UNIVERSITÀ DEGLI STUDI DI PALERMO

Dottorato di Ricerca in INFORMATION AND COMMUNICATION TECHNOLOGIES
Dipartimento di Ingegneria
SSD: ING-INF/05

METHODS FOR COMPUTING EFFECTIVE PERTURBATIONS IN ADVERSARIAL MACHINE LEARNING ATTACKS

IL DOTTORE
Andrea Giammanco

IL COORDINATORE
Ch.ma Prof.ssa Ilenia Tinnirello

IL TUTOR
Ch.mo Prof. Salvatore Gaglio

IL CO-TUTOR
Ch.mo Prof. Giuseppe Lo Re

CICLO XXXVI
ANNO CONSEGUIMENTO TITOLO: 2024

Abstract

In recent years, the widespread adoption of Machine Learning (ML) at the core of complex information technology systems has driven researchers to investigate the security and reliability of ML techniques. A very specific kind of threats concerns the *adversary* mechanisms through which an attacker could induce a classification algorithm to provide the desired output. Such strategies, known as Adversarial Machine Learning (AML), have a twofold purpose: to calculate a perturbation to be applied to the classifier's input such that the outcome is subverted, while maintaining the underlying intent of the original data. Although any manipulation that accomplishes these goals is theoretically acceptable, in real scenarios perturbations must correspond to a set of permissible manipulations of the input, which is rarely considered in the literature.

In this thesis, two different problems are considered related to the matter of generating effective perturbations in an AML attack. First, an e-health scenario is addressed, in which an automatic system for prescriptions can be deceived by inputs forged to subvert the model's prediction. Patients clinical records are typically based on binary features representing the presence/absence of certain symptoms. In this work it is presented an algorithm capable of generating a precise sequence of moves, that the adversary has to take in order to elude the automatic prescription service

Secondly, this thesis outlines an AML technique specifically designed to fool the spam account detection system of an Online Social Network (OSN). The proposed black-box evasion attack is formulated as an optimization problem that computes the adversarial sample while maintaining two important properties of the feature space, namely *statistical correlation* and *semantic dependency.*

Acknowledgments

I would like to thank my advisor, Prof. Salvatore Gaglio, and all of the professors of the NDSLAB research group, Prof. Giuseppe Lo Re, Prof. Alessandra De Paola, for having dedicated time in sharing their knowledge with me.

Thanks in particular to Prof. Marco Morana for having closely followed this work since the beginning.

Thanks to Prof. Marco Ortolani for our collaboration.

I am grateful to my colleagues of the lab for having shared many memories together: Pierluca, Vincenzo, Federico (also for our close collaboration), Antonio.

Thanks in particular to Marlo and Salvatore who have become close friends.

Special thanks to my dear friend Claudio whose presence has been inestimable.

Finally, I am grateful to my parents for their support in the most crucial moments.

Contents

List of Figures

List of Tables

Glossary

AI	Artificial Intelligence
AML	Adversarial Machine Learning
C&W	Carlini and Wagner adversarial attack
DF	DeepFool adversarial attack
FGSM	Fast Gradient Sign Method adversarial attack
LR	Logistic Regression
ML	Machine Learning
NN	Neural Network
OSN	Online Social Network
RF	Random Forest
RR	Ridge Regression
SVM	Support Vector Machine

Chapter 1

Introduction

The need to provide people with more and more sophisticated smart services has recently fostered a renewed interest in the topics of Artificial Intelligence (AI) and Machine Learning (ML). At the same time, it has now been widely accepted that the fallibility of intelligent systems can lead to diffuse, and potentially serious, errors. This is especially true in certain critical scenarios, where the misbehavior of AI and ML can threaten the security of a cyber system. For instance, intelligent algorithms are commonly adopted to analyze large amounts of data and recognize anomalous behaviors, such as network intrusions, cyber-attacks, slander campaigns, or spam activities. In all these cases, a failure will quickly propagate from the cyber space to the real world. The issue is even more severe when the malfunctioning of ML algorithms is induced by attackers exploiting automated Adversarial Machine Learning (AML) strategies, whose applications are as broad as ML itself.

AML [8] techniques aim to exploit the same optimization mechanism at the core of ML with an opposite intent: to let the model be sure, with high confidence, about an erroneous prediction. *Adversarial samples* are defined as "those that change the verdicts of Machine Learning systems but not those of humans" [12]. To be more specific, adversarial attacks can be characterized on the basis of different traits, the most general of which is whether they aim to directly alter the input data, or the corresponding feature values. In the simplest scenario, we can assume that the classifier operates directly on the input data; thus, the adversary calculates a perturbation to be added to the input in order to obtain an adversarial sample. In the case of image analysis, for instance, this would result in modifying individual pixel values directly. Most works consider AML in this scenario since it is straightforward for a human to verify the appearance of a certain image and assess the correctness of the classifier. In this context, the attacks are aimed at creating noise patterns [84] that exhibit two main characteristics: their superimposition over the original image is invisible to the human eye, and they cause an error in the classification algorithm. Moreover, algorithms for adversarial images corruption can heavily exploit the very large number of features (i.e., all the pixels of the image), as well as their scale of variability depending on the adopted encoding, so that the ascent along the gradient can proceed simultaneously in multiple directions at the same time. Moreover, algorithms for adversarial images corruption can heavily exploit the very large number of features (i.e., all the pixels of the image), as well as their scale of variability depending on the adopted encoding, so that the ascent along the gradient can proceed simultaneously in multiple directions at the same time.

In other application domains, understanding the best way to corrupt the input with adversarial noise can be very challenging.

For instance, considering a malware detection algorithm based on the API calls made by a software, one possible adversarial noise may consist in adding innocuous calls [3], while preserving the malicious behavior of the software. In an ambient intelligence scenario, sensors' raw data can be altered through a vector of carefully selected real values, in order to let a smart anomaly detection system raise false irregularity alerts regarding users' behaviors, or interrupt the operation of an intelligent energy-saving system. Assuming the presence of a Reputation Management System capable of identifying malicious entities in a sharing environment, changing the released feedback patterns can refresh the bad reputation of an adversary. Conjugated in Online Social Networks, slight modifications in spammers behavior (e.g., inflating the number of innocuous tweets) can hide their malicious intent to an intelligent detector [20]. Systems relying on smartphones sensors to recognize the activities carried out by users, may be trained on corrupted labeled data and fail in their identification task worsening the end services provided.

1.1 Motivations and Goals

In many scenarios, guaranteeing that the final verdict of the human remains unchanged even in front of input perturbations is not straightforward. In a *healthcare* scenario, for example, it would mean that an expert clinician should not alter his judgment in the face of an altered clinical record. However, if the adversary's move consists in altering the patient's record, it is highly likely that the final decision made by the clinician will change, especially in cases where the clinical record is composed of binary features. Nonetheless, perturbed clinical records may still be regarded as adversarial examples, as they share both the final goal to fool a machine learning algorithm, and the methodology used to get to the specific noise through the formulation of an optimization problem. In this thesis, this issue is addressed in order to show how an adversary may alter binary entries in the clinical record of a patient in order to elude a smart prescription system.

As an additional problem, when ML algorithms operate on complex feature sets that indirectly represent the input, and have no meaning to a human, it is burdensome to find out how to manipulate the feature values while also preserving the nature of the input. In these cases, two alternatives exist. In the easiest one, the features are independent; thus the single feature value can be modified without impacting the others. Otherwise, if the features are dependent, i.e., if they capture aspects that influence each other, only a subset of all possible manipulations ensures that such an interconnection is not broken, and of these, the most effective must be identified. In this thesis, the last setting is considered in the context of a ML system aimed to detect and block spam accounts in Online Social Networks (OSNs). The high *availability* of OSNs, that is, the fact that they can be easily accessed at any time from anywhere, is the key factor in their success and, at the same time, the main reason for the interest of malicious entities. Since spammers may adopt different strategies to achieve their goal, ML algorithms are usually trained on a wide set of features capable of capturing various aspects, such as information concerning the properties of the account, the history of shared content, and the degree to which a user is connected to the rest of the network. While such a comprehensive set of characteristics allows to identify different types of threats, large feature sets may also extend the attack surface of ML systems, making them more easily deceived. From an AML point of view, features describing the user of an OSN are closely interrelated (e.g., adding or deleting a message containing a URL would impact multiple feature values at the same time) and the steps required to fool a classifier cannot be made on a trial-and-error basis. In this context, accomplishing an attack means finding a way for the adversary to automatically alter the feature vector describing a *spammer* so that it is recognized as *genuine*, without impairing the malicious behavior.

1.2 Contributions

The activities carried out during the doctoral studies were focused on the study of AML problems and techniques. The studied problems included the vulnerability of *risk score* models, which are simple linear models useful in scenarios where the explainability is the first aim to reach (e.g., healthcare). Another addressed problem regarded the minimisation of the adversary's moves in perturbing the adversarial samples. Moreover, the study focused on the reduction of queries to the oracle in scenarios where the attacked model is not known. The employed techniques ranged from white- to black-box attacks with different influence on the input data. A preliminary result has been published in the *International Conference of the Italian Association for Artificial Intelligence*. In this work, a system for crafting perturbed records of patients has been described, where the aim of the adversary consists in increasing the prescriptions of a target medicine by changing few binary features of the clinical reports. In order to better understand the dangerousness of adversarial samples in the real world, the focus has been shifted from *white-box* attacks to *black-box* attacks, where the classification model to attack is not known. To this aim, the study focused on techniques to craft *optimal* samples able to elude different classifiers, regardless of their architecture or internal parameters. This study lead to an algorithm which has been tested in a scenario of spam account recognition on social networks. The findings of this study have been published in the *ACM Transactions on Privacy and Security*.

The main contributions of the doctoral work presented in this dissertation are:

- Firstly, a binary features AML attack is proposed, which is based on estimating the most important features that contribute in the variation of the model's loss gradient.
- Secondly, novel AML strategy is proposed, which explicitly preserves the *statistical correlation* among the features of the input space. This is achieved by formulating the attack as an optimization problem in which the search for the adversarial sample is constrained by the maintenance of the correlation coefficients observed in the original data.
- In this strategy the adversarial perturbation is chosen while preserving the *semantic dependency* that occurs when multiple features are computed from the same data. Without this constraint, the algorithm would produce perturbations that are numerically admissible, but not obtainable through real (legitimate) account manipulations.
- Such strategy also allows to deceive unknown classifiers by forging an adversarial sample that has minimum distance from the original sample (so as to preserve the input nature), while showing characteristics that are referable to the desired class samples (so as to maximize the probability of deceiving different target models).
- In order to make the results easily reproducible, the experimental analyses were carried out using two public datasets: one of medical prescriptions for the treatment of urinary tract infections, and one of *spammer* and *genuine* Twitter accounts.
- Experiments involved several models, and the performance were compared with different state-of-the-art attacks. Such a robust evaluation is essential to guarantee the generalisation capacity of the AML techniques.
- Moreover, a concrete case is presented aimed at exploring the manipulations to be applied to a real OSN account in order to carry out the proposed attack.
- The evaluation also includes the analysis of its robustness to three adversarial defense mechanisms, namely, two state-of-the-art approaches and a

confidence-based technique specifically designed to counter the peculiarities of the proposed attack.

- Finally, the source codes of the proposed algorithms have been released to the public [1] [2] in order to practically contribute to the idea of open science.

1.3 Dissertation Outline

The remainder of this dissertation follows this structure.

Chapter 2 provides the essential background to understand the core aspects of an AML strategy. Moreover, it discusses recent and relevant works in the field of AML, providing a categorization based on the adversary attributes and the applications domain explored is also given. In Chapter 3 a binary features AML attack is presented, conjugating it into a e-health application scenario. In Chapter 4 a correlation-aware AML attack is described, conjugating it into a spammer account detection application scenario. Chapter 5 draws the conclusions of this thesis.

1.4 Publications

Parts of the work in this thesis have been published in referred conference proceedings and journals:

- S. Gaglio, A. Giammanco, G. Lo Re, and M. Morana. Adversarial Machine Learning in e-Health: Attacking a Smart Prescription System. *International Conference of the Italian Association for Artificial Intelligence.* Cham: Springer International Publishing, 2021. [37] Github code repository: `https://github.com/agiammanco94/AIxIA2021`

- F. Concone, S. Gaglio, A. Giammanco, G. Lo Re, and M. Morana. AdverSPAM: Adversarial SPam Account Manipulation in Online Social Networks. *ACM Transactions on Privacy and Security* 2024. [23] Github code repository: `https://github.com/agiammanco94/AdverSPAM`

[1]`https://github.com/agiammanco94/AIxIA2021`
[2]`https://github.com/agiammanco94/AdverSPAM`

During the PhD, the following other works were produced:

- V. Agate, F. Concone, S. Gaglio and A. Giammanco. A Hybrid Recommender System for Cultural Heritage Promotion. *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2021. [1]
- A. De Paola, S. Gaglio, A. Giammanco, G. Lo Re and M. Morana. A multi-agent system for itinerary suggestion in smart environments. *CAAI Transactions on Intelligence Technology* 6.4: 377-393, 2021. [28]

Chapter 2

Background and Related Works

In this chapter, an overview on the problems of *Adversarial Machine Learning* is provided, together with the introduction of the notations used in this thesis, and a discussion of the most relevant works in the field.

Properly crafted input perturbations, either in training or testing, can subvert the predictions of machine learning algorithms. The perturbed inputs to ML models are generally referred to *adversarial samples*. The study of the vulnerability of learning algorithms to these adversarial examples, and the study of related countermeasures, defines the field of research of the *Adversarial Machine Learning*. Much attention to this area comes from the study of [76], which shows how deep networks for object recognition can be fooled with imperceptibly perturbed input images.

The security of machine learning algorithms can be imagined as a *arms race* between adversaries producing perturbed samples and defenders of the ML models effectiveness [8]. It has often been thought of countering the actions of malicious actors through ML algorithms, for example in the context of *intrusion* or *anomaly detection*. But ML algorithms themselves introduce new vulnerabilities, which experienced attackers can exploit to their advantage.

Indeed, there is an assumption that is violated in operating in adversarial environments which makes Machine Learning models to become vulnerable. This is the *independent and identically distributed* data assumption: train and test data are assumed to be sampled from the same distribution. This means that the test data, which is encountered in the operational phase of the system, is very similar to the training data, so the algorithm will have good accuracy on the test data if a reasonable amount of noise is present. When working in the presence of adversaries, the noise that the adversary creates is not random, it is created ad-hoc to fool the algorithm. The injection of this noise causes a deviation (shift) between the train and test distributions. If the attacker had no constraints in its actions, then the challenge between learning algorithms and adversaries would always be won by the latter. In the scenario of spam email filtering, the attacker must still preserve the readability of the email in order to convey a message to the human reader.

Supervised learning algorithms consist of a mapping $\mathcal{L} : \mathcal{L}(x) = \hat{y}$, whose aim is to assign the independent variables x with the dependent variables $\hat{y}$ according to the observed ground truth labels y. In order to learn the best mapping from inputs to predictions, the classifier has to be *trained* on a set of measurements; generally, the aim of such a procedure is to find a model $\mathcal{L}$ which is an acceptable approximation of the training data labels. This is pursued by minimising a *loss function*, $\ell(\cdot)$, between the real y and the predicted $\hat{y}$ label:

$$\underset{\theta}{min}\ \ell(\theta, x, y) + \lambda(\theta)\ , \tag{2.1}$$

where θ is the set of $\mathcal{L}$'s adjustable parameters, and λ is a *regularization* term which is typically employed to counter *overfitting*, i.e., the inability of the learnt model to generalize on new instances of the data.

An *adversarial attack* consists of an optimization algorithm which aims at finding the optimal perturbation to add to the input in order to let it change the prediction label of a ML model. Adversarial attacks can be categorized based on three characteristics of the attacker [8]:

1. the aim;

2. the possessed knowledge on the model to attack;

3. the perturbing capability over the inputs.

The attacker, based on these three of his characteristics, defines an optimization problem to identify an optimal attack strategy. The solution of this optimization problem represents the way to manipulate the input data and achieve the attacker's intended goal.

The attacker's purpose is defined in terms of three aspects [8]:

- **Desired security breach:** the attacker may aim to commit several breaches:
  - *integrity violation:* to evade detection without compromising the normal operational flow of the system;
  - *availability violation:* to compromise normal system functionality available to legitimate users;
  - *privacy violation:* to obtain private information about the system, its users, or data through reverse-engineering the learning algorithm.
- **Specificity of attack:** can vary between *targeted* and *untargeted*, depending on the attacker's purpose of causing misclassifications of a specific set of samples, or any sample.
- **Specificity of Error:** can be *specific* if the attacker aims to have a sample misclassified as a specific class; or *generic* if the attacker aims to have a sample misclassified as any among the available classes different from the true class.

The attacker may have different levels of knowledge about the targeted system, may know:

- the training data $\mathcal{D}$;
- the feature set $\mathcal{F}$;
- the learning algorithm $\mathcal{L}$, and the loss function ℓ minimized during training;
- the parameters θ learned by training.

Concisely, it is then possible to denote the attacker's knowledge as $\Omega = (\mathcal{D}, \mathcal{F}, \mathcal{L}, \theta)$. Depending on the nature of Ω, several attack scenarios can be described [8]:

- **Perfect-knowledge (PK) white-box attacks:** the attacker knows everything about the targeted system, i.e., $\Omega_{PK} = (\mathcal{D}, \mathcal{F}, \mathcal{L}, \theta)$. This scenario makes it possible to simulate a worst-case evaluation and assess the security of the learning algorithm by providing upper bounds on the performance degradation of the system under attack under extreme conditions.

- **Limited-knowledge (LK) gray-box attacks:** the attacker knows the feature representation $\mathcal{F}$ and the type of learning algorithm $\mathcal{L}$: the fact that it is a linear classifier, or a neural network with a certain structure), but the attacker does NOT know the training data $\mathcal{D}$, nor the learned parameters θ. However, it is assumed that the attacker is able to collect a surrogate dataset $\hat{\mathcal{D}}$[1] (ideally sampling from the same underlying distribution as the data), and receive feedback from the classifier about the label. The attacker can then estimate the parameters $\hat{\theta}$ from $\hat{\mathcal{D}}$, training the surrogate classifier. This type of attacks is called *LK attacks with Surrogate Data (LK-SD)*, and is denoted by: $\theta_{LK-SD} = (\hat{\mathcal{D}}, \mathcal{F}, \mathcal{L}, \hat{\theta})$. If the attacker does not even know the type of learning algorithm $\mathcal{L}$, the category is *LK attacks with Surrogate Learners (LK-SL)*: $\theta_{LK-SL} = (\hat{\mathcal{D}}, \mathcal{F}, \hat{\mathcal{L}}, \hat{\theta})$. Constructing a surrogate learning algorithm $\hat{\mathcal{L}}$ is also useful to evaluate the *transferability* of attacks between different learning algorithms.

- **Zero-knowledge (ZK) black-box attacks:** it is possible to compromise machine-learning algorithms even without any knowledge of the feature space, as long as the attacker can query the system, as if it were a *black-box*, obtaining feedback on labels or confidence scores. There is, however, a minimum amount of knowledge that the attacker possesses, even in this case: he knows the purpose for which the classifier was designed (e.g., *object recognition*, *malware classification*), and he knows the type of transformation he wants to apply to cause changes in features: for example, if the attack

[1] $\hat{}$ indicates limited knowledge of a certain component

target is a *malware detector* based on dynamic analysis, injecting static code that will never be executed will have no outcome. This setting is denoted as $\theta_{ZK} = (\hat{\mathcal{D}}, \hat{\mathcal{F}}, \hat{\mathcal{L}}, \hat{\theta})$.

A particular class of black box attacks, named *model based*, exploits the *transferability* property [61] of adversarial samples: a surrogate model mimicking the unknown target algorithm is leveraged to synthesize adversarial samples that will transfer to the target black box [50]. Reasons for such a property include orthogonality of the models' gradient directions, the alignment of their decision boundaries as well as geometric correlations between different regions of such boundaries [58], and magnitude of input gradients [33].

The ability of the attacker indicates the influence the attacker has on the input data, and on specific data constraints. An attacker's influence can be **causative** if the attacker can manipulate both training and test data, or *explorative* if the attacker can alter only test data. These scenarios are known as *poisoning* and *evasion* attacks, respectively. In order to effectively conduct attacks, there are still constraints that the attacker must meet: the initial attack samples $\mathcal{D}$ can only be modified by a set of transformations, which constitute the space of possible transformations $\Phi(\mathcal{D})$. In practice, this space of possible transformations gives indications of the amount of perturbation added to create *evasion attacks*, or the number of *poisoning attack points* injected into the training data. The most common *poisoning* attacks are aimed at compromising the availability of a service. *Poisoning* attacks are also spreading to compromise integrity by manipulating training data or the trained model to cause specific misclassifications.

Adversarial Machine Learning stems from the observation that the same optimization procedure can be used by an *adversary* for leading a learning algorithm to a wrong prediction. The strategy is to forge an *adversarial sample* $\tilde{x} = x + \delta$, where the value of the *perturbation* δ is found by subverting Eq. 2.1:

$$\max_{\tilde{x}} \ell(\theta, \tilde{x}, y) \ . \tag{2.2}$$

Indeed, the goal of the adversary is to find the perturbation that maximises the error between the ground truth and the predicted labels. Since the creation of $\tilde{x}$ starts from following the positive direction of the loss function, depending on the magnitude of δ it may happen that $\tilde{x}$ succeeds in escaping $\mathcal{L}$'s decision boundary, so altering the final predicted label $\hat{y}$. Thus, the value of δ should be large enough to allow the decision boundary to be crossed. However, it also have to be small enough not to completely alter the sense of the input x. Both of these requirements are addressed in the methods proposed in this thesis.

Let $\mathcal{T}$ be the target learning algorithm that the adversary wants to evade, and $\mathcal{S}$ the local surrogate model available to the adversary, the corrupted version of sample x, i.e. $\tilde{x}$, built by adding perturbation δ computed from $\mathcal{S}$, is said to transfer towards model $\mathcal{T}$ if:

$$\begin{cases} \mathcal{S}(x) \neq y^* \neq \mathcal{T}(x) \\ \mathcal{S}(\tilde{x}) = y^* = \mathcal{T}(x) \end{cases} , \tag{2.3}$$

where y^* is the adversary's desired output label.

In order to measure the *distance* between the original and the corrupted input samples, several metrics based on the Hölder norm L_p are commonly adopted [2]. The L_p norm between two samples x and $\tilde{x}$, is defined as:

$$L_p(\tilde{x}, x) = ||x - \tilde{x}||_p = \sqrt[p]{\sum_{i=1}^{d}(|x_i - \tilde{x}_i|^p)} , \tag{2.4}$$

where $p \in \mathbb{Z}$, x and $\tilde{x}$ are two vectors of d components with $d \in \mathbb{N}$, and subscript i denotes the i-th component [13]. Starting from defining $0^0 = 0$, the L_0 norm is computed as:

$$L_0(\tilde{x}, x) = \sum_{i=1}^{d}(|x_i - \tilde{x}_i|^0), \tag{2.5}$$

which provides a measure of how many single components are different in the two samples compared. The L_2 norm computes the euclidean distance between samples in order to measure their offset in the feature space:

$$L_2(\tilde{x}, x) = \sqrt{\sum_{i=1}^{d}(|x_i - \tilde{x}_i|^2)} . \tag{2.6}$$

Another type of widely used L_p norm for adversarial attacks is the L_∞ norm, which computes the maximum drift between the original and perturbed components of the input and is defined as:

$$L_\infty(\tilde{x}, x) = \underset{i}{max}\ |x_i, \tilde{x}_i|. \tag{2.7}$$

Using different metrics for determining the imperceptibility of an adversarial perturbation has a strong effect on the attack's outcome. By minimising L_2, for instance, it is possible to forge adversarial samples where multiple features have been slightly perturbed, thus performing a *dense* attack. On the other hand, focusing on L_0 permits to modify only a limited set of features, which results in a *sparse* attack [70].

The adopted notations and abbreviations used throughout the thesis are listed in Table 2.1.

Given the above overview of the core concepts underneath the Adversarial Machine Learning research, in what follows the main scientific works of the field are discussed in order to provide the context to which this manuscript contributes.

Over the past two decades, ML has become one of the core pillars of information technology and has acquired a central role in our daily lives. In this sense, the scientific community has invested a considerable effort in defining learning-based pattern classifiers that, so far, show impressive performance in several application domains. More recently, it has been shown that adversarial perturbations, carefully created in both training and testing phases, can easily subvert predictions made by ML algorithms. The vulnerability of ML to forged *adversarial* patterns, along with the design of appropriate countermeasures, is addressed in a quite novel research area, known as Adversarial Machine Learning (AML).

The effectiveness of AML is frequently demonstrated on computer vision scenarios [84], in which it is easier to visually assess the validity of an adversarial sample. In fact, the *imperceptibility* of the perturbed input is a desired property of any AML attack [39]. Among the most popular approaches in AML, the *Fast Gradient Sign Method* (FGSM) [39] paved the way for many studies exploring evasion strategies against image classifiers. In [61], the FGSM is used for creating adversarial images on the MNIST dataset, and testing their effectiveness against

Table 2.1: Notations used in the thesis.

Symbol	**Description**
$\mathcal{L}$	A general learning model.
$\mathcal{S}$	Surrogate model.
$\mathcal{T}$	Target model.
θ	Set of $\mathcal{L}$'s adjustable parameters.
$\ell(\theta, x, y)$	Loss function of $\mathcal{L}$.
∇_x	Gradient taken w.r.t. x.
$\mathcal{D}$	Training data.
$\mathcal{F}$	Feature set.
$\mathcal{X}$	Set of input samples.
x	Original sample.
$\tilde{\mathcal{X}}$	Set of perturbed samples.
$\tilde{x}$	Adversarial sample.
$\hat{y}$	Predicted label.
δ	Adversarial perturbation.
$L_p(\tilde{x}, x)$	p-norm between original and perturbed samples.
ξ	Real-valued adversarial perturbation vector.
γ	Maximum number of features to perturb.
$db_{\mathcal{S}}$	Decision boundary for $\mathcal{S}$.
α_i	i-th coefficient of $db_{\mathcal{S}}$.
β	Intercept of $db_{\mathcal{S}}$.
$\mathcal{R}_{j,i}$	Linear regression between x_j, x_i.
$m_{j,i}$	Slope of $\mathcal{R}_{j,i}$.
$q_{j,i}$	Intercept of $\mathcal{R}_{j,i}$.
$z(\tilde{x}, x)$	Cost function for $\tilde{x}$.
λ	Controlling factor for $z(\tilde{x}, x)$.
ψ	Maximum distance from the decision boundary.

classifiers hosted on Google and Amazon. A thorough analysis of this property, known as *transferability*, is provided in [33], where the root causes of this phenomenon are connected with the magnitude of the loss' gradient, implying that strongly regularized learning algorithms are more robust to attacks. *DeepFool* (DF) attack [57] introduced the idea of reasoning on the particular structure of the decision boundary to evade, testing approaches against well known convolutional neural network architectures. *Carlini and Wagner* (C&W) approach [13] focuses on the formulation of the adversarial evasion as an optimization problem, where the cost function tends to maximize the confidence of the attacked classifier about the misclassification; such approach, originally tested against image classifiers, has become a milestone for testing the efficiency of novel evasion strategies. The attack proposed in [17] is one of the first effective methods for evading unknown classifiers, where the adversary has the capability to query such models for reasoning on the provided label. AML may also regard the perturbation of audio signals in speech recognition systems. The goal of [68] is to let such a system transcribe a prefixed target sentence by perturbing only those frequencies that are un-listenable by humans. A stochastic compression technique is proposed in [7] for creating more robust models for speech recognition in smart home devices. Attacks are simulated through *Fast Gradient Sign Method* (FGSM) and *Projected Gradient Descent* (PGD), aiming at the execution of unwanted commands by the intelligent assistants deployed in the house. When the target model is unknown and its only observable output is composed of classes probabilities, authors of [79] propose a technique based on the computation of the gradient in a limited set of selected coordinates, with the momentum iterative method for creating the audio adversarial sample.

Other application scenarios include cybersecurity domains in which ML plays a dominant role in threat detection. Malware detection systems, for instance, can be attacked by adversaries through the manipulation of different sections of the source code [31]. In [65], the authors formalize a novel problem-space attack with the aim of automatically generating realistic and inconspicuous evasive adversarial applications for Android devices. Malware can also spread through the infection of PDF files; a comprehensive analysis of PDF-based AML attacks is provided in [53]. On the same topic, a methodology designed to evade structural PDF malware detection systems is presented in [52]. Similarly, Adobe Flash files may be perturbed by acting both on structural features which do not alter the functionalities, and content features through the addition of specific ActionScript commands. In this context, authors of [54] use a bisect line search algorithm for finding the most efficient step along the loss gradient direction for producing the adversarial sample. Windows malware are targeted in [30], where a genetic algorithm is used for generating adversarial executables, in which only a small set of functionality-preserving manipulations have been applied, such as header fields changes, slack space filling, or shifting the content before the start of a program section. Attackers may insert malicious code in users' browser, which will be activated upon visit of webpages and usually imply the gathering of sensible data contained in cookies. By targeting specific aspects of HTML and JavaScript syntaxes, authors of [80] leverage *Soft Q-learning* for creating adversarial samples able to evade cross-site scripting detectors. Considering the robustness of ML models deployed in this scenario, authors of [32] propose a method for learning uniformly distributed feature weights, which have been shown to strengthen the resistance of the model towards adversarial attacks, since adversaries have to perturb a larger number of features in order to succeed in their intent. *Intrusion Detection Systems* (IDSs) are also frequently considered as targets of AML attacks. In [5], the *Jacobian-based Saliency Map Attack* is leveraged in order to raise false alarms for short-circuit faults and other sensible threats. Authors of [4] use several approaches for creating adversarial traffic vectors for camouflaging malicious network flows, such approaches include *Generative Adversarial Networks* and genetic algorithms. It has been observed that features of network traffic present several statistical correlations, and the approach in [64] addresses such aspect by proposing a black-box attack that leverages the Mahalanobis distance between traffic vectors.

Other relevant scenarios include *electronic healthcare* [37], *biometric authentication* [10], *recommender systems* [77], *graph-based ML* [93, 16], *mobile edge computing* [92, 90], *wireless network security* [71, 48], and *spam detection* [26]. With regard to the last topic, it is worth highlighting the difference between *spam* detection, which refers to the identification of *unwanted content*, and *spam account* detection, which instead aims to distinguish *spammers*, be they human or bots, from *genuine users*. With regard to the *healthcare* domain, in [59] the authors evaluated the impact of several attack algorithms against models trained on a dataset containing ten vital signs of patients, showing how both attacks during training and test phase can have perilous implications. However, it is not formulated a precise sequence of steps the adversary has to make in order to achieve his goal, given that the perturbation is a real number which is difficult to interpret, and thus, inject into the data in a realistic scenario.

Over the years, spam detection has evolved from naive systems capable of recognizing common spammy words [20], to more reliable algorithms that consider wide sets of interconnected features. This led to the definition of more sophisticated attack strategies. The authors of [70], for instance, propose a *sparse* evasion attack based on L_1 norm aiming at adding or removing specific terms for letting a *Support Vector Machine* recognize spam emails as genuine. Similarly, by performing several other manipulations such as synonym replacement, ham word injection and spam word spacing, adversaries can fool a Bayesian model trained to detect spam emails [46]. In [89] the impact of feature selection on evasion attacks is evaluated; in particular, it has been observed that a drastic decrease in the number of features easily allow adversaries to fool the ML spam detectors by altering only few words in emails.

Spam account detection exploiting ML algorithms has also been discussed in a number of works [21, 24]. State-of-the art solutions usually exploit feature sets that are aimed to capture different attributes of a spammer, such as its connection with the rest of the social network, or data/metadata associated with the content shared [22]. These characteristics can be analyzed by means of a variety of models that typically include Neural Networks (NNs), Support Vector Machines (SVMs), and Random Forests (RFs), where the last one proved to be the most proper classifier when dealing with large feature sets [83]. Spam account detection in OSNs clearly has unique traits compared to other application domains because of the many ways a user can operate within social networks, hiding its malicious behavior, and thus achieving its disturbance objective. However, the study of the literature has revealed that the only intersections between AML and OSNs analysis regards *fake news* and *social bots* detection systems [25].

A summary of the related work is reported in Table 2.2, which highlights the characteristics of each approach according to the properties discussed in this Chapter.

Table 2.2: Distance norms and threat models of relevant attacks at the state-of-the-art. [2]

Domain	***Ref.***	***Norm***	***Threat Model***		
Image Processing	[39]	L_∞	*I, U*	*W*	*E*
	[33]	L_∞	*I, A, U*	*B*	*P*
	[61]	L_1	*I, U*	*W, B*	*E*
	[57]	L_2	*I, U*	*W*	*E*
	[13]	L_0, L_2, L_∞	*I, T*	*W*	*E*
	[17]	L_2	*I, U*	*B*	*E*
	[84]	L_2	*I, U*	*W*	*E*
Malware Detection	[31]	L_0	*I, T*	*W, B*	*E*
	[65]	L_0	*I, T*	*W*	*E*
	[52]	L_0	*I, T*	*B*	*E*
	[54]	L_1	*I, T*	*W*	*E*
	[32]	L_1	*I, T*	*W*	*E*
	[30]	L_1	*I, T*	*B*	*E*
Intrusion Detection	[4]	L_0	*I, T*	*B*	*E*
	[5]	L_0	*I, T*	G	*E*
	[64]	L_2	*I, U*	B	*E*
Speech Recognition	[68]	L_ϖ	*I, T*	*W*	*E*
	[7]	L_∞	*I, T, U*	*B, W*	*E*
	[79]	L_∞	*I, T*	*G*	*E*
Spam Email Detection	[46]	L_0	*I, U*	*W*	*E*
	[70]	L_1, L_2	*I, T*	*W*	*E*
	[89]	L_1	*I, T*	*W*	*E*
Other	[37]	L_0	*I, U*	*W*	*E*
	[10]	L_1	*I, T*	*B*	*E*
	[77]	L_0	*A, T*	*G*	*P*
	[80]	L_0	*I, T*	*B*	*E*
	[93]	L_0	*A, I, T*	*B*	*E, P*
	[59]	L_2	*I, T, U*	*W, B*	*E, P*
	[16]	L_0	*I, U*	*W*	*E*
	[92]	L_0	*I, T*	B	*E*
	[90]	L_0	*I, T*	W	*E*
	[71]	$L_\dagger$	*A, U*	*B*	*P*
	[48]	L_2	*I, U*	*B*	*E*

[2]Abbreviations: (I)ntegrity/(A)vailability, (T)argeted/(U)ntargeted;(W)hite-/(G)ray-/(B)lack-box; (E)vasion/(P)oisoning.

Chapter 3

Binary Perturbations in e-Health Prescription Classification

This Chapter shows how an adversary may alter binary entries in the clinical record of a patient in order to elude a smart prescription system. In this scenario, given that economic return is one of the most common motivations to conduct adversarial attacks in the healthcare domain [36], we can imagine as adversary an agent of a pharmaceutical company that produces a particular active ingredient, and wants to increase the sales by artificially inflate the number of prescriptions. In order to elude the smart prescription system, it is proposed an algorithm capable of generating the precise sequence of moves that the adversary has to take, i.e., which binary entries on the clinical record of the patient need to be flipped. In particular, it is assumed that the target model to evade is a neural network, whose parameters can be reasonably emulated by probing the smart prescription service as a black box [8]. The remainder of the Chapter is organized as follows. Section 3.1 outlines the *healthcare* scenario considered as case study. Section 3.2 formalizes the model of the adversary. Section 3.3 describes the algorithm to generate the adversarial perturbation for the clinical records of the patients. Section 3.4 presents the experimental analysis to validate the proposal.

3.1 Scenario

Electronic healthcare represents an ideal scenario to describe an adversary attack because of the strong economic interests that move the pharmaceutical production. In this Chapter a typical scenario is considered, schematically represented in Figure 3.1, where Alice and Bob are the doctor and the patient respectively. Bob reports his symptoms to Alice, who compiles a medical record also including his personal information, so that a decision about which treatment to prescribe can be made. In order to refine her decision, Alice relies on a trusted *Smart Prescription Service* on cloud, which is able to reason about the information in the medical record and suggest appropriate treatment. It is assumed that the model beneath this service is a neural network. In this context, a pharmaceutical company infiltrator named Darth, gains an economic return when a drug of its corporation is prescribed. Darth suggests Alice to install a software in the host that will interact with the cloud service, whose stated purpose is to optimize response times and effectiveness of the prescription system. Actually, this program performs an *AML* algorithm able to elude the *Smart Prescription Service* and induce the prescription of Darth's intended treatment. In particular, this middleware software identifies a restricted set of features which have to be altered in order to deceive the predictor in the cloud. Altering just a few features is a characteristic of the utmost importance in this scenario, since the *Smart Prescription Service* may return a detailed report including the altered clinical record received as input, which therefore needs to contain no striking changes in order not to make any doctor suspicious. Finally, Alice will weigh her judgment based on the response of the intelligent service, resulting in the prescription of Darth's intended drug with high probability. The following sections describe the attack.

3.2 Threat Model

Following the guidelines proposed in [8], in this section the model of the adversary is framed according to three main aspects: what is the pursued security breach (attacker's goal); what is the degree of acquired knowledge on the problem do-

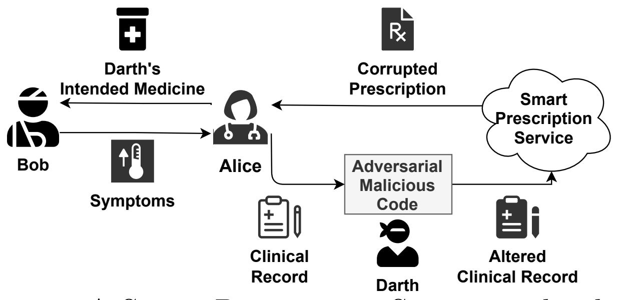


Figure 3.1: A Smart Prescription System with adversary.

main (attacker's knowledge); what are the concrete viable actions to achieve the malevolent intent (attacker's capability).

Attacker's goal: the adversary carries out an *integrity* violation of the predictive algorithm in order to flip its belief without disrupting the system in the whole, thus protecting himself from the risk of being caught. The attack specificity is *indiscriminate*, since the adversary does not make any distinction between the patients he wants to fool at his benefit. Accordingly, the error specificity is *specific*, as the target label, i.e., the active principle, that the adversary wants to be prescribed to raise an economic return for his company, is prefixed.

Attacker's knowledge: it is assumed that the adversary has perfect knowledge on the domain at hand, in other words, he conducts a *white-box* attack. The parameters (weights and biases) and hyperparameters (number of layers, number of neurons per layer) of the neural network under attack are known to the adversary, as well as the feature representation of the data. What needs not to be necessarily known are the portion of samples being part of the training set, and certain hyperparameters of the training process such as the batch size, the number of epochs, the learning rate, the weight decay factor adopted as regularizer, and the momentum coefficient for gradient descent. Although these assumptions may seem highly unlikely, it is common practice to test the strength of a machine learning model against the worst case scenario, so that under real and softer conditions the security of the system should not decrease. Moreover, in light of the transferability property of adversarial attacks [33], the model's parameters can be estimated by querying repeatedly, until a surrogate model capable of providing the same answers as the target model can be built.

Attacker's capability: the attack influence is *exploratory*, as the adversary has no access to the training data, and can only corrupt data belonging to the

Algorithm 1 Binary Adversarial Perturbation

```
Input:
    $x$: the input binary feature vector to perturb;
    $y$: the ground truth label for $x$;
    $mask$: a binary indicator of alterable features;
    $\theta$: trained parameters of the neural network;
    $\gamma$: number of binary features the adversary may perturb;
    $y_{target}$: the desired output label.
Output:
    $\delta$: perturbation to add to the input sample such that: $h_\theta(x + \delta) = y_{target} \neq \hat{y}$.
 1: $\hat{y} \leftarrow h_\theta(x)$
 2: $\delta \leftarrow zeros\_like(x)$
 3: if $\hat{y} == y_{target}$ then
 4:     return $\delta$
 5: $\xi \leftarrow \nabla_x L(\theta, x, y)$
 6: $\xi_{ranked} \leftarrow sort\_descending(\xi,\ key = abs)$
 7: $flippable \leftarrow mask\ \&\ (x \oplus sign(\xi))$
 8: $counter \leftarrow 0$
 9: for $value \in \xi_{ranked}$ do
10:     if $counter == \gamma$ then
11:         break
12:     $idx \leftarrow \xi.index(value)$
13:     if $flippable[idx] == 1$ then
14:         if $x[idx] == 0$ then
15:             $\delta[idx] \leftarrow +1$
16:         else if $x[idx] == 1$ then
17:             $\delta[idx] \leftarrow -1$
18:         $counter \leftarrow counter + 1$
19: return $\delta$
```

test set. The data manipulation constraints strongly depend on the particular scenario we are addressing in this study, where data are in the form of binary feature vectors. It is thus clear that the adversary has to respect the range of allowed values in the clinical records of the patients, i.e., values either of 0 or 1.

3.3 Methodology

The target *Smart Prescription Service* is a neural network, whose structure will be further investigated in Section 3.4.1. The pseudocode for the proposed strategy to create an effective perturbation against this model is provided in Algorithm 1. The proposed approach leverages the logic behind the Fast Gradient Sign Method (FGSM) [39], by first computing the gradient of the model's loss function $L(\theta, x, y)$ with respect to the input vector x, ground truth label y, and trained parameters θ. It then selects the least amount of features whose perturbation leads to the most precipitous step taken along the direction of the gradient. In what follows it is retraced the complete logical flow of the proposed algorithm.

This algorithm is executed by the malicious code injected by Darth into Alice's PC. The first step of the attack consists in computing the forward pass of the neural network w.r.t. the input vector x Darth wants to perturb. If the hypothesis of the model $\hat{y}$ is different from the target label y_{target}, Darth's objective is to flip the predicted label for the input vector x to the intended one. In other terms, the goal of the attack is to find the perturbation δ such that $h_\theta(x+\delta) \neq \hat{y}$. First, Darth puts in place a revised version of FGSM [39]. The traditional approach aims at climbing up the gradient by adding the perturbation $\xi = \epsilon \cdot sign(\nabla_x L(\theta, x, y))$, for a given $\epsilon > 0$, so that the perturbation vector ξ is composed by values equal to $\pm\epsilon$. ∇_x symbolizes the gradient taken w.r.t. the input vector. In this work only the term $\xi = \nabla_x L(\theta, x, y)$ is considered, so that each single $\xi_i \in \mathbb{R}$. The reason lies in the need to select only a small subset of the input features to perturb. Opposed to the image processing domain, where each pixel may be perturbed with a small step in its scale of representation, in different domains where the features may assume a limited set of values (in this case, binary values), each single perturbation added to the input features has to be selected with care. Therefore, from the real-valued perturbations vector ξ, Darth has to craft a binary perturbation mask to add to the input sample in order to flip the neural network's prediction. Initially, Darth sorts the perturbation vector ξ according to the absolute values of its components. The higher the value in ξ for a specific feature, the higher the contribute along the error that its perturbation will induce, thus becoming the optimum target. Let us now suppose that the application domain imposes some constraints on the features that may be altered; these constraints are represented in the form of a binary *mask* as input to the proposed algorithm, where the presence of a 1 indicates that the related feature may be perturbed. This *mask* explicates those features whose alteration is risky, because they can easily lead to the possibility of being disclosed during the attack. Another input parameter to the attack algorithm is the maximum number of features γ that Darth may alter from the input vector. A single input feature x_i may be altered in two cases:

1. $x_i == 1$ and the perturbation which results from ascending along the gradient of the loss function has negative sign, i.e. $sign(\xi_i) = -$, so that x_i may be flipped by adding $\delta_i = -1$. In other words, a feature value of 1 can be altered to 0 only if the sign of the gradient along that feature is negative;

2. $x_i == 0$ and the perturbation has positive sign instead, i.e. $sign(\xi_i) = +$, so that in order to flip x_i, $\delta_i = +1$ may be added.

To achieve this aim, firstly the sign of the perturbation vector ξ is considered, $sign(\xi)$, which is then XORed with the input vector x: $x \oplus sign(\xi)$, so as to obtain a True value only when the input feature is 1 and the perturbation has negative sign, and vice versa. The $sign(\xi)$ is represented as a binary vector where 1 stands for the sign '+' and 0 for '-'. The result of this operation is then processed with a bit-wise AND with the input *mask*: $flippable = mask \ \& \ (x \oplus sign(\xi))$. This operation results in a binary vector, $flippable$, which signals all those features that, if altered, make the neural network increase the error, because their alteration is concordant with the direction of the loss's gradient. Finally, having the list of features he may alter to deceive the neural network, Darth chooses the γ features with maximum absolute value, in order to take the gradient's sharpest stride.

In order to achieve the attack, Darth must have a deep knowledge of the medical domain he is going to infiltrate. This implies the awareness of both the set of alterable features to compose the *mask* and, most importantly, the parameters θ of the model beneath the smart prescription service. The latter can be achieved by probing the service as a black box, and building a surrogate model which responds in the most similar manner to the smart prescription service [8]. This approach finds his justification in the demonstrated transferability property of adversarial attacks [33]. When Alice provides Bob's clinical record to Darth, he first decides a threshold γ of maximum binary feature values to perturb. He computes the loss' gradient ∇_x of the surrogate model's parameters with respect to Bob's record. The most proficient features to alter are those which posses three properties: they do not appear in the *mask* of inconvenient features; they have the highest correspondent module in ∇_x; their alteration is concordant with the respective sign of ∇_x. The perturbed clinical record is then provided to the smart prescription service, which will return, with high probability, a report to Alice containing Darth's intended medicine as the suggested prescription. It is Darth's concern to select γ as a good trade-off between an higher probability of subverting the smart prescriber prediction, and a lower probability of raising Alice's doubts towards the model's outcome.

3.4 Experimental Analysis

In order to validate the proposal, the AMR-UTI dataset[1] [38, 60, 45] has been adopted, which contains electronic health records of patients with urinary tract infections. Each record consists of demographic information, past clinical data such as previous antibiotic exposure or resistance, and the antibiotic prescription chosen by a clinician to treat the patient. This dataset, allows to train a model able to prescribe the so-called "empiric antibiotic treatment", which the patient should take while waiting the necessary three days for the accurate response from his urinal specimen analysis. In this scenario, the interest of the adversary lies in altering the treatment chosen by the model, simultaneously respecting any contraindications w.r.t. the patient's clinical status. In particular, the patients who were treated with a first-line antibiotic have been considered, which is one of two classes: nitrofurantoin (NIT) and trimethoprim-sulfamethoxazole (SXT). The authors of the dataset provided a train/test division based on the years: specimen samples of the train set have been collected during the years 2007-2013, whereas the specimen in the test set refers to the period 2014-2016. Respecting this original division, the train set of first-line prescriptions contains 6815 samples, while the test set contains 2618 samples. Among the train set, 1892 samples received an empirical prescription of nitrofurantoin (NIT), and 4923 the trimethoprim-sulfamethoxazole (SXT). In the test set, 1358 samples where prescribed nitrofurantoin (NIT), the remaining 1260 trimethoprim-sulfamethoxazole (SXT).

Among the features exposed in the AMR-UTI dataset, the patients' demographic information have been considered as *"not-corruptible"* (which have been modeled through the input *mask* in Algorithm 1), in the sense that the adversary has no interest in altering these information in the clinical record of the patient, because of their ease of counter-proofing with reality. By performing other preprocessing steps which are released as part of the source code[2], a set of 564 binary features has been obtained, which are represented with different time granularities. In this work, the analysis is restricted to the features registered within 180 days, also because this is the time window most commonly shared between all the

[1]`https://www.physionet.org/content/antimicrobial-resistance-uti/1.0.0/`
[2]`https://github.com/agiammanco94/AIxIA2021`

features, for a total amount of 135. Finally, the κ best features have been selected according to the chi-square independence test [55], where κ is considered as one of the hyperparameters whose exploration will be further described in the next subsection. Having fixed a specific value for κ, all those samples with equal binary features values but different label have been removed.

3.4.1 The classification network

Experiments were performed starting from an existing neural network[3], which has been extended by adding the cross entropy loss function, the softmax activation layer, the momentum gradient descent, the regularization through weight decay, the random search algorithm for hyperparameters tuning [40], the FGSM [39], and the attack algorithm proposed in this work. The random search approach [40] has been employed to explore different structures for the neural network (in terms of number of layers, and number of neurons per layer) and different configurations of the training phase (the learning rate, weight decay, and momentum factors for the gradient descent algorithm). Table 3.1 shows the range of values explored for each of the hyperparameters: in each experiment, a uniform probability with Min and Max as extremes is sampled for every hyperparameter. In particular, once the number of neurons for the first layer had been selected, the neurons for the subsequent layers are halved, given that a preliminary experimental evaluation proved this architectural choice to be more effective. The number of neurons in the last layer is equal to 2, since there are two classes (NIT and SXT) in the addressed problem. The activation functions employed are the ReLU for all the intermediate layers, and the Softmax for the output layer. This choice led to the adoption of the weights initialization procedure described in [41], which has been proved to be the optimal choice to combine with ReLU layers.

The *f-score* measure [40] has been employed to evaluate the effectiveness of the neural network classification; to be more specific, *f-score* values have been computed for each of the two classes separately, thus by assuming NIT and SXT as the positive class in turn. Then, in order to evaluate the effectiveness of the attack algorithm, the analysis was restricted to the portion of samples of the test set that the neural network identifies correctly, and the error percentage of the

[3]`https://github.com/RafayAK/NothingButNumPy`

Table 3.1: Ranges of values for the hyperparameters explored with Random Search.

Category	Hyperparameter	Min	Max
Network	Number of layers	2	7
	Number of neurons in 1st layer	50	500
Dataset	Number of κ features	20	70
Training	Learning rate α	$1e^{-4}$	$1e^{-3}$
	Weight decay λ	$1e^{-5}$	$1e^{-3}$
	Momentum ν	$1e^{-3}$	$8e^{-1}$

model w.r.t. the corrupted input samples of a specific class has been measured as:

$$error < class >= \frac{|h_\theta(x+\delta) \neq class \;\&\; h_\theta(x) = class = y|}{|h_\theta(x) = class = y|},$$

where h_θ is the hypothesis of the model with the trained parameters θ, x is the set of samples in the test set, δ is the perturbation created with the attack algorithm, and y is the ground truth class.

3.4.2 Results and discussion

50 batches of experiments have been performed where the neural network architecture hyperparameters are sampled from ranges shown in Table 3.1. For each batch, 50 different samplings of training hyperparameters have been explored while keeping fixed the network structure sharing such values in all neurons, thus resulting in a total number of 2500 configurations. Results have been analyzed according to the value of κ; in particular we considered three ranges of values, i.e., κ in $[20; 30]$, $[31; 50]$, and $[51; 70]$. For the sake of clarity, in Table 3.2 the most significant results from each group are presented, while the corresponding hyperparameters are reported in Table 3.3. In particular, Table 3.2 shows the *f-scores* of the selected models w.r.t. the two classes, as well as the error percentage due to the injection of $\gamma \in [1, 2, 3, 4, 5]$ binary feature values into the test data with the proposed attack algorithm. When the dataset is preprocessed in order to select only the $\kappa = 29$ most meaningful features (first row), the proposed attack procedure with $\gamma = 3$ allowed to completely mislead the neural network for all the test data. This result can be due to the extreme sparsity of the AMR-UTI dataset, in which the vast majority of the binary features have value 0. For such a reason, when a binary feature value is flipped from 0 to 1 in the direction of the loss' gradient, it is extremely likely that the new feature becomes "characteristic" for the target class,

Table 3.2: F-score and errors of the best performing experiment in each group.

Exp.	**f-score NIT**	**f-score SXT**	**error NIT**					**error SXT**				
			$\psi = 1$	$\psi = 2$	$\psi = 3$	$\psi = 4$	$\psi = 5$	$\psi = 1$	$\psi = 2$	$\psi = 3$	$\psi = 4$	$\psi = 5$
1	0.80	0.72	0.74	1.00	1.00	1.00	1.00	0.71	0.96	1.00	1.00	1.00
2	0.73	0.70	0.58	0.87	0.97	1.00	1.00	0.78	1.00	1.00	1.00	1.00
3	0.64	0.67	0.31	0.70	0.89	0.96	1.00	0.29	0.60	0.84	0.95	1.00

Table 3.3: Hyperparameters of the most significant experiments.

Experiment	**Neurons**	$\boldsymbol{\kappa}$	$\boldsymbol{\alpha}$	$\boldsymbol{\lambda}$	$\boldsymbol{\nu}$
1	$145, 2$	29	$6.70e^{-4}$	$1.71e^{-4}$	$4.38e^{-1}$
2	$177, 2$	42	$5.40e^{-4}$	$1.36e^{-4}$	$7.49e^{-3}$
3	$215, 2$	59	$5.18e^{-4}$	$3.25e^{-5}$	$7.65e^{-1}$

thus flipping the label with high probability. As the number of features considered increases, an higher quantity of features needs to be perturbed in order to completely subvert the predictions, in particular, for $\kappa = 42$ and $\kappa = 59$ (second and third row of Table 3.2), the best performances of the algorithm are achieved by altering 4 and 5 features respectively.

In order to provide a more in-depth analysis of the features that have actually been altered in the experiments carried out, Figure 3.2 shows the percentage of times that a given feature has been chosen by the proposed algorithm, and the corresponding success rate in deceiving the model. The two heatmaps are computed aggregating the results of the three experiments reported in Table 3.2. It is important to note that the percentages of feature selections depicted in the first heatmap have unitary sum for a fixed value of γ, meaning that the shown set of features contains all the perturbed ones. Instead, the percentages of success due to feature perturbations represented in the second heatmap sum to the error rate of the model, e.g., when $\gamma = 5$ the percentages of success add up to 1, since results shown full model deception in all the experiments by altering 5 features. The most selected feature (38) is related to breathing difficulties, and it has been chosen for the 13.67% of times across both all the experiments and γ values, leading to success in 8.89% of cases w.r.t. the total of all other perturbation attempts. The motivation behind this fact can be traced back to the adverse effects of the two active principles. Indeed, among the side effects of nitrofurantoin assumption[4] there

[4]`https://www.msdmanuals.com/professional/infectious-diseases/bacteria-and-antibacterial-drugs/nitrofurantoin`

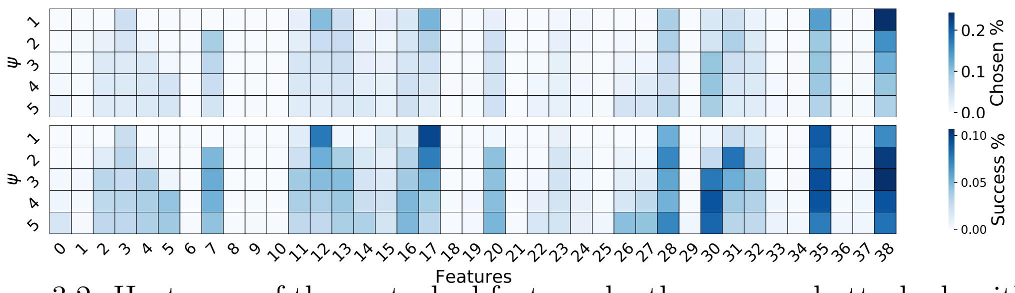


Figure 3.2: Heatmaps of the perturbed features by the proposed attack algorithm, and the respective success percentage in changing the prediction of the model.

is pulmonary toxicity, which is instead absent in trimethoprim-sulfamethoxazole's side effects[5]. In light of this consideration, the proposed approach realized the shrewd behavior of the doctor prescribing SXT treatment for patients who have recently experienced breathing complications.

[5]https://www.msdmanuals.com/professional/infectious-diseases/bacteria-and-antibacterial-drugs/trimethoprim-and-sulfamethoxazole

Chapter 4

Correlations-Aware Perturbations in Spam Account Detection

The high *availability* of OSNs, that is, the fact that they can be easily accessed at any time from anywhere, is the key factor in their success and, at the same time, the main reason for the interest of malicious entities. Since spammers may adopt different strategies to achieve their goal, ML algorithms are usually trained on a wide set of features capable of capturing various aspects, such as information concerning the properties of the account, the history of shared content, and the degree to which a user is connected to the rest of the network. While such a comprehensive set of characteristics allows to identify different types of threats, large feature sets may also extend the attack surface of ML systems, making them more easily deceived.

From an AML point of view, features describing the user of an OSN are closely interrelated (e.g., adding or deleting a message containing a URL would impact multiple feature values at the same time) and the steps required to fool a classifier cannot be made on a trial-and-error basis. In this context, accomplishing an attack means finding a way for the adversary to automatically alter the feature vector describing a *spammer* so that it is recognized as *genuine*, without impairing the malicious behavior.

In order to achieve this goal, here it is proposed *AdverSPAM*[1], an evasion adversarial attack against OSN spam account detection systems that allows to find the optimal perturbation to deceive the target model, while preserving the inter-relationships among the features describing the user behavior.

The remainder of the chapter is organized as follows. Section 4.1 outlines the *OSN* scenario considered as case study. Section 4.2 formalizes the model of the adversary. Section 4.3 describes the algorithm to generate the adversarial perturbation for the spammer account in order to be recognized as genuine. Section 4.4 presents the experimental analysis to validate the proposal. This includes a description of the experimental setup, a preliminary assessment of the attack followed by a comparative analysis with five baselines, a concrete case study with a real example of the modifications made by AdverSPAM, and the evaluation of three mitigation strategies. Discussion on advantages and limitations of the approach is provided in Section 4.5.

4.1 Scenario

OSNs are protected by intelligent systems that are able to identify and block malicious accounts. Usually, the detection algorithms exploit heterogeneous features to describe the different behaviors that spammers may adopt [22]; these can be logically organized into four categories, according to their aims [35]:

- **Metadata-based features** describe the general characteristics of the account, such as its creation date, the geographic location, or the average tweet time. These features can be obtained very easily and can be quite effective in recognizing clear malicious behaviors.

- **Content-based features** are useful to evaluate the quality of the content shared by an account. In order to be effective, spammers need to reach a large number of users; thus, their messages frequently include mentions, hashtags, and URLs. Spam account detection algorithms may look for these elements in order to determine whether an account is genuine or not.

[1]https://github.com/agiammanco94/AdverSPAM

- **Interaction-based features** allow to model the friendship network of the account under analysis. Neighborhood information can be very useful to distinguish influential accounts (characterized by many followers and generally by almost no followings), listener accounts (few followers and many followings), isolated accounts, or even sub-networks that could be used for orchestrated attacks.

- **Community-based features** are able to capture the characteristics of account groups that have similar interests, physical locations, professions or other relevant social aspects. The general idea is that the behavior of an account can be inferred by observing those of the community to which it belongs [35], e.g., an account with a good reputation network is unlikely to be a spammer.

In order to bypass the detection systems, the attacker should make the feature vector describing a spammer resemble that of a genuine user. This goal can be achieved through trial and error strategies, however deceiving the OSN can take a long time, during which actions (e.g., banning or blacklisting) can be taken against the spammer. Altering *metadata*-based features, for instance, would require a shift in the account habits, which is very complicated to achieve. *Content*-based features may be altered by forging ad hoc content in order to re-balance those feature values that might suggest a malicious behavior, e.g., creating new “clean” tweets, or removing those that clearly refer to a spam campaign. These alterations may be implemented manually or automatically, depending on the nature of the user. In either case, a large number of changes are required for the features to undergo a significant change. *Interaction*- and *community*-based features can also be modified by creating new connections within the social network, which is usually done by purchasing followers or followings [85] from third-party providers, by creating fake accounts, or by exchanging followers with other users.

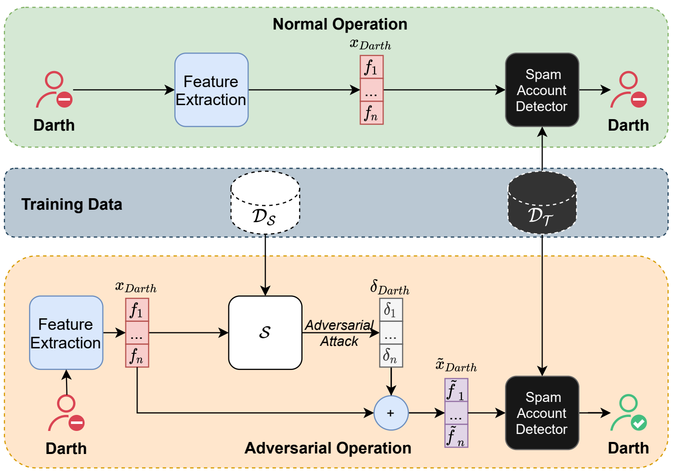


Figure 4.1: The proposed AML strategy in OSN security scenario.

4.2 Threat Model

The proposed attack strategy (see Figure 4.1) follows the general structure of black-box adversarial attacks [61] while capturing the peculiarities of the scenario just described.

The attacker (*Darth*) aims to perform an *integrity* violation of the defense mechanism of the OSN (i.e. *Spam Account Detector*) so as to be *mis*-classified as a genuine user, although showing a typical spammer behavior. The attack specificity is *targeted* since the intent is to hide the spammer behavior to the smart detector.

Even though in a *black-box* scenario the target model $\mathcal{T}$ is not known to Darth, we can make a common assumption [33] by supposing the adversary knows the feature representation $\mathcal{X}$ of the data. The model $\mathcal{T}$ can be queried by Darth, while meeting time constraints and consecutive query limits, in order to collect a training dataset $\mathcal{D}_\mathcal{S}$ for a local *surrogate* model $\mathcal{S}$, which mimics the functioning of the target (unknown) ML system; this configures the proposed approach as a query-limited setting black-box attack [42]. In particular, $\mathcal{D}_\mathcal{S}$ can be obtained by following a strategy known as *mimicry attack* [32, 62], according to which a group of satellite agents is exploited to create a balanced set of spam and genuine accounts. Since Darth knows $\mathcal{S}$, he can build a perturbation vector δ to be added to the original feature vector x so that the perturbed feature vector $\tilde{x}$ is classified

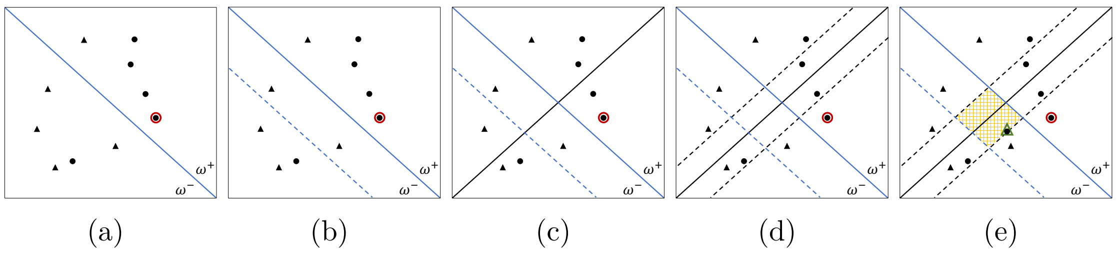


Figure 4.2: Simplified example of binary classification with only two features. (a) The decision boundary $db_{\mathcal{S}}$ (blue line) separates spammers (circles) from genuine users (triangles). The goal of the attack is to project one chosen spammer sample (red) into the opposite region by crossing the decision boundary. (b) The adversarial sample must be generated within a certain distance (dashed blue line) from the decision boundary, which depends on a parameter ψ. (c) The regression line (black) provides a good approximation of the correlation between the spammer samples. (d) In order to preserve the nature of the input, the feasible region (yellow area) for the adversarial sample is further constrained by a margin around the regression line. (e) The adversarial sample (green triangle) is finally computed by solving the optimization problem within the feasible region.

by $\mathcal{S}$ as genuine. Finally, by exploiting the *transferability* property, Darth may obtain that $\tilde{x}$ eludes even the unknown model $\mathcal{T}$.

The computation of δ must satisfy some constraints; indeed, in the scenario considered here, some features may be mutually dependent and therefore not all δ values are valid. Some works, addressing other application scenarios [64], have proposed the generation of perturbations capable of preserving the *statistical* correlation between features; this means that a change in a given feature value should propagate proportionally to the others. However, such a *numerical* dependency, although being able to highlight the hidden relationships between feature values, is not really able to describe the *semantic* dependencies of the elements of the feature set. More specifically, two features are defined to be *semantically dependent* if their computations require one or more common raw data.

The remainder of the section describes the proposed attack algorithm. For the sake of clarity, the adopted notations and abbreviations are listed in Table 2.1.

4.3 Methodology

The requirements discussed so far translate into ensuring the fulfilment of three constraints, namely (i) allowing the computation of an adversarial sample, $\tilde{x}$, beyond the decision boundary of the surrogate model, $\mathcal{S}$, while maintaining the (ii) the *statistical correlation* and (iii) the *semantic dependency* of the features. In order to better explain the meaning of these constraints, the description is complemented with figures that illustrate a simplified classification example in which only two features are employed (see Figure 4.2).

Given the problem of associating an observation $\{x_1, \ldots, x_\kappa\}$ with a class from the binary set $\Omega = \{\omega^-, \omega^+\}$, the goal of the attacker is to take an original sample, $x \in \mathcal{X}$, that lies in the region ω^+ and project it into ω^- so obtaining a perturbed sample $\tilde{x}$. Since low-complexity surrogate models have been shown to transfer attacks more effectively [33], $\mathcal{S}$ it is assumed to be any model based on a *linear* decision function:

$$db_{\mathcal{S}} : \sum_i \alpha_i \, x_i + \beta = 0, \tag{4.1}$$

where α_i and β are the learned coefficients and intercept, respectively. Therefore, crossing the decision boundary $db_{\mathcal{S}}$ means performing a search in one of the two regions (see Figure 4.2a) delimited by Eq. 4.1. To be more specific, every perturbed sample $\tilde{x}$ must satisfy one of the following conditions:

$$\begin{cases} \sum_i \alpha_i \, \tilde{x}_i + \beta > 0 \ \text{ if } \omega^- \text{ is above } db_{\mathcal{S}}, \\ \sum_i \alpha_i \, \tilde{x}_i + \beta < 0 \ \text{ otherwise.} \end{cases} \tag{4.2}$$

It is also useful to choose $\tilde{x}$ on the basis of its distance from the decision boundary. In fact, the farther $\tilde{x}$ is from $db_{\mathcal{S}}$, the greater the probability of evasion success on the target model. On the other hand, points that are too far from the decision boundary would exhibit characteristics too dissimilar from x, resulting in the attack being meaningless. Thus, in order to regulate the maximum allowed distance from the decision boundary (Figure 4.2b) is chosen through a parameter ψ, whose

Algorithm 2 Crossing the Decision Boundary

```
Input:
    $\mathcal{S}$: Surrogate model to attack;
    $\mathcal{X}$: The set of input samples for $\mathcal{S}$.
Output:
    $db_c$: The decision boundary constraint;
    $db_\psi$: The constraint on the parallel to $db_\mathcal{S}$.
 1: $db_\mathcal{S} \leftarrow \mathcal{S}$.getDecisionBoundary()
 2: $[\alpha_i, \beta] \leftarrow db_\mathcal{S}$.getParameters()
 3: $x \leftarrow \mathcal{X}$.getRandomSample()
 4: $\hat{y} \leftarrow \mathcal{S}$.predict($x$)
 5: $\psi \leftarrow$ getOffset($db_\mathcal{S}$)
 6: if test($x, [\alpha_i, \beta]$) $\geq 0$ then
 7:     if $\hat{y} \in \omega^-$ then
 8:         $db_c \leftarrow$ "$\sum_i \alpha_i \tilde{x}_i + \beta > 0$"
 9:         $db_\psi \leftarrow$ "$\sum_i \alpha_i \tilde{x}_i + \beta - \psi < 0$"
10:     else
11:         $db_c \leftarrow$ "$\sum_i \alpha_i \tilde{x}_i + \beta < 0$"
12:         $db_c \leftarrow$ "$\sum_i \alpha_i \tilde{x}_i + \beta + \psi > 0$"
13: else if test($x, [\alpha_i, \beta]$) $< 0$ then
14:     if $\hat{y} \in \omega^-$ then
15:         $db_c \leftarrow$ "$\sum_i \alpha_i \tilde{x}_i + \beta + \psi < 0$"
16:         $db_\psi \leftarrow$ "$\sum_i \alpha_i \tilde{x}_i + \beta + \psi > 0$"
17:     else
18:         $db_c \leftarrow$ "$\sum_i \alpha_i \tilde{x}_i + \beta - \psi > 0$"
19:         $db_\psi \leftarrow$ "$\sum_i \alpha_i \tilde{x}_i + \beta - \psi < 0$"
20: return $db_c, db_\psi$
```

sign depends on the position of ω^- with respect to the $db_\mathcal{S}$:

$$\begin{cases} \sum_i \alpha_i \tilde{x}_i + \beta - \psi < 0 & \text{if } \omega^- \text{ is above } db_\mathcal{S}, \\ \sum_i \alpha_i \tilde{x}_i + \beta + \psi > 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

The procedure that implements this last constraint is described by Algorithm 2.

The search in the region beyond the decision boundary is further driven by the need to ensure that the *statistical correlation* and the *semantic dependency* of the features originally extracted from x are preserved in the forged adversarial sample $\tilde{x}$. These two properties are defined as follows.

Definition 1. *Statistical Correlation: two features are statistically correlated if having a strong linear relationship with each other.*

Definition 2. *Semantic Dependency: two features are semantically dependent if their computations require one or more common raw data.*

In *AdverSPAM*, the *statistical correlation* among features is leveraged by considering the regression line fitting the data. In particular, since the aim of the adversary is to preserve the core properties of the *spammers*, only the samples of the positive class ω^+ are considered (see Figure 4.2c). Formally, the regression line regarding features j and i for all samples $x \in \omega^+$, is defined by:

$$\mathcal{R}^{\omega^+}_{j,i}(x_i) = m^{\omega^+}_{j,i} x_i + q^{\omega^+}_{j,i}\,, \tag{4.4}$$

where $m^{\omega^+}_{j,i}$ and $q^{\omega^+}_{j,i}$ represent its slope and intercept, respectively. In order to maintain the *statistical correlation* between the features, it is therefore necessary that the manipulated sample does not deviate too far from $\mathcal{R}^{\omega^+}_{j,i}$. Hence, $\tilde{x}$ must lie within a certain *margin* from this line (see Figure 4.2d), that it is computed as the squared root of the Residual Sum of Squares (RSS) of the regression model:

$$margin_{j,i} = \sqrt{\sum_{x \in \mathcal{X}} (\mathcal{R}^{\omega^+}_{j,i}(x_i) - x_j)^2}, \tag{4.5}$$

where the squared root is adopted for dimensional homogeneity. When RSS is approximately 0, the regression line is a good predictor of the data, resulting in a particularly tight margin of allowable displacement; conversely, a high value of RSS means $\mathcal{R}^{\omega^+}_{j,i}$ is an unreliable model of the data, thus leading to a wide margin of possible perturbations. In other words, this margin actually outlines the minimum and maximum perturbations permitted on the j-th feature of $\tilde{x}$:

$$\begin{cases} \tilde{x}_j \leq m^{\omega^+}_{j,i}\,\tilde{x}_i + q^{\omega^+}_{j,i} + (margin_{j,i}) \\ \tilde{x}_j \geq m^{\omega^+}_{j,i}\,\tilde{x}_i + q^{\omega^+}_{j,i} - (margin_{j,i}). \end{cases} \tag{4.6}$$

Furthermore, the notation $sd(i, j)$ is adopted to indicate whether a *semantic dependency* between the features i and j exists. As stated in *Definition 2*, such a relationship occurs when multiple features capture similar traits of the account, so requiring the same information (e.g., the number of followers, or the amount of

Algorithm 3 Maintaining Correlations and Dependencies

```
Input:
    $\mathcal{X}$: The set of input samples for $\mathcal{S}$.
Output:
    $\tilde{\mathcal{C}}$: List of constraints.
 1: $\mathcal{C} \leftarrow []$
 2: for all $x \in \mathcal{X}$ do
 3:     for all $i \in \mathcal{F}$ do
 4:         for all $j \in \mathcal{F}$ do
 5:             $sd \leftarrow$checkSemanticDependence($i$, $j$)
 6:             if $i \neq j$ and $sd == True$ then
 7:                 $\mathcal{R}^{\omega^+}_{j,i} \leftarrow$ getRegressionLine($\mathcal{X}$, $j$, $i$)
 8:                 $[m^{\omega^+}_{j,i}, q^{\omega^+}_{j,i}] \leftarrow \mathcal{R}^{\omega^+}_{j,i}$.getParameters()
 9:                 $margin_{j,i} = \mathrm{sqrt}(\sum_{x \in \mathcal{X}} (\mathcal{R}^{\omega^+}_{j,i}(x_i) - x_j)^2)$
10:                 $\mathcal{C}$.add("$\tilde{x}_j \leq m^{\omega^+}_{j,i}\ \tilde{x}_i + q^{\omega^+}_{j,i} + (margin_{j,i})$")
11:                 $\mathcal{C}$.add("$\tilde{x}_j \geq m^{\omega^+}_{j,i}\ \tilde{x}_i + q^{\omega^+}_{j,i} - (margin_{j,i})$")
12: return $\mathcal{C}$
```

posted URLs) to be computed. Therefore, the maintenance of *statistical correlation* is strictly related to the existence of *semantic dependency*:

$$\begin{cases} \tilde{x}_j \leq m^{\omega^+}_{j,i}\ \tilde{x}_i + q^{\omega^+}_{j,i} + (margin_{j,i}) & \forall i,j\ s.t.\ sd(i,j) = true, \\ \tilde{x}_j \geq m^{\omega^+}_{j,i}\ \tilde{x}_i + q^{\omega^+}_{j,i} - (margin_{j,i}) & \forall i,j\ s.t.\ sd(i,j) = true. \end{cases} \tag{4.7}$$

Conversely, features that are not in a direct cause-effect relationship can be manipulated independently of each other. The steps required to compute these last constraints are described in Algorithm 3.

Once the feasible region has been defined (Figure 4.2e), the attack proceeds by determining the *best* adversarial sample within it. This corresponds to finding a set of new feature values which pursues a twofold purpose. Firstly, the overall distance of the adversarial sample $\tilde{x}$ from the input x has to be limited, in order to ensure that the (malicious) nature of x is not actually altered. Secondly, the adversarial sample has to be transferable, that is, $\tilde{x}$ has to be general enough to evade the unknown target classifiers. These two aspects have been modeled as a weighted sum [29] of two objectives controlled by a factor $\lambda \in [0, 1]$:

$$z(\tilde{x}, x) : \lambda L_2(\tilde{x}, x) + (1 - \lambda) L_2(\tilde{x}, x_{\psi_\perp}), \tag{4.8}$$

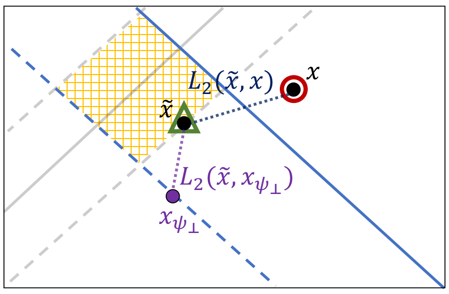


Figure 4.3: The objective function conjugating the minimisation of the distance imposed over the adversarial samples and the maximization of the evasion capability against unknown classifiers.

where $x_{\psi_\perp}$ is the projection of the input sample x over the parallel to the decision boundary.

In order to understand the meaning of this objective function, let us consider Figure 4.3. If the need of the adversary is to evade the local surrogate model with the minimum effort possible, then, setting $\lambda = 1$ allows to only perform the minimization of the euclidean distance between the input and the adversarial sample. Conversely, when the adversary wants to deceive unknown black-box models, since their decision boundaries may greatly differ from the one of the local surrogate, a reasonable amount of additional distance has to be imposed over the adversarial sample; this can be achieved by setting $\lambda = 0$. Any other value of $\lambda \in [0, 1]$ will constitute a trade-off between such two opposing situations. Hence, *AdverSPAM* calculates the final adversarial sample by solving the following

Algorithm 4 AdverSPAM

```
Input:
    $\mathcal{S}$: Surrogate model to attack;
    $\mathcal{X}$: The set of input samples for $\mathcal{S}$;
Output:
    $\tilde{\mathcal{X}}$: The set of adversarial samples.
 1: $\tilde{\mathcal{X}} \leftarrow []$
 2: $db_c, db_\psi \leftarrow$ Algorithm1($\mathcal{S}$, $\mathcal{X}$)
 3: for all $x \in \mathcal{X}$ do
 4:     $\mathcal{C} \leftarrow []$
 5:     $\mathcal{C}$.append($db_c$)
 6:     $\mathcal{C}$.append($db_\psi$)
 7:     $\mathcal{C}$.append(Algorithm2($\mathcal{X}$))
 8:     $z \leftarrow$ getCostFunction($\tilde{x}$, $x$)
 9:     $\tilde{x} \leftarrow$ solveoptimizationProblem($'min'$, $x$, $z$, $\mathcal{C}$)
10:     $\tilde{\mathcal{X}}$.append($\tilde{x}$)
11: return $\tilde{\mathcal{X}}$
```

optimization problem:

$$\begin{aligned}
&\tilde{x} = \min_{\tilde{x}} \; \lambda L_2(\tilde{x}, x) + (1-\lambda) L_2(\tilde{x}, x_{\psi_\perp}),\\
&s.t.\\
\text{(as reported in Eq. 4.2):}\quad &\sum_i \alpha_i \tilde{x}_i + \beta > 0 \quad \textbf{or} \quad \sum_i \alpha_i \tilde{x}_i + \beta < 0\\
\text{(as reported in Eq. 4.3):}\quad &\sum_i \alpha_i \tilde{x}_i + \beta - \psi < 0 \quad \textbf{or} \quad \sum_i \alpha_i \tilde{x}_i + \beta + \psi > 0 \qquad (4.9)\\
\text{(as reported in Eq. 4.7):}\quad &\tilde{x}_j \le m^{\omega^+}_{j,i}\, \tilde{x}_i + q^{\omega^+}_{j,i} + (margin_{j,i}) \quad \forall i,j \; s.t. \; sd(i,j) = true\\
&\tilde{x}_j \ge m^{\omega^+}_{j,i}\, \tilde{x}_i + q^{\omega^+}_{j,i} - (margin_{j,i}) \quad \forall i,j \; s.t. \; sd(i,j) = true\\
&\tilde{x} \in [0,1]^n
\end{aligned}$$

The entire procedure is described by Algorithm 4, which exploits the other two algorithms discussed in this section. In the realized implementation, the problem was solved using COBYLA [67], which operates iteratively by generating local linear approximations of the objective function and constraints. The solution $\tilde{x}$ is searched in $[0,1]^n$, which represents the range of admissible values for the feature set of n elements that will be described in the next section.

Table 4.1: The list of features used to model the accounts.

Category	Acronym	Description
Metadata	RR	Retweet Ratio
	AR	Automated Tweet Ratio
	TSD	Tweet Time Standard Deviation
	TISD	Tweet Time Interval Standard Deviation
Content	UUR	Unique URL Ratio
	UMR	Unique Mention Ratio
	CHS	Content and Hashtag Similarity Ratio
	UR	URL Ratio
	MR	Mention Ratio
	HTR	Hashtag Ratio
	AUR	Automated Tweet URL Ratio
	ATS	Automated Tweet Similarity
Interaction	FR	Follower Ratio
	MFFFR	Mean Follower's Followings to Follower Ratio
	FBR	Follower-based Reputation
	R	Reputation
	CC	Clustering Coefficient
Community	CBR	Community-based Reputation
	CBCC	Community-based Cluster Coefficient

4.4 Experimental Analysis

The effectiveness of the proposed technique, whose code is publicly available [2], has been evaluated in different steps. The first part of this section describes the experimental setup, the metrics adopted for the assessment, as well as the tuning of *AdverSPAM* internal parameters. Then, comparisons with five state-of-the-art attack techniques are presented, discussing both the performance obtained and the quality of the forged adversarial samples. The section continues with the introduction of a concrete case study aimed at exploring the manipulations that should be applied to a real OSN account in order to accomplish the considered attacks. The results of adopting three mitigation strategies follow; finally, advantages and limitations of *AdverSPAM* are outlined.

4.4.1 Experimental Setup

The choice of the classifiers to adopt as *target* and *surrogate* models was driven by the analysis of the literature in the field of AML. A preliminary consideration concerned whether to select ML or Deep Learning methods. When low-dimensional features spaces are considered, it has been shown that classical ML algorithms can produce superior results, which also tend to be better interpretable than those based on deep neural networks [43]. Conversely, the latter are recommended especially in domains characterized by large, high-dimensional data, such as image, video, and audio data processing [27]. Hence, this work focused on five classifiers that are the most commonly chosen for the final assessment of adversarial attacks [9, 75], namely: Neural Network (NN), Support Vector Machine (SVM), Logistic Regression (LR), Ridge Regression (RR), and Random Forest (RF). All the considered models has been examined as possible targets, while only SVM (with linear kernel) and LR were chosen as surrogates since they satisfy the requirement of linear decision boundary needed in *AdverSPAM*.

Moreover, since the attack transferability strongly depends on the complexity of the target model [33], this assessment considers high-complexity (H) and low-complexity (L) variants. The complexity of a ML model is measured by the number of hyperparameters it has. In general, a model characterized by a large set of hyperparameters (high degree of complexity) may be able to capture more variations in the data, but it will also be more difficult to train and may be more prone to overfitting. Conversely, a low complexity model may be easier to train, but may not be able to capture all the relevant information in the data.

In order to properly tune the parameters of the classifiers, a 10-fold Cross-Validation has been performed with the objective of letting all the models achieve a f-score of about 90%. To this aim, a public dataset [35] consisting of 10.000 *genuine* users and 1.000 *spammers*, each described by a set of 19 features (see Table 4.1), was used. Being this dataset unbalanced, we employed *SMOTE* augmentation technique [15] so as to obtain a new dataset of 10.000 users per class. This was split with 80:20 ratio leading to 16.000 accounts for the training and 4.000 for the test set. The former was further split in two parts representing the datasets

[2]https://github.com/agiammanco94/AdverSPAM

Table 4.2: The surrogate and target models used in the experiments, and the corresponding tuned parameters. For all the models, low (L) and high (H) complexity variants are evaluated. Furthermore, two different SVM kernels are considered, namely, linear (lnr) and radial basis function (rbf).

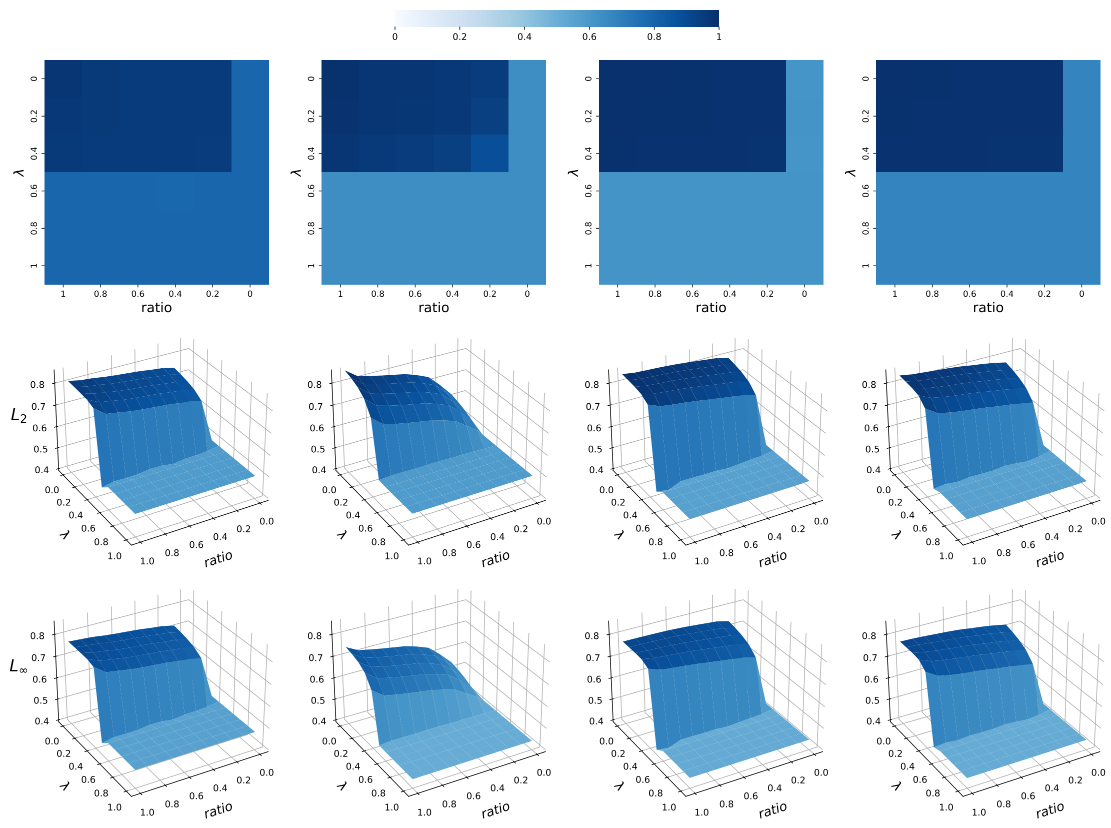
Surrogates	SVM^L_{lnr}, SVM^H_{lnr}, LR^L, LR^H	
Targets	RF^L, RF^H, NN^L, NN^H, SVM^L_{lnr}, SVM^H_{lnr}, SVM^L_{rbf}, SVM^H_{rbf}, LR^L, LR^H, RR^L, LR^H	
Parameters	RF^L	trees = 30; max_depth = 8
	RF^H	trees = 30; max_depth = no-limit
	NN^L	learning_rate = 0.01; weight_decay = 0.01; neurons_layers = [50, 50, 2]
	NN^H	learning_rate = 0.01; weight_decay = 0; neurons_layers = [50, 50, 2]
	SVM^L_{lnr}	C = 1
	SVM^H_{lnr}	C = 100
	SVM^L_{rbf}	C = 1
	SVM^H_{rbf}	C = 100
	LR^L	C = 1
	LR^H	C = 10
	RR^L	$\alpha = 10$
	RR^H	$\alpha = 1$

$\mathcal{D}_{\mathcal{S}}$ and $\mathcal{D}_{\mathcal{T}}$ (see Figure 4.1), each containing 4.000 *genuine* and 4.000 *spammer* accounts. Finally, $\mathcal{D}_{\mathcal{T}}$ was also used to perform the adversarial training described in Section 4.4.5.

A summary of the adopted models and their tuned parameters is provided in Table 4.2.

4.4.2 AdverSPAM Assessment

The behavior of *AdverSPAM* is mainly influenced by two parameters, namely ψ (Eq. 4.3) and λ (Eq. 4.8). Given the definition of ψ, it is calculated as the value that guarantees a desired percentage (*ratio*) of samples of the opposite class ω^- lies between the decision boundary and its shift by a quantity ψ. The higher the *ratio*, the greater the probability that the adversarial sample will be located in a region with a higher density of ω^- samples. On the other hand, greater distances



(a) SVM^L_{lnr} (b) SVM^H_{lnr} (c) LR^L (d) LR^H

Figure 4.4: AdverSPAM tuning. Mean FNRs over the 12 targets (top row), and L_2 (middle) and L_∞ (bottom) distances measured while varying the parameters *ratio* and λ for every surrogate model (columns).

from the decision boundary would lead to an over-distortion of the original sample. For this reason, the choice of the best ψ value was determined on the basis of a set of evaluation metrics.

In particular, being the main goal of the algorithm to create a perturbed sample capable of deceiving the target model, a good measure of its effectiveness is the percentage of actual *spammers* that are misclassified as *genuines*. This information is provided by the *False Negative Rate* (FNR), which is defined as the ratio between false negatives and true positives. Moreover, the L_2 and L_∞ distance norms, can be exploited to evaluate the distortion introduced in the adversarial samples.

Tests were run on four surrogate models while varying both the *ratio* and the value of λ, i.e., the weight of the two components of the objective function. Results are shown in Figure 4.4, organized as a matrix in which each column refers to a different surrogate. The first row summarizes the mean FNR values obtained on the 12 target models for every pair $(ratio, \lambda)$; highest values are represented with darker colors. The 3-D plots in the second and third rows reveal how varying the parameters *ratio* and λ impacts on the values of L_2 and L_∞. As a general trend we can notice that both the FNRs and the distance values grow proportionally to *ratio*, but inversely to λ. Indeed, small values of λ (i.e., $\lambda \leq 0.4$) cause the objective function to push the adversarial sample away from the decision boundary, which leads to both a greater transferability and higher peaks in the distance metrics.

As the considered surrogate model varies, the specific optimal values of the parameters change accordingly. In particular, by analyzing the last two columns of Figure 4.4 (i.e., LR^L and LR^H models), it can be noticed that the best FNRs values are obtained when the $ratio \in [0.2, 1]$ and $\lambda \in [0, 0.4]$. Considering the L_2 and L_∞ values measured in these ranges, a good trade-off between success rate and perturbation degree is reached by choosing $ratio = 0.2$ and $\lambda = 0.4$. The same assessment can be made for the SVM^L_{lnr} and SVM^H_{lnr} models, resulting in $ratio = 1$ and $\lambda = 0.2$. The obtained *ratio* values correspond to ψ equals to 1.91, 5.44, 3.78, and 5.47 for SVM^L_{lnr}, SVM^H_{lnr}, LR^L, and LR^H, respectively.

Further experiments were carried out in order to evaluate the capability of *AdverSPAM* to preserve the statistical correlations of the features. In literature, statistical correlation is expressed in terms of linear correlation between pairs of features; an indicator typically used in this regard is the Pearson's coefficient [6, 49, 47, 82]. In order to assess what type of correlation exists between the 19 features considered, some preliminary tests were carried out on the original dataset by representing all the pairs of features in a two-dimensional space and fitting them with polynomials of degree from 1 to 4. For each pair, the Mean Square Deviation (MSD) of the points w.r.t. the fitting polynomials was calculated as an indicator of approximation quality. The results shown in Table 4.3 indicate that polynomials of a higher degree correspond to a smaller fitting error, which is intrinsic in the fact that the higher the degree the greater the freedom in fitting the data. Thus, the variance of the deviations was also computed, which should decrease if a certain

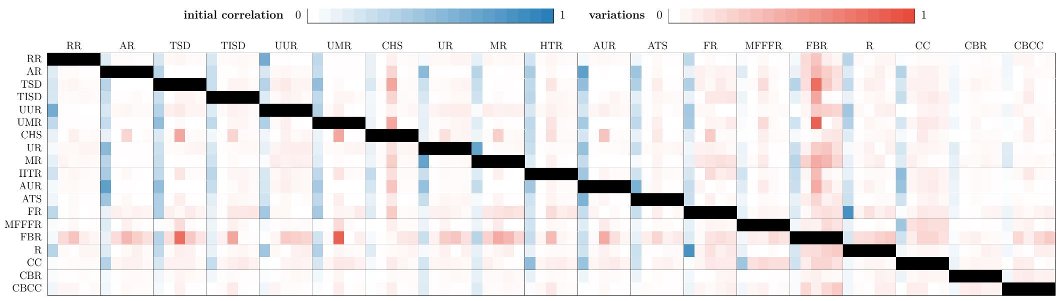
Table 4.3: Average and variance of the Mean Square Deviation (MSD) computed by different degree polynomials fitted on the features of the *spammer* class.

metric \ degree	I	II	III	IV
avg(MSD)	.032	.029	.028	.027
var(MSD)	.002	.002	.002	.002

curve was actually able to better follow the distribution of the data. The results, instead, indicate that the variance is stable as the degree of the polynomial changes; thus, it can be concluded that linear dependency fairly accurately represents the distribution of the data.

The correlation matrix of the feature set before and after the adversarial attack was performed is shown in Table 4.4, where, for each pair of features, the following 5 values (columns) are reported: original Pearson correlation coefficients (blue); variations of these coefficients (red) after creating the adversarial samples with SVM^L_{lnr}, SVM^H_{lnr}, LR^L, and LR^H, respectively. As a general rule, the stronger the correlation between the features, the smaller the changes in Pearson's coefficients caused by the attack should be. Thus, if correlations are maintained, light-red colored cells are expected in correspondence with dark-blue ones, and vice versa; this trend is clearly visible for almost any feature in Table 4.4. Moreover, because of the constraint in Eq. 4.7, features that are not semantically dependent on each other are more likely to be strongly modified. This is particularly evident when looking at FBR (Follower-based Reputation), which belongs to the *interaction* category (see Table 4.1). Features ranging from RR to ATS are not semantically dependent with FBR because they belong to different categories; moreover, they are lowly correlated with FBR as indicated by the corresponding Pearson coefficients. Hence, these features are good candidates for manipulation. This is indeed confirmed by the results, which show greater variations (darker red values) in the rows from RR to ATS, regardless of the surrogate model used.

Table 4.4: Correlation matrix. For each feature, the following 5 values (columns) are shown: Pearson correlation coefficients (blue) measured on the samples of the spammer class before the attack is launched; variations of these values (red) after creating the adversarial samples with SVM^{L}_{lnr}, SVM^{H}_{lnr}, LR^{L}, and LR^{H}, respectively.



4.4.3 Comparing AdverSPAM with the baselines

State-of-the-art AML attacks can be classified into two main categories, namely white-box and black-box [34, 69]. In order to cover both classes, AdverSPAM was compared with both gradient-based white-box attacks (i.e., *FGSM*, *DF*, and *C&W*), and decision-based black-box attacks (i.e., *Cheng* and *Peng*). While *FGSM*, *DF*, and *C&W* attacks are widely adopted in the literature for the comparative analyzes [86, 63, 88, 19, 44, 91, 7, 31, 59], the other two methods are less known but no less important. In particular, *Peng* is the only technique that addresses the problem of statistical correlation between features, just like AdverSPAM. Thus, the comparative analysis is based on the following baselines:

- *Fast Gradient Sign Method* (**FGSM**) [39], where the adversarial perturbation is computed considering the sign of the targeted classifier loss function's gradient, and projecting the adversarial sample on a sphere of radius ϵ_{FGSM} around the input sample.

- *DeepFool* (**DF**) [57], where the adversarial sample is shifted along the direction of the gradient of the target model loss w.r.t. the input, until the decision boundary is crossed. The obtained perturbation is then scaled by a factor $(1 + \eta_{DF})$, which is useful for getting adversarial samples farther from the decision boundary.

Table 4.5: The chosen parameters for the adopted algorithms.

	FGSM	**DF**	**C&W**
Parameter / Model	ϵ_{FGSM}	η_{DF}	$c_{C\&W}$
SVM^{L}_{lnr}	0.37	0.35	0.5
SVM^{H}_{lnr}	0.42	0.75	0.5
LR^{L}	0.45	0.5	0.85
LR^{H}	0.45	0.2	0.95

- *Carlini and Wagner* L_2 (**C&W**) [13], where an optimization problem is formulated with a loss function leveraging the representation in the *logits* layer (the layer prior to the final softmax layer, containing the probabilities that a sample belongs to a specific class) as a measure of attack success. The properties of such a loss function depend on a regularization parameter $c_{C\&W} > 0$, which controls the confidence that the adversarial sample belongs to the opposite class. When the attacked model is not a Neural Network, the equivalent of the logits layer is represented by the class memberships probability.

- *Cheng et al. method* (**Cheng**) [17], where the model to attack is considered as a black box, whose decision boundary is estimated by considering the vector connecting the input sample to a sample belonging to the desired class. The adversarial perturbation is computed taking incremental steps along such direction until the decision boundary is crossed; then, a gradient descent procedure is followed in order to consider the direction minimizing the euclidean distance between the input and the adversarial sample.

- *Peng et al. method* (**Peng**) [64], which is an extension of the *Cheng* method where the optimal direction on which to project the adversarial sample is searched by minimizing the Mahalanobis distance, thus preserving the statistical correlations between features.

The methods just introduced require a calibration phase to tune the corresponding parameters. Similarly to the analysis described in Section 4.4.2, they were computed by maximizing the FNRs of the target models, while minimizing the distance metrics. The chosen values are listed in Table 4.5, in which *Cheng* and *Peng* methods are omitted because their functioning parameters are indicated in the respective original works.

Table 4.6 reports the transfer matrices of the six attacks launched against twelve targets (columns), while using the surrogates previously discussed. For each subtable, the last column and row indicate the average transfer rate of the attacks and the average FNR against every target, respectively.

A first consideration can be made about the performance achieved using SVMs and LRs. Indeed, basing the attacks on logistic regression models, regardless of their complexity, results in a stronger transferability capacity than SVMs, as revealed by the values highlighted in green. This corroborates the thesis that simpler sources of adversarial samples are preferable for transferring against unknown models [33], as they are less specialized and thus less prone to drive the perturbations toward regions which will not translate in miss-classification against black boxes.

By analyzing Table 4.6 by columns, it is also possible to observe some dependency between the measured FNRs and the complexity of the target models. In particular, high-complexity targets seem to be less resistant to transferred adversarial samples, as pointed out by the average values highlighted in pink. For instance, when the surrogate is SVM^L_{lnr}, the average FNRs against NN^L and NN^H are 0.48 and 0.81, respectively. This could depend on the learned decision curve of higher complexity targets, which is highly fitted (very close) to the input data; thus, a small perturbation on the input sample is often sufficient to cross the decision boundary. The same consideration applies to almost all surrogates and targets, except for a few cases where the performance of the low- and high-complexity versions are equivalent. Among these are the FNRs measured against Random Forest (RF), which are quite low (in average) for all attacks.

Nevertheless, the transfer capacity of AdverSPAM over *RF* is about 80% for SVM^L_{lnr} and 90% considering all other surrogates. These values are better than all competitors except *FGSM* and *DF*. The reason is that *RF* does not use a differentiable learning function, but rather relies on a set of decision trees that establish the class of a sample based on the values assumed by individual features. Thus, if a group of features is altered significantly (i.e., exceeds the threshold value learned by the tree), the probability of misclassification is very high. This

Table 4.6: Transfer matrices (FNR values) of five attack strategies compared with AdverSPAM exploiting four different surrogate models, namely low and high complexity SVMs (a) and LRs (b). Each attack is carried out against 12 target models (columns).

		NN^L	NN^H	RF^L	RF^H	SVM^L_{lnr}	SVM^H_{lnr}	SVM^L_{rbf}	SVM^H_{rbf}	LR^L	LR^H	RR^L	RR^H	*average*
$\mathbf{SVM^L_{lnr}}$	AdverSPAM	1	1	.81	.81	1	1	1	1	1	1	1	1	.97
	FGSM	.56	1	1	1	.99	1	.19	.99	.68	1	.92	1	.86
	DF	1	.98	.74	.74	1	.99	1	1	1	1	1	1	.95
	C&W	.27	.69	.03	.03	.73	.73	.10	.73	.42	.73	.71	.72	.49
	Cheng	.02	.73	.43	.43	1	.98	0	1	0	.99	1	1	.63
	Peng	.03	.48	.37	.37	.63	.58	0	.63	0	.63	.25	.57	.38
	average	.48	.81	.56	.56	.89	.88	.38	.89	.52	.89	.81	.88	
$\mathbf{SVM^H_{lnr}}$	AdverSPAM	1	1	.91	.91	1	1	.98	1	1	1	1	1	.98
	FGSM	.76	1	1	1	1	1	.35	1	.94	1	.99	1	.92
	DF	1	.99	.93	.93	1	1	.98	1	1	1	1	1	.99
	C&W	.19	.36	.03	.03	.29	.68	.10	.34	.18	.36	.26	.66	.29
	Cheng	0	.51	.48	.48	.88	.93	0	.95	0	.55	.54	1	.53
	Peng	0	0	.06	.06	0	.02	0	.05	0	0	0	0	.02
	average	.49	.64	.57	.57	.70	.77	.40	.72	.52	.65	.63	.78	

(a)

		NN^L	NN^H	RF^L	RF^H	SVM^L_{lnr}	SVM^H_{lnr}	SVM^L_{rbf}	SVM^H_{rbf}	LR^L	LR^H	RR^L	RR^H	*average*
$\mathbf{LR^L}$	AdverSPAM	1	.99	.91	.91	.99	.99	1	.99	1	1	1	1	.98
	FGSM	1	1	1	1	.80	1	.95	.72	1	1	1	1	.96
	DF	1	.99	.99	.99	1	.99	1	.99	1	1	1	.99	1
	C&W	.88	.97	.42	.42	.93	.99	.98	.90	.99	.99	.99	.99	.87
	Cheng	.80	.95	.68	.68	.99	.99	.33	.99	.99	.99	.99	.99	.86
	Peng	.23	.23	.15	.15	.23	.23	0	.23	.23	.23	.23	.23	.20
	average	.82	.86	.69	.69	.82	.87	.71	.80	.87	.87	.87	.87	
$\mathbf{LR^H}$	AdverSPAM	1	1	.90	.90	.99	.99	1	.99	1	1	1	1	.98
	FGSM	.99	1	1	1	.86	1	.96	.78	1	1	1	1	.97
	DF	1	1	1	1	1	1	1	1	1	1	1	1	1
	C&W	.91	.97	.44	.44	.96	.98	.97	.91	.99	.99	.99	.99	.88
	Cheng	.26	.99	.59	.59	1	1	0	1	.02	1	1	1	.70
	Peng	.40	.41	.23	.23	.44	.43	0	.44	.44	.44	.44	.44	.36
	average	.76	.90	.69	.69	.88	.90	.66	.85	.74	.91	.91	.91	

(b)

is the case of *FGSM* and *DF*, that apply to each feature a perturbation equal to ϵ_{FGSM} and proportional to η_{DF}, respectively. Then, the altered feature values differ greatly from the original ones (defined in $[0, 1]$), so producing a high transfer rate.

By observing the results of the *C&W* attack, it can be noticed that this is quite effective when the surrogate is LR, while it frequently fails in the case of SVMs. This can be explained because the decision boundaries learned from Logistic models are farther from the ω^+ samples than those computed by SVMs; this results in higher L_2 values and thus greater transferability of the samples. The low performances of both *Cheng* and *Peng* methods depend on the idea at their basis: in order to moderately perturb the input, they generate adversarial samples that slightly cross the decision boundary. This impacts on transferability as the examples will fail to evade the target model whenever its decision boundary is different from that of the surrogate.

Actually, FNRs alone are not good predictors of the attack's performance as excessively altered samples could still lead to high transfer rates. To better investigate this aspect, we measured the quality of the manipulated samples in terms of their distance from the original ones. The results shown in Figure 4.5 summarize the average L_2 and L_∞ values calculated for the six attacks, while varying the underlying surrogate model.

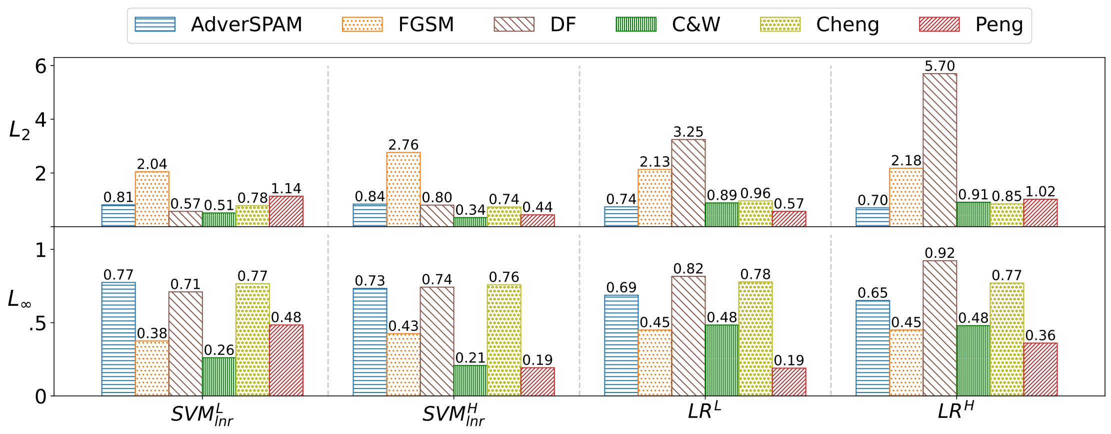


Figure 4.5: The average L_2 and L_∞ distances calculated for the six attacks while varying the surrogate models.

By considering the results of Figure 4.5 and Table 4.6 together, it can noticed that the two methods having FNRs comparable with $AdverSPAM$ are characterized by $L2$ values that are always worse than the proposed attack (as in the case of *FGSM*), or highly variable depending on the surrogate used (*DF*). On the contrary, *AdverSPAM* exhibits fairly controlled L_2 variations for every surrogate; this is due to the intrinsic nature of the proposed algorithm that adjusts the maximum allowed distance of the adversarial samples through the ψ parameter, which is automatically tuned on the specific surrogate.

As regards L_∞, the values measured on *DF* and *Cheng* are comparable with *AdverSPAM*, and in some cases higher (i.e., worse); conversely, the distances obtained by applying *FGSM*, *Peng*, and *C&W* attacks are generally lower. These differences depend on how each approach defines the search region for the adversarial sample. For instance, the L_∞ distances observed for *FGSM* and *DF* are very similar to the values of ϵ_{FGSM} and η_{DF}. The same considerations apply to *C&W* and *Cheng* approaches, whereas a more thorough evaluation is required for *Peng*, which exhibits the lowest L_∞ among all the baselines and is the only method explicitly accounting for feature correlation maintenance.

In order to further examine this aspect, a final comparative analysis was carried out between *AdverSPAM* and *Peng* to measure their ability to preserve the *statistical correlation* between features. As indicator, the Frobenius norm has been

Table 4.7: Comparing the correlation distances induced by *AdverSPAM* and *Peng* methods. The best values are shown in **bold**, whereas the second best values in *italics*. In the header of the table, a single **s** stands for statistical correlation preservation, whereas **ss** signals both statistical and semantic maintainment.

Surrogate \ Attack	**AdverSPAM (s)**	**AdverSPAM (ss)**	**Peng (s)**
SVM^{L}_{lnr}	*1.28*	**0.49**	3.59
SVM^{H}_{lnr}	*2.15*	**1.11**	2.57
LR^{L}	*1.27*	**0.91**	3.06
LR^{H}	*1.08*	**0.89**	3.71
average	*1.45*	**0.85**	3.23

chosen:

$$||A - B||_F = \sqrt{\sum_i \sum_j |a_{i,j} - b_{i,j}|^2}, \tag{4.10}$$

where A and B are the Pearson correlation matrices of the features in the actual *spammers* and in the generated adversarial samples, respectively. Furthermore, in order to highlight the contribution of the *semantic dependency*, the *AdverSPAM* analysis is differentiated by whether *statistical correlation* alone, or both properties are considered. Table 4.7 summarizes the obtained results, where higher Frobenius values correspond to stronger changes in the correlations among features of the adversarial samples. It can be observed that *AdverSPAM* exhibits a lower average correlation distance w.r.t. *Peng* (1.45 vs 3.23); this also holds for the single chosen surrogate models. Moreover, when the semantic dependencies are considered, the distortion is even lower (0.85).

In conclusion, AdverSPAM provides better average FNR values than the considered baselines, which translates to higher success rates on most of the tested target models. When the performances are equivalent to the competitors, the analysis of distance metrics suggests that AdverSPAM perturbs the samples less drastically, which is a desired property for any AML strategy. Finally, maintaining statistical correlation and semantic dependency helps to keep the core properties of the feature set unchanged before and after the attack.

Table 4.8: Example of features describing a spammer account. Original values (first row) and variations produced by AdverSPAM and the five baselines. Stronger differences are highlighted with darker colors.

Features	*Metedata*				*Content*								*Interaction*					*Community*	
	RR	AR	TSD	TISD	UUR	UMR	CHS	UR	MR	HTR	AUR	ATS	FR	MFFFR	FBR	R	CC	CBR	CBCC
Original	0	0	0	0	0	0	0	.5	1	0	0	0	.04	0	.75	.03	0	.44	.08
AdverSPAM	.04	.05	.02	0	.04	.04	.01	.44	.84	0	0	.03	0	0	.2	.02	0	.39	0
FGSM	.45	.45	.45	0	.45	.45	.45	.05	.55	0	0	.45	0	0	.3	0	0	0	0
DF	1	1	1	0	1	1	.67	0	1	0	0	1	0	0	0	0	0	0	0
C&W	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	.03	0	0	0
Cheng	.03	.04	.01	0	.03	.03	.01	.45	1	0	0	.02	0	0	.31	.02	0	.4	0
Peng	.04	.22	.15	0	.03	.28	0	.45	1	.01	0	.03	0	0	.34	0	0	.39	0

4.4.4 Impact of AdverSpam: a concrete case study

AdverSPAM belongs to that category of adversarial attacks that aim to perturb the input of the target model, without directly addressing the so-called *inverse feature-mapping problem* [53], i.e., how to modify the original data given the perturbed feature vector. Hence, the attacker must find on his own the set of actions to manipulate the input (the OSN account in the considered case) in order to obtain the feature vector suggested by AdverSPAM. This can be achieved by relying either on a manual or an automatic strategy.

The former, also known as *opportunistic* [72], consists in performing a certain action (e.g., add/remove data), measuring the resulting feature values, and reiterating until the desired values are reached. As an example, let us consider Table 4.8 which reports the changes made by AdverSpam and its competitors to the feature vector describing one of the spammers available in the dataset. The AdverSPAM row indicates that the most perturbed features are the *Mention Ratio* (MR) and the *Follower-based Reputation* (FBR), while the other values are quite close to the original ones. The feature MR, which is defined as:

$$MR(account) = \frac{number\_of\_mentions}{number\_of\_posts}, \tag{4.11}$$

is quite simple to alter. Since the modification suggested by AdverSPAM is to lower MR to 0.84, the attacker could replace tweets containing a mention with others that have the same content, but not the mention. As the number of posts do not change, the other content-based features are not influenced by this action. However, altering the tweets may change some metadata-based features that capture the tweet time, such as TSD and $TISD$; then, tweet replacement should follow the same posting frequency that characterized the original account. Please note that the attacker could also use an existing account only to get $\tilde{x}$ from AdverSPAM, and then create a *new* account that is described by that perturbed feature vector.

A similar strategy can be adopted to alter the FBR of an account, which is defined as the average of the reputation of its followers [35]:

$$FBR(account) = \frac{\sum_{followers} reputation(follower)}{|followers|}. \tag{4.12}$$

Then, in order to change FBR from 0.75 to 0.2, the attacker could i) reduce the overall reputation of its followers (numerator), or ii) increase the number of followers (denominator). The reputation of the followers can be modified by *replacing* a subset of n followers with others having a lower reputation; this allows to keep the number of followers unaltered, which is essential to not impact on the other interaction-based features. Such a modification is quite simple to achieve by exploiting black markets, where followers having specific characteristics can be bought [73, 74]. Conversely, increasing the number of followers is somewhat more complicated as it should be done while guaranteeing that the new followers have a similar reputation to those already existing, so that the numerator of Eq. 4.12 does not change, nor do other related features such as $MFFFR$ or R.

It is worth noting that attacks modifying the feature vectors in a more extensive way are difficult to manage with an opportunistic strategy, because the more values are changed the more constraints must be met while altering the account. In these cases, automatic strategies can help in finding the optimal set of actions that lead to the desired feature values. A common approach, for instance, is to model a numerical optimization algorithm which follows the negative gradient of the objective function [66]. However, it is not possible to directly apply gradient descent-based techniques when the feature space is not invertible, nor differentiable. A more general strategy consists in modeling the inverse mapping problem as a game. In [11], for instance, Monte Carlo Tree Search (MCTS) is adopted to generate a chain of mutations to be applied on malware with the goal of bypassing the target API-based classifier. In the scenario considered in this work, the possible mutations could be those provided by the Twitter's API to post and remove tweets, or to control the followers and followings of an account.

4.4.5 Mitigation Strategies

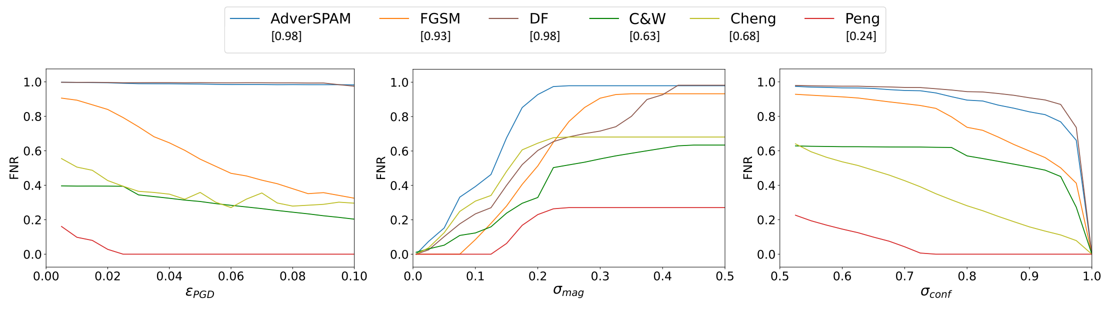
As a final evaluation of AdverSPAM, its ability to accomplish the attack when a defense mechanism is available has been tested. The remainder of this section discusses the results obtained by considering three mitigation strategies, namely, Adversarial Training, Magnet, and a confidence-based defense designed to counter the specificities of AdverSPAM.

Adversarial Training

Adversarial Training (AT) [39] is one of the most widely adopted [19, 86, 14, 63, 18, 88, 87, 78, 91] defenses against adversarial attacks because of its effectiveness and theoretical simplicity. The idea is to enable the target model to identify adversarial samples by incorporating them into the training process. The creation of AT samples is commonly based on the Projected Gradient Descent (PGD) attack [51], which is an iterative variant of FGSM. This approach has been applied in this work to strengthen the target models and test whether their robustness to adversarial attacks is improved.

Since the maximum perturbation amount that PGD can apply to a sample depends on a parameter ϵ_{PGD}, the assessment was repeated while varying ϵ_{PGD} in $[0.005, 0.1]$, with step 0.005. Results are shown in Figure 4.6a, where the different curves indicate the average FNRs achieved by AdverSPAM and the five baselines (see Section 4.4.3 for references) when AT is performed.

In terms of the reference FNRs, i.e., those measured without using any defense, the three best performing attacks are AdverSPAM, DF and FGSM (see the figure legend). After applying AT, FGSM is heavily penalized since, like PGD, exploits the gradient of the loss function; thus, the samples generated by PGD are very similar to those created by FGSM. On the other hand, the performances of AdverSPAM and DF do not change significantly. This may be because the samples generated by PGD are very different from those created by the two attacks, and therefore the AT-based defense cannot exploit them to recognize the adversarial



(a) (b) (c)

Figure 4.6: Average FNRs achieved by each attack when (a) Adversarial Training, (b) Magnet, and (c) Confidence-Based defenses are used. The legend also reports the reference FNRs (indicated between square brackets) achieved by the attacks without using any mitigation technique.

samples. As regards the other baselines, all trends are decreasing as the perturbations increase. Similarly to FGSM, this may depend on the fact that PGD samples are alike to those generated by C&W, Cheng, and Peng.

It is worth noting that, since PGD needs to access the gradient of the loss function of the model being attacked, *RF* and *RR* classifiers are not eligible for this type of attack. In order to face this limitation, two other defense techniques have been tested, which can be applied to all the adopted target models; the results are discussed in the following subsections.

Magnet

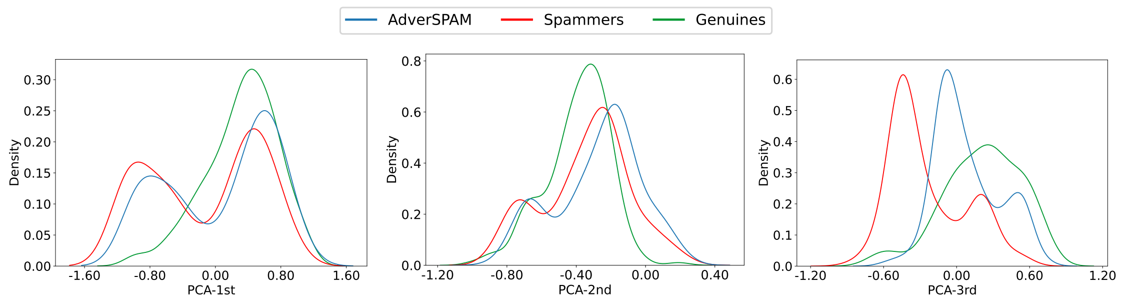
Magnet [56] aims to detect adversarial samples by leveraging autoencoders, i.e., neural networks trained for learning and reproducing the exact same distribution of the data while minimizing the reconstruction error of the inputs in the training set. In particular, Magnet pre-trains an autoencoder over the distribution of the ground truth samples; then, when a new sample is proposed to the target model, this is fed into the autoencoder in order to compute its reconstruction error, which roughly represents its similarity to the data distribution observed during the training phase. If the reconstruction error exceeds a prefixed input threshold σ_{mag}, the input sample is recognized as an out-of-distribution sample and will be rejected; this prevents adversarial samples from being evaluated by the target model. Magnet was implemented according to the specifications provided in [56], and $\sigma_{mag} \in [0.05, 0.5]$ has been explored with step 0.025.

Figure 4.6b shows that higher values of FNR are obtained as σ_{mag} increases; this means that if the threshold is set too high, the defense mechanism will not be able to reject any input and therefore its effect is null. In fact, for $\sigma_{mag} > 0.3$, all attacks reach the reference FNRs. On the other hand, choosing a value of σ_{mag} too restrictive, although it guarantees to cope with the adversarial attacks, might be impractical in a real scenario as it would result in a high number of non adversarial data rejection.

Confidence-Based Defense

As a third defense mechanism, the attention was focused on one of the characteristics that most differentiates AdverSPAM from other baselines, namely the fact that perturbed samples are chosen as close as possible to the decision boundary. However, some target classifiers could implement a mechanism to check the *confidence* of their prediction: the higher the distance of the predicted sample from the learned decision boundary, the greater the confidence in the prediction. According to these considerations, the proposed *Confidence-Based Defense* (CBD) aims to reject the "genuine" class predictions of the target model when the confidence lies below a fixed threshold σ_{conf}.

Results shown in Figure 4.6c indicate that as the confidence threshold increases, the effectiveness of the attacks decreases. For AdverSPAM, FGSM, DF and C&W it is possible to identify a cutoff when σ_{conf} is about 0.75. Since Cheng and Peng produce samples that lie close to the targeted decision boundary, the confidence in the prediction of such adversarial samples is relatively low, and a moderate cutoff threshold is sufficient for defending against these attacks. Finally, as in the case of Magnet, choosing high σ_{conf} values could result in high rejection rate. Therefore, good trade-off values could be those in the range $\sigma_{conf} \in [0.8, 0.9]$, where adversarial samples may be effectively filtered out while non discarding non adversarial data.



(a) (b) (c)

Figure 4.7: Kernel Density Estimation (KDE) of the AdverSPAM samples compared to the ground genuines and spammers, computed for the first (a) second (b) and third (c) principal components describing the feature space.

4.5 AdverSPAM: advantages and limitations

The experimental analysis presented so far allowed to measure the performances of AdverSPAM and compare them with some significant baselines. The results showed the effectiveness of the proposed approach in attacking different target models while applying perturbations that preserve both statistical correlation and semantic dependence between features.

Besides these results, a further aspect to consider is whether the perturbation of one or more features could impair the malicious behavior of spam accounts. In order to make it easier to visualize the properties of the feature space, Principal Component Analysis (PCA) [81] was used to reduce its dimensionality. Experiments were conducted by considering the three principal components that characterize the set of feature vectors; then, the Kernel Density Estimation (KDE) of each component was computed in order to evaluate the distributions representing both the original samples (*genuines* and *spammers*) and those generated by AdverSPAM.

By observing Figure 4.7, it is possible to appreciate how the densities of the genuine samples (green) strongly differ from the others. Conversely, the density estimation of the spammers (red) and AdverSPAM (blue) samples are highly similar, apart from a slight shift introduced by the adversarial perturbations. Such analysis confirms the capability of AdverSPAM in generating adversarial samples that, while preserving the key characteristics of the spam accounts, are able to evade unknown classifiers.

Although experimentally proved to be a promising adversarial attack, the proposed approach exhibits some intrinsic limitations. The first is related to the need for choosing a surrogate model with a linear decision boundary. Such requirement is closely tied to the necessity of formulating the attack by considering linear constraints, which makes AdverSPAM particularly efficient.

Adapting AdverSPAM to different scenarios, or even different feature sets, may be computationally burdensome, as the number of constraints in the proposed optimization problem increases according to the number of features pairs standing in a semantic dependence relationship. Moreover, operating in scenarios where there are no significant statistical correlations would tamper the effectiveness of the proposed approach. Furthermore, if the relationships between features are non-linear, AdverSPAM would require a change in its constraints and a different statistical indicator from the Pearson's Correlation coefficient. For instance, the accuracy of the best-fitting second-order polynomial can be used in the case of quadratic correlations between data.

The robustness of AdverSPAM has been studied against three different adversarial defenses, however, another interesting approach a defender may follow is the one outlined in [31], where an analysis of the sequence of consecutive requests originated from the same source is conducted. Such a countermeasure could inhibit the attack when an offline surrogate model cannot be built, and it is therefore necessary to query the online target model directly. In this extreme case, the iterative nature of the optimization solver would imply making multiple queries to the target model, which would result in a straightforward attack detection.

Chapter 5

Conclusions

The widespread diffusion of Machine Learning algorithms in modern society poses serious problems related to the robustness of such algorithms against malevolent intents. This is witnessed by the rise of the Adversarial Machine Learning research field, whose focus is the analysis of techniques to perturb the input data for altering the predictions of ML models. Such techniques exploit the optimization procedure at the core of ML models in order to find the smallest perturbation to add into the inputs for changing the predicted labels.

Research on Adversarial Machine Learning highlighted risks that may rise in several applications. Object detection models can be deceived with carefully selected noise added into images or videos, which can lead to conceal dangerous activity to a video surveillance system. Electronic healthcare systems can be circumvented with perturbations on the clinical records for the benefit of financial interests of private entities. Threat detection models can be bypassed for spreading new typologies of viruses, and spam filters can be deceived in order to spread unsolicited messages to acquire attention on social media.

These problems can be counteracted through defense mechanisms which are generally focused on understanding the key characteristics defining the nature of the samples, and detecting any malicious attempt to alter it. However, perturbations can be effective and not noticeable, carefully designed not to alter the statistical properties of the samples belonging to a class. The higher the awareness of the scientific community on this kind of threat, the more efficient the defense mechanisms will be, resulting in the design of safer ML systems, which are robust against modern dangers.

This thesis represents an attempt at addressing some of the challenges related to this research area. In particular, it is described the design and implementation of two Adversarial Machine Learning attack strategies. Such strategies dealt with the problem of generating effective perturbations in two different ways.

Firstly, it has been proposed an algorithm for the generation of adversarial examples in scenarios with electronic health records in the form of binary data. In particular, it has been studied how an adversary may alter the medical record of a patient in order to fool an intelligent system for antibiotic prescription. The experimental results showed that even only modifying three fields in the patient record, a trained neural network can almost always be induced into suggesting a prearranged treatment. As part of future works, the time granularity as input parameter to filter the dataset can be eliminated. For example, if the adversarial noise produced by an attack algorithm suggests to modify a feature with time granularity equal to 14 days from 0 to 1, then, all the features falling in the same category and with a granularity > 14 should be set to 1. Moreover, an automatic strategy for dynamically choosing the number of γ features to perturb based on the magnitude of the gradient can be defined, so that γ does not need to be specified as input to the approach. Finally, the feasibility of the approach can be investigated in other smart environments such as university campuses, where adversarial attacks aim at disrupting the provision of intelligent services to users [28].

This dissertation also presented an AML algorithm, *AdverSPAM*, designed to deceive Online Social Networks spammer detection systems. The strategy attack is formulated as an optimization problem whose constraints capture the essence of AML, i.e., finding the optimal perturbation to fool the target model, while preserving the inter-relationships among the features that describe the user behavior. The proposed algorithm explicitly imposes the maintenance of the statistical correlation and the semantic dependency of related features. By neglecting this aspect, the attack would produce numerically admissible perturbations, which could not actually be implemented by the attacker. The effectiveness of *AdverSPAM* was assessed by considering Twitter as case study, although the attack can be easily applied to any other AML scenario where preserving the consistency of the feature space is mandatory. Experimental analysis was conducted on a well-known public dataset of spammer and genuine accounts. Experiments involved the assessment of the method by testing four distinct surrogate models and twelve possible target models, as well as an overall evaluation of the performance of *AdverSPAM* as compared to five state-of-the-art attacks. Results revealed how *AdverSPAM* is a solid competitor in terms of transferability, with FNR indicators that outperformed all the baselines on 10 out of 12 considered target (black-box) models. With regard to the forged adversarial features, *AdverSPAM* exhibited the lowest distortion degree introduced in data, thanks to the formulation of constraints explicitly accounting for the maintenance of features dependencies. Moreover, the effectiveness of AdverSPAM has been tested against three different adversarial defenses, proving the robustness of the approach to state-of-the-art mitigations. As part of future work, it is planned to test *AdverSPAM* effectiveness on multiple public datasets, and evaluate its ability to be more widely applicable to different scenarios characterized by similar AML requirements, such as Network Intrusion Detection. It is also planned to extend the constraint on the decision boundary of the surrogate classifier to include also non-linear classifiers, and test their efficiency as sources of adversarial samples.

Bibliography

[1] V. Agate, F. Concone, S. Gaglio, and A. Giammanco. A hybrid recommender system for cultural heritage promotion. In *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 203–208, 2021.

[2] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional space. In J. Van den Bussche and V. Vianu, editors, *Database Theory — ICDT 2001*, pages 420–434, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[3] A. Al-Dujaili, A. Huang, E. Hemberg, and U.-M. O'Reilly. Adversarial deep learning for robust detection of binary encoded malware. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 76–82, 2018.

[4] E. Alhajjar, P. Maxwell, and N. Bastian. Adversarial machine learning in network intrusion detection systems. *Expert Systems with Applications*, 186: 115782, 2021.

[5] E. Anthi, L. Williams, M. Rhode, P. Burnap, and A. Wedgbury. Adversarial attacks on machine learning cybersecurity defences in industrial control systems. *Journal of Information Security and Applications*, 58:102717, 2021.

[6] J. Benesty, J. Chen, Y. Huang, and I. Cohen. *Pearson Correlation Coefficient*, pages 1–4. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[7] S. Bhattacharya, D. Manousakas, A. G. C. P. Ramos, S. I. Venieris, N. D. Lane, and C. Mascolo. Countering acoustic adversarial attacks in microphone-

equipped smart home devices. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 4(2), jun 2020.

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

[9] B. Biggio, G. Fumera, and F. Roli. Multiple classifier systems for robust classifier design in adversarial environments. *International Journal of Machine Learning and Cybernetics*, 1(1):27–41, Dec 2010.

[10] B. Biggio, G. Fumera, G. L. Marcialis, and F. Roli. Statistical meta-analysis of presentation attacks for secure multibiometric systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(3):561–575, 2017.

[11] J. Boutsikas, M. E. Eren, C. K. Varga, E. Raff, C. Matuszek, and C. Nicholas. Evading malware classifiers via monte carlo mutant feature discovery. In *12th Annual Malware Technical Exchange Meeting*, 2021.

[12] C. Buckner. Understanding adversarial examples requires a theory of artefacts for deep learning. *Nature Machine Intelligence*, 2(12):731–736, Dec 2020.

[13] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, Los Alamitos, CA, USA, may 2017. IEEE Computer Society.

[14] G. Chang, H. Gao, Z. Yao, and H. Xiong. Textguise: Adaptive adversarial example attacks on text classification model. *Neurocomputing*, 529:190–203, 2023.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[16] J. Chen, X. Lin, Z. Shi, and Y. Liu. Link prediction adversarial attack via iterative gradient attack. *IEEE Transactions on Computational Social Systems*, 7(4):1081–1094, 2020.

[17] M. Cheng, T. Le, P. Chen, H. Zhang, J. Yi, and C. Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[18] Y. Cheng, Q. Guo, F. Juefei-Xu, S.-W. Lin, W. Feng, W. Lin, and Y. Liu. Pasadena: Perceptually aware and stealthy adversarial denoise attack. *IEEE Transactions on Multimedia*, 24:3807–3822, 2022.

[19] A. Chernikova and A. Oprea. Fence: Feasible evasion attacks on neural networks in constrained environments. *ACM Trans. Priv. Secur.*, 25(4), jul 2022.

[20] F. Concone, G. Lo Re, M. Morana, and C. Ruocco. Twitter spam account detection by effective labeling. In P. Degano and R. Zunino, editors, *Proceedings of the Third Italian Conference on Cyber Security, Pisa, Italy, February 13-15, 2019*, volume 2315 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.

[21] F. Concone, G. Lo Re, M. Morana, and C. Ruocco. Assisted labeling for spam account detection on twitter. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 359–366, 2019.

[22] F. Concone, G. L. Re, M. Morana, and S. K. Das. Spade: Multi-stage spam account detection for online social networks. *IEEE Transactions on Dependable and Secure Computing*, 20(4):3128–3143, 2023.

[23] F. Concone, S. Gaglio, A. Giammanco, G. L. Re, and M. Morana. Adverspam: Adversarial spam account manipulation in online social networks. *ACM Trans. Priv. Secur.*, 27(2), mar 2024.

[24] S. Cresci, R. D. Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi. Social fingerprinting: Detection of spambot groups through dna-inspired behavioral modeling. *IEEE Transactions on Dependable and Secure Computing*, 15(4): 561–576, 2018.

[25] S. Cresci, M. Petrocchi, A. Spognardi, and S. Tognazzi. Adversarial machine learning for protecting against online manipulation. *IEEE Internet Computing*, 26(2):47–52, 2022.

[26] E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi, and O. E. Ajibuwa. Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon*, 5(6):e01802, 2019.

[27] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar. A survey of deep learning and its applications: A new paradigm to machine learning. *Archives of Computational Methods in Engineering*, 27(4):1071–1092, Sep 2020.

[28] A. De Paola, S. Gaglio, A. Giammanco, G. Lo Re, and M. Morana. A multi-agent system for itinerary suggestion in smart environments. *CAAI Transactions on Intelligence Technology*, 6(4):377–393, 2021.

[29] K. Deb. Multi-objective optimization. In E. K. Burke and G. Kendall, editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pages 273–316, Boston, MA, 2005. Springer US.

[30] L. Demetrio, B. Biggio, G. Lagorio, F. Roli, and A. Armando. Functionality-preserving black-box optimization of adversarial windows malware. *IEEE Transactions on Information Forensics and Security*, 16:3469–3478, 2021.

[31] L. Demetrio, S. E. Coull, B. Biggio, G. Lagorio, A. Armando, and F. Roli. Adversarial exemples: A survey and experimental evaluation of practical attacks on machine learning for windows malware detection. *ACM Trans. Priv. Secur.*, 24(4), sep 2021.

[32] A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, and F. Roli. Yes, machine learning can be more secure! a case study on android malware detection. *IEEE Transactions on Dependable and Secure Computing*, 16(4):711–724, 2019.

[33] A. Demontis, M. Melis, M. Pintor, M. Jagielski, B. Biggio, A. Oprea, C. Nita-Rotaru, and F. Roli. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *Proceedings of the 28th*

USENIX Conference on Security Symposium, SEC'19, pages 321–338, USA, 2019. USENIX Association.

[34] Y. Dong, Q.-A. Fu, X. Yang, T. Pang, H. Su, Z. Xiao, and J. Zhu. Benchmarking adversarial robustness on image classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 318–328, 2020.

[35] M. Fazil and M. Abulaish. A hybrid approach for detecting automated spammers in twitter. *IEEE Transactions on Information Forensics and Security*, 13(11):2707–2719, 2018.

[36] S. G. Finlayson, J. D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, and I. S. Kohane. Adversarial attacks on medical machine learning. *Science*, 363(6433): 1287–1289, 2019.

[37] S. Gaglio, A. Giammanco, G. Lo Re, and M. Morana. Adversarial machine learning in e-health: Attacking a smart prescription system. In S. Bandini, F. Gasparini, V. Mascardi, M. Palmonari, and G. Vizzari, editors, *AIxIA 2021 – Advances in Artificial Intelligence*, pages 490–502, Cham, 2021. Springer International Publishing.

[38] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[39] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples, 2015.

[40] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[41] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[42] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Black-box adversarial attacks with limited queries and information. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2137–2146. PMLR, 10–15 Jul 2018.

[43] C. Janiesch, P. Zschech, and K. Heinrich. Machine learning and deep learning. *Electronic Markets*, 31(3):685–695, Sep 2021.

[44] A. Jati, C.-C. Hsu, M. Pal, R. Peri, W. AbdAlmageed, and S. Narayanan. Adversarial attack and defense strategies for deep speaker recognition systems. *Computer Speech & Language*, 68:101199, 2021.

[45] S. Kanjilal, M. Oberst, S. Boominathan, H. Zhou, D. C. Hooper, and D. Sontag. A decision algorithm to promote outpatient antimicrobial stewardship for uncomplicated urinary tract infection. *Science Translational Medicine*, 12 (568), 2020.

[46] B. Kuchipudi, R. T. Nannapaneni, and Q. Liao. Adversarial machine learning for spam filters. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ARES '20, New York, NY, USA, 2020. Association for Computing Machinery.

[47] R. Kumar, R. Arora, V. Bansal, V. J. Sahayasheela, H. Buckchash, J. Imran, N. Narayanan, G. N. Pandian, and B. Raman. Classification of COVID-19 from chest x-ray images using deep features and correlation coefficient. *Multimedia Tools and Applications*, 81(19):27631–27655, Aug. 2022.

[48] W. Lalouani, M. Younis, and U. Baroudi. Countering radiometric signature exploitation using adversarial machine learning based protocol switching. *Computer Communications*, 174:109–121, 2021.

[49] G. Li, A. Zhang, Q. Zhang, D. Wu, and C. Zhan. Pearson correlation coefficient-based performance enhancement of broad learning system for stock price prediction. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 69(5):2413–2417, 2022.

[50] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[51] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[52] D. Maiorca, I. Corona, and G. Giacinto. Looking at the bag is not enough to find the bomb: An evasion of structural methods for malicious pdf files detection. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ASIA CCS '13, pages 119–130, New York, NY, USA, 2013. Association for Computing Machinery.

[53] D. Maiorca, B. Biggio, and G. Giacinto. Towards adversarial malware detection: Lessons learned from pdf-based attacks. *ACM Comput. Surv.*, 52(4), aug 2019.

[54] D. Maiorca, A. Demontis, B. Biggio, F. Roli, and G. Giacinto. Adversarial detection of flash malware: Limitations and open issues. *Computers & Security*, 96:101901, 2020.

[55] M. L. McHugh. The chi-square test of independence. *Biochemia medica*, 23 (2):143–149, 2013.

[56] D. Meng and H. Chen. Magnet: A two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 135–147, New York, NY, USA, 2017. Association for Computing Machinery.

[57] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society.

[58] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 86–94, 2017.

[59] A. I. Newaz, N. I. Haque, A. K. Sikder, M. A. Rahman, and A. S. Uluagac. Adversarial attacks to machine learning-based smart healthcare systems. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 1–6, 2020.

[60] M. Oberst, S. Boominathan, H. Zhou, S. Kanjilal, and D. Sontag. Amr-uti: Antimicrobial resistance in urinary tract infections (version 1.0.0). *Physionet*, 2020.

[61] N. Papernot, P. D. McDaniel, and I. J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[62] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pages 506–519, New York, NY, USA, 2017. Association for Computing Machinery.

[63] B. Peng, B. Peng, J. Zhou, J. Xie, and L. Liu. Scattering model guided adversarial examples for sar target recognition: Attack and defense. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–17, 2022.

[64] X. Peng, W. Huang, and Z. Shi. Adversarial attack against dos intrusion detection: An improved boundary-based method. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1288–1295, 2019.

[65] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro. Intriguing properties of adversarial ml attacks in the problem space. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1332–1349, 2020.

[66] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro. Intriguing properties of adversarial ml attacks in the problem space. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1308–1325. IEEE Computer Society, 2020.

[67] M. J. D. Powell. *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*, pages 51–67. Springer Netherlands, Dordrecht, 1994.

[68] Y. Qin, N. Carlini, G. Cottrell, I. Goodfellow, and C. Raffel. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5231–5240. PMLR, 09–15 Jun 2019.

[69] J. Rauber, R. Zimmermann, M. Bethge, and W. Brendel. Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax. *Journal of Open Source Software*, 5 (53):2607, 2020.

[70] P. Russu, A. Demontis, B. Biggio, G. Fumera, and F. Roli. Secure kernel machines against evasion attacks. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, AISec '16, pages 59–69, New York, NY, USA, 2016. Association for Computing Machinery.

[71] Y. Shi, K. Davaslioglu, and Y. E. Sagduyu. Generative adversarial network in the air: Deep adversarial learning for wireless signal spoofing. *IEEE Transactions on Cognitive Communications and Networking*, 7(1):294–303, 2021.

[72] N. Srndic and P. Laskov. Practical evasion of a learning-based classifier: A case study. In *2014 IEEE Symposium on Security and Privacy*, pages 197–211, 2014.

[73] G. Stringhini, M. Egele, C. Kruegel, and G. Vigna. Poultry markets: On the underground economy of twitter followers. *SIGCOMM Comput. Commun. Rev.*, 42(4):527–532, sep 2012.

[74] G. Stringhini, G. Wang, M. Egele, C. Kruegel, G. Vigna, H. Zheng, and B. Y. Zhao. Follow the green: Growth and dynamics in twitter follower markets. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 163–176, New York, NY, USA, 2013. Association for Computing Machinery.

[75] O. Suciu, R. Mărginean, Y. Kaya, H. Daumé, and T. Dumitraş. When does machine learning fail? generalized transferability for evasion and poisoning attacks. In *Proceedings of the 27th USENIX Conference on Security Symposium*, SEC'18, pages 1299–1316, USA, 2018. USENIX Association.

[76] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks, 2013.

[77] J. Tang, H. Wen, and K. Wang. Revisiting adversarially learned injection attacks against recommender systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, pages 318–327, New York, NY, USA, 2020. Association for Computing Machinery.

[78] S. Tang, X. Huang, M. Chen, C. Sun, and J. Yang. Adversarial attack type i: Cheat classifiers by significant changes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(3):1100–1109, 2021.

[79] Q. Wang, B. Zheng, Q. Li, C. Shen, and Z. Ba. Towards query-efficient adversarial attacks against automatic speech recognition systems. *IEEE Transactions on Information Forensics and Security*, 16:896–908, 2021.

[80] Q. Wang, H. Yang, G. Wu, K.-K. R. Choo, Z. Zhang, G. Miao, and Y. Ren. Black-box adversarial attacks on xss attack detection model. *Computers & Security*, 113:102554, 2022.

[81] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1):37–52, 1987. Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

[82] J. Wu, N. Li, Y. Zhao, and J. Wang. Usage of correlation analysis and hypothesis test in optimizing the gated recurrent unit network for wind speed forecasting. *Energy*, 242:122960, 2022.

[83] T. Wu, S. Wen, Y. Xiang, and W. Zhou. Twitter spam detection: survey of new approaches and comparative study. *Computers and Security*, 76:265–284, July 2018.

[84] M. Xue, C. Yuan, C. He, J. Wang, and W. Liu. Naturalae: Natural and robust physical adversarial examples for object detectors. *Journal of Information Security and Applications*, 57:102694, 2021.

[85] C. Yang, R. Harkreader, and G. Gu. Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Transactions on Information Forensics and Security*, 8(8):1280–1293, 2013.

[86] Z. Yin, Y. Zhuo, and Z. Ge. Transfer adversarial attacks across industrial intelligent systems. *Reliability Engineering & System Safety*, 237:109299, 2023.

[87] D. Zhan, Y. Duan, Y. Hu, L. Yin, Z. Pan, and S. Guo. Amgmal: Adaptive mask-guided adversarial attack against malware detection with minimal perturbation. *Computers & Security*, 127:103103, 2023.

[88] C. Zhang, X. Luo, and P. Han. On-manifold adversarial attack based on latent space substitute model. *Computers & Security*, 120:102770, 2022.

[89] F. Zhang, P. P. K. Chan, B. Biggio, D. S. Yeung, and F. Roli. Adversarial feature selection against evasion attacks. *IEEE Transactions on Cybernetics*, 46(3):766–777, 2016.

[90] J. Zhang, B. Chen, X. Cheng, H. T. T. Binh, and S. Yu. Poisongan: Generative poisoning attacks against federated learning in edge computing systems. *IEEE Internet of Things Journal*, 8(5):3310–3322, 2021.

[91] X. Zhang, Y. Xu, S. Zhang, and X. Li. A highly stealthy adaptive decay attack against speaker recognition. *IEEE Access*, 10:118789–118805, 2022.

[92] P. Zhao, H. Huang, X. Zhao, and D. Huang. P3: Privacy-preserving scheme against poisoning attacks in mobile-edge computing. *IEEE Transactions on Computational Social Systems*, 7(3):818–826, 2020.

[93] D. Zügner, O. Borchert, A. Akbarnejad, and S. Günnemann. Adversarial attacks on graph neural networks: Perturbations and their patterns. *ACM Trans. Knowl. Discov. Data*, 14(5), jun 2020.