available data and the business goal, different ML techniques can be employed (which will be discussed

in more detail in Section 2.1.2). In this thesis, our focus is on *supervised* learning scenarios. In the training process of a supervised ML-based system, the learning algorithm generates by trial-and-error an ML model that works well (i.e. delivers the expected output) on a baseline dataset for which the desired output is known. Essentially, training is done by computing the error made by the model on the training set data and using it to adjust the model's internal parameters to minimize the error. The achieved error reduction is continuously verified during training by feeding the ML model with some data put aside for testing. At this stage, it is critical that the training set is of high quality and trustworthy to avoid inaccuracies or inconsistencies in the data. While learning sets the values of the internal parameters of ML models, the so-called *hyper-parameters*, which control how the training is conducted (e.g., how the error is used to modify the internal parameters), are set separately during model tuning.

While being tuned, the ML model is also validated to determine whether it works properly when fed with data collected independently from the original dataset. The *Model Testing* phase includes all the activities that provide evidence of the model's ability to generalise to data not seen during training.

The transition from model development to production is handled by the *Model Deployment* phase. The trained/validated ML model is integrated into a production environment, where it can make practical decisions based on the data presented to it. Since the production data landscape may change over time, in-production ML models require continuous monitoring.

The *Model Maintenance* phase serves precisely to monitor the ML model. It feeds into earlier stages of the ML lifecycle to allow the model to be recalibrated and retrained as needed.

2.1.2 Learning Paradigms

Learning paradigms pertain to the possible scenarios under which ML-based systems can learn when data is fed to them. These paradigms differ primarily based on the type of the training data available to the learner and the way it is received, as well as on the nature of the test data used to evaluate the learning algorithm.

Other parameters for further differentiating learning approaches relate to whether learners are active or passive, and whether an online or batch (offline) learning strategy is applied. Based on the degree of interaction between the learner and the environment, three major learning paradigms arise:

- ◇ *Supervised learning:* In a supervised scenario, the training set consists of pairs of input and desired output variables (usually in the form of labels), and the goal is to learn a mapping function from the input to the output by the learning algorithm. Supervised learning problems can be further divided into two distinct groups depending on whether the output domain is categorical (*classification* problem) or cardinal (*regression* problem).

- ◇ *Unsupervised learning:* In an unsupervised scenario, the training set consists solely of unlabelled input data and no corresponding output, and the goal is to discover interesting properties/structures in the data. Unsupervised learning problems can be further grouped into *clustering* and *association* problems according to whether the aim is to infer intrinsic groupings in the data or relationships (usually represented in form of rules) hidden in large datasets.

- ◇ *Reinforcement learning:* In this paradigm, learning is conducted in an exploratory fashion with some form of supervision. The learning algorithm actively interacts with the environment and receives a feedback (reward) whenever it performs an action (i.e., selects an output for a given observation) in an attempt to accomplish the specific goal for which it is rewarded and to receive maximum reward at the same time.

2.2 Adversarial Machine Learning

One of the basic assumptions in learning theory is that training data accurately represent the underlying phenomenon addressed by learning [19]. This assumption is obviously violated when data is altered, either intentionally or unintentionally, to the extent that the statistical distribution of training data differs from that

of test data. The problem of diverging distributions, also referred to as *dataset shift*, is commonly considered the root cause of performance degradation of trained models [20]. In practical situations, potential differences in the two distributions may be the result of natural drifts due to the effect of time, the presence of biased samples or changes in trends and customer behavior – classical examples can be found in recommender systems [21], natural language processing [22] and speech recognition [23]. In this scenario, mechanisms for effectively handling natural low-dimensional and geometrically simple distribution shifts have been reported in the literature [24, 25]. By contrast, considerably more challenging are the adversarial cases, where one has to deal with malicious data modifications. This is due to the fact that ML techniques have not been originally conceived to cope with cunning adversaries who, in principle, may undermine the whole system security by exploiting the intrinsic vulnerabilities of ML technologies (e.g., learning algorithms or generated models) through careful data manipulation. Moreover, the need to periodically retrain ML models to adapt their decision-making capabilities to changes gives adversaries additional room to interfere with the learning process.

A number of attack surface and attack vectors can be identified along a typical ML lifecycle. On one hand, some of the potential vulnerabilities are already known to exist in conventional IT systems and still remain part of the ML attack surface, though perhaps they can be seen in a new light when examined through the ML lens. On the other hand, this traditional attack surface expands along new axes when considering the specific, multifaceted and dynamic nature of ML processes. The resulting surface is therefore extremely complex, and mapping it requires going through all the various steps of the ML lifecycle and explaining the different security threats, a task that is inherently challenging due to the large amount of vectors that an adversary can target. Regardless of the ML stage targeted by the adversary, attacks against ML-based systems have negative impacts that generally result in performance decrease, system misbehavior, and/or privacy breach.

Far from being a mere hypothesis, adversarial exploitation of ML vulnerabilities have become a reality in various applications. These include antivirus

engines, autonomous bots, visual recognition and social networks, among others [26, 15, 27, 28]. These attacks have motivated the investigation of ML security properties, leading to the novel research field of *adversarial machine learning*, which lies at the intersection of machine learning and computer security. In an effort towards improving the robustness of ML models so that learning is successful despite operating in adversarial settings, this emerging field aims to address the following main open issues: *(i)* identifying potential weaknesses of ML-based systems, *(ii)* devising the corresponding attacks and evaluating their impact on the attacked system, and *(iii)* proposing countermeasures against the considered attacks. An overview of the evolution of active research in this emerging area over the last ten years can be found in [29], where the authors presented a historical picture of the work related to the security of machine learning from a technical perspective.

2.2.1 Threat Modeling

Defining an accurate threat model is a key requirement to proper risk analysis of any system. A threat model is an integral component of any defence strategy because it specifies the conditions under which attacks are carried out and the defence is supposed to operate. Barreno et al. [30, 31] and Huang et al. [32] were among the first to propose a security framework specific for the ML domain. The framework, along with its subsequent extensions by other authors [33, 34], is intended to serve as a guidance for correctly identifying where and how an ML model may be attacked by providing careful profiling of the adversary who wish to subvert the system. In this pioneering work, the discussion was centered on the particular characteristics of the affected application, i.e., intrusion detection and spam filtering. More recent work focuses on the security properties of deep learning algorithms in the computer vision and cybersecurity domains. For example, Yuan et al. [35] proposed a taxonomy of attack approaches for generating adversarial examples. Pitropakis et al. [36] further provided an extensive survey of ML vulnerabilities and associated potential attack strategies, while Papernot [37]

covered ML security and privacy through the lens of Saltzer and Schroeder's principles [38]. The common denominator among these threat models is the abstraction of the underlying system to characterize possible attack vectors along multiple axes. Typically, a threefold approach is used based on the *goals*, *capabilities* and *knowledge* of the attacker. In the following, we briefly characterize each of these axes.

2.2.1.1 Adversarial Goals

The first axis, i.e., adversarial goals, relates to the type of security violation the attacker may cause, and the specificity of the attack and error the adversary intends to produce. As for the former aspect, following the classical CIA triad (confidentiality, integrity, availability), the attacker is supposed to undermine the functionality of the system under attack, or to deduce sensitive information about it. More in detail, violating integrity implies performing malicious activities without compromising normal system operation (e.g. evade a spam email detection system via false negative). Typically, integrity is compromised when an adversary is capable to manipulate model inputs so as to control model outputs. By contrast, an availability violation interferes with normal operation and consists of preventing access to a resource or system functionality by legitimate users. Here, the goal is to make the model inconsistent with respect to the target environment. By violating privacy, an attacker gains unauthorized access to sensitive/confidential information about the system, such as parameters or data used to train the model. This aspect is essentially linked to the need of preventing the exposure of sensitive information in environments where users have different degrees of trust. As for attack specificity, an attacker may launch either targeted attacks focusing on specific samples or indiscriminate attacks focusing on a broad range of samples. In the context of ML classifier systems, error specificity is a characteristic introduced in [34] to disambiguate the notion of misclassification in multiclass problems. An attacker may aim to mislead the system to incorrectly classify an input sample into a specific class (error-specific attacks) or into any of the classes except the correct class (error-generic attacks).

2.2.1.2 Adversarial Capabilities

This second axis is defined based on the extent to which the adversary can influence the training data or input samples, or observe the output of a trained model, and on the presence of data manipulation constraints. The attack influence can be categorized as *causative* if the attacker has the ability to influence the learning process by manipulating the training data, or *exploratory* if the attacker can only manipulate input samples during the prediction phase, possibly observing the model's decisions on these carefully crafted instances. Thus, exploratory attacks aim to cause the model to produce erroneous output, rather than to tamper with it. The potential presence of constraints for data manipulation by the attacker is strongly related to the application domain, but in general, we can distinguish two models of adversarial corruption: data insertion and data alteration. At one side, we may consider an adversary with unlimited control over a small fraction of the data. In this scenario, the attacker is restricted to alter only a limited amount of data points, but is allowed to modify them arbitrarily. An example is when the attacker crafts a small number of attack instances that he then inserts into the dataset for training or evaluation. At the other side, we may assume that the attacker can manipulate any of the data points, but with a limited degree of alteration.

2.2.1.3 Adversarial Knowledge

Within the ML context, it is possible to identify several data and information that are considered sensitive in view of possible attacks. These include training data, learning algorithms and architecture, hyperparameters and weights, among others. Formally, the attacker's knowledge can be described in terms of a vector $\theta = (D, X, f, w)$ consisting of four elements representing his level of access to the different ML system components under attack: *(i)* the dataset used for training D; *(ii)* the set of features X; *(iii)* the learning algorithm f, along with the objective function optimized during training; *(iv)* the parameters of the ML algorithm w. This representation enables the definition of different attack settings, ranging from *white-box* attacks to *black-box* attacks, with varying degrees of black-box access.

In the case of a white-box attack, the adversary has full knowledge of the target system. This scenario is quite unrealistic, but it allows one to perform a worst-case evaluation of ML system security, enabling the estimation of the upper bounds of the performance degradation that is likely to be incurred by the system under attack. In the more challenging black-box setting, the adversary has no or limited knowledge about the targeted ML system. Typically, in limited knowledge attacks with surrogate data, the attacker is assumed to know the feature set X, the model architecture and the learning algorithms f, but not the training data D and the trained parameters w. The attacker may be able to collect a surrogate training set from a similar source having analogous characteristics and data distribution and then estimate the parameters of f by leveraging the surrogate dataset. On the contrary, limited knowledge attacks with surrogate models imply that the attacker can use a surrogate model (which may differ from the targeted model) to craft the attack points, since he knows D and X, but not the learning algorithm f.

In addition to the three dimensions just mentioned, another aspect to consider is the strategy adopted by the attacker, which in many cases can be formulated as an optimization problem taking into account the various aspects of the threat model.

2.2.2 Training Under Adversarial Conditions

Attacks against ML-based systems exist at every stage of the ML lifecycle, and an attack launched at a certain stage of the learning process has the potential to cause cascading effects at subsequent phases. Current research focuses primarily on offensive approaches targeting the two core phases of learning, namely the model training and testing phases. In a test-time attack, also known as *evasion attack*, the goal is to evade the trained ML model by elaborately modifying clean target instances. Conversely, the so called *poisoning attack*, takes place either during the initial training phase of the ML model or during the re-training phase, and the goal is to adversely affect the performance of an ML-based system by inserting, editing or removing points into the training set. Below we examine poisoning attacks in more detail, as they are the focus of this thesis.

Poisoning is considered one of the most effective attacks against an ML-based system [31, 39]. In this type of attack, the attacker is supposed to have some control over a portion of the training data used by the learning algorithm, and the consequences of the attack depend on the attacker's ability to access/manipulate the training data. The obvious reason for assuming that the attacker is able to modify only a moderate fraction of the data is that an unbounded adversary can cause the leaner to learn any arbitrary function. Thus, in general, all attack scenarios bound the effort required by the adversary to achieve his desired goal [40]. The high-level objective of poisoning, which is categorized as a causative attack, is to influence or corrupt the ML model itself, resulting in a degradation of the system's performance that may in turn facilitate subsequent system evasion [41]. Specifically, an attacker may launch either an error-generic poisoning attack or an error-specific attack. In the former case, the objective is to induce the ML model to produce a massive number of false outputs such that the learning process is subverted and eventually the system becomes unusable for end users. In the latter case, the attacker seeks to induce the ML model to produce specific kinds of errors, such as a specific incorrect classification [42, 8]. Let us consider attacks involving manipulation of training set labels. In these scenarios, the attackers could randomly draw new labels for a part of the training pool or choose them to cause maximum disruption. For example, in a *label-flipping attack*, the attacker introduces label noise into the training set by modifying the labeling information contained therein. Basically, the attacker selects the subset of the training data for which he wants to change the label either randomly or by following specific criteria based on his aims. Biggio et al. [43] studied the effect of label noise in support vector machines, performing both random and adversarial label flipping, and showed how model performance decreases when varying the percentage of flipping performed. In the case of adversarial label flips, the adversary aims to find the combination of flips maximizing the classification error on uncontaminated test data, which corresponds to designing an optimal attack strategy. The main technical difficulty in devising a poisoning attack is the computation of the poisoning

samples. Several studies have shown that with such a strategy, even a small percentage of carefully poisoned data can dramatically decrease the performance of the ML model under attack [44, 10, 45]. Another attack is manipulating feature values in the training set — either by perturbing them to shift the classification boundary or by adding an invisible watermark that can later be used to "backdoor" into the model. The most common defense technique against these attacks is *outlier detection*. Unfortunately, attackers can generate poisoning points that are very similar to the true data distribution (called *inliers*) but that still successfully mislead the model. An interesting approach are *micro-model* protocols [46] that partition the training set horizontally and train classifiers on non-overlapping epochs (called *micro-models*), evaluated on the entire training set. By taking majority voting of the micro-models, training data items can be classified as either safe or suspicious. Intuition suggests that there is safety in numbers, as attackers could only affect a few micro-models at a time. Another common type of defense is to analyze *a priori* the impact of newly added training data on the model's accuracy. The idea is that if an input is poisonous, it will destroy the model's accuracy on the test set. This can be spotted by doing a sandbox execution of the model (or of a simplified version of it) with the new sample before adding it to the production training pool. Below we discuss common defenses in more detail.

2.3 Defence Strategies Against Poisoning Attacks

As mentioned, adversarial ML research has two main branches. One branch actively designs ingenious attacks to defeat ML-based systems. The other branch studies ways to enhance their capability in coping with such attacks. Recently, there hes been a flurry of activity focused on designing techniques to improve the robustness of ML-based systems against training-time attacks [47, 48, 49, 50, 51]. Previous work has investigated both empirical and theoretical defence strategies for mitigating data poisoning attacks at different stages of the ML lifecycle. Many of the proposed techniques have limitations in terms of applicability, type of attack they protect against, effect on accuracy, and increase in training complexity.

In the following, we provide an overview of the most significant approaches proposed in the literature. Our defense technique, which will be detailed in Chapter 4, falls into the model enhancement category, and in particular is based on model composition.

2.3.1 Detection-based Schemes

A common defence strategy against poisoning attacks involves the use of detection-based schemes that seek to identify possible directions along which poisoned data deviate from their uncorrupted counterparts, and then sanitize or exclude the suspicious points (outliers) from the final dataset used for training.

2.3.1.1 Data Sanitization

The Reject on Negative Impact (RONI) defense [52], which was proposed against spam filter poisoning attacks, assesses the impact of each individual suspicious instance on training and discards points exhibiting a significant negative effect on the model's performance. Albeit this technique has been proven effective against some specific types of poisoning attacks, its main limitation is the high run-time overhead due to frequent retraining and the occurrence of overfitting in cases where the dataset is small compared to the number of features. In [53], Paudice et al. proposed a countermeasure against optimal poisoning attacks based on pre-filtering with outlier detection for linear classifier, and showed that their method can mitigate the effect of the attacks even when the data is scarce compared to the number of features. In another work by the same authors [45], a sanitization-based mechanism was presented to identify and re-label training points suspected of being malicious. Their approach makes use of k-Nearest-Neighbours (kNN) to detect samples having a negative impact on the performance of ML classifiers and assigns to each data point the most common label among its k nearest neighbours in feature space. Similarly, in [54], the authors proposed a defence strategy that filters out outliers by solving an optimization problem. This proposal requires providing as a parameter a value corresponding to the estimated percentage of points that are expected to be outliers. However, this task is very challenging, and in

the case where this estimate does not match actual conditions, the performance of the algorithm decreases dramatically, especially when the system is not under attack. Other interesting works that have addressed the problem of removing contaminative effect due to poisoning include [46] and [55]. Defenses based on point filtering are easy to deploy as they consist of an additional pre-processing step to be added to the learning procedure, but necessitate extensive hyperparameter tuning. Moreover, some recent research has questioned whether data sanitization defenses are vulnerable to attackers who explicitly seek to evade anomaly detection [56, 57]. The affirmative answer was given by Koh et al. [58], who showed that certain outlier-based defences are effectively vulnerable to adaptive attackers who explicitly attempt to evade anomaly detection. In particular, they showed how to bypass common data sanitization techniques such as anomaly detectors based on nearest neighbors, training loss, and singular-value decomposition.

2.3.1.2 Robust Estimation

The idea behind pre-filtering solutions draws on the notion of *robust statistics*, a line of work which has been studying the fundamental problem of learning in the presence of outliers since the 1960s [59, 60, 61]. The main objective of robust learning is to harden ML models by improving their generalization capability. Recently, the problem has received considerable attention due to the pressing need to design modern ML models for high-dimensional datasets that are robust and computationally efficient [62, 63, 64]. For instance, Steinhardt et al. [65] introduced a simple criterion –*resilience*– which, if satisfied, ensures that properties of a dataset, such as its mean, can be robustly estimated even in the presence of a large fraction of arbitrary extra samples. Some work such as [62] analyzed mean and covariance estimation, while other focused on estimating Gaussian and binary product distributions, obtaining dimension-independent errors, and in many cases errors almost linearly dependent on the fraction of adversarially corrupted samples [66]. A number of additional results have also been published. An overview of the recent developments on algorithmic aspects of high dimensional robust statistics can be found in [67].

2.3.2 Model Enhancement Mechanisms

Unlike the schemes described above, model enhancement defences do not aim to remove the points that are supposed to have been attacked (by adopting pre-filtering), rather they act directly during the training phase and aim to prevent poisoning from taking effect by leveraging various methods.

2.3.2.1 Adversarial Poisoning

Borgnia et al. [68, 69] investigated the effects of multiple augmentation schemes on data poisoning attacks and demonstrated that strong data augmentations such as mixup [70] and cutout [71] can desensitize models to triggers and data perturbations. The principle is that by modifying model input via input pre-processing techniques, backdoor trigger recognition is prevented. These strategies can be viewed as special cases of *adversarial poisoning*, where the *adversarial training* technique (one of the primary defenses against adversarial example [72]) is adapted to defend against training-time attacks [73, 74]. In its original form, adversarial training involves augmenting the training data with on-the-fly crafted adversarial examples so as to desensitize neural networks (NN) to test-time adversarial perturbations [75]. In adversarial poisoning, training data are modified in a similar fashion, but for the purpose of desensitizing NNs to the specific types of perturbations caused by data poisoning [76]. Some work has also explored the in-depth interaction of adversarial training with noisy labels, focusing on the smoothing effects of adversarial training under label noise [77]. Albeit promising, research in this regard is in its infancy. At present, these techniques are heuristic approaches that lack formal guarantees on convergence and robustness properties, and thus deserve further investigation.

2.3.2.2 Model Composition

Another line of work relates to the use of ensemble learning to reduce the influence of poisoning samples via partitioning the training set. Most studies focused on the popular *Bootstrap aggregation* (or bagging) framework [78], arguing that, in

addition to accuracy, bagging can also improve robustness in adversarial settings. In their preliminary work, Biggio et al. [79] 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. Similarly, other work [80] investigated the use of bagging and random subspace methods [81] for constructing robust systems of multiple classifiers, extending the preliminary results presented in [82, 83]. Apart from these studies that investigate the potential benefits of classical model composition schemes for defensive purposes, research proposing the use of ensemble models with new data partitioning schemes that explicitly account for poisoning attacks is rather limited. Many of the existing studies actually target inference-time attacks [84, 85, 86, 87]. With regard to training-time attacks, one approach that is emerging in the research community is based on the conjunction of ensembles and certifiable robustness of ML models [88, 89, 90] for developing *provably robust defences* against data poisoning [91, 88, 92, 93]. Specifically, some works focused on distributional robustness guarantees [94, 62, 95], while others focused on pointwise certified robustness [96, 97, 98]. Jia et al. [99] leveraged the intrinsic majority vote mechanism of kNN and rNN (radius Nearest Neighbors [100]) and showed they provide deterministic certified accuracy against both data poisoning and backdoor attacks. Levine and Feizi [97] proposed a certifiable ensemble-based method where the partitioning of the training set into disjoint subsets is deterministically performed via a hash function. However, there is no consensus on the metrics to be used, although some quantitative metrics of model robustness in face of label-flipping attacks have been proposed [101].

2.4 Limitations and Gaps

For the purpose of determining intervention points on which to focus our research work, we have performed a gap analysis along the lines of [102]. Specifically, we have identified three shortcomings in the techniques proposed in the literature

(reviewed in the previous Section), which are as follows:

- ◇ *Difficult applicability in case of limited access to the ML model to be defended* [75, 54, 29]. Most existing defense schemes are model-dependent as they require some degree of knowledge of the underlying ML model details, and thus cannot be applied blindly to all models. By contrast, approaches that treat the protected model as a black box are inherently more robust to the transferability of attacks.

- ◇ *Tenuous connections to basic cyber-security concepts (especially threat, vulnerability, and risk)* [47, 51]. The risk definition from statistical decision theory need to be tailored to the ML context throughout multiple domain of applications. Assessing the risk of compromise associated with specific input data enables well-founded decisions about which defense strategy to adopt and which data to consider most sensitive towards building a secure ML model.

- ◇ *Lack of a shared quantitative definition of the severity of attacks and the effectiveness of defense measures* [103, 48, 104, 105]. No standardized or officially approved risk indicators and attack magnitude indices are available that can guide ML practitioners in estimating ML models' performance degradation before and after the deployment of a defense mechanism.

Chapter 3

Risk Analysis of ML Data Assets

This chapter outlines a method for assessing the risk associated with ML training data which is based on assigning a risk index to points in relation to their proximity to the separation surfaces identified with a linear model (specifically, a SVM classifier).

3.1 Risk Estimation Techniques

In security, the assessment of risks is the first step towards the adoption of appropriate security measures for the protection of assets. Traditionally, assessing the risk of an attack to a given asset requires two estimations: the one of the attack's *severity*, based on the available information on the affected assets' value, and the one of the attack's *likelihood* based on available information about the threats and the asset's vulnerabilities. The product of severity and likelihood is used as the reference model for risk quantification [106], writing $R = S \times L$, where L is a measure of likelihood (for example, a probability or a possibility value [107]) and S is a severity value in monetary units.

Some studies [108] introduced the concept of *risk index*. The risk index (or risk degree) is a joint quantification of severity and likelihood that can be quantized into discrete levels, e.g., *very high risk*, *high risk*, *medium risk*, *low risk*, and *very low risk*, or computed as a continuous *risk score*. In terms of the data tampering risk, computing the risk score of individual data points with respect to data tampering

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 have been proposed to address learning risk from examples [109], most of them assuming a linear regression model [108] 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 [110], as close data points may have different severity or likelihood of attacks.

3.2 A Risk Score for Training Data

In terms of data tampering risk in the training and deployment of ML models, 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) [11, 111]. When the specific data asset under attack is the training set, one can follow the $R = S \times L$ risk model. In principle, the data tampering attack's severity can be linked to model's performance degradation after attack. This would however require consensus on the performance degradation metrics to use (see Section 2.3.2.2) and on the procedure to be used for computing it. Research is ongoing on extracting "gold standard" data sets from input data spaces, providing held-out benchmarks suitable for measuring ML performance degradation (see Chapter 6). However, such held-out data sets are forcibly problem-dependent.

In terms of the 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 [112]. This type of likelihood estimate applies to an entire data asset rather than to individual data items, and is not yet suitable for the risk assessment concerning individual data points.

In order to escape the pitfall of linear risk models [110] (and enable fine-granularity risk assessment), we model the training data risk index as a non-linear function of the features, enabling a greater variance of risk index values across the training set. Low-degree polynomials with decreasing coefficients (in the form

$a + bx + cx^2$) have been used since long in risk modelling [113], as they support capturing 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.

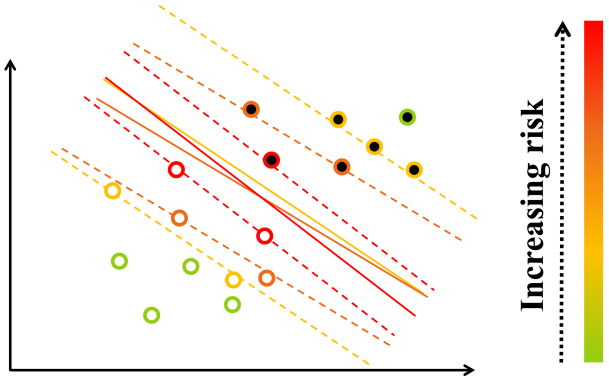


Figure 3: A simplified 2D representation of risk-related color assignment according to Algorithm 1.

3.3 Calculating Our Risk Index

A key statement of ML research is that not all points in a training set have the same relevance. This is stemmed, for instance, by the *Vapnik-Chervonenkis dimension* [114] and the analogous *Sentry points* notion [115], where only a relatively small number of items of a training set are responsible for the Boolean function being learned with no tolerance on errors (*consistent function*). In particular, for the separating hyperplanes considered in this thesis, this number is less or equal to the dimensionality of the points plus 1.

The general idea is to relate the risk index of the training points to their belonging or not to the set of the support vectors identified using a Support Vector Machine (SVM) algorithm. We consider more "relevant" the points not far from the hyperplane, with proper graduation (corresponding to the colors of the map shown in Figure 3). To this end, we apply an iterated process that allows us to identify the support vectors of the hyperplane separating classes according to

a progressive pruning of the points supporting the separator in a previous iteration (see the pseudo-code in Algorithm 1). In practice, 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 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. Figure 4 illustrates the implementation of the algorithm in a simple two-dimensional space.

Algorithm 1 Generating color graduation

Input: Original training set $D = \{(x_i, y_i)\}_{i=1}^{n}$, linear separator $\mathcal{L}$, color set $\boldsymbol{c}$
Output: Color map $\boldsymbol{xc}$

$D' \leftarrow D$
$j \leftarrow 1$
$\boldsymbol{xc} \leftarrow \emptyset$
$t^{max} \leftarrow |\boldsymbol{c}| - 1$
while $j < t^{max}$ **do**
    Identify support vectors sv_j of $\mathcal{L}_j$ separating data points in D'
    Associate color c_j to sv_j
    Add $\{sv_j, c_j\}$ to $\boldsymbol{xc}$
    Remove sv_j from D'
    $j \leftarrow j + 1$
end while
$\forall x \in D'$:
    Associate color c_0 to x
    Add $\{x, c_0\}$ to $\boldsymbol{xc}$
return $\boldsymbol{xc}$

In this way we characterize points which are close to the joint boundary of the classes, the mislabeling of which may induce severe errors in the learning procedure. In principle, points more internal to the classes are useless *per se*; their addition has the sole role of increasing the number of points considered.

As will be illustrated in Chapter 4, the calculation of this risk index in the adversarial context is performed by both the defender and the attacker, but the

use they will make of it is different. Furthermore, we assume that both the defender and the attacker do not know the attack strategy adopted by the attacker and the learning algorithm employed by the defender, respectively. This obviously introduces a degree of uncertainty about the impact of the indices themselves. In summary, it is necessary to consider the following two aspects: 1) the attacker does not know the separator that will be exactly used by the defender and vice versa, 2) when two sets are not linearly separable, many more points than just the support vectors must be considered to determine this separation hyperplane.

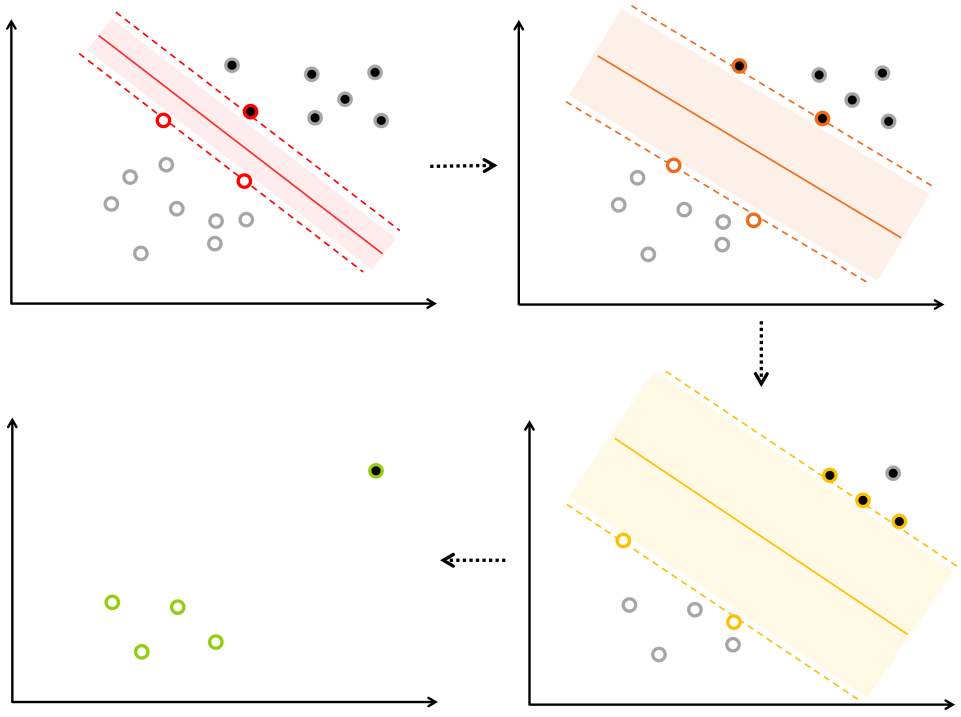


Figure 4: Assigning different gradations of risk-related colors based on proximity to separator hyperplanes.

In turn, the bordering condition depends on the function we adopt to separate the classes. Thus, for instance a point that is far from a bordering hyperplane may result in being close to another bordering surface. Since we assume that neither

the attacker nor the defender knows the *true* classifier surface, adopting separation hyperplanes according to the Occam's razor principle has a twofold advantage: we do not introduce arbitrary features into the classifier, and we rely on a simple and robust separator, which normally fosters learning generalization to the test set.

The next Section introduces the problem definition, along with general information about the attack strategy and the proposed defence approach that will be detailed in Chapter 4, and their link with the use of the reference linear model and the risk indices.

3.4 Problem Definition and Solution Outline

Consistent with the vast majority of existing literature in poisoning attacks, here we consider binary linear classification problems. Furthermore, we assume the attacker is able to manipulate the labels of some training data, i.e., he can perform label-flipping poisoning attacks. In principle, the game between the defender looking for a classification setting that is more robust against label flips and the attacker looking for a flipping strategy that mostly degrades the performance of the classifier under constraints on the flipping budget can be formalized as a bi-level optimization problem [116]

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

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

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

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 f_S which minimizes V, as in (2). However, this optimization problem proves very difficult to solve in general [117], depending on the complexity of f and on the discreteness of y. A rich literature proposes various approaches to address this issue, including the simplification of f, typically by reverting it to linear functions formalized via SVMs [116, 118, 32, 119, 120, 34], and by smoothing the labels, either in terms of continuous variables to be discretized [116], or in terms of the probabilities with which y takes on its values [118], or by looking for greedy solutions, which identify the points to be poisoned one at a time [6]. Unfortunately, the diversity of these techniques makes comparison between them essentially based on numerical experiments whose run times are relatively high. Also, extending the model from binary to multi-class classification is not obvious. For these reasons, we devised an alternative attack-defense framework that shows some benefits in terms of feasibility and efficiency.

Given the assumptions made about the attacker's and defender's knowledge, the respective actions taken to solve the problem are as follows.

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

  Rationale: As support vectors are the points which determine the position of the hyperplane separating a pair of classes, either in the original space or in a kernelized 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. Obviously, one may expect that few points farther away from the separator may damage the classifier even more. However, the more distant they are from the separator the more they can be disabled by both the robustness of the classification algorithms being employed and by sanitization algorithms. Correspondingly, points close to the boundary may fly under the defender's radar. Thus, the attacker may be interested in flipping the labels of these points.

◇ The defender's strategy takes advantages of ensemble model composition (consisting of training different learner on disjoint subsets of the entire training set and merging their outputs according to a consensus algorithm), whose strong theoretical foundation is widely appreciated in the literature [121]. Here we specialize the partitioning technique so as to make the individual models diverse enough to maintain high accuracy and to be resilient to poisoning.

Rationale: If the attacker flips the label of a point with high risk index, this will degrade the performance of only one sub-model. If another point, which is close to the first one, is assigned to another partition, thus affecting a different sub-model, we can expect two benefits: (*i*) the classification of the latter point is not degraded by the label flip of the former, since it obeys the second sub-model, (*ii*) the second sub-model is expected to correctly classify the former point, contributing to a correct result in the majority voting mechanism. Thus, our interest is to put into different partitions the points that are close to each other in some feature space of interest, and this is exactly the goal of the anti-clustering algorithm.

Chapter 4

Anti-clustering Partitioning and Model Composition Against Training-time Attacks

In this chapter, we present our novel defence framework against poisoning attacks, describing the various components into which it is articulated along with the threat model, which defines the information the adversary has at his disposal and the type of attack he can perform. The key idea behind the proposed defense scheme is to make an attack less effective through a data partitioning technique guided by the risk analysis described in the previous chapter.

4.1 Adversarial Threat Model

In the following, we provide our set of assumptions about the influence the attacker has on the data used by the learning algorithm and his level of knowledge about the targeted ML model. Then, we provide a description of the strategy and of the type of poisoning attack he can perform.

4.1.1 Attacker's Power

As discussed in Section 2.2.1, 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. Instead of assuming the attacker has perfect knowledge about the targeted system, here we consider a more realistic scenario where he launches his attack disposing of limited information about the system under attack. In particular, the adversary knows the input feature representation X and the training data D, but not the learning algorithm f. The adversary builds his own *surrogate model* $\hat{f}$ (in our case, a linear SVM model) that he uses to estimate which points to attack. Therefore, the black-box attack with limited knowledge can be denoted with $\hat{\kappa} = (D, X, \hat{f}, \hat{w})$.

Since any system can be trivially bypassed by an unconstrained attacker taking full control of the training data, imposing restrictions on the attacker is necessary to design meaningful defenses, but also to reflect the actual (constrained) circumstances under which the attacker is likely to operate. In line with the existing literature in poisoning attacks [122], we assume here 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., he can perform a label-flipping attack. Furthermore, he aims to produce specific types of error, which means that he can decide the *direction* of the flip (e.g., causing only a certain label of his interest to flip). We denote the altered training set with $D' = \{(x_i, y_i)\}_{i=1}^{n}$.

4.1.2 Attack Algorithm

As mentioned above, the attacker knows nothing about the target model. In order to select the samples that, if modified, would cause the maximum decrease in the accuracy of the target model w.r.t. his goal, he uses the surrogate model to approximate the target discriminating function and to generate the probability distribution that serves in selecting the points to attack. Therefore, before performing any flip, the attacker applies Algorithm 1 (see Section 3.3). Obviously, a rational attacker's intention would be to preferentially manipulate the most risky

points identified by the surrogate model. One natural way of implementing this strategy is to employ a *probability weighting function* (p_ℓ). In particular, we model the above behaviour with a nonlinear function of probability 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. Note that, for now, in Eq. (4) we assume that the attacker chooses to flip the labels of only the points corresponding to the identified support vectors, thus excluding the remaining points, i.e, those 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 identified risky points having the same level of risk as it. From this it follows that 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 sample is flipped.

Algorithm 2 describes the procedure for the flipping attack strategy. This algorithm is fully operational if the percentage of flipped points is relatively moderate. If the attacker can flip many labels, the result is that all the labels of the risky points are flipped, plus the labels of the points not identified as support vectors in any of the iterations of the risk index calculation algorithm are flipped randomly.

A second aspect concerns the direction of the label-flipping attack, which can be either symmetrical or asymmetrical. Provided that in any case a proper check must be be put in place to avoid multiple flips of the same label that could bring back the original value of the label, the first option (symmetric flip) turns out to be less effective from the attacker's point of view, since the flip on the two halves can offset their effect, thus leaving the separating hyperplane almost unchanged. The second option circumvents this drawback (in Algorithm 2, the check at line 7

indicates the use of the asymmetric technique (monodirectional attack).

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 ϵ
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'

4.2 The Proposed Defence Framework Under the Hood

Our novel defence strategy consists of three main components: (*i*) calculation and polynomization of risk indices associated with training data, (*ii*) anti-clustering partitioning, and (*iii*) ensemble composition for increasing diversity and attack resilience. Flowchart in Figure 5 illustrates the different components of the proposed framework. The rest of this Chapter defines the characteristics of each component and how they relate to each other.

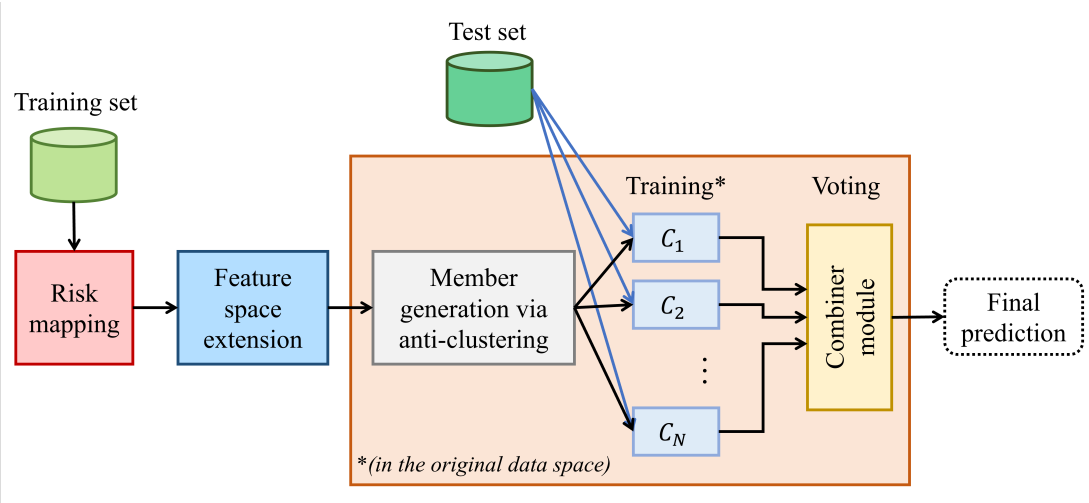


Figure 5: Flowchart of the proposed defence framework.

4.2.1 Feature Space Extension

The first component of our defence framework is our risk index. In line with the general remarks made in Chapter 3, it can be defined in this setting as an explicit polynomial function of the color, which is a quantization of point's distance from a separation hyperplane of our reference SVM model. It is important to remark that, should attack data be available in the form of reports on previously attacked data points, some alternative (and conceivably non-linear) risk index could be learnt from them. In case of *rational attacks* [123], which tamper data based on the damage, intuition suggests that the learnt risk index landscape would be close to the graph of our explicit function of the color map. The idea here is to use the risk index as part of the information that will be used to assign the data points to the training sets of multiple learners, in order to minimize the learners' exposure to poisoned data. In principle, we can 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 4.2.3) on the extended data space to group points into partitions that are highly diverse in terms of both risk and position. Each partition, stripped of the risk feature, can then be used for training

a separate learner, minimizing its exposure to potentially poisoned data. For the sake of flexibility, however, in addition to computing the risk index as a polynomial function of the color, then we will use this additional feature for rescaling when computing the distance in the expanded feature space at (anti-)clustering time (Section 4.2.3).

4.2.2 Ensemble Structuring and Composition

The second component of the proposed 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 (which are usually called *base learners*) and combine their predictions based on a consensus algorithm, treating them as a committee of decision makers. Numerous empirical and theoretical studies have demonstrated that ensemble models very often attain higher accuracy than single models [124]. The principle is that when individual predictions from different base learners are combined appropriately, the committee's decision has on average better overall accuracy than that of any individual constituent member. Our intuition is 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 resilience to poisoning attacks. Our approach relies on two key assumptions:

1. Individual models are sufficiently diverse as to maintain high accuracy and be resistant to training-time attacks;

2. There exist enough intact models in the ensemble such that the consensus output is not corrupted.

Assumption (1) hinges on the way the base models are trained, and comes as a result of our proposed approach for the training phase. Assumption (2) depends on several factors. The most significant ones are the power of the adversary and the strategy used for partitioning the training data. Obviously, if the adversary can tamper with a very large number of points, there is no way for an ML model to learn any meaningful correlation. On the other hand, if the adversary can only

compromise a limited portion of the data, then the way training data is partitioned is key to ensure that compromised data does not adversely affect the behavior of the model and that a sufficient number of models are kept pristine.

4.2.2.1 Ensemble Member Generation and Combination Rule

When designing an ensemble-based system, three interrelated aspects need to be considered: (*i*) the partitioning method used to select the training data for each base model, (*ii*) the specific procedure used for generating the ensemble members, and (*iii*) the combination rule for obtaining the ensemble decision. Our technique for partitioning the training data based on robustness criteria will be described in Section 4.2.3, while details on our choices regarding the other two aspects are provided below.

In a sense, ensemble-based systems can be seen as *algorithm-free-algorithms*, meaning that they are generally independent of the type of base learner used to create the ensemble. This feature is of great benefit as it allows complete discretion in using a specific 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. In our framework, we use the homogeneous ensemble approach, which involves using the same base learner multiple times to generate the family of learners. In this case, therefore, we do not introduce diversity at model level, but at data level used by each model. Once the base models have been trained, it is necessary to determine how to aggregate their individual outputs in order to obtain a single final prediction. In the literature, there exist several different approaches for model combination [125]. The most commonly used in practice are the *linear combiner*, the *product combiner*, and the *voting combiner*. Here, we adopt the third type of combination rule as it is known to exhibit consistently good behavior in many applications and is amenable when models output class labels.

Although there are different flavors of the voting combination method, the most popular one (known as *hard voting* or *majority voting*) requires every individual classifier to vote for a class, then the class that received the most votes by the

classifiers is chosen as the ensemble prediction (in statistical terms, the ensemble decision corresponds to the mode of the distribution of individually predicted labels). Assume that N is the number of classifiers in the ensemble and L is the number of classes. The decision of the t^{th} classifier (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 calculated as

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

Interestingly, the voting-based rule can be proven to be an optimal approach for combining classifiers, if they are independent [126]. In fact, under the minor assumptions of: 1) N is odd, and 2) each classifier has probability p of correctly classifying a given instance, the ensemble predicts the correct (uncorrupted) label if at least $\lfloor N/2 \rfloor + 1$ base classifiers output the correct label.

4.2.2.2 The Accuracy-Diversity Breakdown

It turned out that diversity is one of the essential properties governing how well an ensemble can perform. Frequently, the term *accuracy-diversity* breakdown [127, 128] is used to express the fact that an ensemble error is primarily impacted by two distinct elements, namely the accuracy of the individual models and their interaction once combined. As stated by Dietterich [129], an ensemble of learners succeeds in achieving better accuracy if and only if its individual members are accurate and diverse. However, the exact relationship between diversity and accuracy still remains an open research issue, in part due to the fact that there is no consensus in the community regarding the definition or measurement for diversity (which basically depends on the problem to be solved) [130, 131], as opposed to accuracy. On one hand, if each base learner makes the same errors for new instances, there is no utility in combining their outputs. On the other hand, if the 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. In the case of regression ensembles with linear combiners, error decomposition schemes such as the *bias-variance-covariance decomposition* [132] and the *ambiguity decomposition* [133] are commonly used to split the error into an accuracy term and a diversity term. For instance, with the latter decomposition scheme, the squared error of a linearly combined ensemble is broken into the sum of two components quantifying the average squared error of the individual models and the ambiguity level, respectively. Since the ambiguity term (which expresses the interactions between the predictions) is guaranteed to always be positive, for an arbitrary data point, the ensemble squared error invariably turns out to be less than or equal to the average of the individual squared errors. However, this type of analysis is only suitable for regression tasks with quadratic loss. In the case of classification ensembles, there is a partial theory that relates the classifier correlation to the ensemble error rate [134], but the results hold only when a linear combiner classification ensemble is applied. Conversely, the case of a classification problem with a majority vote combiner is more challenging, and no single accepted theoretical framework capturing the accuracy-diversity breakdown currently exists for it [135]. In spite of this, simple intuition given by a Binomial experiment can prove that correlation between models does indeed affect performance. In practice, if we have N different base classifiers each with identical error probability $p = P(h_t(x) \neq y)$, assuming their errors are statistically independent, the following error probability of the majority voting ensemble holds:

$$P(H(x) \neq y) = \sum_{k>(N/2)}^{N} \binom{N}{k} p^k (1-p)^{(N\ k)} \tag{6}$$

Nevertheless, if the decisions of the base classifiers are dependent, we lack a series of independent Bernoulli trials, and thus we have no guarantee on the amount of the error reduction when a classification ensemble with a voting combiner is employed.

As for robustness concerns, besides benefits shown by experimental results of the many papers we mentioned in Section 2.3.2.2, Levine and Feizi [97] 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:

$$t(x) = \lfloor \frac{n_{g(x)} - \max_{c \neq (x)}(c_c(x))}{2} \rfloor \tag{7}$$

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

The above 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 (7) highlights a crucial trade-off between robustness and accuracy rooted on the number k of partitions. Actually, 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 datasets such as MNIST [12], to exhibit the robustness of their strategy. By contrast, we will employ a dozen partitions for the same dataset to provide a proper balance between robustness and accuracy (Chapter 5).

4.2.3 Anti-clustering for Training Set Partitioning

As discussed earlier, the concept of diversity plays an important role in the ML process. Specifically, diversity of the training data ensures they can provide more discriminatory information, thereby resulting in improved performance of the learned model [136]. Besides increasing the model's representational ability, data diversification schemes can also be exploited for defensive purposes in the context of adversarial learning. The rationale behind the use of the ensemble-based defense strategy is that an attacker is required to expend a lot of effort to compromise the system as he must fool multiple models in the ensemble instead of just a single one. By using diverse data to train each individual model, the models' errors on the test set are more likely to be statistically independent of each other and this

allows for improved robustness to attacks.

4.2.3.1 Data Diversification

Starting from the observation that by simply reversing the logic behind the popular k-means clustering method it is possible to generate partitions that closely resemble each other, Späth [137] and Valev [138, 139] 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:

$$\bigcup_{i=1}^{k} P_i = T \tag{8}$$

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

Condition (8) ensures that every element in T is assigned to at least one of the anti-clusters. On the other hand, condition (9) requires pairwise disjoint anti-clusters. This means that no pair of two anti-clusters can contain the same element.
It is worth noting that having partitions of equal size is not mandatory for anti-clustering methods [140]. However, obtaining comparable sized partitions is a desirable property, and in classical graph-partitioning based clustering this practice is a standard constraint that avoids trivial partitions [141, 142]. It is well understood that 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 [143]. For these reasons, we give the following additional restriction:

$$\mid P_i \mid = \mid P_k \mid, \forall i, k \in \{1, ...k\} \tag{10}$$

From condition (10) 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.

4.2.3.2 Dissimilarity Measure and Objective Function

As previously mentioned, anti-clustering partitioning possesses in itself the ability to ensure that points close to each other are put into different partitions. In order to take advantage of this inherent characteristic for our defensive purposes, we need to choose a specific objective criterion for anti-clustering based on information in a dissimilarity matrix that accounts for the relevance of the points.

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*

$$f(\pi^*) = max\{f(\pi) \mid \forall \pi \in \Pi\} \tag{11}$$

As for classical clustering methods, there is a wide range of criteria for anti-clustering as well [144]. The most intuitive criterion, which, as mentioned earlier, is the one that gave rise to the concept of anti-clustering, is based on the direct reversal of the k-means clustering logic [145]. Thus, 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. For the purpose of illustration, Figure 6 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 anti-clustering partitioning we adopt another type of criterion: *anti-cluster editing*, which is the reverse of the cluster editing clustering paradigm [146].

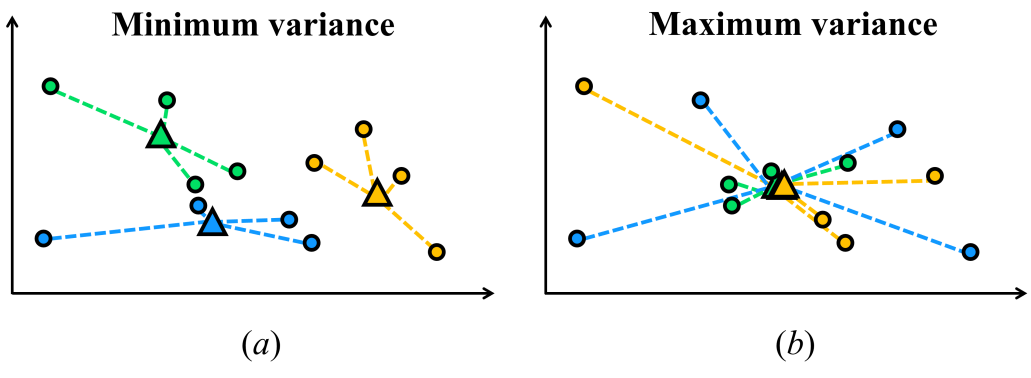


Figure 6: k-means objectives. In (a) we see that k-means clustering minimizes the variance within partitions. The logic is reversed in (b), where anti-clustering k-means maximizes the variance within partitions.

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:

$$D_{diversity} = \Sigma_{1 \leq i \leq j \leq n} d_{ij} x_{ij} \tag{12}$$

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

$$x_{ij} = \begin{cases} 1, & \text{if } x_i \in P_k \wedge x_j \in P_k \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

Figure 7 shows the different assignment of points according to whether the objective function minimizes (cluster editing) or maximizes (anti-cluster editing) the diversity. Differently from Figure 6, 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 Figure 7 the lines are drawn between pairs of points within the same cluster. This reflects the different definitions of within-cluster heterogeneity used by the two criteria.

In Eq. (12) the dissimilarity measure d_{ij} usually corresponds to the Euclidean

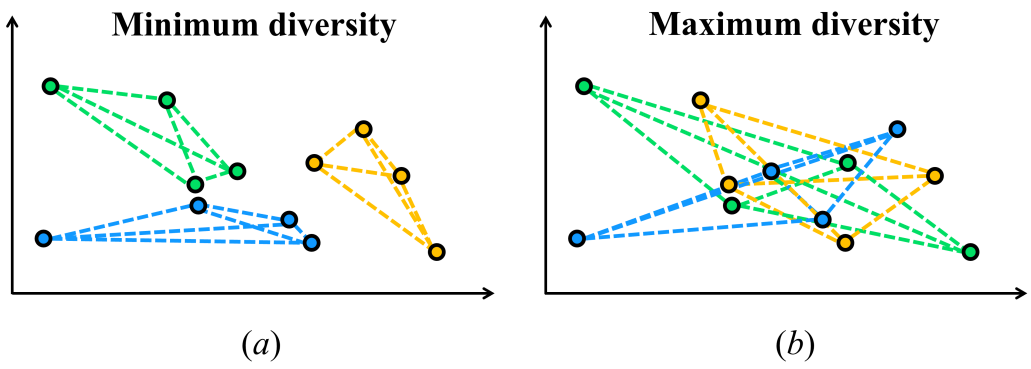


Figure 7: Cluster editing objectives. (a) Cluster editing minimizes the diversity objective (sum of pairwise distances within each cluster), whereas (b) anti-cluster editing maximizes it.

distance (or squared Euclidean distance), though, in principle, any theoretically sound 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 4.2.1) associated with each point. We assume that an $N \times N$ non-negative symmetric matrix $D = [d_{ij}]$ is used to represent pairwise dissimilarity measurements, whose entries are computed as follows:

$$d_{ij} = \frac{\parallel i - j \parallel}{max(r_i, r_j)} \tag{14}$$

$\parallel i-j \parallel$ 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.

The effectiveness of the partitioning strategy clearly depends on the notion of distance between two points. Here, we rescale the Euclidean distance so as to take into account the closeness of the points to the separator hyperplane. The general criterion is: two points are closer the higher their risk index. Thus, a simple distortion can be induced via Eq. (14).

Chapter 5

Experimental Evaluation

In this chapter, we report empirical results evaluating the performance of our defense technique against label-flipping poisoning attacks under the assumption of symmetric lack of information between the attacker and the defender. The experimental results show that our method is more effective compared to both an ensemble method based on random partitioning and to a traditional model trained on the whole training set for moderate poisoning rates.

5.1 MNIST Benchmark

We evaluated the effectiveness of our proposed method on a benchmark for handwritten digit recognition, the MNIST [12] dataset, which contains a collection of 70000 images of handwritten digits from 0 to 9, where the training set and the test set include 60000 and 10000 samples, respectively. This dataset is considered a typical benchmark for classification algorithms because of its high variability of available 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 rescaling them to the range $[0, 1]$. In line with most of the existing literature in adversarial attacks, here we considered a binary classification problem, where the task is to distinguish between digits 1 and 7. The result is a total of 15170 images of dimension 784, with 13007 images in the training set and 2163 in the test set.

5.2 Heuristic

Finding a partitioning that maximizes the diversity criteria is computationally challenging for $k \geq 2$ and an arbitrary M-dimensional Euclidean space problem. Especially when the number of elements is very 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 [140], which is an adaptation of the procedure defined in [147]. 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.

5.3 Practical Implementation Details

Regarding the learners, as mentioned earlier, the advantage of our defense technique is that it is not intended for a specific ML model. Thus, on the defender side we opted for deep neural networks with no great sophistication as the base learners for the ensemble. In addition, we used the reference SVM model for identifying the risk levels on the part of the defender and the points to attack on the part of the adversary. The SVM is a standard two-classes linear separator that have been implemented via the CRANE library of Python. The training of those learners has been carried out in TensorFlow-Keras.

Metrics. Our evaluation metric is distilled into two components. On one hand, we use *clean accuracy* calculated on testing data to evaluate how models perform on pristine data. On the other hand, the second metric – *after poisoning accuracy* – describes how models perform on label-flipped data with varying poisoning rate. This allows us to investigate how the algorithm performs against

different corruption ratio. In addition to these two, we also use an additional metric – *certified accuracy* – again evaluated on the test set, 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. *Ensemble gap* is 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.

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

Attacker's budget. We used the label-flipping strategy described in Algorithm 2 (see Section 4.1.2), with the attack performed monodirectionally, deciding to change the 1 labels to 7. We assessed the efficacy of our strategy for a flipping rate 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 which proves effective is between support vectors and non support vectors (see Section 4.1.2).

5.4 Shared Knowledge

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 Figure 8, 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.

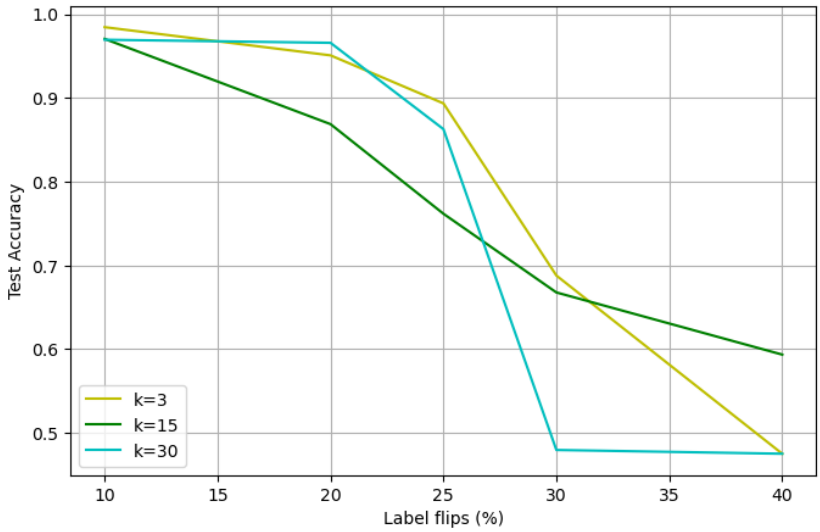


Figure 8: Accuracy curves for percentage of label-flipping ranging from 10% to 40% and $k \in \{3, 15, 30\}$ partitions.

5.5 Experimental Results and Analysis

The approach most similar to the one we propose is that by Levine and Feizi [97], which uses an ensemble-based method where the partitioning of the training set into disjoint subsets is performed via a hash function, so data are randomly shuffled among partitions based on their hash. In all experiments, we compared the performance of our ensemble-based defense (where partitioning is performed according to our anti-clustering technique) – hereafter referred to as *anticl* – with that of an ensemble method using a randomized partitioning mimicking the effect of hash-based partitions selection [97]. Specifically, in the technique we will henceforth refer to as *classic*, the training set is partitioned randomly, without

repetition, and regardless of the risk levels identified by means of Algorithm 1. Also, we denote by *mon*, (short for "monolithic"), the model trained on the whole training set with the traditional ensemble-less architecture. 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.

5.5.1 Absence of an Attack

First of all, Table 1 clearly 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 rather obvious, since, in the absence of local minima, learning based on the whole dataset is more efficient than composing multiple learners on partial training sets, where the huge dimensionality of the training points prevents the local minima mentioned above. The second inequality confirms that under the above conditions the random partitioning underlying *classic_clean*, defeats any "smarter" partitioning strategy (the value of probabilistic strategies). However, as Table 1 shows, the difference in accuracy is quite shallow. Although the *classic* technique exhibits slightly higher accuracy than the *anticl* one, and thus closer to that of *mon*, taking additional factors into account when performing partitioning resulted in only a slight overall degradation in accuracy when no attack is performed.

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

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.

5.5.2 Under Attack

In order to evaluate worst-case behavior, we started the experiments by checking which would be the most efficient strategy on the part of the attacker.

Attack Strategy Selection. The graph in Figure 9 shows a natural order for the experiments where:

a) We monodirectionally flip the labels of 270 images belonging to risk levels $1-10$, with probabilities calculated according to (4);

b) Then, we monodirectionally flip the labels of 215 images belonging to the risk level 0 (lying far from the linear separator);

c) Again, we monodirectionally flip 485 images, but this time all belonging to risk levels $1-10$, with probabilities calculated according to (4).

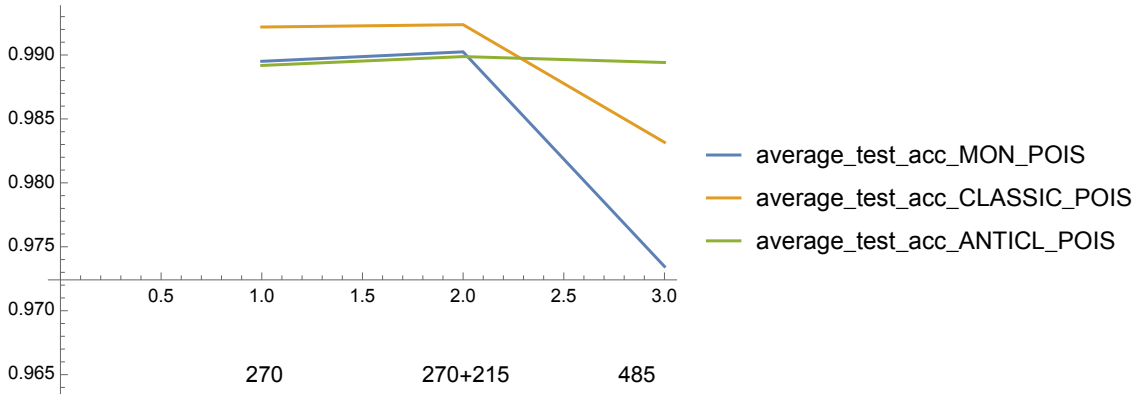


Figure 9: Accuracy trend with increased number of flips for both bordering (risk levels $1-10$) and inner data points (risk level 0).

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*), now for low flipping rates we have

$$classic\_pois > mon\_pois > anticl\_pois$$

whereas for higher rates of poisoning, *anticl* turns out to be superior to the others:

$$anticl\_pois > classic\_pois > mon\_pois$$

From this it follows that 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.

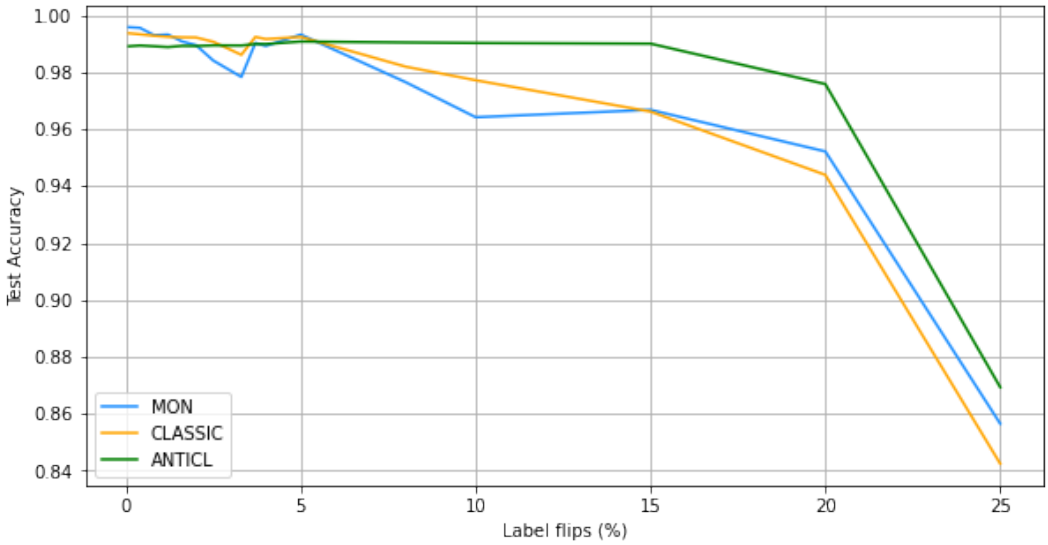


Figure 10: Accuracy comparison for $k = 15$ partitions and poisoning rate ranging from 0.08% to 25% of the training set size.

To confirm the above observations, first we carried out an experiment where all the flip percentages are less than 3.5%, and all the poisoned points are selected from risk levels $1 - 10$. In particular, percentages range from 0.8% to 3.3% of the training set. This corresponds to percentages from 2% to 81% of the points having risk levels $1 - 10$, with a number of flipped points ranging from 11 to 431. In a second

experiment, we considered higher percentages of poisoning, i.e., between 10% and 25%. Clearly, in this second case, we adopted the strategy of monodirectionally flipping 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.

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

Table 2: Results 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 $0-10$).

As we can observe from the graph in Figure 10, the results confirms 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.

Effect of large percentages of poisoned data. Although in the literature on poisoning attacks the budget of the attacker, in terms of the number of points he is able to modify, almost never exceeds $20-25\%$ of the size of the training set [29], in a further experiment we assessed the degree of degradation reached by the different methods with percentages equal to 30% and 40% 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 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.

5.5.2.1 Certified Accuracy

Certified accuracy is 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 [103]. Here, we consider the definition we provided in Section 5.3, i.e., a measure that, for a given poisoning rate, certifies whether the predicted label stays unchanged or not for each testing input. In Table 3 we report the results up to the poisoning percentage of 40% also in terms of the actual number of points

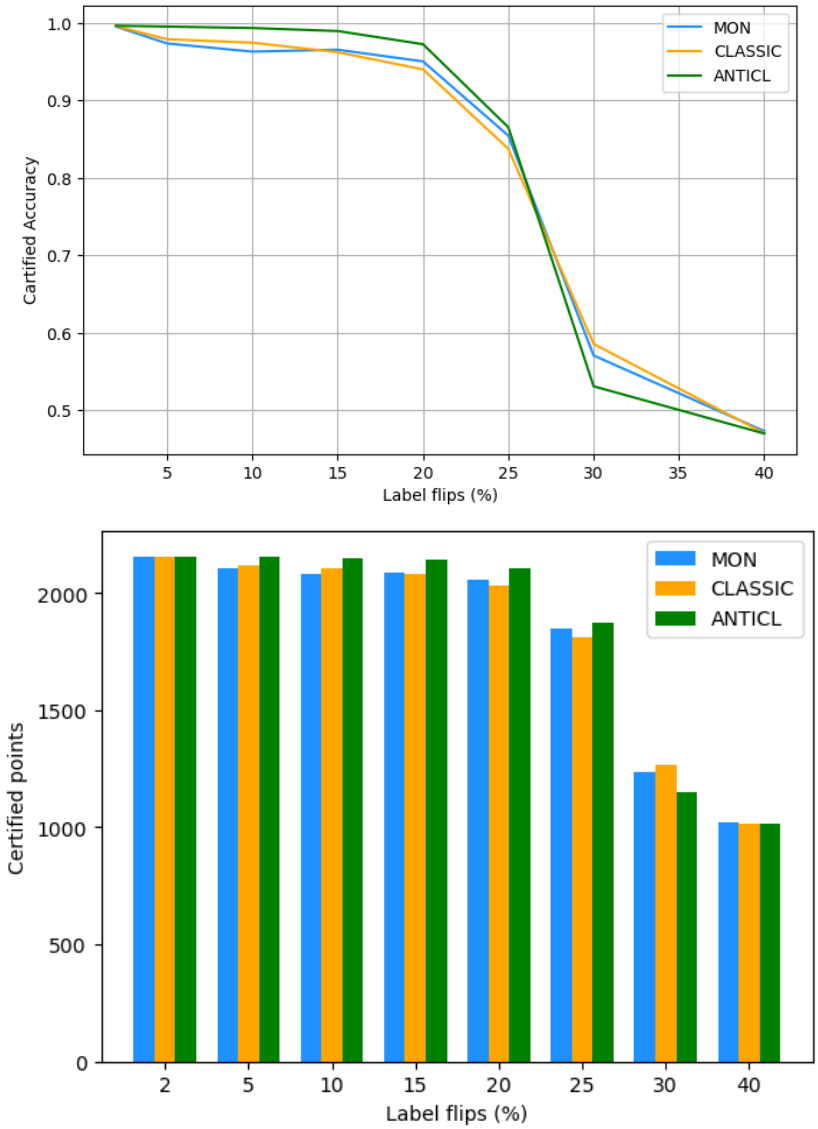


Figure 11: Certified accuracy comparison, with $k = 15$ partitions for *anticl* and *classic* methods. The figure on the top shows the certified accuracy to label-flipping poisoning attacks. The figure on the bottom shows how the corresponding number of certified points changes as the flip percentage increases.

in the test set that did not change their labels after the label-flipping attack. The corresponding graphs are given in Figure 11. We note a benefit similar to that observed with 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, where training corruption leads to essentially random classifications.

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

Table 3: Certified accuracy, 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.

5.5.2.2 Ensemble Gap

As mentioned earlier, 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 (7). 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.

However, a smaller gap with the poisoned training set may again denote a superiority of our method, which raises more contested voters when the majority result is wrong. In fact, 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 Figure 12).

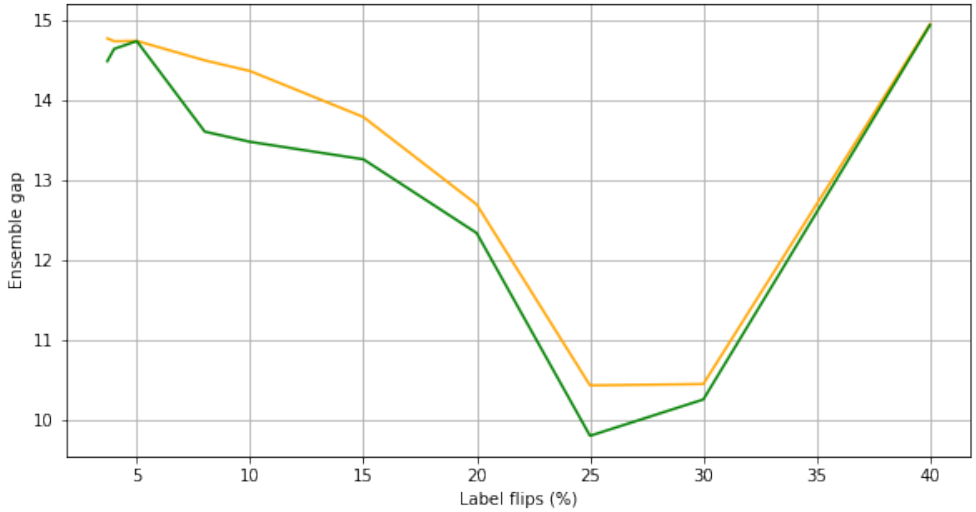


Figure 12: Gap comparison between the *anticl* and *classic* ensemble methods.

5.5.3 Discussion

Our results come from a double accuracy vs. robustness trade-off (with regard to both the number of partitions and the poisoning rate), where, in order to ensure good performance with high, though informational feasible, poisoning of the training set labels, we sacrifice some accuracy in the near-unpoisoned conditions. Note that this is a common effect of introducing relatively random noise into algorithms. This occurs, for instance, with randomized smoothing algorithms [88], where pure noise is introduced to make the algorithm more robust so as to learn the *substance* of the classification logic and avoid pursuing details which induce over-fitting.

Part II

Auxiliary Techniques

Chapter 6

Instantiating Ensemble Parameters Based on Model Degradation Index

This chapter outlines a method for assessing the degradation of an ML model using held-out data which is computed combining training set points in relation to their proximity to the separation surfaces identified with a reference model (specifically, a Convex Hull classifier).

6.1 Introduction and Background

In the previous Chapters, we discussed how training data can play a crucial role in shaping the learning process of ML models and determining the quality of decision making [148]. In Chapter 4, we focused on alleviating the consequences of training set poisoning. In this Chapter, following [149], we focus on a distinct though related topic, i.e. how to estimate the degradation of models. This is a frequent scenario in practical environments - for example, in the Internet of Things, where faulty or compromised sensors may lower the overall quality of the data on which inference is based [150]. Being able to quantify ML model degradation (which can be due to the injection of spurious, adversarial or low quality data in

periodic re-training) provides context knowledge useful to decide whether to deploy the robustness-aimed measures based on ensemble, and even to set the parameters of such measures, including the number of training partitions, for subsequent re-training. We start from the notion that all ML data assets, including the training set, can be considered artefacts with a certain quality. In the context of Artificial Intelligence (AI), it is possible to consider data quality as a set of properties the information fed to (and produced by) AI-ML models should have [151]:

- ◇ *Accuracy*: Reflects the difference between the data and an underlying "true value". This difference includes sensor errors and any error introduced at ingestion (e.g., by quantization).
- ◇ *Consistency*: Expresses compliance of data to user-defined rules; for example, the age of a sibling cannot be equal or greater than the one of a parent.
- ◇ *Timeliness*: Expresses difference between the time when data is made available and a deadline. It denotes if data is being received in time for performing a task.
- ◇ *Completeness*: Expresses whether the number of data points available is enough for the intended purpose (e.g., computing a formula or training an ML model).

For the purposes of this Chapter, we focus on two aspects of ML data assets' quality: data *uncertainty* and *trustworthiness*, which can be seen as two different aspects of *indeterminacy* (how much it is known about the consequences of using a data item). Uncertainty is mainly related to imprecision affecting raw data, while indeterminacy is an inherent issue for pre-processed data assets (for example, training sets) that are targeted by modifications aimed at changing the ML model behavior in production. In all cases, spurious items (adversarial, poisonous or simply low-quality data points) that become part of data assets can impair ML model accuracy, requiring filtering or other alleviation measures.

6.1.1 Model Degradation

Even if there is no tampering with data assets, ML models' predictive performance can decrease over time as they are fed with new data. This phenomenon is known as *model degradation*. There are two major heuristics for identifying and tracking model degradation:

- ◇ *Explicit quantification*: Conceptually, the simplest way to identify model degradation is to explicitly quantify the amount of decrease in model performance or its trend. However, measuring the accuracy of a deployed ML model on live input data is a notoriously difficult problem. This difficulty arises because for each input one needs access to both the model's output, i.e. the classification or prediction value proposed by the model, and the ground truth, i.e. the true value. Even when the model's outputs and the ground truth values are both available, it may be difficult to synchronize them to check the performance decrease's trend. Consider an ML model that predicts the total amount of rain that will fall in a month. The actual value will only be observed monthly, so appreciation of the rate at which the model's performance is decreasing may involve considerable delay.

- ◇ *Distribution comparison*: These techniques compare the probability distributions of the input features to those of the training data to infer model degradation. This *a priori* technique can be helpful when it is not possible to observe the ground-truth values. This method can be improved by issuing alerts when the divergence between key features distribution is significant. However, in production it is not always possible to continuously monitor all features' distributions.

The remainder of the Chapter is organized as follows: Section 6.2 describes a technique for estimating the severity of AI-ML model performance degradation due to spurious additions to training sets, computing an *held-out data set* (Section 6.2.1) to be used as a "gold standard" for the data assets (training and validation data) used for ML training. Section 6.3 discusses how *Convex Hulls* (CHs) can be

used to approximate class regions of classification models and introduces our CH-based degradation severity index for ML data assets, while Section 6.4 explains how to compute it. Section 6.5 reports the experimental evaluation of our approach on the *Belgium Traffic Sign Classification Benchmark* (BTSC). Finally, Section 6.6 draws our conclusions.

6.2 Estimating Severity of AI-ML Models' Data Assets Degradation

ML model degradation is noticed when the accuracy in production deviates sensibly with respect to the accuracy measured in validation. Our approach allows a fast estimate of the severity of models' performance degradation due to spurious additions to training sets. It is designed to be ML model-independent and computationally lightweight. The approach consists of two major components: an *held-out data set generator* and a *convex-hull engine*. Our notion of held-out data set should not be confused with the *simple hold-out* [152] technique used for ML model validation, where some manually labeled data kept aside from the training set are used for model validation purposes. Rather, we take a validation-agnostic point of view: ML model training can use either *simple hold-out* or, more probably, *cross-validation* alternating training and test data.

Whatever the validation technique, checking the performance of ML models against an additional *held-out data set* has become customary when models are trained using uncertain data, i.e. data that may or may not be representative of the data space at deployment [153]. Our approach is to start from the training set to generate an held-out data set that can be used as a "gold standard" [153] for quick degradation assessment. Our convex hull engine comes in at this point: we use changes in convex hull of the classes computed by the ML model under assessment on the held-out data set to estimate the severity of the model's degradation. It is important to remark that if the training set is later added to, our technique allows to fuse successive severity estimates, in the line of data uncertainty measures based on Dempster-Shafer theory of evidence [151]. In the ML lifecycle, held-out data

sets are mostly used *during* the training phase for *early stopping*, i.e. stopping training when the error on the held-out data set increases, in order to prevent model over-fitting. For the purposes of this thesis, we refrain from interactive use of the held-out data set, as the error on the held-out data set may fluctuate during training. We use held-out data only *after* training, in order to provide a quality benchmark for trained models.

6.2.1 Using Latent Variables to build the Held-out Data Set

In principle, the held-out data set can be built by manually selecting labeled data points that are considered certain (e.g., because they come from trusted sources or were measured at a time where no spurious data injection was possible). Also, held-out data may be composed of hand-picked data points whose classification is going to be verified during ML model audit or certification at the request of an auditing authority [154]. However, manual selection does not scale well, and cannot easily generate held-out sets comparable to the training set size. In the following, we discuss how, for the purpose of quality assessment, the held-out data set can also be built in an unsupervised way, modeling training data as *observables*, or *manifest variables*, and extracting held-out data as *latent variables*, which are not directly observable.

Latent class models were first introduced by Lazarsfeld in the Fifties of the last century [155] and were later made computationally efficient by Goodman [156]. Often, the observed variables are modeled as classifications assigned by different judges. We follow the idea of using latent class models to correct symmetric disagreements that appear to result from multi-rater bias [157]. The latent class model underlies various unsupervised ML techniques, starting from the seminal Auto Class model [158]. We propose an unsupervised latent class model to build the held-out data set for the degradation assessment of AI-ML data assets.

Let us call D the ML data asset whose degradation we want to assess. We define a *window of observation* W of k-dimensional (complete or partial) data points taken from D. The scope of this window is decided heuristically, according

to temporal locality (e.g., the data points may be repeated sensor readings) or to spatial locality (e.g., the data points may be subset of pixels sampled from a set of images showing the same object).

In order to distillate a single "golden" data point out of W, we consider k latent variables, one for each component of the data points in W. We apply the classic naive Bayes approach [156]: we assume the values of data in W (the observed variables) to be all conditionally independent of one another, given the value of each latent variable[1]. Let H, K, J, and L be the observed variables, i.e. the components of the data points in W, and let X_H, X_K,X_J and X_L be the corresponding latent variables, i.e. the components of the "golden" data point we want to compute. For each latent variable X_I ($I \in \{H, K, J, L\}$), our latent model is:

$$p(i_1, i_2, ..., i_k, x_i) = p(i_1|x)p(i_2|x)...p(i_k|x)p(x_i) \tag{15}$$

where the equality holds by our assumption of observable variables' independence. The estimate of the conditional probabilities in Eq. (15) can be carried out by replacing the probabilities on the right-hand side by the corresponding estimates.

Several competing techniques (including *max-likelihood*) are available for computing such an estimate. Recent research [159] has proposed the *Laplacian rule of succession* as an alternative method to estimate the conditional probabilities of the candidate x values. The classic Laplacian succession rule [160] provides a setting where, repeating n times an experiment that can result in a success or failure, and getting s successes, and $n - s$ failures, we can estimate the probability of success of the next time. In our setting, "success" corresponds to $x = i_j$. For the sake of computational speed, we use a pre-computed Laplace distribution [161], and estimate each $p(i_j|x)$ as the probability associated to i_j by the Laplace distribution centred in x (see Figure 13).

[1]This assumption may of course require filtering out highly correlated features. This is a customary practice in data pre-processing, and lies outside the scope of this dissertation.

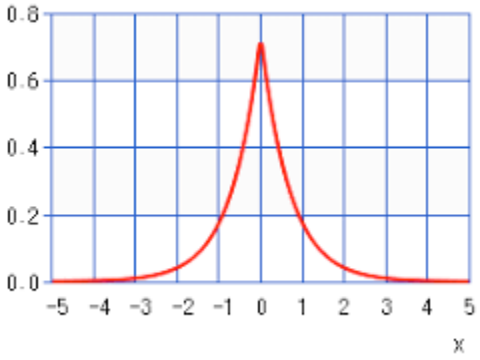


Figure 13: A differential Laplacian distribution. Values on the x-axis are the difference between x and i_j.

Once estimates of $p(i_j|x)$ have been computed for all possible values of x in the representation interval, the most probable latent value is assigned to x. This way, the values chosen to build the held-out data set are the ones for which the agreement among the observable variables within the observation window W is highest.

i_1	i_2	i_3
1	2	0
1	1	2
2	2	1
0	0	0

Table 4: Three observable variables.

As a simple example (the complete computation carried out for our experiments is described in Section 6.5.1), let us consider three integer variables (i.e., $k = 3$) taking values in the interval $[0 - 2]$ and an observation window W containing 4 tri-dimensional points. For each possible value of the latent variable x (one of the 3 components of the golden data point), we compute $p(x)$ and estimate $p(i_j|x)$ by querying the Laplacian distribution centered in x (we obtain $L(i_j, x)$). Let us now estimate x_{i_1} using Eq. 15. Looking at the first column of Table 4, for $x_{i_1} = 0$ we get $p(x_{i_1}) = 0.25$, to be multiplied by $L(1,0)L(1,0)L(2,0)L(0,0)$. For $x_{i_1} = 1$ we get $p(x_{i_1}) = 0.5$, to be multiplied by $L(1,1)L(1,1)L(2,1)L(0,1)$.

For $x_{i_1} = 2$ we get $p(x_{i_1}) = 0.25$, to be multiplied by $L(1,2)L(1,2)L(2,2)L(0,2)$. By comparison, the golden value for the first latent variable is $x_{i_1} = 1$. The computation can be repeated for x_{i_2} and x_{i_3}, obtaining the "golden" tri-dimensional vector $GV = (1, 2, 0)$.

6.3 A Degradation Severity Index based on Classes' Convex Hulls

Most classification models make use of a distance metric in some n-dimensional space to compute separation surfaces between classes. Generally speaking, poisoning attacks try to deform such surfaces at training time, in order to "push" future data points across them at run-time. Intuitively, the severity of an adversarial input to an ML model is linked to the attack's effectiveness in achieving such deformation; but if the structure of the ML model being attacked is unknown, it is difficulty to quantify it *a priori*. To achieve model independence for our severity index, we look at the classes computed on the held-out data set by a virtual *Nearest Convex-Hull* (NCH) classifier, which plays a similar role to the SVM reference classifier we introduced in Chapter 3.

6.3.1 The Reference Model

Computational geometry models have been used since long to approximate class regions of data classification models. In geometric terms, we can represent each class by the smallest convex region enclosing all points in the class. Such an envelope corresponds to the so-called *Convex Hull* (CH). Formally, the CH of a set of points S in a n-dimensional space is the smallest convex set that contains S. A point in S is an *extreme* point (with respect to S) if it is a vertex of the convex hull of S. If S is finite then the convex hull of S is a convex polytope with vertices, edges and facets making up its boundary. Figure 14 shows the convex hull of a set of bi-dimensional points.

The "hard-margin" version of the NCH classifier model assigns a data point

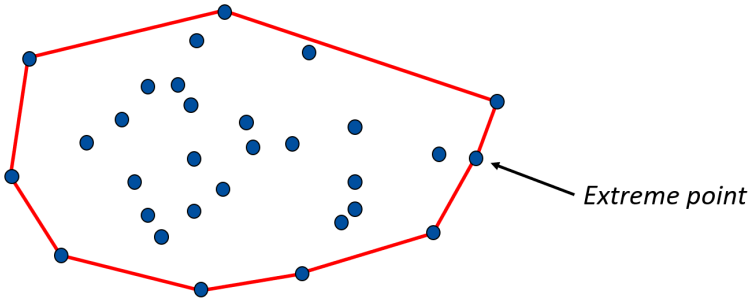


Figure 14: The convex hull of a set of bi-dimensional points.

x to the group of training points whose CH is closest to x. Performing NCH classification involves solving an optimization problem to find the distance of the input object to each class. Starting from the Eighties, several algorithms for doing so have been proposed under the general heading of finding the minimum distance between convex sets [162]. The NCH model has the property that the extent of proximity of a test point x to a given class is determined without taking into consideration objects from other classes. In classic separable cases, however, a problem arises if the object x to be classified lies inside the CHs of two or more classes, since its distance to these CHs is equal to zero, leaving the classification of x undetermined. This problem was solved by introducing the *soft-margin* version of the NCH classifier [163], where overlap between a data point x and a given class C's CH is penalized linearly or polynomially.

6.3.2 CH Deformation

For the sake of speed, we do not directly apply NCH classification to compute our severity metrics. Rather, we compute the deformation of the CHs of the classes computed by the ML model M under evaluation on an held-out data set derived from M's training set. Intuition suggests that for each point p to be classified, the best choice for an attacker would be to cut the CH point that is closest to p. Figure 15 shows the effect of cutting a CH point: the closest CH to the candidate point x, originally the one depicted in red (upper part of Figure 15), becomes the one depicted in green after one point of the red class' CH has been removed

(lower part of Figure 15). This cut corresponds to a CH deformation, as stated in Theorem below.

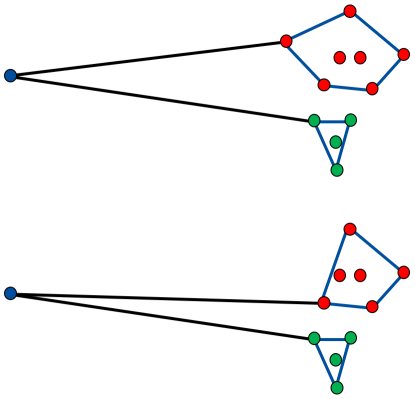


Figure 15: The link between classes' CH deformation and NCH classification accuracy.

Theorem. *For any point p outside the CH of a class C, the point $c \in C$ closest to p lies on the CH of C.*

Proof. Without loss of generality, let us consider p as the origin of a system of coordinates, and let us fix the direction of the x-axis along the line connecting p and c. Then, c is the point of C with the smallest x coordinate. Therefore c lies on the CH of C. □

In the following, we will assess the severity of the degradation of a generic ML model M via the deformation of the CHs of the classes computed by M on a held-out data set.

Definition. *Given a model M, let $CH(C)$ be the convex hull of a class C of the VNCH model corresponding to M. If M is modified by a change in a data asset, resulting in a model M', we define the severity S of the degradation as follows:*

$$S = \frac{| CH(C) \cup CH(C') |}{| CH(C) \cup CH(C') | + | CH(C) \cap CH(C') |} \tag{16}$$

where $|CH|$ denotes the area within the convex hull CH and C' is computed by the VNCH model corresponding to M' on the same set of data points belonging to model M's class C.

Of course $C' \neq C$, as different data assets were used for training M'. Specifically, C' may miss some points that were in C and include some points that were not in C. Note that, when no point of $CH(C)$ is missing in C', and any additional point of C' fall inside $CH(C)$ (for example, when $C = C'$) $S = 1/2$. Our severity index S is modeled after the Dempster-Schafer measures used in data quality [151]. Indeed, such measures are able to represent incomplete knowledge, and update it as new information comes. We looked for the simplest representation that was at the same time easy to compute and provided the following properties:

- ◇ *Representation interval*: Our severity index S takes values inside the interval $[0, 1]$, which helps comparing it to individual data assets' quality measures.
- ◇ *Weak Monotonicity*: S is weakly monotonic with respect to random points' addition. Given two sets D and D' of random data points with $D \subseteq D'$, $S(D) \leq S(D)$. This property is also held by most data quality indexes [151].

6.4 Computing the S index

There exist several methods for computing the CH of a set of finite number of points in a Euclidean space. Discussions on which is the best performing CH algorithm are outside the scope of this thesis. We used Quickhull [164], which is able to compute the convex hull in 2D, 3D and higher dimensions with the average time complexity of $O(nlogn)$. The recursive nature of the Quickhull algorithm allows a fast implementation and proves efficient in practice, though there are also variants of the algorithm that can make it run much faster[2].

[2]When needed, it is possible to approximate the CH of the union with the union of the CHs, and the CH of the intersection with the intersection of the CHs. This yields $O(|CH|)$ complexity, where usually $|CH| << n$.

6.4.1 Aggregation

Once class-based S indexes have been computed, they need to be aggregated to provide the degradation metrics for the entire model. Several aggregation techniques for uncertainty measures have been proposed [165], which can be briefly summarized as follows:

- ◇ *Averaging*: This technique aggregates numerical data points by taking their average or median.
- ◇ *Bayesian Systems*. This technique takes data points as samples and computes their *typical value* by means of a conditional Probability Density Function (PDF). The Bayesian approach is widely used in recommender systems [166]. It also underlies the held-out data set generation proposed in this dissertation.
- ◇ *Belief Models*: In belief-based aggregation, data points are represented as beliefs and some technique (usually, *max-likelihood*) is used to select the most probable one. As the sum of beliefs over all possible data points does not necessarily add up to 1, belief values are easier to generate and process than probabilities.
- ◇ *Fuzzy arithmetics*: Fuzzy numbers can be used to represent uncertain values; in this case, aggregation is computed as fuzzy average.
- ◇ *Flow Models*: Data points are modeled as paths within probabilistic processes like Markov chains. Aggregation is performed by taking the most probable.

In order to estimate the overall level of degradation of the classification model M, we need to compute an aggregation putting together the degradation indexes S_{C_i} associated to M's classes $C_1, C_2, ..., C_n$. We follow the literature [166] in using the classic *Ordered Weighted Averaging* operator [167], where the order takes into account the importance of each source. The overall severity index S_M is computed

as follows:

$$S_M = \frac{1}{n}\sum_{i=1}^{n} \omega_i S_{C_i} \tag{17}$$

where $\omega_i = \frac{n-i-1}{n+1}$ and n is the number of classes.

6.5 Experimental Evaluation

We carried out an experimental evaluation of our approach on the *Belgium Traffic Sign Classification Benchmark* (BTSC) dataset [13], using a Convolutional Neural Network (CNN). BTSC is part of the widely adopted Belgium Traffic Sign Dataset, which features multiple, calibrated images of traffic signs. The subset we used for our experiments originally contains 62 classes of traffic signs with 4591 images in the training set and 2534 images in the test set. On average for each physically distinct traffic sign three images are available with different position/orientation, scale and illumination. The actual data set we used to train the CNN is a gray-scale remapping of the original BTSC data. Based on both shape and color features and types of message they communicate, we split BTSC into five macro classes of traffic signs, namely *warning*, *mandatory*, *stop and yield*, *prohibitory* and *information*. Table 5 summarizes the specification of the data set used for training according to our remapping, whereas our manual prioritization of the five classes is given in Table 6.

Class	Type	Shape	Color
C_1	Warning	Triangular (pointing upward)	Red
C_2	Mandatory	Circular	Blue
C_3	Stop and Yield	Octagonal or Triangular (pointing downward)	Red
C_4	Prohibitory	Circular	Red
C_5	Information	Square or Rectangular or Diamond	Blue or Red

Table 5: Summary information of traffic sign data set.

Prioritization	Type	Class
1	Stop and Yield	C_3
2	Prohibitory	C_4
3	Mandatory	C_2
4	Warning	C_1
5	Information	C_5

Table 6: Prioritization of the classes.

6.5.1 Building the held-out data set

Here, we show the computation of an image belonging to the held-out data set for our traffic sign classifier, applying the approach described in Section 6.2.1. For the sake of conciseness, we show the generation process on binary versions of the BTSC images. We define an observation window W composed of four data points. Each data point is a sub-area of a training image, namely a square whose side is four pixels, at the center of the 64-pixel side image. It is important to remark that the definition of an observation window W based on spatial locality may require zooming or other pre-processing activities, in order to normalize the window's size with respect to the entire image. However, this is not the case for the BTSC training set, where cropping and zooming were applied, as customary, in the training set construction phase of the ML lifecycle, yielding a uniform set of images.

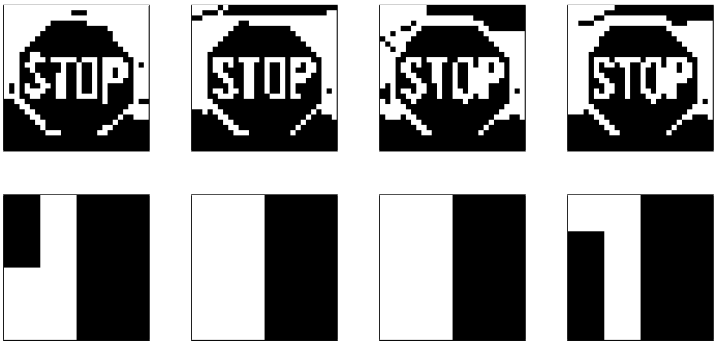


Figure 16: A binary image from the training set (above) and 16 pixel areas included in the window of observation W (below). We used the classic Otsu's algorithm for thresholding [1].

Figure 16 shows four binary images taken from the classifier training set (above), and the 16- pixel ($4x4$) sub-areas included in W (below). Figure 17 shows a "golden point" distilled from the window W (right-hand side) and the image including it[3] (left-hand side). This lightweight technique allows for fast computation of large held-out data sets.

Figure 17: A "golden" data point (left) and the held-out set image embedding it (right).

6.5.2 Noise Insertion

In our experiments, we considered degradation due to the insertion of random noise data into the training set of an ML model. This type of poisoning attack degrades the performance of the ML classifier by adding new (spurious) data to the training set that may cause some held-out points to be mis-classified.

In order to assess degradation severity, we calculated the CH of each class before and after the insertion, computing CH deformation. We repeated our experiment considering varying percentages of spurious additions to the training set. Some relevant results are shown in Figure 18, where additions of spurious points amount to about 10 and 20 percent of the original training set. Each row shows the CH of a class *i)* before the insertion, *ii)* after the insertion, and *iii)* their overlap. For cases where the noise insertion has caused changes in the classification, one of the mis-classified images is also shown.

Case *(a)* shows the circumstance where 10% spurious additions resulted in no change of the convex hull of the class under consideration (i.e. C_1). All the points in this class were correctly classified and eventual changes in classification suffered by the other classes did not affect it. Looking at case *(b)*, also concerning

[3]The other pixels of the image were computed by simple majority voting.

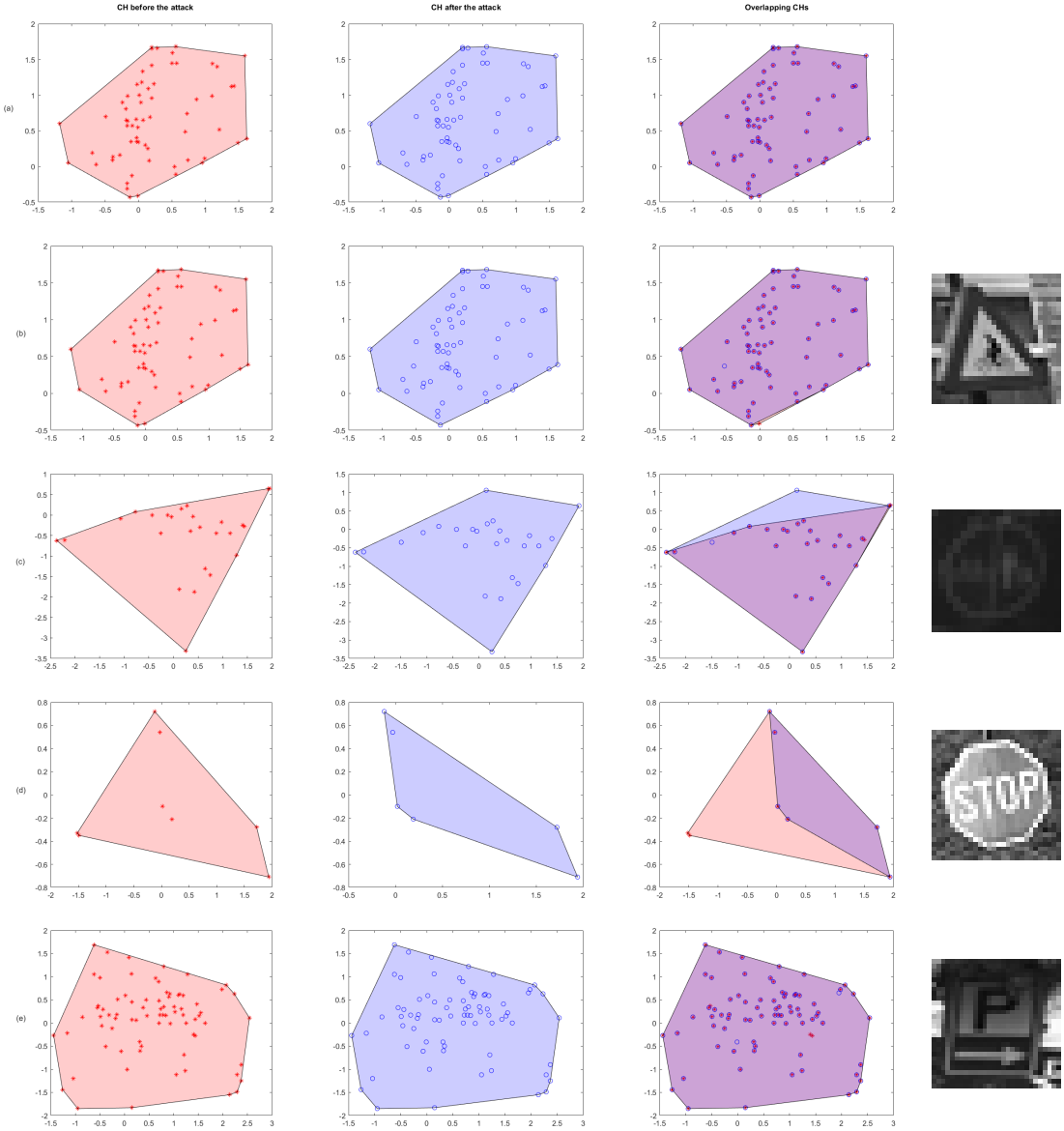


Figure 18: Example of convex hull (CH) deformations as a result of poisonous additions. Rows (a)–(e) show the CH of the class under evaluation *i)* before the attack, *ii)* after the attack, *iii)* their overlap, and *iv)* greyscale representation of one of the mis-classified images (when applicable).

class C_1, we notice that by adding 20% spurious data, the CH of the class has slightly changed. In particular, a point that before the noise insertion was on the class' CH has changed class after it, causing a decrease in the area defined by the hull. The image of the traffic sign for which the classification is no longer C_1 is shown. In cases *(c)* and *(d)*, which refer respectively to classes C_2 and C_3, the CH deformation is more pronounced. Of particular interest is the deformation for class C_3. This set of points is relatively small, but it is clearly visible that the classification change of an extreme point with very limited support (i.e., which is far from the other points of the class) has led to a large deformation of the CH. Lastly, like what happened in case *(a)* for class C_1, in case *(e)* the CH of class C_5 has not changed, but the cause is different: the few mis-classified points fell all within the CH and hence, mis-classification did not affect the class perimeter.

Results in Table 7 show that doubling the insertion of spurious data only increases marginally the index. Indeed, this trend is in accordance with the negligible variations in the per-class and overall accuracy in classification of the ML model. The Pearson correlation index between variations of our index and accuracy variations over four randomly chosen runs of the model was $\rho = 0.75$, suggesting that our index's changes can be used to predict accuracy variations.

	S Index	
Class	10% Spurious Data (Accuracy: 0.972)	20% Spurious Data (Accuracy: 0.972)
C_1	0.501	0.501
C_2	0.500	0.501
C_3	0.500	0.503
C_4	0.501	0.500
C_5	0.500	0.500
S_M	**0.2499**	**0.2504**

Table 7: OWA-based aggregation of S indexes associated to each class for a sample run.

6.6 Conclusion and Outlook

In this Chapter, we have proposed a quantitative technique for assessing the degradation of ML models' data assets due to injection of spurious training data. Some work remains to be done to bridge the Index computation with the setting of the hyper-parameters of the ensemble model introduced in Chapter 4.

6.6.1 Setting Ensemble Hyper-parameters based on the Index

Generally speaking, ML models' performance depends primarily on the selection of hyper-parameters for that particular problem. In the case of ensembles, determining the exact hyper-parameters for the base learners and the meta-learner is difficult and complex as the hyper-parameter space is larger [168]. Regarding training data, research has focused on the hyper-parameters driving the training process (e.g, determining the training to test data ratio that maximises the accuracy of ensemble models [169]). The hyper-parameter of interest in our case is the number k of training set partitions, and the optimization target is certified accuracy rather than accuracy. In our approach, the index's changes are used to trigger partition adjustment as they predict accuracy variations. The idea is to use a threshold on index-predicted accuracy variation to trigger re-partitioning of training data. For the computation, we plan to follow the approach of previous work [170] in using simple yet effective evolutionary methods to tackle the ensemble optimization. We plan to map the ensemble partitions to virtual particles, define the position of each particle by the hyper-parameter value (the number of parts k), and then use an evolutionary optimization algorithm to proceed towards the best value in terms of certified accuracy, integrated with the standard metric Mean Magnitude of Relative Error (MMRE) used for ensemble hyper-parameters optimization [171]:

$$MMRE = \frac{1}{n \sum_{t=1}^{n} MRE} \tag{18}$$

where n is number of samples.
We will further develop this approach in our future work.

6.6.2 Countermeasures

Of course, once degradation has been detected and measured, some countermeasures should be taken. Filtering would require applying the usual criteria for outlier identification; but, as our experimentation suggests, spurious data values can be "inliers", whose filtering is burdensome and error-prone [172], unless humans are kept in the loop [173]. We argue that a more viable strategy for alleviating ML data assets' degradation is the addition of *compensation data* that counterbalance the damage caused by the insertion of spurious data [174]. The acceptable amount (and collection cost) of compensation data depends on the specific data asset to be restored. In principle, contributing l labeled data that result in an increase ΔS_M of the severity index for a model M should require the addition of enough compensation data items to offset ΔS_M, bringing S_M back to the original value. We believe that the notion of compensation can be at the basis of collaborative protocols for ensuring ML data asset quality, as the actor doing the compensation needs not be the same who contributed the initial data. The next Chapter will present a Distributed Ledger-based approach based on the notion of *reciprocity*, where the computation of compensation is used as *Proof-of-Useful-Work* (PoUW) [174].

Chapter 7

Integrating DLT for Training Data Trustworthiness

This chapter outlines a method for improving the trustworthiness and quality of training set data, which is based on using a Distributed Ledger (DL) based on Proof-of-Useful Work (PoUW), where the work required from a candidate contributor to the DL holding training data set improves the value for ML inference of the other points already in the ledger.

7.1 Introduction

THE identification of threats to ML models' data assets is pointless if it does not lead to choosing safeguards or countermeasures (a.k.a. security controls) to counteract or alleviate the attacks' impact. In this Chapter, we put forward the novel idea of using an ML-oriented distributed consensus mechanism to support ML models' training set selection, using *Distributed Ledger Technology* (DLT) as a data protection framework. We envision a radical change in the domain, when organizations adopting ML models will start to use DLT technology to gain assurance about the integrity and trustworthiness of their models' training data, building robust ML-based systems that are immune to external interference. Developing on this idea, we investigate the notion of *Reciprocally Useful Work* (RUW) across ML

data assets, forcing proposed additions to the training set to "pay the price" of useful updates to other assets. Even if an attacker succeeds in inserting spurious labeled data into the training set, the work he has performed to that end will have benefited other labeled data to compensate for the effects of the poisoned value. Under reasonable assumptions about the ratio between malicious and honest users, improved data values will offset the effect of insertion and append attacks.

7.1.1 Distributed Ledger Technology

DLTs, often going under the collective name of *blockchain*, support a community of users in agreeing on the current state of a distributed data structure, the Distributed Ledger (DL). Permissioned DLTs support access control policies to the ledger content, while non-permissioned DLTs allow any user to take part to the protocol. The DL state is updated via a distributed consensus protocol, which conditions any update to the ledger by having reached an agreement among participants. In the literature, many consensus mechanisms have been proposed. The first use of a consensus mechanism was implicitly defined by Bitcoin [175], which introduced the *Proof of Work* (PoW) mechanism as consensus. PoW is difficult to generate but easy to verify. The idea came from Hashcash [176], a protocol originally designed for spam prevention. The Hashcash Protocol works as follows. Suppose a client wants to send an email to a server. At the beginning, the client and the server both agree on a hash function $H()$ which maps an input string to an output string of length n. Then, the email server sends a challenge string c to the client, who must find a string x such that $H(c||x)$ starts with k zeros. Since $H()$ has pseudorandom outputs, the probability of success in a single trial of this PoW puzzle is very low. So providing a solution is a proof of having spent CPU time (work) in multiple trials. The operation of a DLT consensus protocol is only slightly more complicated. Depending on the network architecture and blockchain type, some or all DLT users update the ledger by adding blocks to it. The creation of each new block to be added to the ledger is performed by a participant who is known as the leader of the consensus protocol in that execution. This leader is elected by a mechanism of election, consisting of a PoW puzzle competition

followed by implicit voting to reach an agreement, a sequence called *Nakamoto Consensus*.

7.2 Outline

In this Chapter, following [174], we discuss the use of DLTs to increase the quality of training data and the trust in their integrity. Regardless of the alleviation techniques used to improve the robustness of ML models, training data could be poisoned by stealthy attackers like compromised sensors, which can contribute to training set with poisoned points. Our scheme complements the alleviation procedure described in Chapter 4, inasmuch it is aimed at making less convenient for the attacker to insert poisoned data in the training set.

Specifically, we elaborate on the idea of achieving *trustworthiness of ML training data* via a DL, which supports community-wide agreement on the data used to express reputation [177] or to train ML models. Blockchain's original consensus mechanism based on the PoW notion supports trust in properties of *ledger transactions* like order and, indirectly, provenance but was not designed for establishing collective trust in properties of *ledger content* (e.g., being genuine representation of an external world phenomenon). In fact, as previously mentioned, PoWs have been introduced to prove that a certain amount of effort was spent and force malicious users of the ledger to shoulder a too large computational burden. PoW schemes are decoupled from the task they are attached to. The work (and energy) expended is generally not useful for anything except proving that some work has been done. Alternative lines of research focused on Proofs of Space (PoS) [178] where any user wishing to add to the ledger must prove she has set aside some disk space. Recently, the notion of *Proof-of-Useful-Work* [179] has emerged, where the proof-of-computation needed to access the ledger concerns a problem whose solution is actually useful for the application(s) supported by the ledger itself.

A major difference between our RUW protocol and other approaches to PoUW reported in the literature (see Section 7.3) is the notion of *reciprocity*. Namely, rather to have DL miners computing a collectively useful PoUW, we rely on work

exchange: each agent is asked to perform a RUW computation that benefits someone else, and will be able to update the ledger only if someone else (not necessarily the same person as the beneficiary) reciprocates. In our approach, instead of doing a part of a single collaborative task, each miner executes a task useful for someone else. This way, a single DL can be used for supporting multiple applications or ML models, each having its implementation of RUW. Our approach also confines the scope of an attack: each poisoned value added to the ledger will only affect the outcome of the application in which the attacker is directly involved. Finally, our approach directly compensates the decrease of data utility due to hostile content injected on the ledger with an increase of the utility of some other data. We claim that, under reasonable assumptions about the ratio between malicious and honest users, the data whose utility will increase will be most of the time an honest contribution, compensating the disruption due to poisoning. Specifically, in our consensus protocol, any participant x has to satisfy two conditions in order to be able to contribute a sample s_x to the ledger:

(1) x having delivered the result r_y of a hard computation $r_y = C(s_y)$ that improves the quality of some other contributor y' sample s_y;

(2) Some participant z having delivered the result of $r_x = C(s_x)$.

The remainder of this Chapter is structured as follows. Section 7.3 discusses in depth the ideas behind Proof-of-Useful-Work; our proposed reference problem (Hamiltonian path computation) is discusses in Section 7.3.1. Section 7.4 describes our DL consensus protocol based on RUW, while Section 7.5 presents an evaluation of its properties. Finally Section 7.6 draws our conclusions.

7.3 Proofs of Useful Work in the Wild

The original PoW mechanism of blockchain involves a considerable expenditure of both energy and computation and has neither meaningful applications nor interest in itself. In fact, the computational work done by blockchain miners serves no

useful purpose beyond determining accounting rights among participants and securing the system itself, thereby ensuring Sybil-resistance [180]. In order to tackle the environmental drawbacks of PoWs, there have been several attempts to construct consensus schemes that convert the meaningless work into practical tasks. Early efforts include Primecoin [181], which is based on searching for Cunningham and bi-twin chains of prime numbers, and Permacoin [182], which repurposed Bitcoin's mining resources as a distributed archival storage system. In particular, Permacoin relies on Proof-of-Retrievability, which requires miners to prove having access to a given copy of a file to successfully mint new coins. Both Primecoin and Permacoin have failed to reach wide adoption. For the former, it is still under discussion whether it is actually useful to find long sequences of prime numbers, whereas for the latter it is claimed that the protocol recovers only a small fraction of mining resources, so it cannot completely replace PoW.

The works proposed in [183] and [184] took a different approach to PoUWs, where the computation invested by miners is used to solve classes of problems of general utility. More specifically, the study in [183] focused on the construction of a PoUW scheme whose hardness is based on different computational problems that can be represented by low-degree polynomials, such as Orthogonal Vectors (OV), 3SUM and All-Pairs Shortest Path (APSP). Although the authors argue their approach may be seen as a delegation of computation that can provide PoUWs useful for any type of graph problems reducible to OV, 3SUM or APSP, the actual link between the work they propose and its subsequent useful utilization is somewhat feeble. The idea behind the Resource-Efficient-Mining (REM) framework [184] is to achieve PoUW by leveraging the Intel Software Guard Extensions (SGX), which enable process execution in an isolated environment conferring hardware protections on user-level code. Indeed, miners use or outsource SGX CPUs for computationally intensive workloads and have to prove that a certain amount of useful work has been devoted to a specific branch of the DL. Then, trustworthiness of the workloads is ensured by means of a hierarchical attestation mechanism. Also, the authors show that REM addresses two major limitations identified in the competing Proof-of-Elapsed-Time consensus mechanism (originally invented by Intel

and currently supported in the Hyperledger Sawtooth blockchain platform [185]), namely the stale and broken-chip problems.

The idea of directing computational resources to the solution of some ML-related task is currently emerging as a major line of research on PoUW. The protocol proposed in [186] has put forward the notion of Proof-of-Learning by drawing inspiration from ML competitions, such as the ones hosted in Kaggle and Codalab platforms, and its ultimate goal is to create an open repository of state-of-the-art ML models and datasets. The process involves three different types of actors, namely suppliers, trainers and validators, who, depending on the role, can either host an ML competition, train and propose models, or make decisions about new blocks and ranking. The mechanism used to select the members of the committee of validators is built upon Algorand's *cryptographic sortition* [187], which randomly chooses a subset of nodes according to per-user weights, providing each node with a priority. As a consequence, the scheme in [186] inherits some of Algorand's open issues related to preventing collusion between actors. The solution in [188] is also based on ML competition, but the focus is on privacy-preserving data mining. In [189], a draft design of Proof-of-Deep-Learning (PoDL) is discussed. PoDL is an improved PoW-like scheme where miners are asked to perform deep learning training and submit blocks along with their trained DL models as proofs. The block containing the DL model which provides the highest accuracy is then fully accepted by the network. The protocol deals with the problem of model overfitting by means of a block acceptance policy. However, PoDL has some inherent limitations. First, the verification process is not efficient, as full nodes are required to repeat the training. Second, strong assumptions are made on the honesty of the data provider. Coin.AI [190] is another theoretical proposal in which mining involves training a Deep Neural Network (DNN). The procedure for generating the mapping from the input (the hash of the last mined block, the transaction list and a nonce) to the DNN architecture is expressed as a context-free grammar. Similarly, BlockML [191] proposes a system for supervised training of Neural Networks. Also, there are works investigating the concept of PoUW with the aim of either integrating federated learning in the consensus process of DLs [192, 193] or

of helping developers build their DApps while promoting cloud-based computing [194]. Interest in developing PoUW-based solutions for AI also extends to projects beyond academia. For example, in [195] a hybrid proof-of-work/proof-of-stake DL called Personalised Artificial Intelligence (PAI) is used to train a DNN. In the proposed system, miners process a separately allocated mini-batch and improve their local models by sharing weight updates with their peers, while supervisors record all messages and detect possible malicious behaviour during training. Other actors involved in the protocol are verifiers and evaluators. At the core of the protocol lies a novel way to derive nonces by means of a formula that takes into consideration inputs and by-products of the ML training.

In the next Section, we will outline an alternative type of difficult mathematical problem upon which we based our RUW scheme.

7.3.1 The Hamiltonian Path Problem

The *Hamiltonian Path Problem* (HPP) is one of the best-known traversal problems which involves finding a path passing through all the vertices of a graph. According to the standard graph notation, a graph $\mathcal{G}$ is a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges. Formally, a *Hamiltonian path* (HP) is a spanning path in a graph $\mathcal{G}$, i.e. a simple traversal that touches each vertex of $\mathcal{G}$ exactly once. If the endpoints of the visit are adjacent, the path can be extended to a cycle and the corresponding variant of the problem is to find a *Hamiltonian cycle* (HC). The origins of the Hamiltonian problems date back to the eighteenth century, when William Rowan Hamilton proposed a puzzle known as the Icosian game, whose objective was finding a HC along the edges of a dodecahedron.

Over time, the Hamiltonian problem has been examined through different lens, and several variants of the original version have been proposed for both directed and undirected graphs in different computational frameworks. A widely studied extension is represented by the *Travelling Salesman Problem* (TSP), where a cost function assigns an integer cost to each edge in the graph. Essentially, the TSP is concerned with computing the most cost-efficient HC on a given weighted graph.

Thus, an instance $\mathcal{G}$ of the Hamiltonian cycle problem can be easily turned into an instance $\mathcal{G}'$ of the TSP by modeling the problem as a complete graph where all edges of the original graph obtain weight 1 and all other edges obtain weigh 2. If the original graph $\mathcal{G}$ has a HC, then the cost function assigns to each edge of such cycle a cost of 1, and so $\mathcal{G}'$ contains a tour of cost $|\mathcal{V}|$. Conversely, if $\mathcal{G}$ is not Hamiltonian, then any tour of $\mathcal{G}'$ must use some edge not in $\mathcal{E}$, i.e. has a length of at least $|\mathcal{V}| + 2$.

Despite its simple definition, the problem of finding a Hamiltonian path, or determining whether or not one exists, is computationally difficult for general graphs [196]. In fact, both the directed and non-directed variants were two of Karp's celebrated 21 $\mathcal{NP}$-complete problems [197]. The main difficulty of Hamiltonian problems lies in the fact that the large volume of data that needs to be analyzed makes it extremely unlikely the existence of a polynomial-time algorithm capable of solving the HPP for an arbitrary graph. An evidence of the complexity is that determining *Hamiltonicity*[1] by inspection is not trivial even for small-size instances, and slight increases in the size of the problem result in a massive increase in computational complexity. Therefore, the Hamiltonian problems are interesting on their own because they are closely related to the long-standing question on the relationship between the two complexity classes $\mathcal{P}$ and $\mathcal{NP}$. These problems have attracted the interest of computer scientists for many years due to their fundamental theoretical importance, as well as their broad spectrum of applications in different fields, including computer graphics, cryptology, operations research and genomics. Pre-computation of Hamiltonian paths across raster bit planes (e.g., portions having uniform pixel luminance) is a classic image pre-processing task. It satisfies two properties of PoUW: being *hard* for all and being *useful* for someone. Its computation adds values to the image because makes standard image pre-processing simpler, in view of training computer vision models [198, 199, 200].

In order to generally state the concept of Hamiltonian path in the context of

[1]Here we use the term Hamiltonicity to refer to graph properties that are related to both HCs and HPs. Though, to be precise, the proper term used to designate graphs possessing only HPs is semi-Hamiltonicity.

digital image pre-processing, we adopt a graph-based approach. Since graphs represent a set of elements and a set of pairwise relationships between those elements, these structures provide a natural way of representing images. In this thesis, we consider only the non-directed graph version of HPP. Therefore, given an image represented by means of a non-directed graph $\mathcal{G}$, HPP determines whether a HP exists in $\mathcal{G}$ (with an additional constraint, which will be discussed later). Intuitively, we treat every pixel in the image as a vertex in a graph. In addition, we interpret the image as a complete graph, where each vertex is connected to every other vertex by an edge (that is, $\mathcal{E} = \mathcal{V} \times \mathcal{V}$). Accordingly, the adjacency matrix associated with these graphs contains all 1s with 0s on its main diagonal. The reason why it is convenient to use a complete graph is that for such a graph the existence of a feasible solution is always guaranteed. Since every vertex is connected to every other vertex, it is always possible to find a HP or HC. Actually, the number of HPs and HCs in a complete graph with $n = |\mathcal{V}|$ vertices is $n!$ and $\frac{(n-1)!}{2}$, respectively.

7.4 The Proposed Protocol

For the sake of conciseness, we describe our protocol as a fully on-chain process, i.e. using the DL as the support for executing operation on data as well as the storage for the final data points. Our data operation are modeled as special primitives on the chain (Figure 19). We assume all contributors to have access to these primitives, and their code to have been supplied by some trusted third party (which is offline when the protocol is executed). This choice will allow us to focus on the protocol's steps rather than on precautions for ensuring code to be trustworthy. We remark that the hard Hamiltonian computation has to be implemented off-chain if it has to play the full role of PoUW, as on-chain computation would affect the CPU time of the DL node being updated rather than the one of the updater. However, as the result of the computation is checked on-chain, pushing the contract off-chain can be done via a simple challenge pattern [201]. Also, on-chain computation of the Hamiltonian will provide the desired effect in terms of

improving the data utility. We are also well aware that off-chaining other steps of the protocol may be preferable for performance reasons as well as for preventing Denial-of-Service (DoS) attacks by imposing some marginal effort also on who proposes the Hamiltonian challenge, but we will not deal with this problem in this thesis.

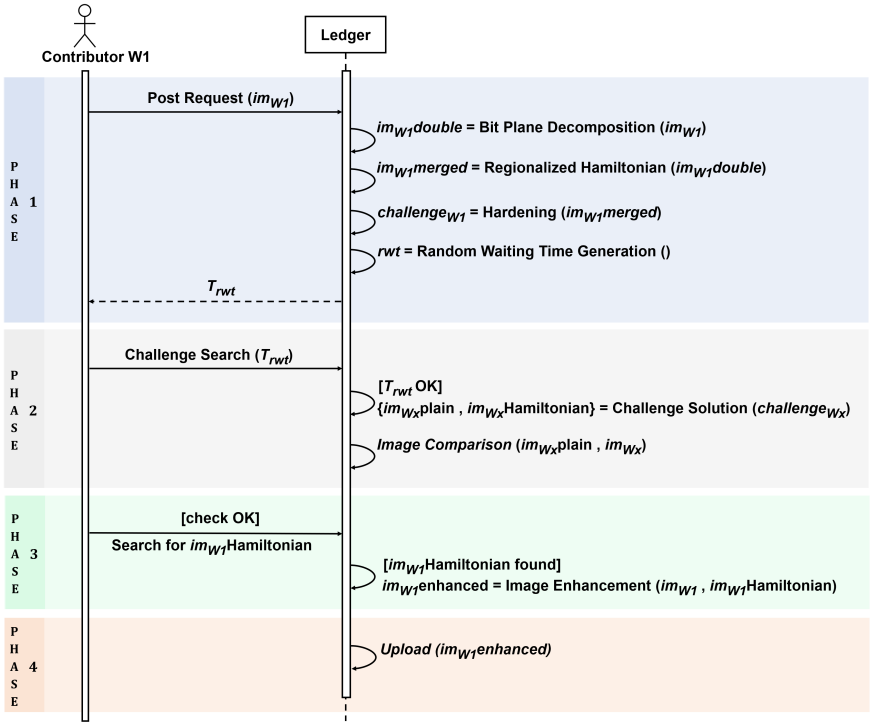


Figure 19: Phases of the proposed protocol.

In order to illustrate the image operations carried out during the first phase of the proposed protocol, we make use of a 256-level gray-scale image taken from the BTSC dataset [13]. In gray-scale images, pixel intensity values are integers ranging from 0 (black) to 255 (white). The intensity of each pixel is composed of 1 byte, so they can be represented by an 8-bit binary sequence. The proposed protocol consists of four phases, which are detailed hereinafter.

7.4.1 Phases

7.4.1.1 Challenge preparation

The first phase of our protocol is *challenge preparation* on the part of a potential contributor. The generation of the challenge involves three main steps: *(i)* changing image representation, *(ii)* regionalized permutation, and *(iii)* problem hardening. Suppose the 8-bit gray-scale image shown in Figure 20 is the one to be used to create the challenge.

Figure 20: An 8-bit gray-scale image of size 194 × 193 pixels.

The first step consists in changing the representation of the image through bit-plane decomposition and enlargement [202, 203]. An 8-bit image may be considered as being composed of eight 1-bit planes with plane 1 containing the lowest-order bit of all pixels in the image and plane 8 all the highest-order bits. For ease of presentation, henceforth we consider only the 5 × 5 pixel portion of the original image shown in Figure 21.

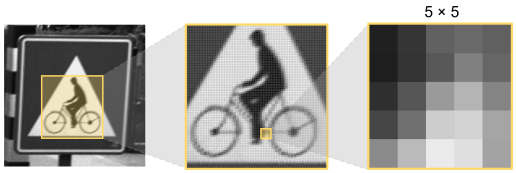


Figure 21: A selected 5 × 5 pixel portion of the image.

By using the traditional binary bit-plane decomposition [204], the image can be split into eight binary bit planes. Figures 22(a) through (h) are the eight bit planes of the selected 5 × 5 pixel portion, with Figure 22(a) corresponding to the highest-order bit. Image enlargement generates a new representation of the image in which

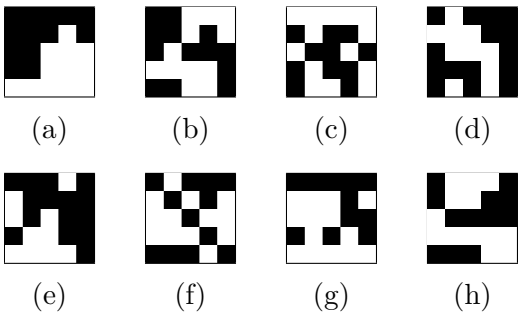


Figure 22: Bit planes 8 through 1.

pairs of consecutive bit planes are grouped together and placed adjacent to each other. The transformed image will be composed of four parts corresponding to the four groups of bit planes created as above. Therefore, starting from an image of size $M \times N$, we create an image of double the size. By applying these operations to our selected 5×5 portion, we obtain four sub-squares which form a 10×10 expanded image. In this new representation, each unit (or pseudo-pixel) contains only two bits.

In the second step, we compute a permutation of the pseudo-pixels of the enlarged image. As previously mentioned, we assume images are represented by a fully connected graph, where pixels correspond to the vertices and every pair of distinct vertices is connected by a unique edge in the graph. Clearly, finding a permutation of the pixels amounts to finding a HP by a one-to-one correspondence between the n elements of the permutation and the n vertices of the graph. In the process of visiting each pseudo-pixel, we enforce the constraint that the Hamiltonian path enters and exits each region exactly once within each of the four sub-squares of the enlarged image, where by *regions* we mean groups of pixels having in common k bits in their binary code. Basically, the idea is to traverse the four sub-squares by visiting all the pseudo-pixels they contain in lexicographic order, starting from the first sub-square and then moving to the others. Since our pseudo-pixels are represented by two bits, k is equal to 2, so there are $2^k = 4$ possible combinations of bits $(00, 01, 10, 11)$. It follows that when $k = n$, the notion of pseudo-pixel coincides with that of pixel, and it is exclusively in this case that the path visits all the pixels in order of intensity values – all the shades of

gray from black to white. Figure 23 shows the HP generated on the expanded image, where the dotted lines delimit the four sub-squares of pseudo-pixels, and the blue cell and the green one are the starting point and the ending point, respectively.

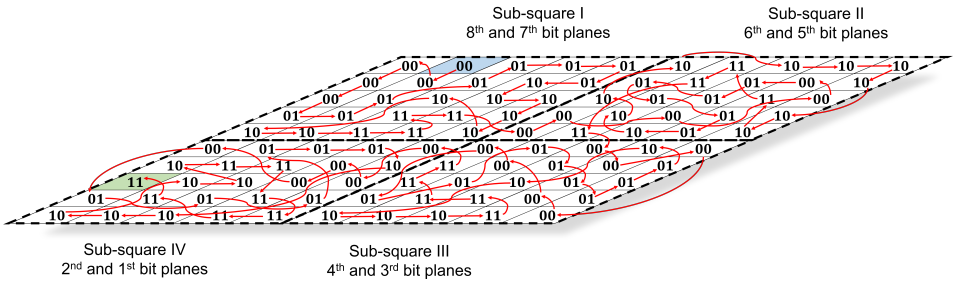


Figure 23: Hamiltonian path within the expanded image.

Once the HP has been generated, the pseudo-pixels locating in each of the four sub-squares have changed their position. At this point, the expanded image is restored to its original size $M \times N$ (5×5, in our case), thereby forming a new permuted image.

The fact that no worst-case efficient algorithm exists to find a HP or HC implies that the only known way to determine whether a given general graph is Hamiltonian is to undertake an exhaustive search. However, we do have to keep in mind that while finding an Hamiltonian path is a $\mathcal{NP}$-complete problem in general, there is no guarantee that finding one in a specific image will be difficult enough for our purposes. Also, it is evident that the instance specific characteristics have a significant impact on the intrinsic difficulty of the problem. So the rationale behind the third step is that in order to achieve the overall hardness property, it is necessary to implement a mechanism to increase the difficulty degree of the instances. To capture the hardness of the problem represented by a particular instance of both the HPP and the closely related TSP, diverse direct and indirect metrics have been investigated in several works [205, 206, 207, 208]. For example, in [209] the authors converted the TSP optimisation problem into a binary decision problem (under the question *can an algorithm find a solution with a tour length*

less than l?) and they correlated difficulty with the existence of a phase transition, suggesting that the dimensionless ratio $\frac{l}{\sqrt{n \cdot A}}$ could be used as a critical parameter controlling the transition from easy to hard, where n and A are the number of cities to visit and the average inter-city distance, respectively. Typically, the phase transition occurs at $\frac{l}{\sqrt{n \cdot A}} \approx 0.75$. Since the strategy in [209] required to find the optimal or a sub-optimal solution, entailing a considerable computational effort, the work carried out in [210] proposed an alternative approach to infer the complexity of the instances by analyzing different spatial properties, including the statistical distribution of distances. Specifically, in order to discriminate easy- and hard-to-solve instances, some parameters were derived from the distribution and correlated with the hardness value obtained from $\left|\frac{l}{\sqrt{n \cdot A}} - 0.75\right|$. In our context, distances between cities can be interpreted as the differences between the binary representation of the pixels within the image. Since in natural binary code the value of each bit depends on its position, we believe it is reasonable to calculate distances by means of the arithmetic distance instead of the Hamming distance. In fact, while the arithmetic distance is proportional to the binary difference of two values expressed in binary code, the Hamming distance does not account for positional values. For instance, consider the difference from a pixel having a gray level 128 (1000000_2) to a pixel having gray level 127 (01111111_2). While the arithmetic distance between the two values is just 1 bit, the Hamming distance is the maximum it can be for 8-bit pixels, i.e. 8 bits. We can roughly estimate the distribution of distances by considering the ratio between the number of difference values in the image (computed using arithmetic distance) and the total number of all possible differences. So, the protocol's step 3 performs a modification of the probability distribution of distances (which increases the value differences in a controlled manner) that will make sure the Hamiltonian problem published as a challenge belongs to a category of hard instances, unsuitable for heuristics or approximate searches. When we use a 256-level gray-scale image, the number of potential differences is modest because pixels may have a maximum difference of 255, thus the higher the image resolution, the harder the problem can be made to solve. Intuitively, the idea is to obtain an image whose pixels tend to occupy the entire range of possible

intensity levels and, in addition, tend to be distributed uniformly. Once the above operations have been performed, the Hamiltonian challenge is declared open on the ledger.

The last step of this first phase is the on-chain generation of a random number corresponding to the time period the contributor has to wait before she is allowed to solve a challenge. Such a number is received by the contributor in the form of a token that becomes valid after the defined random amount of time to wait has elapsed[2]. This ensures the enforcement of a delay (not known to the contributor in question) on the contributor's actual ability to solve another challenge.

7.4.1.2 Challenge solution

In the second phase of the protocol, *challenge solution*, which is asynchronous w.r.t. phases 1 and 3, the contributor searches for an open challenge while presenting the token received at the end of the previous phase. The search fails if the prescribed random time has not yet elapsed, whereas in the positive case, the contributor is allowed to solve the challenge she found. The resolution of the challenge consists in computing a permutation of the pixels in order to obtain an image equivalent to the original one (from which the challenge was derived). In other words, the contributor has to calculate a HP, and once generated, the result is compared with the original image. In case the two images match, the contributor is allowed to publish the solution, at the same time getting the token for condition (2) and leaving a token for condition (1) for the other contributor to retrieve.

7.4.1.3 Token acquisition

In the third phase, *token acquisition*, the contributor searches the ledger for the solution to its own challenge, retrieve the condition (1) token and leaves a condition (2) token.

[2]Attempts to use a token that is not yet valid will not succeed. For the sake of simplicity, this is not shown in Figure 19.

7.4.1.4 Trusted data update

Finally, phase 4 is *trusted data update*, where the contributor officially publishes the original image on the ledger, with both tokens.

7.4.2 Discussion

As can be clearly seen in the representation of Figure 24, our procedure allows us to calculate a Hamiltonian path across four different levels, while preserving information about original bit depth. The fact that our traversal proceeds by visiting each image unit on the basis of internal regions is the key of its usefulness. Once the solution of a given challenge is published, the problem of determining which pixels of the original image have a certain configuration in specific bit positions is greatly simplified. For example, consider the two lowest bits of any 8-bit pixel. In order to identify all the pixels ending with, say, the value '10' (that is, having pixel intensity equal to 10_2), it suffices to access the fourth sub-square, and therein you will find all the pseudo-pixels with the searched value in sequence. Therefore, calculating a region-based Hamiltonian path within digital images is useful both for determining the relative importance of each bit in the image and for having a direct access to the set of pixels that contribute to the total image appearance by the same values in specific bit positions.

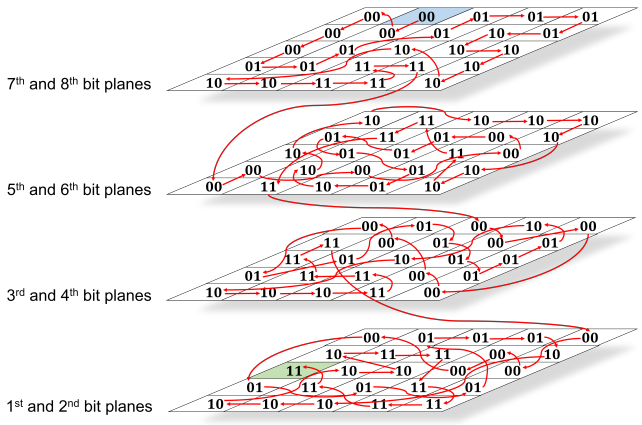


Figure 24: Hamiltonian path across bit planes.

7.5 RUW Protocol Evaluation

The goal of our RUW scheme is to prevent manipulation attacks on training data by imposing *reciprocity* of useful work: any manipulation that affects a data point (e.g., appending a spurious value) will require some work that will improve the quality of another value. In this Section, we first discuss how our protocol delivers reciprocity; then we discuss the impact of reciprocity on data value. Here we are not interested in the mechanisms used for the composition and the concatenation of the blocks to the DL, but we assume that blocks are chained using hashes of previous transactions, as it happens in public DLs such as Bitcoin or Ethereum.

The core of our proposal is that RUW participants are requested to perform computation in a reciprocal way: each participant should satisfy conditions (1) and (2) before updating the ledger, contributing with the solution of a challenge and getting a benefit from the computation performed by someone else. In the standard framework of indirect reciprocity, pairs of participants, one donor and one recipient, cooperate on occasion even if they will not meet again. There are a couple of assumptions we have to make in order to prevent malicious behaviors:

- ◇ Challenges are randomly assigned to participants, each one selected among the available pool of submitted samples;
- ◇ Each participant cannot solve a challenge she submitted.

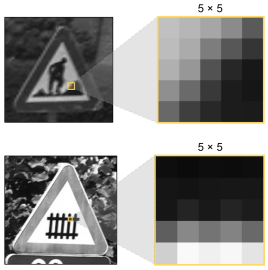
With the first assumption, we want to avoid that malicious participants cooperate in poisoning the training set, by submitting fake samples with known information which could ease the solution of the challenges; in this case one could solve the challenge of the other and then together they will be allowed to modify the distribution ledger. Therefore, we introduce a *random waiting time*, generated by an on-chain trusted module, before the contributor of a challenge can solve another. Being unknown to the participant, this random waiting time makes it unpractical for an attacker both to try to solve its own challenge (which is likely to be intercepted by a honest participant during the delay) and to collude with another attacker (effective collusion would require sharing the waiting time duration, which

is only computed on-chain). In this setting, occasional success of collusion cannot of course be excluded. We argue nonetheless that it can be controlled via classic *exponential back-off*, enlarging the time interval in which the random delay is chosen [211]. We liken our collusion setting to the *collision attack* scenario, where an attacker connected to a CSMA/CD access control network tries to disrupt communication by generating collisions with others. In our case, the "collision" event models the success of a collusion strategy, i.e. the attacker being able to intercept and solve a challenge posed by herself or by an accomplice, contributing nothing to the data point's value. It is well known that the success rate of CSMA/CD collision attacks depends on two factors: *(i)* traffic, expressed as the expected value of the number of competing terminals that attempt to access the network simultaneously, and *(ii)* the attacker's capability of monitoring accesses to the network. The security literature [212] shows that even when accesses (in our case, challenges) are fully monitored by the attacker, the predictability of the collisions (in our case, intercept) is very sensitive to changes in the delay parameters, even with a large number of competing terminals. Following this line of reasoning, we argue that a random token validity will force in most cases malicious participant to solve another party's challenge before being allowed to submit a fake sample. So, the attacker will incur in a cost c to compute the solution, giving a contribute Δ to the improvement of the whole model, that compensate the damage produced by the submitted fake sample.

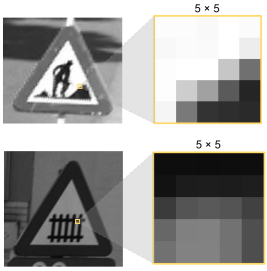
7.5.1 Value Analysis

Let us assume that a training set D has an overall value $V(D)$. The *append* attack adding a random value v changes the dataset value to $V(D + v)$. So we can call $\Delta V = V(D+v) - V(D)$ the disruption achieved by the attacker. In *insert* attacks, we have to pay attention to the position in D where v is inserted, as the decrease in value of the dataset may depend on that; if so, we should consider the dataset's Shapley value [213] by taking the average of the value decreases across all the possible insertion points for v. We also remark that the quantification of $V(D)$ (and of $V(D+v)$) for a given problem P may be done on a simple model rather than on

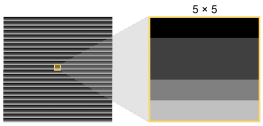
the actual ML model that will be used to solve P, with the advantage of having a simpler computation of the disruption. We use D to build a Nearest-Neighbor model for P and compute the disruption due to the injection of a value v in D by using a test set T, and counting the number of elements in T for which v is the closest value. Figure 25a and Figure 25b show (some) data belonging to our training and test sets.



(a) Training set.



(b) Test set.



(c) Poisoned image.

Figure 25: Training, test and poisoned data items for the reference NN model.

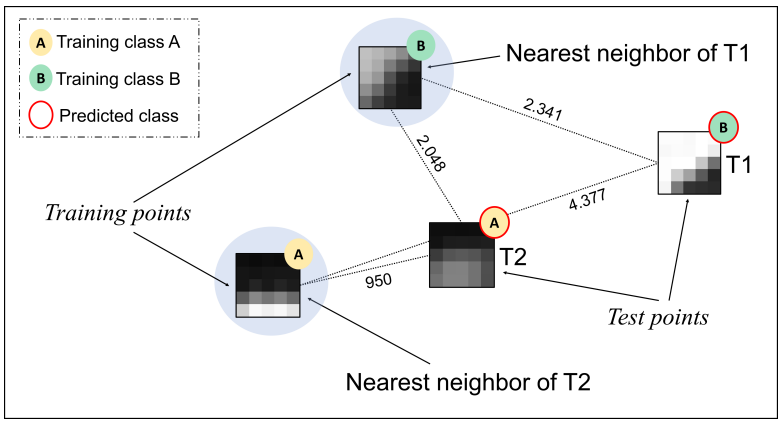


Figure 26: NN-based classification.

Figure 26 shows the operation of the NN model: an image from the test set is classified according to the label of the item in the training set which is closest to it.

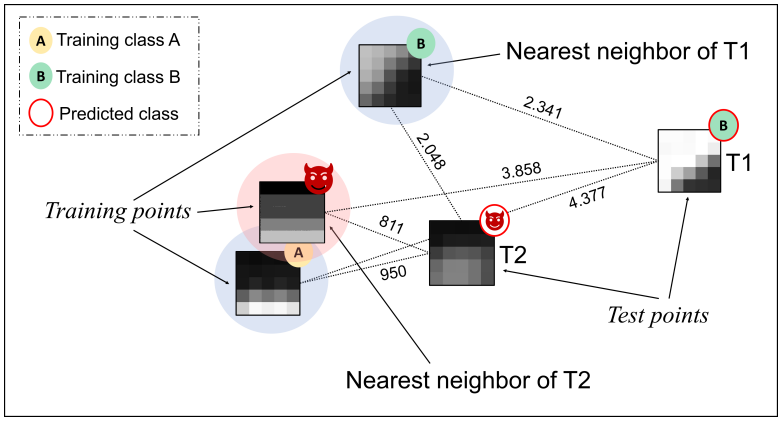


Figure 27: The effect of the poisoning attack on the NN model.

Figure 25c shows a poisoned image prepared by the attacker. As shown in Figure 27, the injection of the poison v introduces an error in the classification, as v becomes the nearest neighbor to the test image.

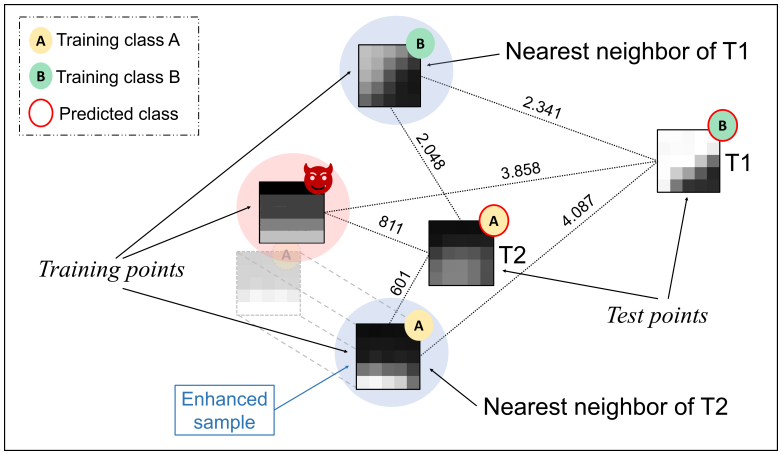


Figure 28: The remediation action by our protocol.

Thanks to our protocol, however, the injection of v also guarantees that some data point v' in the training set is modified to v'_h by adding the Hamiltonian path to it. The total value variation can be written as $\Delta V = V(D + v - v' + v'_h) - V(D)$.

Figure 28 also shows that the increase in value is enough to move the training image v' closer to the test one, offsetting the effect of the poison v. Of course, this would not happen if $V(D * v - v' + v'_h) > V(D)$ when $v = v'$, i.e. were the Hamiltonian unluckily added to the fake image injected by the same, or by another attacker. The probability of the latter event (the attacker's good work not being beneficial to honest participants, because it lands on another attacker) depends on the ratio between attackers and honest participants. However, as previously discussed, we claim that the probability of the former event (an attacker solving its own challenge, or two attackers colluding) can be controlled by the protocol via time randomization.

7.6 Conclusions

In this Chapter we put out the idea of *Reciprocally Useful Work* (RUW), a novel update mechanism for Distributed Ledgers where any agent wishing to add a block to the ledger must first perform an activity that will improve the utility for the DL-supported application of some other agent's block. Our scheme is aimed at making less convenient for stealthy attackers like compromised sensors to contribute to training set with poisoned points. Also we showed how RUW-supported trust in training data can be used to alleviate the problem of poisoning attacks to ML models.

Chapter 8

Conclusion and Future Work

In this chapter, we summarize the contributions of this dissertation and discuss some potential future research directions.

8.1 Summary of Contributions

In recent years, there has been a surge toward integrating Artificial Intelligence-based systems into IT infrastructures, with Machine Learning as one of the most widely used technologies. At the same time, the growth in power and scale of ML models has been accompanied by an expansion of their attack surface with innumerable attack vectors that, due to the uncertain nature of ML algorithms, are not fully understood. With the increasing challenges of adversarial attacks, developing appropriate defense mechanisms and security controls specific for ML applications is of paramount importance to ensure correct decision making.

In this dissertation, we focused on data, which is the core element of any AI-ML systems, and investigated how to make ML models more resilient to poisoning attacks and reduce the vulnerabilities that threats exploit in ML-based systems. One of the main gaps we identified in the literature is the fact that both the attacks and the proposed defense techniques are strictly tied to the model being considered. From the attack standpoint, knowing in advance the ML model to be attached allows for precise modeling of attack strategies that focus on worst-case analysis, i.e., are designed to maximize the damage on the learning algorithm. However,

in most practical situations, the adversary does not know the actual architecture, parameters, or output of the model. Moreover, when considering the entire ML lifecycle, even a more constrained attack (such as a label-flipping poisoning attack) is likely to be as harmful as an optimal one. From the defense standpoint, the fact that defense techniques are designed to protect specific algorithms makes them model-dependent and therefore not applicable on a large scale and in possibly different scenarios. Other gaps we found in the literature include the lack of approaches to estimate the risk of compromise associated with various ML data assets, as well as the lack of a shared quantitative definition of the severity of attacks and the effectiveness of the defense measures put in place.

In an effort toward filling the above gaps, in this thesis we contributed to the study of robust ML from both a theoretical and empirical perspective by proposing three types of security controls tailored for ML functionalities based on ensemble model robustness, index-based degradation detection, and distributed ledger-based training data trustworthiness. Whether they are applied individually or in conjunction, the overall goal of the proposed techniques is to provide a framework to enforce the security and robustness of practical ML-based systems in adversarial environments.

Specifically, in Part I we presented a novel defense mechanism combining ensemble composition and security risk analysis to protect ML models against training-time attacks, under the assumption of black-box knowledge about the attacker's and the defender's strategies. Based on the proximity to the separation surfaces identified with a linear model, each data point is associated with a risk index that is used during partitioning of the training set by an unsupervised technique. In Part II, we proposed two novel auxiliary techniques to assess the severity of degradation of ML model data assets and to improve the trustworthiness and quality of data to be used in training, respectively. The former technique, which relies on convex hull classifiers, serves to monitor the performance of deployed models (a posteriori) and can be useful in establishing whether re-training is needed and to appropriately instantiate the parameters of the ensemble-based approach.

The latter technique leverages inherent benefits of Distributed Ledger Technologies to make it less convenient for attackers to insert poisoned data points into the training set with ad-hoc Proof-of-Useful-Work definition.

8.2 Future Research Directions

As some of our proposed controls are not validated in depth, nor standardized in how they should be implemented, we recognize that further research should focus on creating benchmarks for their effectiveness. Furthermore, this work offers numerous avenues for future research directions. Below we outline some of them.

A possible improvement is concerned with the risk analysis associated with the input data space. In addition to introducing generic risk landscapes, wherein the color and corresponding risk index value of data points depend on contextual information (such as the non-uniform cost of attack for different regions in the input data space), one could better characterize the possibilistic component associated with the risk of tampering with the training data in terms of the uncertainty of data itself. The assessment of uncertainty about the training data is intended to estimate how feasible it is for a certain training data to have been tampered with and is essential to ensure that the defense mechanism employed is proportional to such uncertainty measure. In this regard, delving into how ML-oriented distributed consensus schemes can support ML models' training set selection while decreasing uncertainty on data is a key step towards the convergence of AI and DLT, since applications using both technologies in close integration can benefit from high-quality data for training efficient and robust ML models.

As for the natural trade-off between accuracy and robustness, which serves as a guiding principle in designing defenses against adversarial attacks, there is certainly no single path on how to solve it, and commonly the decision on which factor to prioritize is driven by specific use cases or applications. In the case of the ensemble-based technique, the hyper-parameter controlling the accuracy-robustness dilemma is the number of base learners in the ensemble. Instantiating ensemble parameters based on the model's degradation index may be an easily

applicable option, but further efforts are needed to properly bind the index to ensemble optimization. For example, partition adjustment could be controlled on the basis of a threshold on index-predicted accuracy variation, calculated through evolutionary approaches.

Another appealing perspective for future investigation concerns the action ability of attackers. Relevant literature, from which our own research draws, focuses on depicting adversarial scenarios where attackers operate autonomously, and the extent and success of the corresponding attack depends primarily on the resources and incentives the individual attacker wants to invest in damaging the system. Nowadays many attackers no longer operate exclusively independently, rather they are part of cyber-crime, meaning they act with a strong orientation to their potential gain from the offensive action. In this sense, we argue that it might be interesting to broaden the theoretical perspective by taking into account not only the difficulty and cost of at-scale compromises, but also the connections between malevolent actions and long-range benefits within a criminal group.

In conclusion, educating ML adopters about potential perils and proper design of security controls before ML models are deployed in high-stakes decision domains is a pivotal step. By engaging experts in cybersecurity and machine learning issues, we hope that our work will foster opportunities to design innovative and effective security solutions against emerging threats on ML models.

Bibliography

[1] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[2] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. Adversarial machine learning-industry perspectives. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 69–75. IEEE, 2020.

[3] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *WIREs Data Mining Knowl. Discov.*, 8(4), 2018.

[4] Zhi-Hua Zhou. *Ensemble Learning*, pages 270–273. Springer US, Boston, MA, 2009.

[5] Lior Rokach. *Ensemble Methods for Classifiers*, pages 957–980. Springer US, Boston, MA, 2005.

[6] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 1467–1474, Madison, WI, USA, 2012. Omnipress.

[7] Roei Schuster, Congzheng Song, Eran Tromer, and Vitaly Shmatikov. You autocomplete me: Poisoning vulnerabilities in neural code completion. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1559–1575. USENIX Association, August 2021.

[8] Junfeng Guo and Cong Liu. Practical poisoning attacks on neural networks. In *European Conference on Computer Vision*, pages 142–158. Springer, 2020.

[9] Blaine Nelson, Benjamin I. P. Rubinstein, Ling Huang, Anthony D. Joseph, Steven J. Lee, Satish Rao, and J. D. Tygar. Query strategies for evading convex-inducing classifiers. *J. Mach. Learn. Res.*, 13(1):1293–1332, may 2012.

[10] Shike Mei and Xiaojin 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, page 2871–2877. AAAI Press, 2015.

[11] ENISA. AI Cybersecurity Challenges – Threat Landscape for Artificial Intelligence, December 2020.

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[13] Radu Timofte, Karel Zimmermann, and Luc Van Gool. Multi-view traffic sign detection, recognition, and 3d localisation. In *2009 Workshop on Applications of Computer Vision (WACV)*, pages 1–8, 2009.

[14] Bowei Xi. Adversarial machine learning for cybersecurity and computer vision: Current developments and challenges. *Wiley Interdisciplinary Reviews: Computational Statistics*, 12(5):e1511, 2020.

[15] Gamaleldin Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alexey Kurakin, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial examples that fool both computer vision and time-limited humans. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[16] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on mnist. *arXiv preprint arXiv:1805.09190*, 2018.

[17] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[18] Yujie Li, Xing Xu, Jinhui Xiao, Siyuan Li, and Heng Tao Shen. Adaptive square attack: Fooling autonomous cars with adversarial traffic signs. *IEEE Internet of Things Journal*, 8(8):6337–6347, 2021.

[19] Vladimir Vapnik. *Statistical learning theory*. Wiley, 1998.

[20] Michael Brückner and Tobias Scheffer. Nash equilibria of static prediction games. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NIPS'09, page 171–179, Red Hook, NY, USA, 2009. Curran Associates Inc.

[21] Antônio David Viniski, Jean Paul Barddal, Alceu de Souza Britto Jr., Fabrício Enembreck, and Humberto Vinicius Aparecido de Campos. A case study of batch and incremental recommender systems in supermarket data under concept drifts and cold start. *Expert Syst. Appl.*, 176:114890, 2021.

[22] Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[23] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational*

Linguistics, pages 440–447, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[24] Masashi Sugiyama and Motoaki Kawanabe. *Machine Learning in Non-Stationary Environments - Introduction to Covariate Shift Adaptation.* Adaptive computation and machine learning. MIT Press, 2012.

[25] Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. *Dataset shift in machine learning.* Cambridge, MA: MIT Press, 2009.

[26] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436, 2015.

[27] Qinglong Wang, Wenbo Guo, Kaixuan Zhang, Alexander G. Ororbia, Xinyu Xing, Xue Liu, and C. Lee Giles. Adversary resistant deep neural networks with an application to malware detection. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 1145–1153, New York, NY, USA, 2017. Association for Computing Machinery.

[28] Ishai Rosenberg, Asaf Shabtai, Yuval Elovici, and Lior Rokach. Adversarial machine learning attacks and defense methods in the cyber security domain. *ACM Comput. Surv.*, 54(5), may 2021.

[29] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[30] Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. The security of machine learning. *Mach. Learn.*, 81(2):121–148, 2010.

[31] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM*

Symposium on Information, Computer and Communications Security, ASIACCS 2006, Taipei, Taiwan, March 21-24, 2006, pages 16–25. ACM, 2006.

[32] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, AISec '11, page 43–58, New York, NY, USA, 2011. Association for Computing Machinery.

[33] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE Trans. Knowl. Data Eng.*, 26(4):984–996, 2014.

[34] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C. Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization, 2017.

[35] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Trans. Neural Networks Learn. Syst.*, 30(9):2805–2824, 2019.

[36] Nikolaos Pitropakis, Emmanouil Panaousis, Thanassis Giannetsos, Eleftherios Anastasiadis, and George Loukas. A taxonomy and survey of attacks against machine learning. *Comput. Sci. Rev.*, 34, 2019.

[37] Nicolas Papernot. A marauder's map of security and privacy in machine learning. *CoRR*, abs/1811.01134, 2018.

[38] J.H. Saltzer and M.D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[39] Anthony D. Joseph, Pavel Laskov, Fabio Roli, J. Doug Tygar, and Blaine Nelson. Machine learning methods for computer security (dagstuhl perspectives workshop 12371). *Dagstuhl Manifestos*, 3(1):1–30, 2013.

[40] Blaine Nelson and Anthony D Joseph. Bounding an attack's complexity for a simple learning model. In *Proc. of the First Workshop on Tackling Computer*

Systems Problems with Machine Learning Techniques (SysML), Saint-Malo, France, page 111, 2006.

[41] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 387–402, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[42] Michael Kearns and Ming Li. Learning in the presence of malicious errors. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 267–280, New York, NY, USA, 1988. Association for Computing Machinery.

[43] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. In *Asian conference on machine learning*, pages 97–112. PMLR, 2011.

[44] Han Xiao, Huang Xiao, and Claudia Eckert. Adversarial label flips attack on support vector machines. In *ECAI 2012*, pages 870–875. IOS Press, 2012.

[45] Andrea Paudice, Luis Muñoz-González, and Emil C. Lupu. Label sanitization against label flipping poisoning attacks. In *ECML PKDD 2018 Workshops*, pages 5–15, Cham, 2019. Springer International Publishing.

[46] Gabriela F. Cretu, Angelos Stavrou, Michael E. Locasto, Salvatore J. Stolfo, and Angelos D. Keromytis. Casting out demons: Sanitizing training data for anomaly sensors. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 81–95, 2008.

[47] Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *CoRR*, abs/2012.10544, 2020.

[48] Samuel Henrique Silva and Peyman Najafirad. Opportunities and challenges in deep learning adversarial robustness: A survey. *CoRR*, abs/2007.00753, 2020.

[49] Zixiao Kong, Jingfeng Xue, Yong Wang, Lu Huang, Zequn Niu, and Feng Li. A survey on adversarial attack in the age of artificial intelligence. *Wirel. Commun. Mob. Comput.*, 2021:4907754:1–4907754:22, 2021.

[50] Xianmin Wang, Jing Li, Xiaohui Kuang, Yu an Tan, and Jin Li. The security of machine learning in an adversarial setting: A survey. *Journal of Parallel and Distributed Computing*, 130:12–23, 2019.

[51] Paul Schwerdtner, Florens Greßner, Nikhil Kapoor, Felix Assion, René Sass, Wiebke Günther, Fabian Hüger, and Peter Schlicht. Risk assessment for machine learning models. *CoRR*, abs/2011.04328, 2020.

[52] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I. P. Rubinstein, Udam Saini, Charles Sutton, J. Doug Tygar, and Kai Xia. Exploiting machine learning to subvert your spam filter. In Fabian Monrose, editor, *First USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET '08, San Francisco, CA, USA, April 15, 2008, Proceedings*. USENIX Association, 2008.

[53] Andrea Paudice, Luis Muñoz-González, Andras Gyorgy, and Emil C. Lupu. Detection of adversarial training examples in poisoning attacks through anomaly detection, 2018.

[54] Jiashi Feng, Huan Xu, Shie Mannor, and Shuicheng Yan. Robust logistic regression and classification. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, page 253–261, Cambridge, MA, USA, 2014. MIT Press.

[55] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artif. Intell. Rev.*, 22(2):85–126, oct 2004.

[56] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3520–3532, Red Hook, NY, USA, 2017. Curran Associates Inc.

[57] Christopher Frederickson, Michael Moore, Glenn Dawson, and Robi Polikar. Attack strength vs. detectability dilemma in adversarial machine learning, 2018.

[58] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses, 2021.

[59] John W. Tukey. A survey of sampling from contaminated distributions. 1960.

[60] Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964.

[61] Frank Rudolf Hampel. *Contributions to the theory of robust estimation.* University of California, Berkeley, 1968.

[62] K. A. Lai, A. B. Rao, and S. Vempala. Agnostic estimation of mean and covariance. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 665–674, Los Alamitos, CA, USA, oct 2016. IEEE Computer Society.

[63] Amir Globerson and Sam Roweis. Nightmare at test time: Robust learning by feature deletion. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 353–360, New York, NY, USA, 2006. Association for Computing Machinery.

[64] Adarsh Prasad, Arun Sai Suggala, Sivaraman Balakrishnan, and Pradeep Ravikumar. Robust estimation via robust gradient estimation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82(3):601–627, 2020.

[65] Jacob Steinhardt, Moses Charikar, and Gregory Valiant. Resilience: A criterion for learning in the presence of arbitrary outliers. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 45:1–45:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[66] Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high dimensions without the computational intractability. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 655–664, 2016.

[67] Ilias Diakonikolas and Daniel M. Kane. Recent advances in algorithmic high-dimensional robust statistics, 2019.

[68] Eitan Borgnia, Valeriia Cherepanova, Liam Fowl, Amin Ghiasi, Jonas Geiping, Micah Goldblum, Tom Goldstein, and Arjun Gupta. Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3855–3859, 2021.

[69] Eitan Borgnia, Jonas Geiping, Valeriia Cherepanova, Liam Fowl, Arjun Gupta, Amin Ghiasi, Furong Huang, Micah Goldblum, and Tom Goldstein. Dp-instahide: Provably defusing poisoning and backdoor attacks with differentially private data augmentations, 2021.

[70] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2018.

[71] Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout, 2017.

[72] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346–360, 2020.

[73] Lue Tao, Lei Feng, Jinfeng Yi, Sheng-Jun Huang, and Songcan Chen. Better safe than sorry: Preventing delusive adversaries with adversarial training. *Advances in Neural Information Processing Systems*, 34, 2021.

[74] Evani Radiya-Dixit and Florian Tramèr. Data poisoning won't save you from facial recognition. *arXiv preprint arXiv:2106.14851*, 2021.

[75] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *CoRR*, abs/1706.06083, 2017.

[76] Jonas Geiping, Liam Fowl, Gowthami Somepalli, Micah Goldblum, Michael Moeller, and Tom Goldstein. What doesn't kill you makes you robust(er): Adversarial training against poisons and backdoors, 2021.

[77] Jianing Zhu, Jingfeng Zhang, Bo Han, Tongliang Liu, Gang Niu, Hongxia Yang, Mohan Kankanhalli, and Masashi Sugiyama. Understanding the interaction of adversarial training with noisy labels. *arXiv preprint arXiv:2102.03482*, 2021.

[78] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[79] Battista Biggio, Igino Corona, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. Bagging classifiers for fighting poisoning attacks in adversarial classification tasks. In *Proceedings of the 10th International Conference on Multiple Classifier Systems*, MCS'11, page 350–359, Berlin, Heidelberg, 2011. Springer-Verlag.

[80] Battista Biggio, Giorgio Fumera, and Fabio Roli. Multiple classifier systems for robust classifier design in adversarial environments. *Int. J. Mach. Learn. Cybern.*, 1(1-4):27–41, 2010.

[81] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[82] Battista Biggio, Giorgio Fumera, and Fabio Roli. Multiple classifier systems for adversarial classification tasks. In Jón Atli Benediktsson, Josef Kittler, and Fabio Roli, editors, *Multiple Classifier Systems*, pages 132–141, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[83] Battista Biggio, Giorgio Fumera, and Fabio Roli. Multiple classifier systems under attack. In Neamat El Gayar, Josef Kittler, and Fabio Roli, editors, *Multiple Classifier Systems*, pages 74–83, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[84] Sanjay Kariyappa and Moinuddin K. Qureshi. Improving adversarial robustness of ensembles with diversity training. *CoRR*, abs/1901.09981, 2019.

[85] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 4970–4979. PMLR, 2019.

[86] Huanrui Yang, Jingyang Zhang, Hongliang Dong, Nathan Inkawhich, Andrew Gardner, Andrew Touchet, Wesley Wilkes, Heath Berry, and Hai Li. DVERGE: diversifying vulnerabilities for enhanced robust generation of ensembles. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[87] Thilo Strauss, Markus Hanselmann, Andrej Junginger, and Holger Ulmer. Ensemble methods as a defense to adversarial perturbations against deep neural networks, 2017.

[88] Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and Zico Kolter. Certified robustness to label-flipping attacks via randomized smoothing. In

Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8230–8241. PMLR, 13–18 Jul 2020.

[89] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672, 2019.

[90] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320. PMLR, 09–15 Jun 2019.

[91] Yuzhe Ma, Xiaojin Zhu, and Justin Hsu. Data poisoning against differentially-private learners: Attacks and defenses, 2019.

[92] Binghui Wang, Xiaoyu Cao, Jinyuan jia, and Neil Zhenqiang Gong. On certifying robustness against backdoor attacks via randomized smoothing, 2020.

[93] Maurice Weber, Xiaojun Xu, Bojan Karlaš, Ce Zhang, and Bo Li. Rab: Provable robustness against backdoor attacks, 2021.

[94] Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high dimensions without the computational intractability. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 655–664, 2016.

[95] Ji Gao, Amin Karbasi, and Mohammad Mahmoody. Learning and certification under instance-targeted poisoning, 2021.

[96] Saeed Mahloujifar, Dimitrios I. Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks

from concentration of measure. AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019.

[97] Alexander Levine and Soheil Feizi. Deep partition aggregation: Provable defense against general poisoning attacks, 2021.

[98] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang 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*, pages 7961–7969. AAAI Press, 2021.

[99] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Certified robustness of nearest neighbors against data poisoning attacks. *CoRR*, abs/2012.03765, 2020.

[100] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[101] Adarsh Subbaswamy, Roy Adams, and Suchi Saria. Evaluating model robustness and stability to dataset shift. In *International Conference on Artificial Intelligence and Statistics*, pages 2611–2619. PMLR, 2021.

[102] Nader S. Labib, Matthias R. Brust, Grégoire Danoy, and Pascal Bouvry. Trustworthiness in IoT – A standards gap analysis on security, data protection and privacy. In *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*, pages 1–7, 2019.

[103] Linyi Li, Xiangyu Qi, Tao Xie, and Bo Li. Sok: Certified robustness for deep neural networks. *CoRR*, abs/2009.04131, 2020.

[104] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Duane S. Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. *CoRR*, abs/1906.06316, 2019.

[105] Mislav Balunovic and Martin Vechev. Adversarial training and provable defenses: Bridging the gap. In *International Conference on Learning Representations*, 2019.

[106] Valerio Bellandi, Stelvio Cimato, Ernesto Damiani, Gabriele Gianini, and Antonio Zilli. Toward economic-aware risk assessment on the cloud. *IEEE Security & Privacy*, 13(6):30–37, 2015.

[107] Valerio Bellandi, Stelvio Cimato, Ernesto Damiani, and Gabriele Gianini. Possibilistic assessment of process-related disclosure risks on the cloud. In *Computational Intelligence and Quantitative Software Engineering*, pages 173–207. Springer, 2016.

[108] Zhiru Li, Wei Xu, Huibin Shi, Yuanyuan Zhang, and Yan Yan. Security and privacy risk assessment of energy big data in cloud environment. *Computational Intelligence and Neuroscience*, 2021, 2021.

[109] Jochen Kruppa, Andreas Ziegler, and Inke R König. Risk estimation and risk prediction using machine-learning methods. *Human genetics*, 131(10):1639–1654, 2012.

[110] David M. Johnson, Caiming Xiong, and Jason J. Corso. Semi-supervised nonlinear distance metric learning via forests of max-margin cluster hierarchies. *IEEE Transactions on Knowledge and Data Engineering*, 28(4):1035–1046, 2016.

[111] B Caroline, B Christian, B Stephan, B Luis, D Giuseppe, E Damiani, H Sven, L Caroline, M Jochen, Duy Cu Nguyen, et al. Securing machine learning algorithms. 2021.

[112] Lara Mauri and Ernesto Damiani. Stride-ai: An approach to identifying vulnerabilities of machine learning assets. In *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 147–154. IEEE, 2021.

[113] Patrick Royston, Gareth Ambler, and Willi Sauerbrei. The use of fractional polynomials to model continuous risk variables in epidemiology. *International journal of epidemiology*, 28(5):964–974, 1999.

[114] A. Blumer, A. Ehrenfreucht, D. Haussler, and M. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM*, 36:929–965, 1989.

[115] B. Apolloni, S. Bassis, D. Malchiodi, and P. Witold. *The Puzzle of Granular Computing*, volume 138 of *Studies in Computational Intelligence*. Springer Verlag, 2008.

[116] Han Xiao, Huang Xiao, and Claudia Eckert. Adversarial label flips attack on support vector machines. In *Proceedings of the 20th European Conference on Artificial Intelligence*, ECAI'12, page 870–875, NLD, 2012. IOS Press.

[117] Antonio Emanuele Cinà, Sebastiano Vascon, Ambra Demontis, Battista Biggio, Fabio Roli, and Marcello Pelillo. The hammer and the nut: Is bilevel optimization really needed to poison linear classifiers? *CoRR*, abs/2103.12399, 2021.

[118] Mengmei Zhang, Linmei Hu, Chuan Shi, and Xiao Wang. Adversarial label-flipping attack and defense for graph neural networks. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 791–800, 2020.

[119] Pooya Tavallali, Vahid Behzadan, Peyman Tavallali, and Mukesh Singhal. Adversarial poisoning attacks and defense for general multi-class models based on synthetic reduced nearest neighbors, 2021.

[120] Benoit Frenay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 2014.

[121] Zhuolin Yang, Linyi Li, Xiaojun Xu, Bhavya Kailkhura, Tao Xie, and Bo Li. On the certified robustness for ensemble models and beyond, 2021.

[122] A.D. Joseph, B. Nelson, B.I.P. Rubinstein, and J.D. Tygar. *Adversarial Machine Learning*. Online access: Morgan & Claypool Synthesis Collection Eight. Cambridge University Press, 2019.

[123] Seth James Nielson, Scott A Crosby, and Dan S Wallach. A taxonomy of rational attacks. In *International Workshop on Peer-to-Peer Systems*, pages 36–46. Springer, 2005.

[124] Matteo Ré and Giorgio Valentini. Ensemble methods : a review. 2012.

[125] Robi Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, 2006.

[126] Cha Zhang and Yunqian Ma. *Ensemble Machine Learning: Methods and Applications*. Springer, Boston, MA, 2012.

[127] Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Mach. Learn.*, 51(2):181–207, 2003.

[128] Yijun Bian and Huanhuan Chen. When does diversity help generalization in classification ensembles? *CoRR*, abs/1910.13631, 2019.

[129] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[130] Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, 2004.

[131] Ludmila Kuncheva. That elusive diversity in classifier ensembles. In Francisco José Perales López, Aurélio C. Campilho, Nicolas Pérez de la Blanca, and Alberto Sanfeliu, editors, *Pattern Recognition and Image Analysis, First Iberian Conference, IbPRIA 2003, Puerto de Andratx, Mallorca, Spain, June 4-6, 2003, Proceedings*, volume 2652 of *Lecture Notes in Computer Science*, pages 1126–1138. Springer, 2003.

[132] Naonori Ueda and Ryohei Nakano. Generalization error of ensemble estimators. *Proceedings of International Conference on Neural Networks (ICNN'96)*, 1:90–95 vol.1, 1996.

[133] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994.

[134] Kagan Tumer and Joydeep Ghosh. Error correlation and error reduction in ensemble classifiers. *Connect. Sci.*, 8(3):385–404, 1996.

[135] Gavin Brown. Ensemble learning. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 312–320. Springer, 2010.

[136] Zhiqiang Gong, Ping Zhong, and Weidong Hu. Diversity in machine learning. *IEEE Access*, 7:64323–64350, 2019.

[137] H. Späth. Anticlustering: Maximizing the variance criterion. *Control and Cybernetics*, 15:213–218, 1986.

[138] Venceslav Valev. Set partition principles. In *Transactions of the ninth Prague conference on information theory, statistical decision functions, and random processes (Prague, 1982)*, page 251–256. J. Kozesnik (Ed.), T. Prague: Springer Netherlands, 1983.

[139] Ventzeslav Valev. Set partition principles revisited. In *Advances in Pattern Recognition, Joint IAPR International Workshops SSPR '98 and SPR '98, Sydney, NSW, Australia, August 11-13, 1998, Proceedings*, volume 1451 of *Lecture Notes in Computer Science*, pages 875–881. Springer, 1998.

[140] Martin Papenberg and Gunnar W. Klau. Using anticlustering to partition data sets into equivalent parts. *Psychological Methods*, 26(2):161–174, 2021.

[141] Han G. Karypis, V. Kumar, and B. Mobasher. Clustering in a highdimensional space using hypergraph models. *Technical Report 97-019, University of Minnesota, Department of Computer Science*, 1997.

[142] Alexander Strehl and Joydeep Ghosh. Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing*, 15(2):208–230, 2003.

[143] Ates Dagli, Niall McCarroll, and Dmitry Vasilenko. Data partitioning for ensemble model building. *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, 7(3/4), 2017.

[144] Michael J. Brusco, J. Dennis Cradit, and Douglas Steinley. Combining diversity and dispersion criteria for anticlustering: A bicriterion approach. *The British journal of mathematical and statistical psychology*, 73(3), 2020.

[145] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR).

[146] Ron Shamir, Roded Sharan, and Dekel Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1):173–182, 2004. Discrete Mathematics and Data Mining.

[147] R. R. Weitz and S. Lakshminarayanan. An empirical comparison of heuristic methods for creating maximally diverse groups. *The Journal of the Operational Research Society*, 49(6):635–646, 1998.

[148] Yang Liu, Lei Ma, and Jianjun Zhao. Secure deep learning engineering: A road towards quality assurance of intelligent systems. In *Formal Methods and Software Engineering*, pages 3–15, Cham, 2019. Springer International Publishing.

[149] Lara Mauri and Ernesto Damiani. Estimating degradation of machine learning data assets. *ACM Journal of Data and Information Quality (JDIQ)*, 14(2):1–15, 2021.

[150] Aimad Karkouch, Hajar Mousannif, Hassan Al Moatassime, and Thomas Noel. Data quality in internet of things. *J. Netw. Comput. Appl.*, 73(C):57–81, September 2016.

[151] Horacio Paggi, Javier Soriano, Juan A. Lara, and Ernesto Damiani. Towards the definition of an information quality metric for information fusion models. *Computers Electrical Engineering*, 89:106907, 2021.

[152] S. Yadav and S. Shukla. Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, pages 78–83, 2016.

[153] Roland Roller and Mark Stevenson. Held-out versus gold standard: Comparison of evaluation strategies for distantly supervised relation extraction from Medline abstracts. In *Proceedings of the Sixth International Workshop on Health Text Mining and Information Analysis*, pages 97–102. Association for Computational Linguistics, 2015.

[154] Ernesto Damiani and Claudio A. Ardagna. Certified machine-learning models. In *SOFSEM 2020: Theory and Practice of Computer Science*, pages 3–15, Cham, 2020. Springer International Publishing.

[155] Paul F. Lazarsfeld. Studies in social psychology in world war ii vol. iv: Measurement and prediction. *Journal of information security and applications*, pages 362–4121, 1950.

[156] Leo A. Goodman. Exploratory latent structure analysis using both identifiable and unidentifiable models. *Biometrika*, 61:215–231, 1974.

[157] Abdulhadi Shoufan and Ernesto Damiani. On inter-rater reliability of information security experts. *J. Inf. Secur. Appl.*, 37(C):101–111, December 2017.

[158] Peter Cheeseman and John Stutz. *Bayesian Classification (AutoClass): Theory and Results*, page 153–180. American Association for Artificial Intelligence, USA, 1996.

[159] Marco E. G. V. Cattaneo. Conditional probability estimation. In *Probabilistic Graphical Models - Eighth International Conference, PGM 2016, Lugano, Switzerland, September 6-9, 2016. Proceedings*, volume 52 of *JMLR Workshop and Conference Proceedings*, pages 86–97. JMLR.org, 2016.

[160] E. T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

[161] M. C. K. Tweedie. Functions of a statistical variate with given means, with special reference to laplacian distributions. *Mathematical Proceedings of the Cambridge Philosophical Society*, 43(1):41–49, 1947.

[162] James E. Bobrow. A direct minimization approach for obtaining the distance between convex polyhedra. *The International Journal of Robotics Research*, 8(3):65–76, 1989.

[163] Qiangkui Leng, Zuowei He, Yuqing Liu, Yuping Qin, and Yujian Li. A soft-margin convex polyhedron classifier for nonlinear task with noise tolerance. *Applied Intelligence*, 2020.

[164] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, December 1996.

[165] Ernesto Damiani, Sabrina De Capitani di Vimercati, Pierangela Samarati, and Marco Viviani. A wowa-based aggregation technique on trust values connected to metadata. *Electron. Notes Theor. Comput. Sci.*, 157(3):131–142, May 2006.

[166] Ernesto Damiani, Paolo Ceravolo, Fulvio Frati, Valerio Bellandi, Ronald Maier, Isabella Seeber, and Gabriela Waldhart. Applying recommender systems in collaboration environments. *Comput. Hum. Behav.*, 51(PB):1124–1133, October 2015.

[167] R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):183–190, December 1988.

[168] Răzvan Andonie. Hyperparameter optimization in learning systems. *Journal of Membrane Computing*, 1(4):279–291, 2019.

[169] Pablo Ribalta Lorenzo, Jakub Nalepa, Michal Kawulok, Luciano Sanchez Ramos, and José Ranilla Pastor. Particle swarm optimization for hyperparameter selection in deep neural networks. In *Proceedings of the genetic and evolutionary computation conference*, pages 481–488, 2017.

[170] Sampath Kumar Palaniswamy and R Venkatesan. Hyperparameters tuning of ensemble model for software effort estimation. *Journal of Ambient Intelligence and Humanized Computing*, 12(6):6579–6589, 2021.

[171] Przemyslaw Pospieszny, Beata Czarnacka-Chrobot, and Andrzej Kobylinski. An effective approach for software project effort and duration estimation with machine learning algorithms. *Journal of Systems and Software*, 137:184–196, 2018.

[172] Wei Zhang and Jana Kosecka. A new inlier identification scheme for robust estimation problems. In *Robotics: Science and Systems II, August 16-19, 2006. University of Pennsylvania, Philadelphia, Pennsylvania, USA*. The MIT Press, 2006.

[173] Chengliang Chai, Lei Cao, Guoliang Li, Jian Li, Yuyu Luo, and Samuel Madden. Human-in-the-loop outlier detection. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 19–33, New York, NY, USA, 2020. Association for Computing Machinery.

[174] L. Mauri, E. Damiani, and S. Cimato. Be your neighbor's miner: Building trust in ledger content via reciprocally useful work. In *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*, pages 53–62, 2020.

[175] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008.

[176] Adam Back et al. Hashcash-a denial of service counter-measure. 2002.

[177] M. H. U. Rehman, K. Salah, E. Damiani, and D. Svetinovic. Towards Blockchain-Based Reputation-Aware Federated Learning. In *2020 IEEE International Symposium on Edge Computing Security and Blockchain, (EdgeBlock), in Conjunction with IEEE INFOCOM 2020, Beijing, China, April 27, 2020*, 2020.

[178] G. Ateniese, I. Bonacina, A. Faonio, and N. Galesi. Proofs of Space: When Space Is of the Essence. In Michel Abdalla and Roberto De Prisco, editors, *Security and Cryptography for Networks*, pages 538–557, Cham, 2014. Springer International Publishing.

[179] A. F. Loe and E. A. Quaglia. Conquering Generals: An NP-Hard Proof of Useful Work. In *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, CryBlock'18, page 54–59, New York, NY, USA, 2018. Association for Computing Machinery.

[180] J. R. Douceur. The Sybil Attack. In Peter Druschel, Frans Kaashoek, and Antony Rowstron, editors, *Peer-to-Peer Systems*, pages 251–260, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[181] S. King. Primecoin: Cryptocurrency with Prime Number Proof-of-Work. https://primecoin.io/bin/primecoin-paper.pdf, 2013.

[182] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz. Permacoin: Repurposing Bitcoin Work for Data Preservation. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 475–490. IEEE Computer Society, 2014.

[183] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan. Proofs of Useful Work. *IACR Cryptol. ePrint Arch.*, 2017:203, 2017.

[184] F. Zhang, I. Eyal, R. Escriva, A. Juels, and R. van Renesse. REM: Resource-Efficient Mining for Blockchains. In Engin Kirda and Thomas Ristenpart, editors, *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, pages 1427–1444. USENIX Association, 2017.

[185] Hyperledger Sawtooth. https://sawtooth.hyperledger.org/.

[186] F. Bravo-Marquez, S. Reeves, and M. Ugarte. Proof-of-Learning: A Blockchain Consensus Mechanism Based on Machine Learning Competitions. In *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*, pages 119–124, 2019.

[187] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 51–68. ACM, 2017.

[188] H. Turesson, A. Roatis, M. Laskowski, and H. M. Kim. Privacy-Preserving Blockchain Mining: Sybil-resistance by Proof-of-Useful-Work. *CoRR*, abs/1907.08744, 2019.

[189] C. Chenli, B. Li, Y. Shi, and T. Jung. Energy-recycling Blockchain with Proof-of-Deep-Learning. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 19–23, 2019.

[190] A. Baldominos and Y. Saez. Coin.AI: A Proof-of-Useful-Work Scheme for Blockchain-Based Distributed Deep Learning. *Entropy*, 21(8):723, 2019.

[191] A. Merlina. BlockML: A Useful Proof of Work System based on Machine Learning tasks. In Faisal Nawab and Etienne Riviere, editors, *Proceedings of the 20th International Middleware Conference Doctoral Symposium, Middleware 2019, Davis, CA, USA, December 09-13, 2019*, pages 6–8. ACM, 2019.

[192] X. Qu, S. Wang, Q. Hu, and X. Cheng. Proof of Federated Learning: A Novel Energy-recycling Consensus Algorithm. *CoRR*, abs/1912.11745, 2019.

[193] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang. Blockchain and Federated Learning for Privacy-Preserved Data Sharing in Industrial IoT. *IEEE Transactions on Industrial Informatics*, 16(6):4177–4186, 2020.

[194] Z. Dong, Y. C. Lee, and A. Y. Zomaya. Proofware: Proof of Useful Work Blockchain Consensus Protocol for Decentralized Applications. *CoRR*, abs/1903.09276, 2019.

[195] A. Lihu, J. Du, I. Barjaktarevic, P. Gerzanics, and M. Harvilla. A Proof of Useful Work for Artificial Intelligence on the Blockchain. *CoRR*, abs/2001.09244, 2020.

[196] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[197] R. M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[198] C. Fredembach and G. D. Finlayson. Hamiltonian path based shadow removal. In William F. Clocksin, Andrew W. Fitzgibbon, and Philip H. S. Torr, editors, *Proceedings of the British Machine Vision Conference 2005, Oxford, UK, September 2005*. British Machine Vision Association, 2005.

[199] K.T. Schütt, M. Gastegger, A. Tkatchenko, K. R. Müller, and R. J. Maurer. Unifying machine learning and quantum chemistry with a deep neural network for molecular wavefunctions. *Nature Communications*, 10(1):5024, 2019.

[200] G. Winkler. *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods: A Mathematical Introduction (Stochastic Modelling and Applied Probability)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[201] J. Eberhardt and S. Tai. On or Off the Blockchain? Insights on Off-Chaining Computation and Data. In Flavio De Paoli, Stefan Schulte, and Einar Broch Johnsen, editors, *Service-Oriented and Cloud Computing*, pages 3–15, Cham, 2017. Springer International Publishing.

[202] W. Zhang, KW. Wong, H. Yu, and Z. Zhu. A symmetric color image encryption algorithm using the intrinsic features of bit distributions. *Communications in Nonlinear Science and Numerical Simulation*, 18(3):584 – 600, 2013.

[203] W. Zhang, S. Wang, W. Han, H. Yu, and Z. Zhu. An Image Encryption Algorithm Based on Random Hamiltonian Path. *Entropy*, 22(1), 2020.

[204] R. C. Gonzalez and R. E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., USA, 2006.

[205] M. Haythorpe. FHCP Challenge Set: The First Set of Structurally Difficult Instances of the Hamiltonian Cycle Problem. *CoRR*, abs/1902.10352, 2019.

[206] P. Baniasadi, V. Ejov, M. Haythorpe, and S. Rossomakhine. A new benchmark set for Traveling salesman problem and Hamiltonian cycle problem. *CoRR*, abs/1806.09285, 2018.

[207] P. Krömer, J. Platos, and M. Kudelka. Network Measures and Evaluation of Traveling Salesman Instance Hardness. In *2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017, Honolulu, HI, USA, November 27 - Dec. 1, 2017*, pages 1–7. IEEE, 2017.

[208] M. Cárdenas-Montes. Slope-to-optimal-solution-based evaluation of the hardness of travelling salesman problem instances. *Log. J. IGPL*, 28(1):45–57, 2020.

[209] I. P. Gent and T. Walsh. The TSP phase transition. *Artificial Intelligence*, 88(1):349 – 358, 1996.

[210] M. Cárdenas-Montes. Creating hard-to-solve instances of travelling salesman problem. *Applied Soft Computing*, 71:268 – 276, 2018.

[211] B-J Kwak, N-O Song, and L. E. Miller. Performance Analysis of Exponential Backoff. *IEEE/ACM Transactions on Networking*, 13(2):343–355, 2005.

[212] A. L. Toledo and X. Wang. Robust Detection of MAC Layer Denial-of-Service Attacks in CSMA/CA Wireless Networks. *IEEE Transactions on Information Forensics and Security*, 3(3):347–358, 2008.

[213] R. Jia, D. Dao, B. Wang, F. A. Hubis, N. Hynes, N. M. Gurel, B. Li, C. Zhang, D. Song, and C. J. Spanos. Towards Efficient Data Valuation Based on the Shapley Value. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 1167–1176. PMLR, 2019.