UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Universitat de Barcelona
Facultat de Matemàtiques i Informàtica

ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA
Universitat Rovira i Virgili

MASTER IN ARTIFICIAL INTELLIGENCE

MASTER'S DEGREE THESIS

Adversarial Machine Learning for Cyber Security

Author:
Juan Esteban Fonseca Núñez

Supervisor:
Ignasi Paredes Oliva, PhD
Co-supervisor:
Pere Barlet Ros, PhD
Tutor:
Enrique Romero Merino, PhD

Nestlé Global Cyber SOC
Data Science Team

April, 2022

"A scientist is just a kid who never grew up"

Neil deGrasse Tyson

Abstract

This master thesis aims to take advantage of state of the art and tools that have been developed in Adversarial Machine Learning (AML) and related research branches to strengthen Machine Learning (ML) models used in cyber security. First, it seeks to collect, organize and summarize the most recent and potential state-of-the-art techniques in AML, considering that it is a research branch in an unstable state with a great diversity of difficult to contrast proposals, which rapidly evolve but are quickly replaced by attacks or defenses with greater potential. This summary is important considering that the AML literature is far from being able to create defensive techniques that effectively protect a ML model from all possible attacks, and it is relevant to analyze them both in detail and with criteria in order to apply them in practice. It is also useful to find biases in state-of-the-art to be considered regarding the measurement of the attack or defense effectiveness, which can be addressed by proposing methodologies and metrics to mitigate them.

Additionally, it is considered inappropriate to analyze AML in isolation, considering that the robustness of a ML model to adversarial attacks is totally related to its generalization capacity to in-distribution cases, to its robustness to out-of-distribution cases, and to the possibility of overinterpretation, using spurious (but statistically valid) patterns in the model that may give a false sense of high performance. Therefore, this thesis proposes a methodology to previously evaluate the exposure of a model to these considerations, focusing on improving it in progressive order of priorities in each of its stages, and to guarantee satisfactory overall robustness.

Based on this methodology, two interesting case studies are chosen to be explored in greater depth to evaluate their robustness to adversarial attacks, perform attacks to gain insights about their strengths and weaknesses, and finally propose improvements. In this process, all kinds of approaches are used depending on the type of problem evaluated and its assumptions, performing exploratory analysis, applying AML attacks and detailing their implications, proposing improvements and implementation of defenses such as Adversarial Training, and finally creating and proposing a methodology to correctly evaluate the effectiveness of a defense avoiding the biases of the state of the art.

For each of the case studies, it is possible to create efficient adversarial attacks, analyze the strengths of each model, and in the case of the second case study, it is possible to increase the adversarial robustness of a Classification Convolutional Neural Network using Adversarial Training. This leads to other positive effects on the model, such as a better representation of the data, easier implementation of techniques to detect adversarial cases through anomaly analysis, and insights concerning its performance to reinforce the model from other viewpoints.

Each of the approaches used in the case studies provides multiple important findings and conclusions that are documented to create useful knowledge for creating and improving other ML models. In addition, they provide potential research directions, techniques to be tested, and considerations to be more deeply explored to continue strengthening and enhancing the capabilities of ML models in cyber security.

Acknowledgements

I would like to express my gratitude to the Nestlé CyberSOC Data Science team and in particular to Ignasi Paredes for guiding and supporting me in the development of this interesting and challenging research project, which I enjoyed at every stage while enriching myself in the topics I am most enthusiastic about. I would also like to thank my parents for their motivation and unconditional support that has allowed me to continue studying the topics I am most passionate about.

Contents

List of Figures

List of Tables

Dedicated to my parents and sisters, who have made me happy by encouraging my curiosity and admiration for the flight of a hummingbird facing crosswinds.

Chapter 1

Introduction

This master thesis presents a practical research approach to benefit from the Adversarial Machine Learning (AML) state of the art, concepts, and tools to add value in the development and improvement of Machine Learning (ML) models used in cyber security.

The objectives of this master's thesis are presented, followed by its motivation, its contributions, and finally, an overview of the topics covered.

1.1 Objectives

The main objective in this thesis is to extract value from the Adversarial Machine Learning field from different approaches, in order to take advantage of its techniques and the created knowledge in order to improve the Machine Learning practices in Data Science teams focused on cyber security. This main objective is decomposed into the following objectives to be covered in the thesis:

- Research and document the risks associated with ML models and their respective vulnerabilities when applied in cyber security, in order to propose a qualitative analysis prior to the AML review.
- Study and analyze research branches related to AML and document the importance of considering them simultaneously to create more robust models at a general level.
- Research and document the types of attacks a ML model may receive, their motivation, assumptions, operation and implications, in order to propose a qualitative analysis of vulnerabilities to adversarial attacks to measure the exposure and risk of a model under study.
- Analyze the technical details from the techniques associated with the most relevant attacks, and practically implement adversarial attacks from an attacker's perspective to measure their effectiveness over the ML model and contrast it with their ability to pass undetected by a user.
- Develop exploratory analysis and create useful KPIs for the ML model owner to provide insights about the model's exposure to adversarial attacks, to attempt to measure adversarial robustness with different approximations, and to obtain more detailed information about the model's weaknesses or strengths. Propose potential improvements before exploring specific AML defense techniques.

- Investigate and analyze state-of-the-art defenses to weaknesses that cannot be solved by improving traditional processes, and document the different alternatives, their potential impact, and viability.
- Implement a defense technique designed to mitigate the effects of certain types of adversarial attacks and propose KPIs to correctly evaluate the uplift generated to isolate its real incrementality in terms of performance and adversarial robustness. Adversarial Training defenses and a basic ensemble of models are implemented.
- Propose future techniques to be considered and experimented, to improve the studied models, and future research directions for further improvement.

In addition, transversally to the analyses carried out, another objective is to use Open Source tools, and test recent libraries that allow the incorporation of AML techniques.

1.2 Motivation

In cyber security, it is crucial to develop robust mechanisms to extract the potential of ML models to automate large-scale processes and find patterns to extend the human capabilities and traditional cyber security systems. Attacks on cyber networks evolve at a pace that usually outpaces the ability of defenses to adapt to them. ML provides the opportunity to create methods to adapt systems to new variants of emerging attacks and zero-day attacks, detecting new attacks instantly, and not only after incorporating them into the defense process.

Among these, Bagging and Boosting techniques based on tree ensembles are increasingly used because of their potential to achieve high performance and to be implemented easily. These methods allow dealing with more complex patterns, generally are more stable, reduce variance and bias, allow more flexibility to use both numerical and categorical data, and are able to handle outliers and missing values robustly. Likewise, Deep Neural Networks (DNNs) have become both popular and necessary given their potential for dealing with problems involving unstructured data, such as images and text.

As in all ML applications, ensuring the correct model implementation and training process is very important to obtain the best results. But in particular, in cyber security, it is essential to ensure that the models are robust to previously unseen cases, also called out-of-distribution (OOD) cases, and explicitly designed cases created by attackers to make the model misbehave. These attacks are in fact performed with Adversarial Machine Learning (AML) techniques, which aim to exploit ML models to take advantage of them for different purposes, being one of the most relevant purposes to create malicious attacks.

Consequently, AML is a field of great interest in cyber security given the motivation for attackers to compromise ML systems, attempting to pass unnoticed, and due to ML models' proven weaknesses. In particular, it is difficult to explain how black-box algorithms make predictions and even more when using large numbers of features, narrowing the model to focus on correctly modeling only a particular and small region from the whole space of possible inputs. This may allow adversaries to exploit these not explicitly modeled regions (data gap regions) in the training process to achieve their outcome goal. Additionally, with unstructured data in which ML models must involve preprocessing and automatic feature creation, for example, with DNNs such as Convolutional Neural Networks (CNNs) for images, it is difficult to understand their decision-making, being susceptible to specific difficult to generalize cases, and vulnerable to optimized attacks created with minimal changes in their input that can change their prediction.

Therefore, it is essential to study this topic to be informed about the existing risks associated when implementing a ML model and analyzing its vulnerabilities at each stage of its construction and deployment to improve its design. Also, it is helpful to test its capabilities against adversarial attacks to detect unforeseen weaknesses and recognize mechanisms to make it stronger. Finally, considering that the topics covered are presently at a boom and have grown considerably in the last few years, with more and more proposed technics with different approaches and different ramifications, it is important to be updated with state of art to implement good practices in order to ensure the security measures are constantly updated and improved. For example, the first paper published on Overinterpretation, *Overinterpretation Reveals Image Classification Model Pathologies* (Carter et al., 2020) and the *Securing Machine Learning Algorithms Report* (Malatras, Agrafiotis, and Adamczyk, 2021) created by the European Union Agency for Cybersecurity (ENISA) were published in December 2021, at the same time this document was written, and a summary of state of the art in AML for cyber security with current contributions and challenges was published in June 2021 in *Adversarial Machine Learning for Cybersecurity and Computer Vision: Current Developments and Challenges* (Xi, 2020).

1.3 Overview and Contributions

1.3.1 General Overview

First, a review of the state of the art is made, contemplating the application of ML in cybersecurity, its threats, and vulnerabilities, focusing its documentation to extract practical value from the area of Adversarial Machine Learning, and its relationship with other related topics to generate more robust and properly defined models to improve security and reliability. At the same time, an analysis of the trajectory of these topics in state of the art is documented, the different branches and classifications that have been created, explaining their motivations, their operation, the proposed attacks and defenses, the main conclusions obtained by different authors and the metrics employed. This aims to train people involved in the development of ML models in cybersecurity, allowing them to contemplate in a holistic way all the relevant factors and have foundations to propose better solutions.

This process of documentation and constant review of state of the art is considered of great relevance in this field, considering that there is no apparent consensus or conclusive solutions to handle it. There are papers by different authors that have contradictory conclusions, which are strongly modified over time and can totally change how AML problems are approached.

Second, a methodology is proposed to evaluate and strengthen a ML model against adversarial attacks. This methodology first proposes a qualitative vulnerability analysis to detect weaknesses and improvement opportunities in a ML model, followed by practical tests to measure its adversarial robustness and empirically evaluate its susceptibility to different adversarial attacks. Next, metrics and exploratory analysis are created to generate knowledge based on them to understand weaknesses and possible improvements.

Finally, metrics are proposed to evaluate implemented defenses and their adversarial robustness and performance effect. In the last stage, it should be emphasized that certain traditional metrics biases are aimed to be solved to evaluate the uplift in the adversary robustness, and a methodology to correctly measure the incrementality attributed to the implemented defense is proposed. This is tested with the implementation of two different defenses, which are orthogonal in the way to tackle the problem and that can complement each other in the second case study.

From another perspective, this can also be seen as a methodology that should complement the selection and definition of a model to be used in production. It is important, for example, to not only rely on performance metrics with a test dataset of the same distribution of the training dataset, as in the case of a classifier with its confusion matrix, but also on metrics that indirectly help to measure its performance against out-of-distribution (OOD), adversarial attacks and variations susceptible to overinterpretations in the model.

Two case studies are presented, in which ML models are used for different purposes. The first is a binary classifier (for phishing detection) using more than 50 extracted features from metadata and text, and the second is an image-based logo classifier. Each of the cases follows the same general methodology but applies different techniques and approaches that are better suited to its purpose. In particular, for the second case, two different techniques are tested to increase the robustness of the model, one being Adversarial Training and the other a simple model ensemble.

1.3.2 Document Structure

In addition to this chapter, this document contains the following structure:

- Chapter 2 gives an initial background on some ML applications in cybersecurity, state of the art associated with them, their usability, and their challenges.
- Chapter 3 summarizes state of the art in the field of AML, focusing on its history, its progress, and the most important conclusions and proposals. This chapter summarizes the most relevant attacks and defenses in the literature, considering the focus of this document and the metrics used to measure their effectiveness.
- Chapter 4 provides an overview of ML, AML, and closely related topics to explain concepts and assumptions necessary to consider in analyzing the thesis results.
- Chapter 5 provides a technical summary for each of the statistical, ML, and AML techniques used in the project to explain how they work prior to exposing the two case studies. Also, the Open Source tools used to implement each of the methods are shortly presented.
- Chapter 6 explains the vulnerability analysis proposed to diagnose the difficulties and areas for improvement for a ML model while also analyzing its vulnerability to adversarial attacks.
- Chapter 7 presents the analyses performed on the first case study, associated with a binary tree ensemble classifier. It starts by giving a general context of its creation process, followed by the analyses performed, their purpose, assumptions, methods used, and conclusions. It ends with a summary of the most relevant conclusions drawn from the case study.

- Chapter 8 presents the analyses performed for the second case study, associated with an image-based logo classifier using CNNs. The development of the base model is explained, followed by the presentation of the analyses performed to test it against images with transformations and adversarial attacks. For this case, an adversarial defense is tested, and a methodology is defined to measure its incrementality in performance and adversarial robustness.
- Chapter 9 proposes techniques and analysis to be tested as future work to explore the case studies further and presents the main conclusions obtained. Additionally, an analysis of the available Open Source libraries and their scope is made to determine the ability to implement other techniques easily.

1.4 Comments on the Document's Approach and Restrictions

This project is done in collaboration with the Data Science team from the Cyber Security Operations Center (SOC) from Nestle Global, which has the objective of protecting the security of more than 350,000 employees, focusing on providing value and covering the most challenging problems in which ML delivers the highest potential. Therefore, it is necessary to clarify that the thesis will be oriented to the analysis of adversarial attacks focused exclusively on the vulnerability of ML models at their predictive algorithm level, and not on vulnerabilities in other related processes such as tools, platforms, or other important aspects that are addressed in the SOC.

Considering that the second case study, which was the most deeply explored, is about image classification, this text focuses on documenting and evaluating attacks and defenses in Computer Vision and, in particular, classifiers. Although the methodology and certain sections are generalizable to other types of applications, this was selected as the main one for its potential to explore and measure AML techniques, for the easy attack visualization, direct access to raw data (images), and for allowing the application of state-of-the-art attacks and defenses, being the field in which AML has been explored the most.

In the practical tests in the case studies, the main focus is made on evasion attacks, which seek to obtain an incorrect output in the supervised models used (in this case avoiding detection or correct classification to create false-negative instances). Other types of attacks are not addressed practically in the case studies, considering that in the qualitative risk analysis for the selected models in production, evasion was considered the main risk. But other types of attacks are highly recommended to be analyzed in a qualitative way and also in a practical way when considered relevant, such as the poisoning attacks when automatically collecting training data from real cases where attackers may be responsible for some samples.

Each analysis was performed using Open Source tools, Python as the programming language, and cutting-edge libraries and frameworks in Data Science, Machine Learning, and Deep Learning.

Furthermore, it is necessary to mention that this thesis is approached in a way that does not compromise confidential information according to the company's policies, reason why it will focus more on the description of the analysis performed, the type of conclusions that can be drawn, and presenting results with no real data.

Chapter 2

Machine Learning in Cyber Security

2.1 Introduction to Cyber Security and the Role of ML in its development

Cyber security in *A Survey of Deep Learning Methods for Cyber Security* is defined as:

"The collection of policies, techniques, technologies, and processes that work together to protect the confidentiality, integrity, and availability of computing resources, networks, software programs, and data from attack. "(Berman et al., 2019)

In cybersecurity, there is a wide range of defenses oriented to different IT components and information systems, existing for applications, networks, data, etc. To be effective, they must be used simultaneously or serially, and they must be constantly updated since an attacker could find a minor breach enough to compromise the system. In addition, systems are becoming more exposed every day as they are more interconnected through the Internet and cloud-based tools.

Many defenses are created focused on known attacks, attempting to develop generalizable rules to protect systems against them. However, there are problems that require the generation of more complex and difficult to extract rules, where ML can help to generate them, as it has the ability to extract them from existing abundant data stored in IT systems.

2.2 ML Applications in Cyber Security

In this way, all types of ML algorithms are used in cybersecurity, from distribution analysis, hypothesis testing, and traditional statistical techniques such as linear regression, to supervised techniques with methods with greater predictive potential, unsupervised clustering techniques, anomaly detection techniques, etc. Additionally, they can be used depending on the problem to be solved with traditional structured data, time series, and increasing potential in fields with unstructured data such as NLP, Speech Processing, and Computer Vision. Particularly, there is increasing use of Boosting and Bagging techniques, and DNNs as summarized in A Survey of Deep Learning Methods for Cyber Security, among which Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Generative Adversarial Networks (GANs), Deep Autoencoders, etc. are used.

Some of the applications are the detection and classification of malware, generalizing patterns, and allowing to adapt them with data to new attacks easily. They can also be used, for example, to detect network intrusion, in which data on user behavior can be used to detect anomalies. They are helpful for creating SPAM or Phishing detection models, for which email metadata and text-based features, ranging from Bag of Words to embeddings, can be used. They can be used with all metadata types, and in some cases, it is sufficient to create features that are candidates to be relevant in the analysis, for example, using one-hot encoding. In the case of images, it is typical to use CNNs, RNNs for sequential or time-series data, Autoencoders to create useful representations while reducing their dimensionality, or also to train unsupervised models for anomaly detection.

2.3 Importance of Correctly Defining the Scope and Approach of the ML Models

Additionally, it is essential to know how to approach problems from different perspectives and to understand how to create models with a scope that maximizes their potential, and not necessarily make a single model that does not have the ability to address all cases or provide all the capabilities, being this one of the conclusions in *A Survey of Deep Learning Methods for Cyber Security*:

"Researchers should consider how to generalize or integrate different DL approaches to cover a broader range of attack vectors. Multiple DL detection schemes are needed in parallel and may also benefit from knowledge learned by different methods to improve its performance locally." (Berman et al., 2019)

Additionally, it is crucial to carry out the logical process to consider whether, in the data used in the creation of the ML model, there is relevant information that can correctly discriminate or predict the target variable, being important not to skip the business logic and to consider the assumptions that there is a pattern in the data to be extracted to achieve the purpose of the model. In the case that the most important variables are not covered directly with the available data, it should be considered that the approach to solving the problem will have limitations and difficulties in cases that do not maintain the same correlations so that despite mitigating the problem, it should be prepared to have errors and accept lower performance.

On the other hand, if the variables or patterns to be detected are minimal, the purpose of the model should be correctly defined and not try to directly predict the variable of most significant interest, being clear about the scope of the data. For example, suppose one wants to use only the title of an email to predict phishing. In that case, one must be aware that the information is insufficient to detect all types of phishing. Its objective should be to detect instances of evident phishing that only by the title can generate an alert (that the suspicion of phishing is not external to the title). But suppose one wants to predict phishing completely. In that case, the complete body of the email, metadata on how and who sent it, should be used (with one or multiple models depending on the best way to deal with the problem with modularity). Even in this situation, there may be many cases that imitate correct emails that the model is not able to differentiate with the potential of the data.

2.4 Traditional ML Metrics Biases and Limitations, and Importance of Considering AML

As presented in several papers such as *A Survey of Deep Learning Methods for Cyber Security* (Berman et al., 2019), detection and classification models are evaluated for their predictive performance on a test dataset, using, for example, metrics constructed solely from the confusion matrix in the case of classifiers. This test dataset is usually created with non-training cases to avoid overfitting and attempting to measure the model's generalization capacity. Still, these cases are assumed to come from the same distribution from the training test, generally not including Out-of-Distribution cases, complex new cases, or adversarial examples. Additionally, these algorithms have more significant difficulties in diagnosing the root cause of an error when they are black-box models and understanding their specific weaknesses to be reinforced.

Therefore, it is very important to study Adversarial Machine Learning, which is vital to analyze these weaknesses and strengthen the systems against these types of problems:

"This generally falls under the adversarial examples, an emerging research area that examines the weaknesses and susceptibilities of machine learning models. This will be crucial to hardening systems against zero-day attacks by particularly sophisticated attackers." (Berman et al., 2019)

2.5 Additional Relevant ML Research Fields

The European Union Agency for Cybersecurity (ENISA) published the report *Securing Machine Learning Algorithms* (Malatras, Agrafiotis, and Adamczyk, 2021), in which they explain how ML can be used in cybersecurity, a taxonomy of algorithms, and then analyze their properties and threats, emphasizing the importance of making them secure and trustworthy. It is interesting to note the alignment of this paper with this thesis topic, and notice that it was published on December 14, 2021, while this thesis was being written, highlighting the importance of not only applying ML algorithms in cybersecurity with traditional performance metrics but also being aware of all the kind of threats they entail. Also they cover the topic not only from a purely adversarial point of view as done in AML, but also in other types of vulnerabilities such as ML model generalization to OOD cases, Overinterpretation and the need to create explainability.

These topics will be discussed in greater depth in the following sections, as they are all closely related and should be considered together to create a ML model.

Finally, to end this section, it is important to mention that there are areas of research with very little research so far which have emerged from ML risks in cybersecurity, such as Active Learning. This research field studies how to monitor the detection of adversarial cases in the deployment of the model since it could be important to alert about a reduction in the number of attacks detected by it. This should be analyzed, considering that this pattern may be caused by an attacker who has created a new attack that successfully evades the model. This open problem is also essential to be considered when monitoring and retraining the models.

Chapter 3

Adversarial Machine Learning

Adversarial Machine Learning is a ML research branch that attempts to find the different ways a malicious adversary could attack a ML and the way to prevent or protect it against them. Thus it can be subdivided into two branches, one actively designing attacks to damage ML models, and the second one to improve their ability to resist such attacks.

3.1 AML Context in Computer Vision: Unbalance between Attacks and Defenses

It is essential to clarify that as mentioned in *Adversarial Machine Learning for Cybersecurity and Computer Vision: Current Developments and Challenges* (Xi, 2020), there is a significant difference in adversarial examples for Computer Vision compared to other application areas. In some applications, adversarial examples tend to have different properties and distributions compared to training data, which allows exploiting this phenomenon to create defenses, for example, using anomaly analysis, however for Computer Vision, this property does not seem to be fulfilled in a large number of cases.

In addition, in the Computer Vision field, adversarial examples can be created with minor disturbances that at the human visual level do not represent a change in the image, but that can create a considerable malfunction (as has been demonstrated in several papers, starting with the most popular Explaining and harnessing adversarial examples by the AML pioneers: Goodfellow, Shlens, and Szegedy). This susceptibility makes it possible to create a wide variety of attacks in the Computer Vision field, which seem to have no limitations on their design. These perturbations manage to affect the models easily, while it is complicated to develop robust techniques to resist all types of attacks on the defenses' side. The proposed defenses are usually designed to resist specific attacks in consideration but are easily affected by another kind of attack using different logic or approaches. Or as said in *Recent Advances in Adversarial Training for Adversarial Robustness*:

"Naturally, the defenses also stimulate the development of stronger attacks, seeming like an arms race." (Bai et al., 2021)

In fact, in 2017, Google Brain organized a competition at NIPS to attract researchers to create new adversarial attacks and defenses. In the case of defenses in *Adversarial Attacks and Defences Competition* (Kurakin et al., 2018), it was concluded that no defense is entirely satisfactory. No matter how much you measure it with existing attacks, you can never guarantee its success as you can always have new inputs from different distributions.

Additionally, by being able to easily and directly affect the raw data (image), also being able to visually see the disturbances necessary for an effective attack, and being able to compare it to the human ability to detect them as well (which is often not evident), this is the area where AML has been explored the most. For example, Tesla's systems have had to break down the task of visualizing and understanding the environment into multiple models with multiple sensors with varying functionality to ensure the security achieved today.

Consequently, as mentioned in *Adversarial Machine Learning for Cybersecurity and Computer Vision: Current Developments and Challenges* (Xi, 2020), it is urgent and vital to create ML techniques that are safe, and that can overcome different types of attacks, and at present, although there is not something that achieves this, ideas are coming from various fields such as computer vision, cybersecurity, artificial intelligence, and machine learning.

3.2 AML Attacks Taxonomy

AML attacks can be classified according to a combination of two main factors: the assumptions and knowledge the attacker has about the model, and the second the stage of the ML model they affect. In turn, these can be targeted or non-targeted, being targeted when they seek, for example, to create a specific misclassification, and non-targeted when they seek to affect its performance at a general level. Additionally, in the case of images, attacks can be digital or physical, being in the first case perturbations at the level of images represented as pixels in numerical arrays (which allow detailed changes). In contrast, physical refers to affecting natural objects that are photographed, such as painting a specific shape on a traffic sign.

3.2.1 Model Knowledge

The adversary may know or have access to elements of the model, and therefore attacks can be classified into White-box, Black-box, and Grey-box attacks. White-box attacks are those cases in which the adversary has complete knowledge of the model, its architecture and parameters, and the training data. On the other hand, in Black-box attacks, the adversary does not have access to the internal configuration and can only know the output of a given input, necessarily requiring iteration to accomplish an attack. Finally, Grey-box attacks assume an intermediate scenario, in which the attacker possesses partial information about the model and its training data. In the case of Black-box attacks, it is important to consider that for some models, if the way of testing is optimized, it is possible to take an example and modify it to create an effective adversarial example. For Grey-box attacks, it is also possible to create a substitute model using suitable training data and design attacks with it using a White-box approach, with the interest of transferring them to the real model. This is an advantage considering that there are many efficient ways to create White-box attacks in the state of the art compared to the Black-box scenario (at least in the case of Computer Vision). One way to increase the transferability effectiveness of the attack created with the substitute model over the real model before testing it is to test the transferability between several substitute models as done in *Delving into Transferable Adversarial Examples and Black-box Attacks* (Liu et al., 2017), in which the models are studied geometrically, showing how it is possible to have transferability of adversarial attacks between different models, considering that their decision boundaries tend to be aligned.

Depending on the dimensionality, there may be cases where it is unnecessary to know the training data, and the appropriate substitute model can be created by testing some instances over the model in the Black-box scenario. This analysis is performed in detail in the first case study, in which, due to the curse of dimensionality, this approach is not efficient.

3.2.2 Attack Types

Among the different ways of attacking ML models, the three most studied are poisoning attacks, evasion attacks, and privacy attacks. Poisoning attacks contaminate the training data to make the model ineffective. On the other hand, evasion attacks, as their name suggests, seek to evade the model, avoiding detection or correct classification in its deployment. Finally, privacy attacks seek to infer information from the model training data by analyzing its structure and parameters.

3.2.2.1 Poisoning Attacks

In particular, attackers using poisoning attacks seek to make the ML model learn characteristics designed by them and added to data examples that will be probably be used in the training process while being unnoticed by the ML owner. It is done to bias the model to learn using the introduced characteristics instead of other relevant features. This is why they are also known as backdoor attacks, in analogy to the fact that they are cases that activate the backdoor trigger created by a case designed by the attacker as a backdoor key, which are unusual patterns in the training set that achieve the same final result.

An example may be the use of certain benign words in phishing emails that, when used in the training process by a classifier, will associate these words as phishing cases and will create false positives. This will affect the model's performance and even lead the model owner to increase the decision threshold to reduce the false positives again, indirectly decreasing the defenses against phishing cases. Or, for example, adding specific details to an image from a particular category and then forcing a CNN classification to use it as a spurious feature, causing different images with the same detail to be classified in the same category. Something to note is that in the training process, it is very likely that this vulnerability cannot be known when using the traditional performance metrics of accuracy and metrics based on the confusion matrix since an image of the adversary would not have been tested yet.

For example, in *Bagging Classifiers for Fighting Poisoning Attacks in Adversarial Classification Tasks* (Biggio et al., 2011) propose to use bagging ensembles, arguing that poisoning attack instances can be seen as a particular type of outliers, being important to use techniques that are robust to outliers (that do not vary their parameters significantly with an outlier as in techniques such as linear regression in which these instances have large leverage). In the case of DNNs in *Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering* (Chen et al., 2019) propose to analyze the embeddings of the last representation layer to analyze cases that appear to be in a different cluster from the rest of the examples from the same class.

3.2.2.2 Evasion Attacks

On the other hand, evasion is the most common attack, as they seek to add perturbations to a case (which the adversary tries to keep to a minimum in order to pass unnoticed) in order to fool the model. In models using images, it is usually easy to achieve this with imperceptible perturbations to the human eye, being even easier to accomplish this in White-box scenarios, as explained before. The perturbation can be optimized using different approaches and methods. In the case of Black-box attacks, requests iteration is required to estimate, for example, the gradient of the model, to approximate a good solution as done, for example, with the ZOO method in *ZOO: Zeroth Order Optimization based Black-box Attacks to Deep Neural Networks without Training Substitute Models* (Chen et al., 2017), or to create a substitute model to create White-box attacks expecting them to be transferable to the real model.

These models are the main focus of this document, reason why they will be explained in greater detail in chapter 5.

3.2.2.3 Privacy Attacks

Finally, privacy attacks have a very different focus. They are usually segregated into a separate research area and attempt to obtain information about the training data, motivated to deduct personal or sensitive data. They can be divided into inversion and membership inference attacks, with the inversion case seeking to recreate an input given an output, such as reconstructing a face from a facial recognition system. In contrast, membership inference attacks aim to learn whether a specific example was used inside the training data. Although these attacks are not discussed in greater depth in this document, it is essential to create strategies to avoid and detect them.

3.3 AML Defenses

3.3.1 AML Defenses Current State Overview and Considerations

As mentioned earlier, creating defenses that adapt to all types of attacks is very difficult and varies depending on the type of problem and model. There are defenses designed to improve against specific attacks such as poisoning, evasion, and privacy attacks such as those presented in the previous section, and in turn, within each type of attack, these defenses are often designed to counteract attacks made with very specific methods. In Computer Vision exist a wide variety of attacks, as explained previously. In the competition created by Google Brain in NIPS (Kurakin et al., 2018), it is mentioned that no defense is entirely satisfactory and that the proposals that are not wholly successful are documented as research paths.

Adversaries can be considered static or adaptive, being a static adversary one that manages to create attacks but is not aware of the possible defenses used by the model owner, making attacks without adapting them to overcome any particular defense. Although not all attackers will be static in practice, those who are will try common attacks without much expertise, and some defense methods would be useful to filter their attacks. However, the worst-case scenario concerns adaptive adversaries that can guess the type of defenses that are being added to a model and thus improve their attacks considering them. With adaptive attacks, it is evident that an entirely satisfactory defense cannot be found since this is precisely the scenario simulated in the state of the art trajectory, with researchers constantly succeeding in overcoming defenses proposed by other researchers. An example of this process is shown in the following section.

It is also important to remember that there is usually a trade-off between adversarial robustness and model performance, so it should also be clarified that although there are more robust models, their performance is not as high as other methods in the same task. These metrics, adversarial robustness and performance, should be considered simultaneously when evaluating a model, and in fact there are researchers who focus specifically on analyzing how to achieve this, as in the case of *Adversarial Examples Improve Image Recognition* (Xie et al., 2020), where the authors goal is to ensure that adversarial examples provide real value, taking advantage of their potential to create more human-aligned representations, robust to high-frequency noise and robust to distortions by focusing more on shape information. In this paper, to try to extract this potential, it is proposed to separate the Loss function into a term for legitimate cases and another for adversarial ones (to make them independent and not affect the legitimate cases), and it is proposed Disentangled Learning with Auxiliary Batch Normalization, using two parallel batch normalization layers, one for legitimate images and another for adversarial examples.

In general, there are two directions of research on defenses against evasion adversarial attacks. One seeks to correctly predict adversarial examples without adding or needing an additional model while maintaining the performance over legitimate cases. On the other hand, there is a line of research to detect adversarial examples to only perform the prediction step over the legitimate cases, leaving out of the inference step the already detected adversarial examples (He et al., 2017). When detecting adversarial examples, it should also be tried to avoid increasing the number of false-positive cases. In this last approach, the defense is independent of the predicting model and is not embedded inside it.

3.3.2 AML Evasion Detection Defenses

Among the defenses proposed in the NIPS challenge for evasion using images classifiers, the *feature squeezing defenses* aim to remove noise from images before applying the model to them. For attacks made with small changes in multiple pixels, it counteracts by compressing the images beforehand, and for high-frequency noise attacks by applying median filters (removing these outliers). A possible drawback of this approach is that it may sacrifice details from the images and affect their accuracy. This has also been used to create an adversarial examples detector, using three different versions of the same image (the original, the compressed, and the median smoothing image), and estimating the largest L_1 norm found between them in the output of the final softmax layer of the DNN. A large L_1 norm would represent a high probability of having an adversarial attack. This assumes that legitimate images do not change significantly when applying any of the previous transformations compared to adversarial examples. This decision process requires defining a reasonable threshold to segregate adversarial examples but also avoid false positives.

However, considering the adaptive attacker scenario, in *Adversarial example defense: Ensembles of weak defenses are not strong* (He et al., 2017) found simple ways to attack effectively this type of defenses in the White-box scenario. An example is binary squeezing, in which the image is modified to have a small discrete number of colors by changing the pixels' colors to the nearest ones. Suppose we decided to use only two colors with a threshold in between (for example, white and black images). An adversary can overcome this defense by generating, for example, a gray image with values close to the threshold to avoid detection, having a small L1 norm when comparing the original and the compressed image. Some tests have been made with ensembles of this type of defenses without creating a truly robust defense (Adversarial example defense: Ensembles of weak defenses are not strong).

Also, in *On Detecting Adversarial Perturbations* (Metzen et al., 2017), classifiers are created in Neural Networks by adding subnetworks that emerge as branches from the main network in different depth layers. The process consists of first standardly training the main network with legitimate data, then freezing it to train each subnetwork using legitimate images (with category 0) and images with some adversarial perturbation (with category 1). The adversarial examples can be created with any method, being Fast Gradient Sign Method (FGSM), Basic Iterative Method (which is an iterative version of FGSM), and DeepFool methods tested in the original paper.

An additional and complementary method is proposed in *Mitigating Evasion Attacks to Deep Neural Networks via Region-based Classification* (Cao and Gong, 2017), which tries to detect if an example was created with an attack by making predictions in several examples in a close neighborhood. Suppose there is high variability over these points prediction. In that case, there could be a higher probability of having an adversarial example, assuming that the adversarial attacks will create points near the decision boundary. In this way, it is possible to infer the probability of having an adversarial example, being possible to have natural examples that are difficult to classify over a boundary region too.

This method can be useful when working with images where the perturbations directly represent the intensity of the attack, but it can have difficulties in other types of data instances. For example, with high dimensional data, it would be difficult to create enough examples to cover a relevant region in the neighborhood of the example. Also, it is possible that creating artificial examples in the neighborhood doesn't make sense in the vector representation according to the problem. For example, categorical features would not accept continuous values, some artificial examples may not represent feasible cases for the problem, and there could be a difference in the importance of the features, being necessary to alter the neighborhood region to really represent similar and feasible cases.

3.3.3 AML Evasion Model Defenses

Among the most popular defenses with apparent most significant impact are Adversarial Training, Gradient Masking, and Input Transformation techniques. Each of them is explored according to their positive results in practice and their state-of-the-art in-depth study.

3.3.3.1 Adversarial Training

3.3.3.1.1 Adversarial Regularization

As stated in the *Adversarial Attacks and Defences Competition* (Kurakin et al., 2018) one of the most popular defenses is Adversarial Training, which seeks to inject adversarial examples into the training process, either by using them alone or mixing both legitimate and adversarial ones. The first approaches added adversarial examples in the training dataset as first done in *Explaining and Harnessing Adversarial Examples* (Goodfellow, Shlens, and Szegedy, 2015), but then in *Improving the Robustness of Deep Neural Networks via Stability Training* (Zheng et al., 2016) proposed adding a term to the objective function of the DNN to force the model to create similar representations for the legitimate cases and their corresponding adversarial cases, using KL-divergence.

In general terms, Adversarial Training can be defined at a mathematical level as a min-max problem, seeking to create the best solution (or lowest model performance error) to the worst-case optimum (maximizing the impact of the adversarial example bounded by a constraint on the perturbation to be created).

In *Recent Advances in Adversarial Training for Adversarial Robustness* (Bai et al., 2021), an exhaustive analysis of this technique and its most recent development is made. This survey emphasizes the effectiveness of this technique and its interesting approach to make models more robust intrinsically compared to other types of defenses. However, it is also pointed out: "It is still a long way to go for adversarial training to handle attacks perfectly." One problem with this technique is that it tends to overfit the attack method used to create the adversarial examples and doesn't guarantee an improvement against other attack methods.

In this survey, a taxonomy is created, and the progress made in Adversarial Training is shown. It is explained how Huang (Learning with a strong adversary) is the first to analyze the problem with a min-max approach. It is emphasized that the central importance relies on creating a strong adversary the adversarial example maximization effect achieved.

There are techniques that extend Adversarial Training as in the case of *Metric Learning for Adversarial Robustness* (Mao et al., 2019) in which a triplet loss is proposed for classifiers, taking a base case (positive case), creating an adversarial example (anchor case) and a nearby case with a different ground truth (negative case). The distance between the anchor case and the positive case is minimized (as in traditional Adversarial Training), but also the distance between the anchor case and the negative case is maximized, in order to increase the boundary between classes in the creation of the representation.

3.3.3.1.2 Other Adversarial Training Methods

The previous cases are examples of Adversarial Training through Adversarial Regularization, focusing on modifying the loss function, specifically the cross-entropy loss for classification. However, there are other techniques, such as Curriculum-based Adversarial Training, which has been shown to have better results than traditional training. It uses the same concept of Curriculum Training, starting the training process with easy cases and progressively adding more difficult ones. The transference of this concept to Adversarial Training is done by gradually increasing the strength of the attack in each epoch of the training process (Cai, Liu, and Song, 2018). In *Recent Advances in Adversarial Training for Adversarial Robustness* (Bai et al., 2021) it is concluded that these methods are useful to improve generalization over clean data while preserving adversarial robustness, considering that soft attacks in the early stages improve generalization, and the stronger attacks then help to reduce possible overfitting.

There are also variations in this field with Ensemble Adversarial Training, in which an ensemble of models trained with adversarial training is created, using each model different adversarial examples creation techniques. They usually improve robustness to single-step attacks such as FGSM (Fast Gradient Sign Method). Still, it is not easy to make the different created models provide additional value since they seem to create similar representations. Several techniques have been used to mitigate this as done in *Improving Adversarial Robustness via Promoting Ensemble Diversity* (Pang et al., 2019) and *Improving Adversarial Robustness of Ensembles with Diversity Training* (Kariyappa and Qureshi, 2019).

3.3.3.1.3 Instance Level Adversarial Training

Other authors have considered that not all examples in adversarial training should use the same attack strength since they do not have the same robustness to adversarial attacks (this is evident, for example, for certain classes or variations within certain classes in the case study 2 in this document). Using Instance Adaptive Adversarial Training (IAAT) in *Instance Adaptive Adversarial Training: Improved Accuracy Tradeoffs in Neural Nets* (Balaji, Goldstein, and Hoffman, 2019) proposed to do adversarial training at instance level, in which the perturbation is carried out by setting an epsilon radius as large as possible, with the constraint that all instances within the epsilon-ball are of the same class. This helps to decrease the trade-off between adversarial robustness and performance, mainly by maintaining good performance while sacrificing little robustness. Another instance-level approach is Margin Maximization Adversarial Training (MMT) (Ding et al., 2018) in which the author seeks to maximize the model's decision boundary, justifying that each instance requires a different epsilon, since a too large epsilon will affect the accuracy, and a very small epsilon will not improve robustness, being suboptimal to define a static epsilon for all instances.

3.3.3.1.4 Additional Improvements in Adversarial Training

Among all the variations found in state-of-the-art, some also focus on doing adversarial training using unlabeled data, which can be very useful for cases with few available training instances. Some papers concentrate on improving their computational time, considering that these processes take considerable additional time compared with standard training, as explained in more detail in *Recent Advances in Adversarial Training for Adversarial Robustness* (Bai et al., 2021). An interesting simpler approach is proposed by IBM researchers in *Efficient Defenses Against Adversarial Attacks* (Zantedeschi, Nicolae, and Rawat, 2017), where the training data is augmented using Gaussian filters to create softer images that will produce smoother and more stable models. Authors claim that this technique could increase the model's adversarial robustness without compromising clean images performance, and without needing to create adversarial examples.

Finally, it is important to mention that there is criticism or doubts about the claimed success some authors make about their proposed methods (Bai et al., 2021). In some cases, the proposed methods are not as effective as they appear to be in the paper results when used with other datasets or problems. Some complex methods do not seem to be considerably more effective than other more basic techniques, such as early stopping during the training process.

3.3.3.1.5 Metrics in Adversarial Training

In creating variants and improvements of techniques such as Adversarial Training that seek to improve the model intrinsically, it is crucial to evaluate the model's ability to generalize. This can be analyzed from three different perspectives, standard generalization to examples from the same training distribution, generalization to adversarial examples manufactured in the adversarial training process, and generalization to unseen attacks. There is no direct way to measure each of these properties of a model, but there are certain metrics that under certain assumptions allow having an indirect notion about them.

For the standard generalization of In-Distribution (ID) instances, traditional performance metrics such as the accuracy over a test set of the same training distribution can be used and analyzed in the desired depth. In the case of adversarial robustness, the distance in the representation for an example and its corresponding adversarial example can be measured using, for example, L_2 or L_{inf} norms, being ideally close between each other. In this representation space, it is also possible to measure the distance required to cross the decision boundary for each of the instances. On the other hand, to measure in some way their resistance to new attacks, it is possible to create attacks with substitute models and measure their transferability to different types of attacks and attack intensities.

3.3.3.2 Gradient Masking and Defensive Distillation

These techniques seek to prevent attackers from taking advantage of gradients in a White-box scenario. This is done by using non-differentiable operations or by forcing the network to have null gradients in many locations, or even preventing the gradients from pointing to the decision boundary.

Defensive Distillation (Carlini and Wagner, 2017) is based on the distillation technique, which is normally used to transfer knowledge from models with large architectures to smaller architectures models. Papernot extends this process by adapting it to create more resilient to adversarial examples network. In this process, the variation around the input is reduced, considerably reducing the magnitude of the gradients and making it more difficult for an attacker to use them. However, this defense seems to be easy to overcome by an attacker by slightly increasing the intensity for some attacks and is not helpful for attacks such as Carlini and Wagner's, as demonstrated in *Mitigating Evasion Attacks to Deep Neural Networks via Region-based Classification* (Cao and Gong, 2017).

These techniques do not seem to be efficient when transferring attacks with substitute models, since although they make it difficult for the adversary to have the gradient, the decision boundaries are essentially the same (Papernot et al., 2017). Considering the facility to transfer attacks as demonstrated in the case study 2 in image classification, this defense is not expected to be of great value in that scenario. This is also demonstrated in *Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples* (Athalye, Carlini, and Wagner, 2018) which by name reveals its conclusion quite well, showing that the adversary can in other ways infer the gradients or approximate them, and thus hiding them gives a false sense of security in being deducible in other ways. Consequently, both Adversarial Training papers and other papers on input transformations justify that it is better to replace these defenses with one of these methodologies (Raff et al., 2019).

3.3.3.3 Model Ensembles

As mentioned in *Metric Learning for Adversarial Robustness* (Mao et al., 2019), model-ensemble is a useful method to increase the model robustness, which is orthogonal to other techniques like Adversarial Training. In *Improving Adversarial Robustness via Promoting Ensemble Diversity* (Pang et al., 2019) the importance of not only averaging outputs (regression) or selecting by a majority (classification) the output of any multiple models is mentioned. In this paper, they note that it is important to consider the interaction between the individual models and promote diversity between them to improve overall robustness by making it more difficult to transfer adversarial examples between them. In particular, this is done by adding

Adaptive Diversity Promoting (ADP) regularizer in the loss function, which seeks to make the distribution of the non-maximal predictions (the output of the softmax without counting the dominant class) more diverse. With this method they achieve results that improve robustness while maintaining the accuracy to normal examples (ID examples).

3.3.3.4 Input Transformation Techniques and Other Techniques

The second-best defense in the NIPS 2017 adversarial examples defense challenge (Kurakin et al., 2018) then published in *Mitigating Adversarial Effects Through Randomization* (Xie et al., 2018) proposes to add randomization at the time of inference, without the need to do additional training, with a low computational cost for variations and compatible with other defenses. The method simply consists of random resizing (in a reasonable range) and random padding in the image classification. It works under the assumption that the attacks do overfitting to the model parameters and that these transformations can destroy the adversarial perturbations. Also among the top 4 best defenses of the challenge itself, one was based on applying spatial smoothing with median filters, similar to the feature squeezing techniques explained in the previous section combined with other complementary defenses.

With a different orientation, in *PixelDefend: Leveraging Generative Models to Understand and Defend Against Adversarial Examples* (Song et al., 2018) assumes that adversarial examples are usually outside of the natural data distribution (which has been argued to be not always valid especially in images), and before making a prediction, seeks to bring a new example closer to a natural example before passing it through the model, in order to avoid using it with the added noise from an attacker.

The best defense of the NIPS 2017 adversarial examples defense challenge (Kurakin et al., 2018) uses as a basis the same logic from a Denoising autoencoder (DAE), but to avoid the bottleneck that forces to lose information, they use a U-net and call it DUNET in *Defense against Adversarial Attacks Using High-Level Representation Guided Denoiser* (Liao et al., 2018). The main advantage over the DAE is that it adds lateral connections of the encoder layers in the decoder with the same resolution. They propose a particular architecture and minimize the reconstruction distance in the training process. It is important to note the complexity of this defense and the difficulties of training it and adapting it to other contexts.

An interesting paper seeks to integrate several weak defenses to make a strong defense in *Barrage of Random Transforms for Adversarially Robust Defense* (BaRT) (Raff et al., 2019), which contrasts with the conclusions drawn two years earlier in (Adversarial example defense: Ensembles of weak defenses are not strong) which as can be seen from its name, stated that joining several weak input transformation would not improve the robustness from a model any further. This is demystified with BaRT by adding a random component and using a large set of different defenses that make this method very effective.

The random part is essential so that the adversary cannot deduct the defense used, no matter how "omnipresent" it may be regarding the process, in the most white-box scenario possible. Ten groups of transformations are used, which, when opened represent 25 different possibilities, from which a subset is chosen randomly in a random order creating a large spectrum of combinations. Among the transformations are:

- Color Precision Reduction: Feature squeezing defenses similar to those explained in section 3.3.2, but varying the number of ranges in each channel, and allowing each channel to have different numbers of ranges.
- JPEG Noise: Compressing them with different possible values.
- Swirl: A soft swirl is produced in the image by rotating the pixels in some regions of the image.
- Noise Injection: Adding gaussian noise, salt and pepper, speckle, and applying it to the channels equally or differently.
- FFT (Fast Fourier Transform) Perturbation: Performing the FFT for each of the channels, and converting some random or lower frequency coefficients to 0.
- Zoom Group: Trying to avoid "over zooming" and choosing some area of the image.
- Color Space Group: The RGB image is transformed to another color space such as HSV, XYZ, LAB, or YUV, making modifications and returning it to RGB.
- Contrast Group: Using different histogram equalizations to re-scale and redistribute the values.
- Grey Scale Group: Using different grey scales conversions.
- Denoising Group: Applying filters to remove high-frequency noise, with median, mean, and Gaussian filters.

3.3.3.5 Additional Good Practices Based on Experimentation

Finally, experimental papers also provide interesting conclusions regarding trends and recommendations to obtain more robust models before applying any additional defense. For example, in *Is Robustness the Cost of Accuracy? - A Comprehensive Study on the Robustness of 18 Deep Image Classification Models* (Su et al., 2018) after extensive experimentation, gives recommendations about architectures with a good balance between accuracy and adversarial robustness, showing, for example, higher susceptibility to attacks in architectures such as VGG. This paper demonstrated the importance of not selecting an architecture only by its accuracy, but also to guarantee robustness, also showing some statistics about the most transferred attack methods, the architectures between which they are transferred, etc. Interestingly, this paper shows the trade-off between accuracy and robustness with a Pareto frontier, which is useful to decide with a multi-criteria approach.

3.4 AML Metrics

3.4.1 Analysis of the Difficulty of Measuring Adversarial Robustness

Due to the nature of the analyzed problem, it is difficult to generate metrics that allow to measure in a standard way the adversarial robustness of a ML model, and thus its improvement through the use of a specific defense. There are no direct metrics as it happens with the traditional performance metrics estimated with the accuracy over a test set from the same training distribution, considering that the assumptions are not the same and the cases to be considered in the statistic will never contemplate all possible attacks or scenarios.

After analyzing the literature, three main particularities are found related to the way to measure the adversarial robustness from a model with a defense: **bias to adversarial test set attack method, bias to attacks designed using the original model, and bias to only originally effective attacks**. These will be explained in the following sections.

Additionally, the adversarial robustness is also dependent on the performance of the original model (without defense) over clean examples, as it would be easier to create an adversarial example from an original example that is already difficult to predict.

3.4.1.1 Bias to Adversarial Test Set Attack Method

In many cases, the model's adversarial robustness is estimated with performance statistics over a test set, being the estimation biased to the adversarial examples added to the collection and, in this way, biased to the attack method selected.

The metric is dependent on the type of attack applied to test the model, not being possible to create a metric independent of the attack method used. This does not allow creating a standard straightforward metric, as explained in *Adversarial Attacks and Defences Competition* (Kurakin et al., 2018) regarding the competition realized, where it is mentioned that the participants faced an open-ended problem with attacks from unknown distributions:

"It is not sufficient to benchmark a defense against a single attack or even a suite of attacks prepared ahead of time by the researcher proposing the defense. Even if the defense performs well in such an experiment, it may be defeated by a new attack that works in a way the defender did not anticipate. Ideally, a defense would be provably sound, but machine learning in general and deep neural networks in particular are difficult to analyze theoretically."

In this competition, defenses are tested against attacks from other independent teams as an approximation to a real-life case, but it is made clear that this is evidently not a conclusive evaluation like a theoretical test.

Also, after reviewing state of the art, the metrics used to measure the Adversarial Training improvement are based on the same attack method used in the training process, being this metric to this particular attack. Doing this will logically make the new DNN with Adversarial Training have higher robustness, being totally biased in its analysis and overfitting to the attack algorithm. It should be clarified that this is not the case in all papers, as multiple attacks are tested in the most recent ones.

Also in *Recent Advances in Adversarial Training for Adversarial Robustness* (Bai et al., 2021) it is shown how adversarial robustness is usually estimated by focusing in the performance over an adversarial example dataset. It is mentioned here that the dataset can be created using White-box attacks to measure the worst-case scenario, and that generally the most used method is Projected Gradient Descent (PGD), as done in *Towards Deep Learning Models Resistant to Adversarial Attacks* (Madry et al., 2018). This method is explained in section 5.

3.4.1.2 Bias to Attacks Designed Using the Original Model

In many cases, a defense effectiveness is measured by comparing the performance of the original model against the performance of the model with defense, using the same adversarial test set. The creation of this adversarial test set is created in many cases with White-box attacks using the original model architecture and parameters for its development. This can bias the comparison between the two models as the attack would tend to be more effective over the original model, being specifically designed for it, and probably less effective for the second model with different parameters. In this way, it is not possible to know if the improvement of the metric is due to the defense, or rather just because the model with the defense has different parameters from those used to create the attack.

It is proposed to use an external model as a basis for the White-box adversarial examples created for the adversarial test set. This would be shown in case study 2.

3.4.1.3 Bias to Only Originally Effective Attacks

Finally, when comparing the performance of the original and defense model to an adversarial test set, it is frequent to see papers measuring the improvement by considering only the effective adversarial examples over the original model as a basis, and then estimating the percentage of cases also effective over the defense model. This metric will have values between 0 and 100%, being the first case when the defense model can detect correctly all adversarial examples that were effective over the original model, and the second one when being fooled by all the same adversarial examples. Clearly, any value lower than 100% will appear to show an improvement by the defense model, but it is biased to analyze the vulnerabilities from the original model without measuring new vulnerabilities that the model with defense may have.

This problem can even be seen in experimental papers whose main objective is to measure the transferability of adversarial examples as in *Is Robustness the Cost of Accuracy? - A Comprehensive Study on the Robustness of 18 Deep Image Classification Models* (Su et al., 2018), in which it is observed that the transferability and vulnerability of the models to attacks is measured only based on the effective attacks of the original model. In this paper, this is visualized with a square transferability matrix, with 100% values on the diagonal and values equal to or less than 100% in the rest of the cases. At no point any analysis of the weaknesses or attacks that are now more efficient in the model to which the transfer is made.

This is intended to be analyzed in case study 2, where some vulnerabilities from the original model are improved in the model with defense, but also new weaknesses from the defense model that are not in the original model appear.

3.4.2 Adversarial Robustness Metrics

There are some metrics to approximately estimate the intrinsic robustness of a model (Su et al., 2018). For example, the CLEVER score (Cross-Lipschitz Extreme Value nEtwork Robustness) in *Evaluating the robustness of neural networks: An extreme value theory approach* (Weng et al., 2018), which is an attack-agnostic for neural networks. It seeks to estimate the minimum distortion required to create an adversarial example from an original example, measuring it with some L_p norm with p being greater or equal to 1. Other metrics are oriented to measure the attack success rate, being possible that for some instances, it is not even possible to perform the attack restricted to a maximum perturbation.

Also aligned with CLEVER, it is possible to measure the distortion required to create the attacks and analyze the data's norm statistically. In the case of external attacks, the transferability of attacks created with other models, which may vary in architectures, attack type, and intensity, can be measured. A Pareto frontier can also be used to compare various models on two criteria: accuracy (generalization or standard robustness) vs. distortion required by adversarial examples (similar to CLEVER) as an approximation to adversarial robustness. A similar approach is made in *DeepFool: a simple and accurate method to fool deep neural networks* (Moosavi-Dezfooli, Fawzi, and Frossard, 2016).

Finally, it is essential to emphasize how useful is the visualization of the representations made in DNNs over the data, using techniques such as tSNE or PCA. With them, it is possible, for example, to have a notion of how the categories are grouped in a classification problem. They allow making detailed and exploratory analysis of the cases near a boundary, also about the movement of the adversarial examples over the manifold, and to better assimilate the relationship between different classes, and the variability inside each of them. The importance and potential of these visualization techniques can be seen in *Metric Learning for Adversarial Robustness* (Mao et al., 2019).

Chapter 4

Machine Learning Concepts and Related Branches

To create a high performance ML model, which is also reliable and robust to different scenarios, many considerations must be considered and guaranteed. As explained previously, AML is an interesting field that can be used to create safer models for adversarial attacks. Still, it is essential to contemplate it together with other ML assumptions, statistical phenomena, and other related fields.

To really take advantage of the AML potential to improve a model, it is necessary first to ensure that the base model is created correctly and that all the available improvements are considered. Using AML to improve a model with fundamental assumption flaws, data problems, or poor logic and strategy will make it easier for an attacker to fool it (even without using AML methods). Also, any defense implemented will not create significant improvement, and more traditional ML considerations would make better progress.

Some important ML considerations and assumptions are summarized, and other state-of-the-art modern research branches are presented, showing their purpose and approach and how they interact and share many ideas. These ideas are focused mainly on supervised ML problems.

4.1 Machine Learning Concepts

First, it is important to remember how a supervised Machine Learning model works. It is an algorithm trained to meet a task (for example, classification) using data as input. It is essential to consider that even having complex and advanced algorithms, the task and data must be coherent to make sense using ML and expect reliable results. The ML model will learn from the given data and maximize its performance given a metric to be optimized, such as cross-entropy. By doing so, it is expected to have a model that understands the patterns inside the data to perform the task with new unlabeled data correctly.

Considering the ML model is fed with data, it is essential to guarantee that the relevant variables are included, that the dataset contains enough variability to extract different patterns required to solve the task, and to have a fair distribution to avoid biases in the predictions. Generally, the number of examples is related to the variability needed by the model, but it is crucial to focus on the previous considerations, as there may be large datasets with low variability to represent the different possible scenarios, with biases in the objective variable that may reward the model for predicting better specific scenarios than others, without the most relevant variables not allowing the model to find patterns, etc.

Also, the data must be preprocessed correctly to facilitate the algorithm its use. Some considerations are transformations to the continuous variables, correct categorical variables grouping, missing values handling, etc. More specific preprocessing must be done with unstructured data like images, sound, or text. In these cases, better performance is allowed by designing a proper DNN that is able of creating good representations. On the other hand, more traditional statistical models such as linear regression are based on particular distribution and data assumptions that must be fulfilled to allow reasonable predictions and inference analysis.

The algorithm and training process must be chosen considering the weaknesses and strengths of each of the possibilities. In this process, there is a tradeoff between performance and explainability, being sometimes black-box models better for complex tasks but susceptible to unexplainable errors and difficult to control for logical improvements. Finally, the objective variable or trained task must be correctly defined considering the data and algorithm potential and limitations.

4.2 Related ML Research Branches

Some recent research branches with high potential to enhance ML models, besides AML, are Extreme Reliability (ER), Overinterpretation, and Explainable Artificial Intelligence (XAI). Each of them focuses on analyzing certain behaviors from the ML models, are motivated by different purposes, but in the background are very aligned in the ML aspects to be improved.

4.2.1 Extreme Reliability

Among the "Cambrian Explosion of Machine Learning Research Topics" presented by Ian Goodfellow in the 2019 International Conference of Learning Representations (ICLR), ER was considered one of the most important. Its purpose is to guarantee no errors for specific ML tasks with severe consequences, such as safety in autonomous vehicles, air traffic control, surgery robots, etc. Creating a model to meet this requirement is challenging, and its training process and validation require using all the available knowledge and techniques to add the necessary reliability. In this case, using inside-of-distribution test sets will not be enough, being necessary to evaluate the model with complex new cases if possible, but also to understand how it is making the predictions, which features it is using and detecting its weaknesses. Not understanding a model will blind us from improving complex cases, biases in the data, and the patterns created by the model, and we will fool ourselves by trusting in the traditional performance metrics.

In the same 2019 ICLR, Goodfellow explained how AML could be useful for ER, as there may be out-of-distribution difficult cases, but also adversarial examples affecting a model. Also, the AML attack methods may reveal the weaknesses in the model inference process, knowing that most of the created White-box attack methods optimize the perturbations required to fool the model and, in this way, will highlight the most relevant features and vulnerabilities.

4.2.2 Explainable Artificial Intelligence

Other research field, XAI, focuses on adding interpretability to ML models, to provide explanations on how it performs the task in understandable terms to a human. There are many proposed methods to approach this goal, depending on the type of algorithm and data used, and it is a very broad and creative field. *A Survey on Neural Network Interpretability* (Zhang et al., 2021) proposed an interesting taxonomy to classify the different XAI methods, considering three orthogonal dimensions: passive vs. active interpretation, type of produced explanation, and local vs. global interpretability.

A passive method modifies the model before the training process to promote explanation mechanisms, while an active method tries to explain an already trained model (most used in literature). The type of produced explanation, or how explicit it is, can be divided into four categories. One is the creation of logical rules which show clear patterns that the inference process follows, which may be helpful to have a general understanding of the model, but maybe oversimplifying it, being also challenging to be developed for complex DNNs. A second category is hidden semantics, something useful to give an idea of how the representation of a model is being created (for example showing the patterns maximizing the activation of a convolutional layer in a CNN). Attribution tries to point out more directly the most relevant features used while making a prediction, and finally, there are methods focusing on the creation or selection of examples to show the most representative cases for an output. The third dimension of Zhang's taxonomy differs between global explanations which are focused on explaining the ML model as a whole, which are almost impossible for DNNs, and local and semi-local explanations, which focus on explaining predictions on individual or groups of similar input examples.

Many XAI methods are based on AML methods to highlight small changes in an input example that will change the prediction from the ML model drastically, which will represent decisive features, as it is explained for example with the Contrastive Explanation Methods in *Explanations based on the Missing: Towards Contrastive Explanations with Pertinent Negatives* (Dhurandhar et al., 2018) estimating the minimal present and absent features. Dhurandhar in the same paper proposes including a convolutional autoencoder in the loss function to create local and by example explanations closer to true data (from the training data manifold) to create more human-understandable evidence for explanations. So it is evident how AML and XAI are very related to each other and how understanding how a ML operates can give feedback to reforce vulnerabilities. Also, it is clear that good representations in DNNs will ensure models work logically and reasonably while creating higher reliability to different scenarios as in ER and being robust to adversarial attacks.

4.2.3 Overinterpretation

Finally, another related topic is Overinterpretation (Carter et al., 2020), a problem related to how nonsense can make sense to ML models. This recently discussed problem consists of analyzing how some models such as DNNs may be based on patterns or features that do not make sense to us humans but that at a statistical level are totally valid to achieve their objective with good performance. In the case of images, overinterpretation occurs when a classifier finds strong class-evidence in regions of an image that contain no semantically salient features. This can be related to overfitting, but its problem is that it cannot be diagnosed via reduced test accuracy, as it may happen that these spurious and statistically valid patterns are still present in the test set.

The previous problem show again the risk of measuring a ML model just with performance metrics over a test set, which could even perform well with out-of-distribution cases in a test set, but that will not guarantee to make the process correctly and being robust to unseen before examples. This can happen because for example a CNN is free of choosing the features that work better to achieve a task, and may be finding statistical shortcuts instead of learning more complex semantic relationships between the image pixels and the assigned class label.

Overinterpretation, unlike adversarial examples, relies upon unmodified image pixels, but it is clear that a model with overinterpretation problems will clearly be easier to fool by an attacker, and that adversarial robustness alone will not prevent models from overinterpreting spurious signals. Also, it shares the fact that creating good representations will create more reliable, correctly explained and robust ML models. Actually, in the same paper, the Sufficient Input Subsets (SIS) algorithm is proposed to help humans interpret the decisions of black-box models, and it is reinforced the importance of carefully curating training data to eliminate overinterpretation artifacts, which can be seen as a AML poisoning attack. Finally model ensembles and clever regularization strategies are encouraged to favor salient features, being the last one very related with Adversarial Regularization introduced in chapter 3.

Chapter 5

ML and AML Used Methods

5.1 Machine Learning Techniques

5.1.1 Bagging and Boosting Tree Ensembles

As their name suggests, tree-ensemble methods are based on the combination of multiple simple supervised models (weak learners) such as decision and classification trees to achieve a much more powerful model. These methods have proven their strength by demonstrating very high performances compared to other ML techniques, and their ease of implementation makes them useful tools to achieve successful supervised models quickly.

In fact, these methods have shown outstanding performance in many ML competitions based on structured data, being very versatile due to certain properties they exhibit. By working with decision trees, these methods are more robust to outliers, as the outliers do not have large leverage on the parameter tuning as would happen for example, in linear regression. Additionally, these methods reduce the intensity of the bias-variance trade-off, as they are based on the combination of simple models that complement each other in one way or another.

There are different ways of grouping them, such as Bagging (Bootstrap aggregating) and Boosting techniques. Bagging consists of averaging the output of multiple models (or voting in classification problems) that work in parallel, in this case, trees with a random component in order to create a more reliable final output, with lower variance and high performance. An example of these models are Random Forests, which average trees, each created with a different random sample (bootstrapping) of data and features. This forces each tree to focus on different features to solve the problem and thus reduce risks due to biases in the data and to take advantage of the general average to give a more stable prediction.

On the other hand, there are Boosting techniques, which in tree-ensembles group the output of multiple single trees sequentially in an adaptive way so that each new tree focuses on improving the difficulties of the previous models. These methods thus reduce not only variance but also bias. An example of these models is Adaptive boosting or Gradient boosting methods, among which XGBoost (Chen and Guestrin, 2016) is very popular for its outstanding performance in ML competitions, in which it is analyzed after each tree in the sequence how correctly each example is predicted in order to adjust them according to their error in the next tree in the sequence.

5.1.2 Counterfactual Analysis

Counterfactual analysis is useful for providing local-level explanations of an example of a dataset associated with a ML classification model prediction. Seeking to understand the example $\mathbf{x_0}$ that was classified into class t by a classification model, we can look for its counterfactuals $\{\mathbf{x_{c1}}, \mathbf{x_{c2}}, \mathbf{x_{c3}},...\}$ which are nearby instances in the input space that the same model classifies into a different class than t.

The counterfactuals are used in different techniques of local explainability, since they allow us to perform a sensitivity analysis around the example $\mathbf{x_0}$, and thus to understand its susceptibility to punctual changes in the variables. In many cases, these counterfactuals are artificially created around $\mathbf{x_0}$ to make sampling and be able to apply Counterfactual Explanations for Machine Learning techniques (Verma et al., 2021).

This is even useful for causality analysis since it can be shown that with the change of a variable in $\mathbf{x_0}$, the output of the model would no longer be the same, attributing the current output of $\mathbf{x_0}$ to the absence of that variable. This technique is very important in Ethical Machine Learning issues to identify causality in features that should not be decisive in a decision, which serves as a basis to increase the fairness and transparency of an ML model (Kasirzadeh and Smart, 2021).

Although counterfactuals are a well-studied branch of research in state of the art, and there is a wide variety of methods maximizing their use, in this thesis it is limited to a simple application in case study 1. Among all its uses, one consists in analyzing real counterfactuals, analyzing the proximity of the examples in a dataset, and identifying those that are classified differently according to the ML model. This, as discussed in case study 1, allows guaranteeing in specific scenarios that the nearest counterfactual of the case under analysis x_{c2} is a real case, which is useful when the representation does not directly represent the raw data and it is not possible to guarantee to create viable examples by simulating examples.

5.1.3 Deep Classification Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a specific group of Deep Learning models characterized by their ability to solve Computer Vision problems, among other applications. In the specific case of image classification, they are models characterized by directly taking an image to analyze its patterns with spatial properties. To achieve this, architectures with multiple layers in series (and in certain architectures in parallel) are used, where each layer has the ability to recognize different patterns at a different scale, and based on the output of the previous layer.

These networks use operations such as convolutions, with which different spatial features can be extracted using different convolutions in the same layer and can be compressed with pooling layers to be used as input in the next layer. This allows considering spatial dependencies (in the case of an image) and progressively creating more complex representations, starting with very local and simple features such as edges and textures and then grouping them into larger patterns and objects.

CNNs like other DNNs have activation functions in each layer to model non-linear relationships and guarantee successful learning, then convolutional layers convert the output into a one-dimensional vector and use fully-connected layers, and in the case of classifiers can end up with a soft-max layer that creates a normalized score for each of the possible categories.

These neural networks are trained using techniques such as gradient descent by minimizing a loss function, which in the case of classification can be the cross-entropy loss. There are many variations in the training process using back-propagation to perform the gradient descent, in which the parameters are adjusted by epochs, starting by evaluating the initial errors and then adjusting them according to the gradient, being able to use techniques such as Stochastic Gradient Descent and other techniques with momentum to address them correctly.

This is usually done using a training data set, but in each epoch, analyzing its progress with a validation set not used by the model to know its generalization progress to avoid overfitting to the training data. All kinds of variations can be tested, being important the logic of each hyperparameter but also its empirical knowledge developed in state of the art.

There are a wide variety of architectures that tend to become deeper and deeper and to use new techniques to create certain layers with parallel convolutions, skipping connections, etc. Considering the high computational level required to train a CNN, it is possible to parallelize the processes with GPUs or TPUs, for which it is usually necessary to use mini-batches for training, considering the capacity limitations of each one.

5.1.4 Embeddings and Visualization Techniques

Some models as DNNs can be viewed as models that are directly responsible for creating a data representation that facilitates predictions in a supervised model. In the case of a classification CNN, the neural network can be conceptualized as creating these relevant features with the convolutional layers allowing easy discrimination between categories in the following layers.

Analyzing these representations or embeddings is useful to get an idea of whether they really allow discriminating the classes and to be able to analyze in detail the cases that tend to group together the most, the ones with the highest intra-class variability and their closeness to other classes. To do this visually with multidimensional embeddings, techniques can be used to optimally summarize the distribution of the embeddings in a reduced space. Among these techniques, two are used in the analyses of this thesis, which are Principal Component Analysis (PCA) and t-distributed Stochastic Neighbourhood Embedding (t-SNE).

PCA is an unsupervised technique useful to reduce dimensionality by creating a new space to represent the data, creating the new dimensions so that they have orthogonal directions in decreasing order of explained data variability. This allows a smaller number of directions, or principal components, to summarize a large percentage of the data variability and select the first few to visualize the distribution of the data. Another useful method for this type of visualization is t-SNE, which does not transform the space of variables, but directly reduces the dimensionality on the space of dimensions considered, seeking to maintain the closeness of the data with a local approach.

Using both techniques allows to analyze with different approaches the embeddings in a high dimensionality space, and in the case of PCA many additional analyses can be performed, considering that each of the principal components can be interpreted to understand the underlying variables of the problem, in particular in cases with highly correlated variables. Seeing which features compose each principal component also allows to know which ones are related to each other, what kind of correlation they have, or if on the other hand, they tend to be independent when presented in different principal components (due to their orthogonality).

5.2 Logo Detection and Classification

Logo classification or recognition aims to create an algorithm that can automatically classify an image with a logo with its respective brand. This problem in the literature was approached for many years with traditional Computer Vision techniques, in which there was first a process of creating features in images (or feature descriptors), and then an algorithm that used them to statistically analyze their similarities, either with histograms, bag-of-words approach with handcrafted features, etc. In the era of Deep Learning, Convolutional Neural Networks have been characterized by their facility to address these problems more efficiently and in a more direct way. An interesting paper analyzing the advantages of both approaches is presented in *Deep Learning vs. Traditional Computer Vision* (Mahony et al., 2019). For this reason, most of the proposals to perform state-of-the-art logo classification are based on DNNs, particularly on CNNs, which have the advantage of automatically doing both the image preprocessing to create its representation in a vector and use it to perform the classification process.

It is important to mention that object detection techniques can be used to find multiple logos in an image. Among these, some algorithms directly perform the detection and classification. There are algorithms such as YOLO (Redmon et al., 2015) or SSD (Liu et al., 2015), which for training take as input features that represent the location and dimensions of the bounding box of an object in the image and its class. These models classify multiple anchor-boxes of different proportions and sizes on a grid of points in the image to choose those with the highest confidence, eliminate duplicate cases for the same object, and finally give as output confidence of having found an object, the confidence that it is of the predicted class and the coordinates characterizing its bounding-box.

However, to be more conservative and avoid not detecting logos in an image, techniques that propose a large number of regions of interest and crop the image can be used. Then each of the crops can be analyzed to see if they have a logo of interest or not. This case will be the one assumed in case study 2.

In fact, in *Deep Learning for Logo Recognition* (Bianco et al., 2017), it is proposed to **separate the process into two stages, first doing detection without the need to be very accurate, with a high recall to propose regions, and then a CNN to do classification.** As the detector is recall-oriented the classification model should consider there would be many false positives, which are crops without any logo. For this reason, it is proposed adding crops without logos to **represent explicitly the background as another class in the classification CNN**. Also, class balancing is important to avoid biasing the model to some classes.

Now, it is important to consider the low variability of the logos, compared to other objects in classification models. Even if large volumes of data are used, if the logos are very similar, the CNN will learn to do overfitting and probably will not be able to learn good representations, starting with simple features in the first layers and then more complex representations in deeper layers. Therefore, **it is also proposed to use data augmentation to increase the data set synthetically and add variations that allow it to understand the logo's relevant features.**

Additionally, to compensate for the difficulty of creating adequate representations, **transfer learning of a pre-trained model can be applied** in such a way that it uses the same transformations learned with problems in which there was more significant variability. For example, a CNN trained with Image-Net can be used and re-trained. One option may be to freeze a certain number of initial convolutional layers and re-train the network with only the last layers to adapt it to the logo detection problem. Another option is to do fine-tuning to re-train the whole CNN but starting from the pre-trained parameters, hoping that it will modify only the necessary details to adapt to the logo classification problem.

This thesis uses these considerations to create the enhanced model in case study 2.

5.3 Min-max Adversarial Robustness Paradigm

Complementing section 5, one of the approaches presented in state of the art is described to explain mathematically a possible way of focusing a DNN to be trained in order to be robust against adversarial attacks and not only maximizing its performance with in-distribution cases.

As explained in *Towards Deep Learning Models Resistant to Adversarial Attacks* (Madry et al., 2018), a standard classification task with underlying distribution D, trained with pairs of examples $x \in \mathbb{R}^p$ and corresponding labels $y \in [k]$, using a loss function $L(\theta, x, y)$ such as cross-entropy, where $\theta \in \mathbb{R}^p$ is the set of model parameters, and the goal is to the model parameters θ to minimize the risk $\mathbb{E}_{(x,y)\sim D}[L(x, y, \theta)]$.

In the previous definition of the Empirical Risk Minimization (ERM), it can be seen how it emphasizes the domain over which risk is minimized, which is the underlying distribution of the training data. This is generally useful to create successful classifiers over the data manifold but it is not considering a way to increase out-of-distribution robustness, and in this way adversarial robustness too.

Hence, one way to minimize the risk associated with having incorrect predictions with small perturbations can be approximated as seen in Equation 5.1.

$$min_\theta \ \rho(\theta), \ where \ \rho(\theta) = \mathbb{E}_{(x,y)\sim D}\left[max_{\delta \in S} L(x + \delta, y, \theta)\right] \tag{5.1}$$

Where S is the set of allowed perturbations, and $x + \delta$ is the example with the added perturbation. This formulation unifies an inner maximization problem with an outer minimization problem in a saddle point. Here the inner maximization problem aims to find an adversarial version of a given data point x that achieves high loss, being used as example to train the model for difficult scenarios around x. This inner maximization can be seen directly as the attack created to the DNN, which is the main focus of the evasion attacks in AML. The outer minimization problem will take this optimized attack and help the model find the best parameters to reduce its contribution in the loss function. A small adversarial loss function for a classifier would directly represent its robustness to the attacks created in the inner maximization problem attack.

This representation provides the basis for creating attacks by focusing on the inner maximization problem and serves as a basis for creating defenses such as Adversarial Training. This formulation is adapted and extended to enhance the adversarial robustness of a model.

5.4 Adversarial Attacks

Complementing section 3.2.2.2. on evasion attacks by going into a little more detail on its mathematical definition, focusing on the inner maximization problem of Equation 5.1, explained previously, by seeking to create optimized attacks within a subset of possible perturbations to affect its predictions.

Each method used in the case studies is explained, highlighting its particularities. Therefore, the White-box Fast Gradient Sign Method (FGSM), Projected Gradient Descent (PGD), Carlini & Wagner attacks are explained. Subsequently, the ZOO (Zeroth Order Optimization) attack is explained as a Black-box adaptation based on Carlini & Wagner. Finally, the White-box Efficient Adversarial Attack for Tree Ensembles method is explained. Each of these methods was designed for supervised classification models.

5.4.1 Fast Gradient Sign Method (FGSM)

Fast Gradient Sign Method (FGSM) was one of the first methods created to demonstrate the adversarial vulnerability of DNNs and in particular CNNs, presented in *Explaining and Harnessing Adversarial Examples* (Goodfellow, Shlens, and Szegedy, 2015). Following the definitions from section 5.3, this method creates a l_∞-bound attack, meaning that the adversarial example cannot exceed this norm from the original example. It computes the adversarial examples as shown in Equation 5.2.

$$x + \varepsilon * sgn(\nabla_x L(\theta, x, y)) \tag{5.2}$$

This attack intends to maximize the inner part of the saddle point formulation in Equation 5.1, by creating a one-step attack in direction of the gradient of the loss function, being the strength of the pass determined by the scalar hyperparameter ϵ.

5.4.2 Projected Gradient Descent (PGD)

Projected Gradient Descent (PGD) can be seen as an extension of the FGSM attack method (Goodfellow, Shlens, and Szegedy, 2015). PGD is the multi-step variant which makes smaller steps in the direction of the loss function, being capable of following more smoothly, and being l_∞-bounded too. The iterative process optimization search can be expressed as in Equation 5.3.

$$x^{t+1} = \Pi_{x+S}(x^t + \alpha * sgn(\nabla_x L(\theta, x, y))) \tag{5.3}$$

It is important to mention that this method is usually the most used in papers to test defenses as explained earlier in the AML metrics, and also to train defenses as in Adversarial Training.

5.4.3 Carlini & Wagner Attack

Carlini & Wagner, presented in *Towards Evaluating the Robustness of Neural Networks* (Carlini and Wagner, 2017) is considered one of the strongest White-box adversarial attack methods (Chen et al., 2017). It optimizes the way of creating a successful adversarial example for classification DNNs, by taking a base example $\mathbf{x_0}$ and creating an adversarial example $\mathbf{x}$ to change its prediction to the class label t.

This is performed following the optimization problem in Formula 5.4. The first term enforces creating an adversarial example similar to the original example by minimizing the Euclidean distance between the different of both examples. The second term is a loss function showing how unsuccessful was the adversarial attack trying to make the model classify the adversarial example into target class t, multiplied by a scalar factor c to weigh the importance of the first and second term from the minimization equation. This second term in the Carlini & Wagner attack for targeted attacks is suggested to be estimated as in Formula 5.5.

$$minimize_{\mathbf{x}} \left\|\mathbf{x} - \mathbf{x}_0\right\|_2 + c \cdot f(\mathbf{x}, t) \quad subject\ to\ \mathbf{x} \in [0,1]^p \tag{5.4}$$

$$f(\mathbf{x}, t) = max\left\{max_{i \neq t}\left[Z(\mathbf{x})\right]_i - \left[Z(\mathbf{x})\right]_t), -\kappa\right\} \tag{5.5}$$

Z($\mathbf{x}$) represents the logits in the representation layer for the example x, and $[Z(\mathbf{x})]_k$ represents the predicted probability of $\mathbf{x}$ belonging to class k. This means that the subtraction tries to create an x with a higher predicted probability in class t than in any other class i, selecting the class i achieving a maximum value. Achieving an attack will make the subtraction output a negative number. κ is a tuning parameter limiting how strong the attack should be or how clear should it be that the adversarial example will change the prediction class, being interpreted as a minimal gap between the prediction for class t and the second higher prediction class. κ is generally set to be 0, meaning that the method is just interested in achieving changing the target class with the minimal probability required, but a larger κ is suggested to create stronger and more transferable attacks.

Finally, the box constraining $\mathbf{x}$ to have variables with values between 0 and 1 is added considering that this method was created for attacks over images, which are assumed to have normalized pixel values between 0 and 1.

In the paper, the author suggests using this attack as a better baseline for evaluating candidate defenses, considering it is a stronger attack that defense distillation defenses can not resist. They also demonstrate that adversarial examples using this method are transferable from the unsecured model to the defensively distilled model, recommending checking the transferability of the attacks to measure a defense success.

5.4.4 ZOO: Zeroth Order Optimization based Black-box Attacks to Deep Neural Networks without Training Substitute Models

Zeroth Order Optimization, or ZOO (Chen et al., 2017) is a Black-box method that assumes only having access to the input and output of a targeted model. It was designed mainly for DNNs, and instead of trying to create a substitute model, it directly estimates the gradients of the targeted model for generating adversarial examples. This method is one of the stronger Black-box attack methods in the state-of-the-art, outperforming existing other Black-box methods based on substitute models, having comparable efficiency in the creation of adversarial examples as strong White-box methods as Carlini and Wagner attack (Carlini and Wagner, 2017).

This method uses zeroth order stochastic coordinate descent, dimension reduction, hierarchical attack, and importance sampling techniques together to create the attacks. Zeroth order means that the method is derivative-free and can achieve the optimization process just by using the input **x** and its function value *f(x)*, by approximating the gradient evaluating two close points, *f(x + hv)* and *f(x - hv)* where h is a small scalar and **v** a properly estimated gradient vector. This method's convergence to a stationary point has been proved, arriving at a local optimum.

This method takes the Carlini & Wagner attack (Carlini and Wagner, 2017) and modifies it to a Black-box scenario by modifying Formula 5.4 and Formula 5.5. First, the loss function to represent how unsuccessful was the adversarial attack is modified to depend only on the output distribution, using the softmax results given by the model to check that the confidence score from one class significantly dominates the confidence scores of other classes, being Formula 5.5 replaced by Formula 5.6. In this equation F represents the output from the model, and the log operator is added to reduce the dominance effect while preserving the order of the confidence scores given its monotonicity property.

$$f(\mathbf{x},t) = max\left\{max_{i\neq t}log\left[F(\mathbf{x})\right]_i - log\left[F(\mathbf{x})\right]_t), -\kappa\right\} \tag{5.6}$$

Also, to approach the optimization problem from Formula 5.4, not being able to use back propagation, the gradient is approximated using stochastic coordinate descent which is explained in detail in the paper, even showing a variation of Zeroth Order Stochastic Coordinate Descent with Coordinate-wise ADAM, which was much faster than Newton's method.

The dimension reduction, hierarchical attacks, and importance sampling are used to speed up the computational time by reducing the number of queries needed. The dimension reduction is made to represent **x** in an m space rather than the original p space, helping the method estimate less number of gradients to propose a new **x**. For images, it is easy as the image can be changed from a 299x299x3 tensor to a compressed 32x32x3 image, but this may not be coherent with tabular data in other contexts. Also, discrete cosine transformation is tested.

The hierarchical attack consists of applying dimension reduction but progressively increasing m while the algorithm converges to improve the search space according to the detail needed, starting with larger changes with a small m space, and making more detailed modifications by increasing m, being m always equal or less than p. Finally, in this method is possible to choose which coordinates to update first, which is why importance sampling is used to estimate the importance from pixel regions and start with the most important ones first.

5.4.5 An Efficient Adversarial Attack for Tree Ensembles

An Efficient Adversarial Attack for Tree Ensembles (Zhang, Zhang, and Hsieh, 2020), as its name implies was created specifically for Tree-Ensemble models such as gradient boosting decision trees and random forests, and focused on classification problems.

Its importance lies in the fact that it does not make sense to use gradients in these models, which are essential in most attacks designed for DNNs. This means that the methods for DNNs do not adapt efficiently to tree-ensemble models, have problems in their estimation, and do not share the same assumptions to optimize the creation of adversarial attacks. This phenomenon is experimentally verified in case study 1 of this paper.

This White-box attack transforms the attack problem into a discrete search problem specially designed for tree-ensembles. Its goal is to find a valid "leaf tuple" that leads to misclassification while having the shortest distance to the original input. It is performed using a greedy algorithm to iteratively optimize the adversarial example by moving the leaf tuple to its neighborhood in the input space, constraining the movement to a Hamming distance 1 (maximum feature change of 1). This approach shows an efficient way to create faster adversarial examples compared to Mixed-Integer Linear Programming approaches.

Briefly summarized, this method takes an example x0 and seeks to create an adversarial example x. It initially selects an initial adversarial example x′ that the model classifies with a different target class to meet its objective, but that still doesn't minimize the perturbations. Subsequently, x′ is iteratively modified, approaching in similarity to x0, taking small steps in the input space, and keeping the model classified in the wrong target class. Considering that tree-ensembles are based on discrete decisions, one can mark decision boundaries on the continuous input space and transform them into discrete leaf tuples on which to iterate efficiently through the vertices of the decision boundaries. This is demonstrated in detail in the paper. An example to understand the logic of the algorithm is presented below in Figure 5.1, using an original representation of the paper (Zhang, Zhang, and Hsieh, 2020).

5.4.6 Adversarial Patch

Adversarial Patch (Brown et al., 2017) is an attack method that consists of predefining a batch of a specific size and shape to be superimposed on the image to maximize its adversarial impact. In the case of evasion, it seeks to determine the ideal location to make the model misclassify an image.

Although this technique can optimize the creation of the patch design to be an ideally universal perturbation, it can also be tested with any type of patch of interest. One goal of this method is to be able to create an attack that is replicable on physical objects so that the patch can be printed and placed at a given location (e.g. a stop sign). But another use of the adversarial patch proposed in this thesis is to use it to visually analyze possible relevant areas in the explanation of an image.

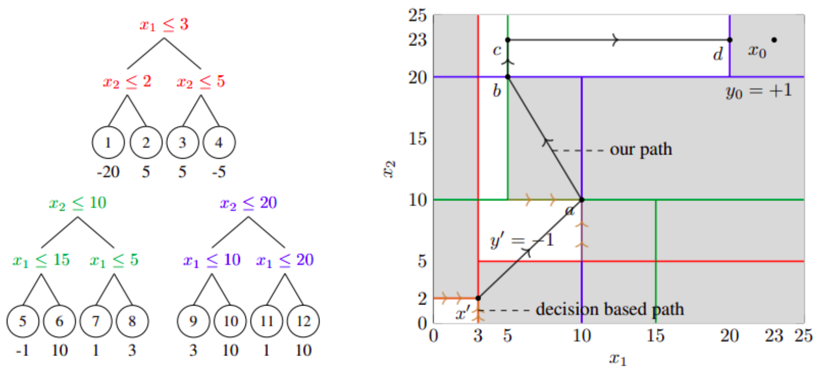


Figure 1: An ensemble defined on $[0, 25]^2$ and its corresponding decision boundaries on the input space. Numbers inside circles are indexes of leaves, and the number below each leaf is the corresponding prediction label v^i. For clarity we mark boundaries belong to tree 1, 2, 3 with red, green, and blue respectively, and fill $+1$ area with gray. x_0 is the victim example and x' is an initial adversarial example. Assume we are optimizing ℓ_1 perturbation, our method can reach d by changing one leaf of the tuple at a time (black arrows). On the other hand, decision based attacks update the solution along the decision boundary, and easily fall into local minimums such as x' and a (brown arrows) since they look only at the continuous neighborhood. To move from a to b, the path on decision boundary is $a \rightarrow (5, 10) \rightarrow b$, but since $a \rightarrow (5, 10)$ increases the distortion they won't find this path.

FIGURE 5.1: Example from *An Efficient Adversarial Attack for Tree Ensembles* (Zhang, Zhang, and Hsieh, 2020)

5.4.7 Spatial Transformation Attack

Spatial Transformation Attack method (Engstrom et al., 2017) was created to measure the **spatial robustness** of a model by limiting its attack purely to spatial transformations of the image. This method iterates over different values (using Grid-search) to perform an optimal image rotation and translation to make the classification model misclassify the image.

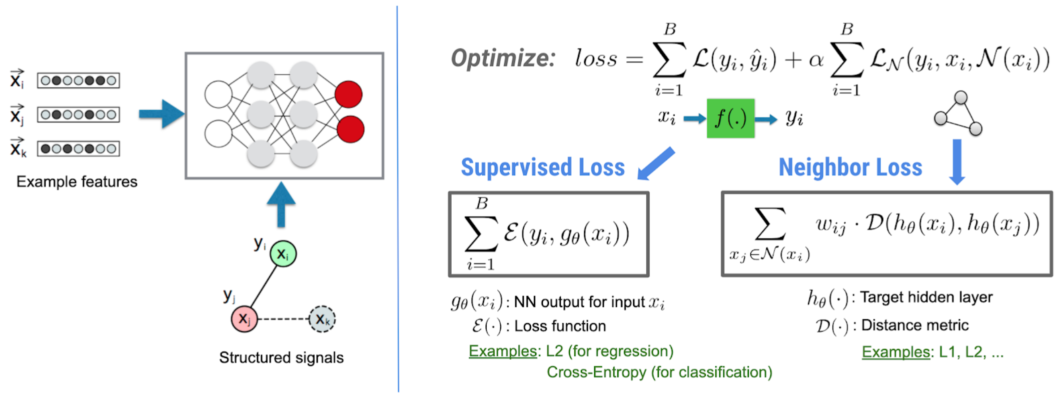
This method is used in the second case study to visually analyze the model's susceptibility to image rotations. This can be used to intuit about the features that the model uses to represent each logo, if it gives importance to its shape and proportions, or if it does not seem to use this type of properties.

5.5 Adversarial Defenses

5.5.1 Adversarial Training using Adversarial Regularization

Adversarial Regularization is another approach to the min-max problem explained in section 5.3, and in particular in Equation 5.1. Although there are different ways to perform adversarial regularization, here it is explained the approach used in case study 2, which is the TensorFlow approach proposed in the Neural Structured Learning library, presented formally in the paper *Neural Structured Learning: Training Neural Networks with Structured Signals* (Gopalan et al., 2021), and explained in greater detail in the TensorFlow Blog *Introducing Neural Structured Learning in TensorFlow* (Ravi, n.d.).

Figure 5.2 summarizes the method used, which modifies the loss function to train the DNN. This method seeks to create a similar representation for the original example and its adversarial example created with PGD, minimizing the distance between them. This is done so



$\vec{x}_i$
$\vec{x}_j$
$\vec{x}_k$
Example features
y_i
x_i
y_j
x_j
x_k
Structured signals
Optimize: $loss = \sum_{i=1}^{B} \mathcal{L}(y_i, \hat{y}_i) + \alpha \sum_{i=1}^{B} \mathcal{L}_{\mathcal{N}}(y_i, x_i, \mathcal{N}(x_i))$
$x_i \rightarrow f(.) \rightarrow y_i$
Supervised Loss
Neighbor Loss
$\sum_{i=1}^{B} \mathcal{E}(y_i, g_\theta(x_i))$
$\sum_{x_j \in \mathcal{N}(x_i)} w_{ij} \cdot \mathcal{D}(h_\theta(x_i), h_\theta(x_j))$
$g_\theta(x_i)$: NN output for input x_i
$\mathcal{E}(\cdot)$: Loss function
Examples: L2 (for regression)
Cross-Entropy (for classification)
$h_\theta(\cdot)$: Target hidden layer
$\mathcal{D}(\cdot)$: Distance metric
Examples: L1, L2, ...

FIGURE 5.2: Neural Structured Learning Adversarial Regularization, taken from *Introducing Neural Structured Learning in TensorFlow* (Ravi, n.d.)

that the representation can be oriented to more important features that do not differ by perturbing only certain features (or pixels in images). This is done by adding to the loss function a term minimizing the distance in the representation of the example and its adversarial example with PGD, which is added to the traditional loss function used for performance with a factor to define its importance and balance between both purposes. By default, the L_2 norm is used with a multiplicative factor of 0.2, and for classification purposes the first term is the Cross-entropy loss.

Chapter 6

AML Diagnosis Methodology

Making a correct diagnosis of the model is very important before applying any AML defense to increase its robustness. Applying a defense or any complex method prematurely, without first assessing failures in other more fundamental dimensions of the ML model, will result in no significant improvements and misdirected efforts. It is also important to spend some time creatively and openly analyzing ways to solve the problem at hand, before being tempted to apply complex state-of-the-art models quickly, and blindly automate the data collection and model creation processes. In the case of having the urgency to create an initial model to mitigate a problem, it is recommended to document the aspects that need to be considered for improvement and continually re-evaluate the models used in production.

This chapter presents a methodology to analyze the construction of a ML model qualitatively to evaluate each of its components and create a general diagnosis of it. At the same time, it is used to examine possible improvements considering their potential impact on the model's performance, their business impact, and the effort required for their implementation. It should focus its documentation on considering each of the risks and vulnerabilities related to AML, ER, and Overinterpretation. This methodology should be considered together with good practices and considerations from the Cross Industry Standard Process for Data Mining (CRISP-DM), seen as an additional dimension to be considered, being useful for the model's refinement.

The analysis was carried out holistically, starting with a business vision to understand the motivation for implementing each model in the company's processes, the specific purpose to be solved and its scope, and the potential information that may exist in the data and the techniques necessary to carry it out. Considering the models are in production, ideas of greater potential impact are proposed and separated by the efforts required, especially in terms of time, in order to start with quick improvements to current models, but at the same time to develop more elaborated ideas that can represent leaps in progress in the medium and long term.

Each component was analyzed considering its impact on reliability and robustness over cases from the same distribution, out-of-distribution possible cases (considering Overinterpretation, explained in chapter 4), and adversarial robustness. The adversarial robustness of the model is evaluated algorithmically but also technically to analyze how easy could it be for an attacker to achieve an adversarial attack, and the main motivation to do it, to understand which type of attacks to study further. Finally, this process is considered valuable to document the technical aspects of each ML model and the considerations evaluated so that in the future, a new member can more easily understand how to contribute with new ideas and confidently take ownership of it.

As a result of the analysis, two matrices are proposed. The first matrix will document the business motivation and purpose, together with technical details for the development of the model. The second matrix will analyze qualitatively different relevant factors to get a notion about the present risks and possible improvements.

6.1 General ML Components Analysis

The purpose of the first matrix or report is to properly document each of the components and steps followed in the creation of the ML model. Starting from understanding the context, business purpose, tasks intended to be solved, utility and use of its output, and assumptions for its creation. Considering its purpose is to improve already conceived models, it is important to focus the analysis on the dimensions to be improved: out-of-distribution robustness and adversarial robustness.

The first task is to understand the current model and its motivation always having as a north its business purpose but also considering how each step is affected in reliability and adversarial robustness, starting with general information as:

- Model's name: Making sure it is unique, clear, and explicit.
- Purpose inside the company processes: Considering the related areas, processes, when would it be operating, amount of requests needed and how the output would be used. Also documenting the associated risks inside the process considering new unseen cases and adversarial vulnerabilities.
- Task to be solved: Regarding the exact output and its interpretation
- Motivation and assumptions: Considering the cyber security incident to solve, the intentions of the attacker or expected scenarios to be detected, and the information to be extracted directly or indirectly from the data.

Then the documentation of the main components for the development of the model is done considering:

1. Data

- Format: Being possible having tables, strings, JSON logs, images, time series, etc.
- Content: Considering the information collected with the data. It is essential to evaluate how examples are being selected, if the data is collected following a standard process with restrictions and business considerations. Also, which cases are not being involved in the process, which processes affect the way it is created, and relevant cases that are not included or are not reachable.
- Criteria to choose training examples: How is the data filtered correctly to avoid adding noise, avoid biases, consider all relevant cases and variations, and contemplate the creation of simulated data or data augmentation. Also given the purpose of the model how many cases are collected, if it is historical data which is the time range used, etc.

- Amount of examples used and its distribution regarding the objective variable: For classification models check the number of examples for each class to know if it is balanced, analyze the intra-class variability, and record possible biases.
- Definition of the objective variable: Consider the best way to give the output to take advantage of it, but also to improve the statistical and ML properties. In a classification model analyze how to define each class, how to group classes if necessary according to their similarity, etc.
- Deployment data particularities: Compare similarities and dissimilarities between training data and new deployment data, possible new variations or difficulties, and analyze how the model will handle them.

2. Features

- Feature creation and origin: When managing structured data it should be considered how the features were extracted from its source, to consider if it done properly to represent the most relevant characteristics. Also considering its source is relevant to consider the type of data an attacker can modify and how it is propagated in the feature engineering process.
- Feature representation: Document how the feature engineering was done, considering the transformations made to numerical and categorical variables. Consider how this representation promotes better model creation and how easy would it be for an attacker to modify it abruptly affecting the raw data source.
- Features distribution: It is important to check if an analysis of tthe distribution of the features was done before the model's creation to detect possible biases and potential improvements. Also, if any distribution analysis was made over the predictions considering the target and input variables to check their influence.

3. Model

- Statistical purpose: It is important to consider which is the task that the model should perform, now focusing on its statistical potential to be extracted using the given data. This step is important to evaluate if other types of techniques can solve the same problem or if it makes sense to dissolve the model into several complementary models that work in series or in parallel. This in such a way that each one concentrates on solving specific tasks by exploiting the potential of the data and the algorithms to the maximum.
- ML algorithm: The algorithm used must be documented, detailing the reason for its choice, if possible the advantages it has over other techniques, and the reason for the selection of its architecture, hyperparameters, and particularities considered.

4. Model Evaluation:

- Measured capabilities: The type of techniques used to measure the model's performance should be documented. These include how the performance metrics are evaluated, the creation of the test set, which metrics or statistics were prioritized in its development, and whether fuller exploratory analyses of its performance in production were done. It should also be analyzed if readjustments have been made to improve these metrics, if traditional cases of greater difficulty have been found for the model, and if it has been tested with unusual cases.

6.2 Extreme Reliability and Adversarial Robustness Diagnosis

Subsequently, an additional matrix is prepared, which focuses on a more detailed analysis of the risks to which the model is exposed in terms of reliability, adversarial attacks, and the consequences of failure. The matrix is created having over the rows each of the possible risks or dimensions to evaluate and in the columns the relevant aspects to consider. Over the columns, it is expected to document a description of the possible associated vulnerabilities and the expected probability of leading with them, the possible cause for having those scenarios, the business impact, and possible solutions. Also, it is recommended to add an additional column with additional doubts and questions to be further considered or extended.

Each of the dimensions should be critically analyzed considering all the previously mentioned concepts about ML fundamentals, AML, ER, and Overinterpretation. The dimensions analyzed are:

1. Standard Generalization

Starting with the model's standard robustness to in-distribution cases by analyzing the probability and repercussion of each of the possible errors. For example, in a binary classification algorithm used for detection, as in many cases in cyber security, the false positives and false negatives should be described.

Depending on the scenario, it may be easier to collect false negative or false positive cases. If positive cases are discarded automatically, it would not be possible to collect false positives, but false negatives would be identified when doing further analysis in the not discarded cases. But if the logic is contrary to the previous scenario, and an analyst analyzes only positive cases, then false positives would be identified, and false negatives could have overpassed the model without being detected. If the model is trained or can be evaluated only with a certain part of the confusion matrix, there is a risk of not being able to measure the real performance of the model, and that the training set is not incorporating relevant cases to be corrected.

In this case, it is also necessary to analyze what purpose an attacker would seek to achieve to perpetuate an attack, probably the most common being to evade the model and pass undetected. In addition, the cost of each error must be evaluated, which in the case of a detector, for example, will be to be exposed to an attack if it is a false negative, and to create unnecessary alerts and extra work for analysts if it is a false positive.

2. Out-of-distribution Robustness

It is necessary to analyze what possible variations not seen before by the model could occur in the use of the model. Among them, one can think of real cases that could not have been used in the model's training or that may not have had the proper importance. For example, in the case of a logo detector, it may be known that the training set does not contain certain variations of logos that could be encountered in practice.

On the other hand, it should also be considered that there may be cases never seen before by the model but also not known at the business level. Therefore, it is necessary to think about what variations an example could have that the model may not be able to handle. Continuing with the example of a logo classifier, one could consider variations to new colors, font changes, and adaptations of logos for special events. At a qualitative level, it should be analyzed what kind of features to reinforce in the model to consider these possible scenarios, and in the case of complex variations, their risk of occurrence should be documented.

3. Adversarial attacks

The primary motivation for an attacker to affect the model should be evaluated to focus improvement efforts, but at the same time, document the level of risk of the other types of attacks. In this process, the type of model used should also be analyzed to consider how susceptible it is to each kind of attack. For example, it is known that a CNN that works with images will be very vulnerable to evasion attacks due to the weaknesses of this type of black-box architectures, and in the case of a model trained with phishing emails from the past, poisoning attacks may occur as explained in section 3.2.2.1.

4. White-box and black-box scale analysis

As mentioned in section 3.2.1, the attacks created will depend on the attacker's knowledge of the model and its development. The possibility that each component may be known or intuited must be analyzed. In this process, the attacker may know about the training data used, know how to collect it, for example, if it is based on public sources, or even deduce the kind of examples that it has to create a substitute model with them.

It should also be analyzed whether the attacker can know the type of features used in the model, without needing to share the same exact definition. In this process, it is very important to consider which features are the most important in the model and which of these can be easily modified by the attacker, known as **leverage features**. Finally, it is necessary to analyze how likely the attacker can know the architecture or the type of model used, either because he uses a common algorithm for the task, a popular tool, or because he can deduce it by making requests. To this must be added the possibility of knowing the model's parameters, which could happen, for example, if a standard tool offered in the cloud is used, or in the case of using a pre-trained model, or simply being accessible in some other way.

6.3 Extreme Reliability and Adversarial Robustness Improvements

Once the analysis of the previous two matrices has been carried out, improvements can be proposed to focus on the areas of greatest weakness and parts with the greatest potential for improvement. It is important to consider which improvements can be made easily to enhance the current model, but also which changes that require more effort can be considered in the long term.

Data issues should be the priority to try to have correct, quality, unbiased information and to consider as much as possible the greatest variability of cases of interest. Subsequently, the handling of variables, their creation and the analysis of problems such as the curse of dimensionality, which can open doors so that there are regions of the input outside the manifold of the data used, which an attacker can explore, must be analyzed. If it is possible to directly create variables aligned with the business logic, it will be useful to validate that they are used directly and that there are no spurious correlations that the model uses, causing Overinterpretation.

Other techniques can also be considered to complement the model, using for example model ensembles, operations to select a smaller number of variables, using unsupervised models to detect anomalies, for example or with other logic to solve the same problem to use it in parallel. Even very simple improvements that can protect the model can be analyzed, such as tracking and limiting the number of times someone can make requests, validating if it is a case not seen before that must be checked before, that the format and range of valid values for each variable is complied with, etc.

After analyzing the traditional improvements, AML techniques can be used to measure the adversarial robustness of the model, and propose improvements or defenses according to the result. To carry out the chosen attacks, in addition to selecting the attack methods to be used, one must also analyze the assumptions of each one and take the role of the attacker. It will be necessary to measure how to deceive the model, see if it is possible and what perturbation or changes are necessary to achieve it, obtain metrics if possible and analyze the critical cases. With the insights obtained, the analysis process can be repeated, and efforts can be redirected, taking advantage of the knowledge created by simulating the attacker's case.

Finally, if the opportunities to improve the model in a traditional way are exhausted, AML defenses can be applied according to the context, for which the technique should be selected according to the potential improvement, and if possible, several alternatives should be tested. In this process, it will be important to correctly measure the adversarial robustness metrics to know the impact on each of the aspects of the defense or defenses applied. An example of how to do this is shown in case study 2.

Chapter 7

Case Study 1

A model with evasion attacks risks was chosen as the first case study to make a further diagnosis following the methodology followed in chapter 6. This case study begins by explaining the reason for choosing this model, some relevant considerations for its creation, and some context to understand how to approach the practical experiments. Subsequently, pretending to be an attacker, experiments intending to evade the ML model are made, gradually increasing their complexity. In each of the experiments, the assumptions, the technique used, the results obtained, and the associated conclusions are explained.

7.1 Model's Context

A binary classification supervised model is selected, which has the purpose of detecting email phishing attacks done to employees. This model is performed after a sequence of cybersecurity filters, made to analyze remaining complex cases that previously had to be reviewed manually by analysts. This model is intended to help the analysts as a previous filter to reduce the number of suspicious emails analyzed manually.

The entire email metadata is recorded, and different features are created based on it, with more than 100 initial features considered. The detail from this features can't be revealed in this document for security purposes explained in section 1.4. With this type of data the examples can be organized as structured data, making it possible to test traditional supervised ML algorithms to achieve the model's goal. This study focuses on the tree-ensemble models used, which were selected due to their reduced variance and bias properties, their ability to handle outliers, their ability to handle better unrelevant features, the possibility of estimating the feature importance for indirect explainability purposes, and for their good performance.

The model's refinement was done by selecting the most important features from an initial model, in order to achieve better performance and discard those not adding relevant information. The final feature set was composed of 50 variables, composed by continuous, binary, categorical, and ordinal variables. Each of the variables was transformed to enhance their statistical properties, creating a representation that suits the input restrictions of the model, using for example one-hot encoding for categorical variables, a sequence of integers for ordinal variables, etc.

Finally, the model's training process was performed following an iterative process with different hyperparameters using cross-validation, using a hold-out methodology to test its generalization capabilities and avoid overfitting. The performance was measured with the test set, not considering the output as a discrete value (0 or 1) but using the score in order to select a conservative threshold for the problem. The decision threshold is selected with a value lower than 0.5, according to the distribution of the false-negative cases, which are considered the riskier errors to have in the model, being phishing cases not classified as such.

Considering that the model was created using a dataset constructed from emails, there may be potential for an adversary to look for unanticipated variations that could evade the model. Thus, the purpose of this analysis is to explore the relevance of the different variables in the model output, in order to detect leverage variables that can significantly affect the model prediction. The results obtained can be used to prevent the model from learning patterns that are easy to exploit by an adversary. Or in the case of not finding easy to modify patterns, the analysis will be useful to validate that the model has certain robustness to adversary attacks, being a complementary measure to select among candidate models.

7.2 Attack Approaches

Six types of approximations are experimented to measure the model's robustness, starting with simple ideas and progressively performing more complex attacks and analysis. The approaches are linked, as each one is a product of the analysis of the results of the previous one, taking advantage of the conclusions found to improve the way of dealing with the case study.

The first approach performs random requests to the model to save its outputs and try to find flaws to be exploited by an attacker. This approach tries to create a substitute model with the collected information to analyze the feasibility of using it to generate attacks efficiently.

The second approach assumes that the attacker possesses certain real extreme cases of the model and performs an exploratory analysis to evaluate opportunities to create attacks based on them. This approach focuses on analyzing the similarity in the input of the different cases.

The third approach assumes that the adversary takes the same cases and performs minimal perturbations to measure the local influence of each of the variables, looking for high potential leverage variables.

The fourth approach simulates an adversary using a state-of-the-art Black-box method (ZOO), following its iterative process, looking for ways to create adversarial attacks optimizing changes in the variables.

The fifth approach uses a recent state-of-the-art White-box attack optimized for tree ensemble models, which has better properties than ZOO with respect to the type of model under study.

Finally, the sixth approach performs a counterfactuals analysis to explore the similarity between different real cases, following a similar logic to the second approach but using a larger data set.

7.2.1 Black-box Random Requests

Assuming an attacker can only make Black-box attacks by sending requests iteratively, this simple approach measures the difficulty of performing successful evasion attacks to the binary classifier. It is assumed that the attacker knows the variables' names to create consistent examples with each of the input variables type and domain. Random examples are massively created and tested, recording their output score to generate a substitute model based on them for further analysis.

7.2.1.1 Results and Conclusions

Despite not being able to create a substitute model with high performance, an interesting conclusion was that the models created showed a similar distribution of variable importance to that of the original model, using only 9,000 points. **This could allow an attacker to deduce with a substitute model created with random examples which variables to focus on, giving him insights to find leverage variables.**

On the other hand, it is found that **for this case study, it would not be possible to create a valid substitute model by feeding it with randomly generated cases and their output in the real model.** This can be seen in the following analysis performed:

Output score distribution

A comparison between the distribution of the output score from the simulated cases versus the real cases is performed. It could be seen in Figure 7.1 that the simulated cases had a normal-curve form distribution centered in 0.42, very close to the center of the range, instead of showing two differentiated curves as it happens with real cases. This will mean that using simulated cases to create a substitute model will not have enough information to characterize cases near 0 or 1, which are probably the most relevant cases in the creation of patterns by the original ML model. Not using cases from the real data manifold will make the substitute model represent regions of not interest. Many ML models would succeed in creating a model that fits the random samples, but they will not guarantee good performance when dealing with real data.

Substitute model creation

Substitute regressor models (using the recorded output score from the original model) were created using the simulated data, creating tree-ensemble models with cross-validation and analyzing the variability in their performance, checking the model's robustness creation over different data partitions, and iterating this process over a hyper-parameters grid-search. Each of them managed to have good performance over the used simulated training data (tending to do overffiting to it easily), but showed poor performance when trying to predict some real data examples. As seen before, the output score distribution lacks extreme values (close to 0 and 1), reason why the used data will surely not help the substitute model create patterns with discrimination potential.

Computational cost for a broader input space search

Knowing that the distribution of simulated cases doesn't seem to be covering the real data manifold properly, the possibility of performing an exhaustive sampling process to cover most of the input space was analyzed. Considering most of the variables are categorical or binary, and evaluating a small subset of values for the continuous variables, it was estimated

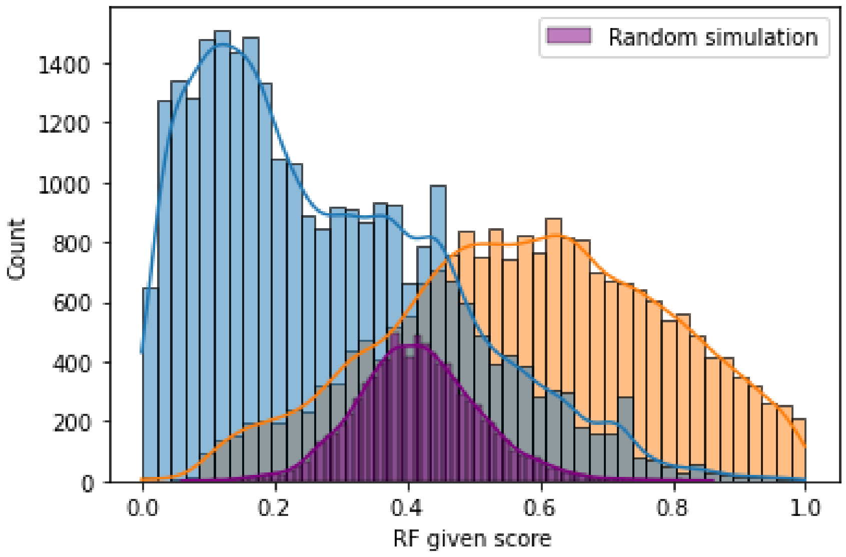


FIGURE 7.1: Original model's output score over real data vs simulated data. Blue distribution is for real no phishing cases (label 0), orange for real phishing cases (label 1), and purple for simulated data.

that to cover the input space, 105×10^{12} cases were needed to be simulated. It would not be possible for an attacker to cover more than the $8.5 \times 10^{-9}\%$ of the cases making requests for an entire day. This means that the input space must be narrowed somehow to reduce the number of cases to be evaluated, being still difficult to deal with the curse of dimensionality.

Difficulty for analyzing feasible examples

Finally, trying to narrow the search space given the previous analysis, it should be tried to create only feasible examples representing the real data manifold. But given that the features were created based on many characteristics evaluated from the raw data, it is complicated to logically analyze how to add helpful restrictions to focus the areas for the simulated examples.

7.2.2 Comparison of a Sample of Extreme Cases

Based on the previous conclusions, the scenario in which the attacker manages to collect a small set of real examples is presented. An exploratory analysis of cases collected for each of the four positions of the confusion matrix was performed, intending to measure the similarity between these extreme cases. The intention is to understand if there are few variable changes able to transform, for example, a true positive into a false negative, performing an evasion attack.

To measure the main differences between these cases, a PCA was done to easily visualize the importance of the 50 variables in the variability of the extreme cases, checking over the first principal components the variables with higher loadings, and also creating a scores plot to have intuition about the similarity between the different examples.

Additionally, the difference between the examples was measured by creating a matrix representing the Manhattan distance between each of the examples. This metric was selected

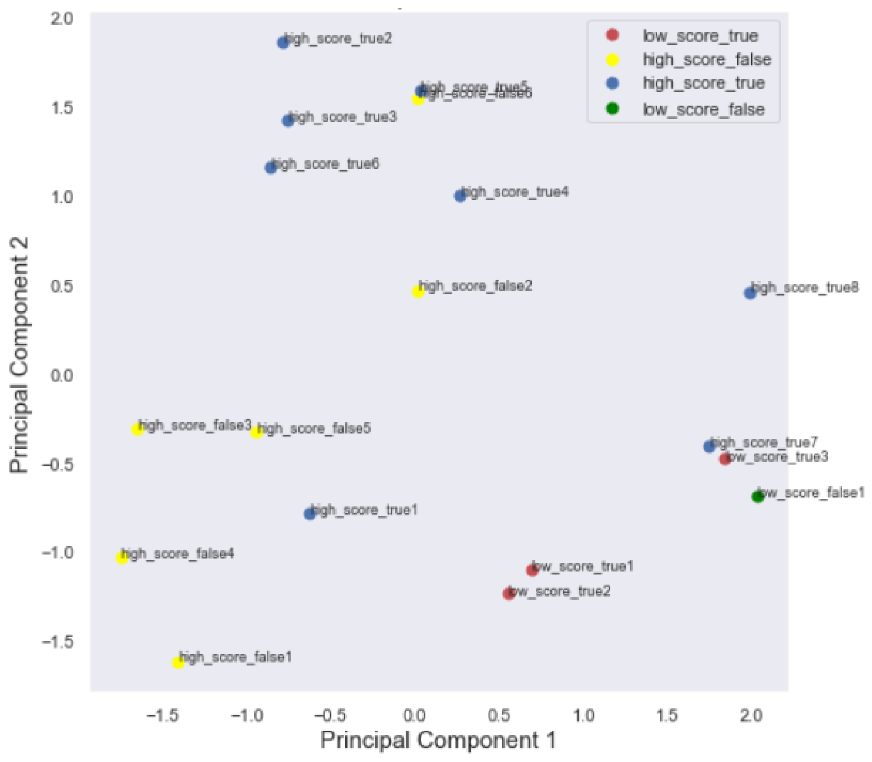


FIGURE 7.2: Extreme cases first scores over first two principal components. Red examples are real phishing cases with low prediction score (false negatives or evasion cases), yellow examples are non-phishing cases with high prediction score (false positives), blue examples are phishing cases with high prediction score (true positives) and green examples are non-phishing cases with low prediction score (true negatives).

considering that most of the variables were binary, making it useful to approximate the discrete number of changes required to transform one example into another.

7.2.2.1 Results and Conclusions

First, by analyzing the loadings of the first principal components using a heatmap table on top, it was possible to detect a small number of variables with great importance. However, when analyzing the definition of these variables in detail, it was considered logical that they were the most influential ones, being important variables with little leverage for an attacker. In addition, although the number of data is very small, it was analyzed that the first two principal components represent 20% of the variability of the selected cases, so they are not considered to have sufficient potential to summarize the variability of the data.

The previous table, together with the scores plot of the examples on the first principal components allows analyzing if there are close cases and concerning which variables. As seen in Figure 7.2, red examples are false negative or evasion cases, so finding a true positive (blue example) close could help check the most relevant features between correctly and incorrectly classified phishing cases. Following the same logic, close false positives (yellow) and true negatives (green) may help understand relevant variables for correctly classifying non-phishing cases. These variables must be detailed to check if it is valid to consider them essential and difficult to modify variables by an attacker, or if they may represent spurious variables easy to be changed to fool the model.

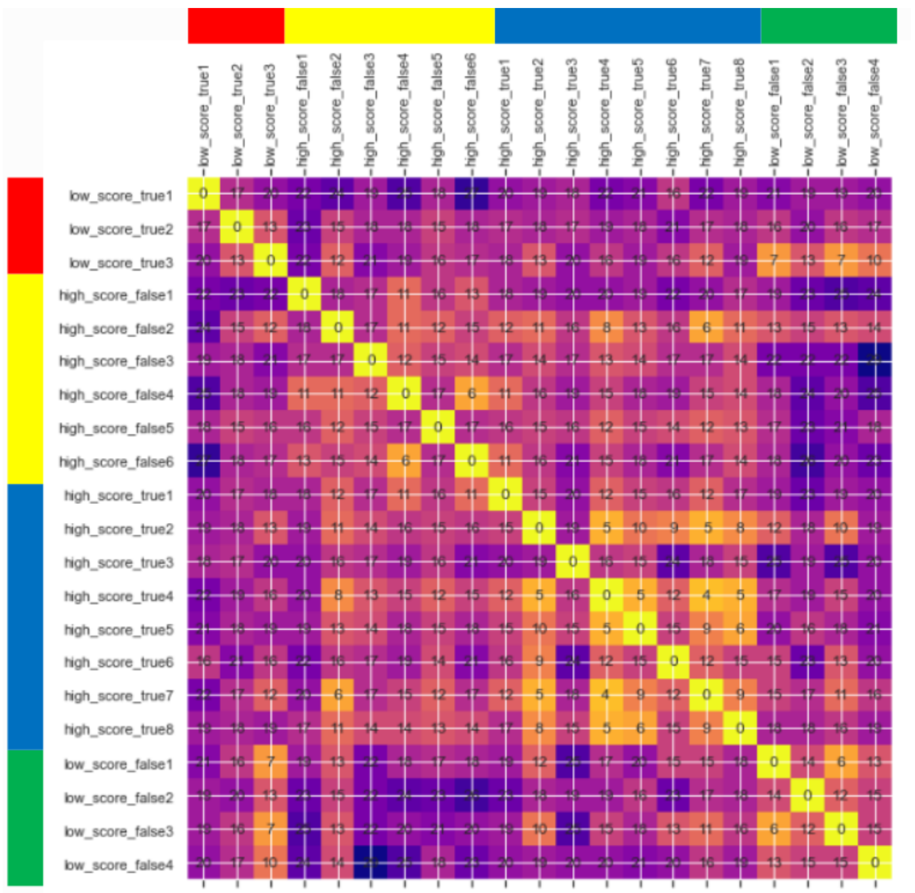


FIGURE 7.3: Extreme cases Manhattan distance matrix. Light values show examples with low Manhattan distance or few number of differences in the 50 variables representation used as model's input. Lateral colors represent the type of example from each of the four regions of the confusion matrix. Red examples are real phishing cases with low prediction score (false negatives or evasion cases), yellow examples are non-phishing cases with high prediction score (false positives), blue examples are phishing cases with high prediction score (true positives) and green examples are non-phishing cases with low prediction score (true negatives).

Also, analyzing the Manhattan distance matrix it was possible to check for similar cases (with low values) from different regions of the confusion matrix. There were not considerably close cases when comparing correctly or incorrectly predicted phishing cases (blue vs. red regions), nor correctly or incorrectly predicted non-phishing cases (green vs. yellow), as they seem to require at least 12 variable modifications. Cases with closer similarity from different type of examples were found between true-positives and false-positives (blue vs. yellow) and some between true-negatives and false-negatives (green vs. red). These cases were further analyzed to check the differentiating features, giving insights about the critical features in the model's decision.

The model appears to be robust with the analysis performed, so it is decided to create more complex analyses to try to look for weaknesses in the following approaches.

7.2.3 Single Perturbation Analysis

For each of the gathered real cases from the previous approach, all the possible single perturbations were created. This process consisted of changing a single feature at a time until covering all the 50 features for each of the considered examples, and evaluating the perturbated examples with the model. These results were analyzed to check the influence of each of the variables for cases from each position of the confusion matrix, also considering their output score change.

In the analysis, it was important to keep in mind the threshold used to make model decisions on the model output. There can be variables that generate significant changes in the output score of specific examples, but do not really change the region in the confusion matrix delimited by the model's threshold chosen, while other variables may only minimally affect the output score but make an example surpass the threshold and create an effective attack.

7.2.3.1 Results and Conclusions

From all the possible combinations for each of the examples, only 26 single perturbation cases managed to change from an output score above the threshold to a value below it, being evasion candidates. There were found that many cases ocurred when changing a feature which was considered logic to be important in reasoning for detecting a phishing case, meaning that these evasion cases would not be feasible to be created by an attacker. But there were found few feasible cases, that by modifying a single feature may evade the model. **This variable is difficult to change, but could be considered by an attacker to improve a previous tried attack.**

Also, some statistics were performed for each of the variables, to have an idea of the minimum, maximum and average change created when modified, and also the variance from these changes. Finally, some visualizations were created comparing the score from the original example in the horizontal axis and the score of a perturbation made in the vertical axis for all the perturbations created. An example of the visualizations performed is shown in Figure 7.4, where a perturbation without changing the output score should be placed over the 1:1 green line starting from the origin. Those relevant cases would be those surpassing the threshold line vertically (used as 0.2 in this example) starting from the green line. This visualization was performed for each of the features to make a detailed analysis.

This approach was useful to detect some potential attacks using a specific base example and modifying a single feature, allowing making a **feature sensibility analysis as an indirect approach to detect features to be considered by an attacker**, also **being useful to improve the feature engineering step to reduce these vulnerabilities.** A double perturbation analysis could be done, but it will need an exponential increase in computational efforts.

7.2.4 Black-box ZOO Method

In this approach, the Black-box ZOO (Zeroth Order Optimization) attack method is used to create candidate adversarial examples that are able to fool the model, and in particular to evade it. This is performed using this public method implemented in the Adversarial Robustness Toolbox (ART) Python library created by IBM but currently Open Source.

The previously mentioned candidates are useful to analyze the adversarial robustness of the model, reason why different metrics or KPIs are proposed, as it will be seen later. This methodology can be applied in a generic way to other models, allowing it to be used to create decisive metrics together with other performance metrics to select between models by

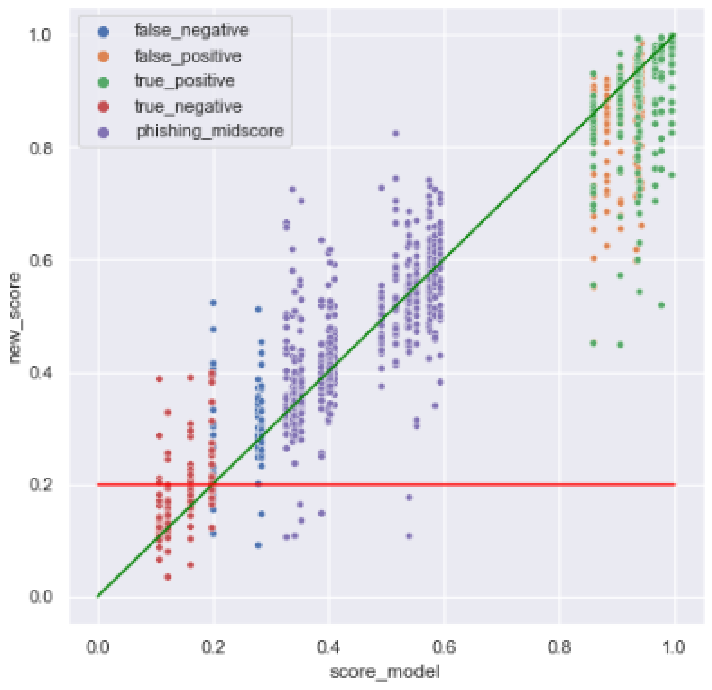


FIGURE 7.4: Single perturbation output score change analysis. Horizontal axis present the original example output score, and the vertical axis the perturbated example output score

measuring their robustness and not only the performance with in-distribution cases. Additionally, a more detailed analysis of the candidates allows the creation of a report that helps to know in greater detail the model's operation and its vulnerabilities.

7.2.4.1 Attack Purpose and Assumptions

As explained in section 5.3.4, the ZOO attack method only needs the input and output to solve the optimization problem to create adversarial examples, being assumed that the attacker has no further knowledge of the model. But in this case, it is assumed that the attacker owns real data to take as a base to create adversarial examples, being a test set from the same training data distribution used for this adversarial robustness estimation. It also assumes that the attacker can create a considerable amount of requests to the model to use the iterative optimization method.

In practice, the test set of examples is used, and for each of them an adversarial example is optimized using the ZOO method. This method was considered given its strengths explained in chapter 5, and because it is available in the ART library created by the same IBM authors of the paper.

It is important to note that the optimization method and logic from this method are based on images problems, being possible to measure the adversarial examples perturbations using the Euclidean distance, assuming that all input features, which are pixels, have the same relevance and scale. This will clearly not be valid in this context, as there may be very important binary features and complementary features that help only in the inference of specific scenarios. Also, this method was created for DNNs, which work using gradients, but not tree-ensembles with discrete partitions. This will mean that the method attack will not differ between continuous and integer features, and would try to estimate gradients in the neighborhood of integer features and may propose adversarial examples not meeting the features data type. This is furtherly analyzed in the conclusions from this approach.

7.2.4.2 Attack Methodology and Considerations

The methodology followed consists of proposing one by one an adversarial example for each of the cases of the test data set. The test set is divided into X_{test} and y_{test}, being the first one an array with n rows (number of examples) and p columns (one for each variable), and y_{test} being a vector of length n with binary values according to the real class of the associated example (1 for phishing and 0 for non-phishing). Using X_{test} and y_{test} it will be possible to create the adversarial test set, with X_{adv_test} containing the adversarial examples candidates, with n rows (associated in the same order as X_{test}) and the same p variables. Additionally, y_{adv_test} is estimated, which will have the output score estimated with the model predictions, with continuous values between 0 and 1.

With the above data sets, the metrics can be estimated and the corresponding analyses can be performed, as shown in the next section.

7.2.4.3 Adversarial Robustness Proposed KPIs

Statistics are created to measure an approximation of the adversarial robustness of the model to this particular type of attack. This methodology is generalizable to other types of attacks, but the KPIs created will be associated with a particular attack method, model, and dataset. This implies that it will not make sense to compare these KPIs by changing the dataset or the attack method, and will be useful only to compare two different models with the same base test dataset and attack method.

The KPIs created can be separated into two categories. The first one analyzes only the predictor variables (input), for example, X_{test} against X_{adv_test} to analyze the number of cases in which the method succeeded in proposing an adversarial example (without yet analyzing whether it was effective) and to analyze the required perturbation. In the second category, the results in the target variable are analyzed to evaluate the attack's effectiveness, together with the type of attack performed (false negative or false positive) with different thresholds, and how feasible it is.

7.2.4.3.1 Input KPIs

The first thing to note is that the method fails to propose adversarial candidates for some examples. For this reason, the first KPI estimated is the Stability index.

- **Stability index:** Percentage of X_{test} examples that remain exactly the same in X_{adv_test} (percentage of unchanged cases). A high value in this index will be aligned with a higher adversarial robustness of the model.

Additionally, there may be used complementary KPIs to measure the perturbation aggressiveness over the examples:

- **Number of changed features:** Estimating the mean, minimum, maximum, and standard deviation of the number of features with changes in the example when comparing

X_{adv_test} with X_{test}. This is useful to measure the aggressiveness of the perturbations proposed by the method.

- **Average mean feature perturbation:** Estimating the mean (or even the median) feature perturbation for each of the examples and then averaging over the changed cases (leaving the unchanged cases out).

Finally, the previous two KPIs can be estimated but counting examples for each feature to estimate the most decisive features. In this way, it is possible to measure the number of adversarial examples candidates with changes for each of the features. Also, the average, minimum and maximum perturbation made to the feature for the adversarial examples with modifications in this feature. This will allow analyzing the most relevant features in the creation of adversarial examples to analyze if any of them may represent a leverage feature.

7.2.4.3.2 Target KPIs

Having measured the proposed perturbations in the input, it is analyzed whether the purpose of the attack was achieved. For this purpose, we measure how many cases were able to be switched from one side of the threshold to the other. This metric is linked to the selected threshold, so it is recommended to analyze it for different ones and thus make a threshold sensitivity analysis.

The following metrics can be estimated:

- **Evasion cases (7.1):** Cases that were originally predicted as positive, but with the perturbations are now predicted as negative. They are dependent on the output score selected threshold.

$$Num.\ Evasion\ cases_{threshold} = count(y_{test} \geq threshold \cap y_{adv\_test} < threshold) \tag{7.1}$$

- **Evasion cases proportion (7.2:** Evasion cases divided by the number of cases that were originally predicted as positive. This will show the proportion of positive cases that the attack method was able to turn into evasion attacks.

$$Prop.\ Evasion\ cases_{threshold} = \frac{Num.\ Evasion\ cases_{threshold}}{count(y_{test} \geq threshold)} \tag{7.2}$$

- **False alarm cases (7.3):** Contrary to evasion cases, these cases were originally predicted as negative, but with the perturbations are now predicted as positive. They are dependent on the output score selected threshold.

$$Num.\ False\ alarm\ cases_{threshold} = count(y_{test} \leq threshold \cap y_{adv\_test} > threshold) \tag{7.3}$$

- **False alarm cases proportion (7.4:** False alarm cases divided by the number of cases that were originally predicted as negative. This will show the proportion of negative cases that the attack method was able to turn into false alarm cases.

$$Prop.\ False\ alarm\ cases_{threshold} = \frac{Num.\ False\ alarm\ cases_{threshold}}{count(y_{test} < threshold)} \tag{7.4}$$

Finally, considering that for this problem, the evasion attacks are the most relevant, the following KPI was created to measure its resilience to them.

- **Resilience index 7.5:** Proportion of original positive cases that the attack method was not able to transform into negative examples, being the complement of the Evasion cases proportion. The Resilience index is dependent on the output score threshold selected.

$$Resilience\ index_{threshold} = 1\ - (Prop.\ Evasion\ cases_{threshold}) \tag{7.5}$$

7.2.4.3.3 KPIs Integration and Presentation

Having defined the input and target KPIs, it is relevant to integrate them to measure the effectiveness of the attack in the target variable but also its feasibility.

In order to make an initial filter to determine which candidate adversarial examples are feasible, it was important to analyze which cases met the data type and rank. So an initial table analyzing each of the features was created, showing the number of candidate adversarial examples with perturbations on each feature, how many met the data type, and how many were below the minimum or above the maximum value of its logical range. This table will be called the *Feature perturbation feasibility table*, and the example is not shown for security reasons.

Finally, a *Threshold sensitivity table* estimates the target KPIs presented previously, for different thresholds. This table will clearly show more evasion cases for higher output score thresholds and more false alarm cases for lower ones. This table will help understand the adversarial robustness of the model for different thresholds, being useful to analyze both performance and robustness to select the more appropriate threshold for the problem. An example of a simple *Threshold sensitivity table* is presented in Table 7.1

This table can be created first without filtering cases not meeting data type and range, but also another table filtering them. Comparing the previous two tables would help to understand the proportion of proposed adversarial examples that can be really be used in practice.

7.2.4.4 Results and Conclusions

Before concluding on the adversarial robustness of the model to the ZOO attack, it is important to mention certain particularities encountered. First, when analyzing the *Feature perturbation feasibility table*, it was found that **the method was not able to propose changes for integer variables**, meaning that it was only able to modify continuous variables. This can be explained as mentioned in section 5, because ZOO is a method designed for DNNs that works with gradients, which it approximates with values in the neighborhood of the analyzed point. Considering that this model is a tree-ensemble, small changes will not cause the branch in a decision tree to change, and the estimated gradient will probably be zero.

To overcome the previous problem, **different hyperparameters from the ZOO attack were tested, logically tuning them to reach a broader space to estimate the gradients**, but no improvements were achieved.With this first conclusion, it is considered important to look for an attack that fits this type of model, as will be done in the following approach.

Additionally, of the **continuous variables** capable of modifying the method, **some cases did not meet the logical range of the features**, thus reducing the number of feasible candidate adversarial examples. To solve this problem, the ZOO algorithm could be modified to adjust these constraints, which involves considerable development time, being considered a possible future improvement in the ART library.

With the previous considerations, a **high Stability index was achieved** for cases meeting the data type and range or without it. Also, it was interesting to find that the method **was able to create false alarms more easily than evasion attacks, having no feasible evasion attacks for most the most relevant thresholds analyzed** in the *Threshold sensitivity table*. Considering these facts, it could be concluded that the model has considerable adversarial robustness to adversarial evasion attacks performed with the Black-box method ZOO.

Finally, although in this case it was not necessary to analyze the attacks in greater depth, since there were no effective evasion cases for the thresholds considered, **it is crucial to continue analyzing the feasibility of the candidate adversarial examples**. A proposed attack could fulfill its evasion purpose and additionally fulfill the data type and range, but for this reason, it could not be considered feasible ye. An attempt should be made to **analyze which raw example can represent the candidate adversarial example, analyzing its multivariable and business logic.**

7.2.5 White-box Efficient Adversarial Attack for Tree Ensembles

7.2.5.1 Purpose and Implementation Adjustments

This last approach uses exactly the same methodology as the previous approach, with the difference that instead of using the ZOO attack method, it now uses the *Efficient Adversarial Attack for Tree Ensembles* method (Zhang, Zhang, and Hsieh, 2020). This White-box attack that was already explained in section 5 is adapted for tree-ensembles like the ones used in this case study, and allows proposing adversarial examples that modify both continuous and integer features and use a more efficient algorithm taking advantage of the internal structure of the decision trees.

This approach seeks to test the model with a more appropriate attack that could damage much more. The implementation was made with the code created by the authors of the paper, which is openly shared in the paper. This code was modified because it was not as versatile as the ART library used to implement ZOO. The code was adjusted to allow the use of the type of model to be evaluated, which had a different format than the example. Also, to save each of the candidate adversarial examples that the code produced only to generate statistics at the end. Finally, **it was also modified to allow the use of null values**, considering that the model under study allows them. In this way, the null values were adjusted to have high values and thus force them to take the right branch in the decision trees they are involved in, doing it in a similar way as the Python implementation of XGBoost does.

TABLE 7.1: Threshold Sensitivity Table Example

Output Score Threshold	Evasion Cases	False Alarm Cases	Evasion Mean Number of Perturbated Features	Evasion Minimum Number of Perturbated Features	Resilience Index
0.1	0	1033	NA	NA	1.00
0.2	0	1046	NA	NA	1.00
0.3	2	1054	2	2	0.99
0.4	17	1055	3	1	0.99
0.5	108	1055	4.1	1	0.95
0.6	104	193	4.2	1	0.95
0.7	99	35	4.4	1	0.95

7.2.5.2 Results and Conclusions

The Stability Index, in this case, is much lower than that achieved with ZOO, as the method almost always manages to propose some adversarial candidate example, showing the effectiveness of the method used. However, although the attack was successful in a large number of examples, most of them were to create false alarms. Thus, a high proportion of false alarm cases and a very low proportion of evasion cases were identified, having the model a high Resilience Index.

This analysis was performed for multiple output score thresholds and by analyzing data type and range fulfillment. It was found that approximately half of the candidate adversarial examples did not meet the ranges for any of the features. Still, the same proportion remained for evasion and false alarm cases. An example of the Threshold Sensitivity Table can be seen in Table 7.1, showing for each output score threshold the evasion and false alarm cases, together with statistics from the number of perturbations needed in the evasion cases and the Resilience index. This table can be used together with more metrics as explained before, to use it to select the most favorable threshold.

Finally, it should be noted that this method is capable of modifying all features regardless of their type, that it was able to propose attacks with changes in a few features, and to use an algorithm optimized for tree-ensemble models. It was also useful to analyze in greater detail the candidate adversarial examples in order of priority and to analyze which features were important and how viable they were.

It should be remembered that it should be further analyzed how the representation proposed by each candidate adversarial example can be translated into a raw data example used before preprocessing for the model. Although this requires an analysis process that can be time-consuming, this analysis reveals the most promising cases on which to focus efforts.

7.2.6 Counterfactual and Simmilarity Analysis

To complement the previous approaches, similarity analysis of real examples is carried out to evaluate close cases classified differently by the model. This is done by first visualizing the data using dimensionality reduction methods to sense the overlap of the two classes evaluated. Subsequently, tools are used to analyze the real nearest counterfactual of the data in order to identify potential analysis cases for an attacker to evade the model. Finally, the nearest counterfactuals are explored in an automated way to obtain useful metrics to understand the vulnerabilities of the model.

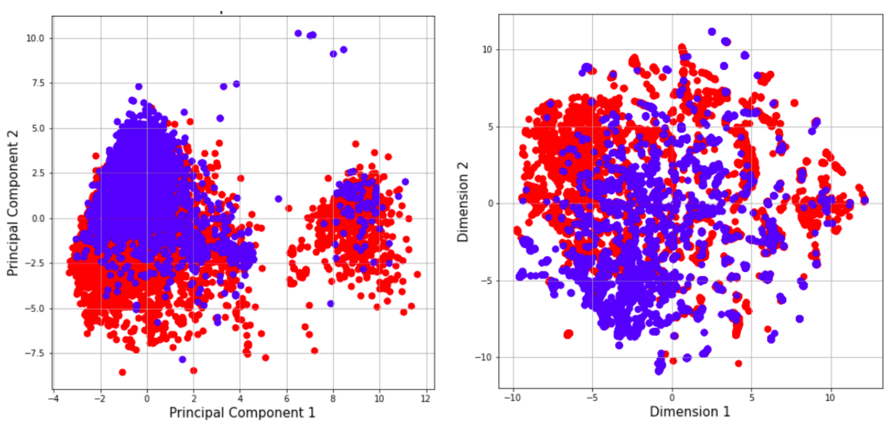


FIGURE 7.5: Data Representation PCA and t-SNE Visualization Example. Red dots represent cases classified as positive and blue dots as negative.

7.2.6.1 Results and Conclusions

First, the PCA dimensionality reduction technique is used to see how the examples of the two classes overlap according to the input representation. The projection of the data on the first two principal components can be seen in Figure 7.5. However, it should be considered that these two principal components, although they give intuition about the distribution of the data, only explain 12% of the variance of the data. Additionally, the t-SNE technique is applied to visualize the proximity of the data in a summarized form over two dimensions, also presented in Figure 7.5.

In addition, Google's *What-if-Tool* was used to explore the dataset examples in a more versatile way. This tool allows estimating the nearest counterfactual for each case and visually contrasting their differences. This tool is very useful to analyze specific cases. It is recommended to use it frequently to validate and investigate atypical cases of the model daily, considering it important to perform an audit of the model periodically.

However, to know which examples to focus on when using the tool, the determination of the nearest counterfactuals of the data was automated using Python. In this way a list of pairs of nearest counterfactuals $(\mathbf{x_1}, \mathbf{x_2})$, $(\mathbf{x_3}, \mathbf{x_4})$, ... was made where the first case is classified as non-phishing by the model, and the second case as phishing. These tuples were ordered according to their proximity in Manhattan distance (considering most features are binary), but also testing other metrics such as Euclidean distance and Cosine similarity. This analysis was repeated for different output score thresholds of interest, similar to what was done with the *Threshold Sensitivity Table*.

Several more detailed exploratory analyses can be derived from these results. For example, it is possible to take the first n closest to each other nearest counterfactuals pairs from the dataset and to create statistics to determine the most relevant features to differentiate the example pairs. **This can be used as a different methodology to determine important features to be analyzed as leverage features by an adversary. The analysis of real nearest counterfactuals is very useful to complement other analyses since it guarantees to be analyzing real cases.** This implies that if a pair of examples with potential is found, it will be possible to evaluate how to exchange them with small changes in the variables to carry out a feasible attack.

7.3 Case Study 1 General Conclusions

This case study was helpful to show the **importance of using different approaches to analyze the adversarial robustness of a model**, mixing exploratory and similarity techniques, adversarial attack methods and KPIs from them, and, importantly, business logic. Additionally, it is recommended to address the approaches in increasing order of complexity, starting with approaches with a solid analytical and experimental orientation, progressing to more automated and complex approaches. Each of these stages is highly recommended to really be able to detail the vulnerabilities of a model without losing control of the analysis, meaning that it is not recommended to directly evaluate attacks on a model to assess its adversarial robustness.

In the development of each approach, it is very important to **keep in mind the assumptions involved from the attacker's point of view and make hypotheses and conclusions according to the techniques' capabilities and logic used**. Thus, it is important to consider that each of the approaches used is useful to approximately determine the robustness of the model and raise possible vulnerabilities, but that a detailed analysis for its validation should always complement it.

A useful methodology was created to measure the degree of vulnerability of an attack method to the model, which is generalizable to all types of attacks . This methodology was automated in AzureDevOps pipelines in order first to create adversarial examples optimized with a chosen attack method and then create all the KPIs explained in the fourth approach. **Its use can be useful to quickly get an intuition about the adversarial robustness of a model to specific attacks, which is helpful to choose between candidate models and create a report about the most relevant features in the creation of attacks.**

Additionally, the importance of evaluating adversarial examples was analyzed, **considering their effectiveness over the target variable and the perturbation required to achieve it**. These two factors are important to be analyzed together since it is possible to create an adversarial example that manages to change the output of the model but requires a very high perturbation that makes the example no longer feasible, or simply no longer resembles the base example. For example, a phishing email can be created such that the model does not detect it, nonetheless, it is illogical and generates distrust for the user, without resembling a benign email. In the opposite case, it may happen that an adversarial example does not achieve its objective in the model output because it has limited itself to minimal perturbations. In this case, it may happen that the method achieves its goal with a slightly larger perturbation, and the required perturbation is still not sufficiently noticeable to be detected by a user.

Given the type of problem considered in case study 1, which creates a representation from the raw data with manually designed features, **it is difficult to propose candidate adversarial examples that directly represent a valid raw adversarial example**. For this reason, it is important to analyze how feasible, starting with data type and range conditions, but also later on in a multivariate and business logic manner. Additionally, **the methods used assume that each of the features has equal importance when optimizing the perturbations**. However, there may be very important features at the business level, which if changed will represent very strong logical perturbations. This will cause attacks to focus on features that are difficult to change (which makes sense to be the most relevant in the model inference process), instead of focusing directly on leverage features.

When analyzing adversarial candidate examples, **it is important to consider the type of attack achieved and its impact at the business level**. For example, in this case, greater importance was given to evasion cases than to false alarms, and the problem was oriented to the output score thresholds selected in the model deployment.

Additionally, certain conclusions could be drawn at the technical level in this case study. First, it was seen that **due to the curse of dimensionality, it is not easy to simulate random data to create a substitute model in a high dimensionality problem**. It is important to know the real data manifold to be able to create real approximations and to be able to reduce the Black-box search space to create a substitute model.

The ZOO method is helpful to guarantee a Black-box scenario, being easy to implement with the ART library in Python. However, since it is based on gradients approximations to attack DNNs, it has problems when creating adversarial examples in tree-ensemble models. **ZOO fails to change integer features that can represent binary or categorical characteristics and is limited to using only continuous variables in its optimization**. An Efficient Adversarial Attack for Tree Ensembles solves this problem and adapts specifically to tree-ensemble models, with the small difficulty of requiring code modifications as it is implemented outside a more robust library such as ART.

Finally, The use of counterfactuals helps ensure that real cases are analyzed on tabular data on which it is difficult to add knowledge of the logic of the problem to the search for adversarial examples.

Chapter 8

Case Study 2

With a different perspective, case study 2 studies the adversarial robustness of a logo classification model, which can be used in cybersecurity together with other serial and parallel models to identify fake web pages. This model is selected due to the great vulnerability demonstrated that image classification models based on CNNs have, to evaluate its robustness, find vulnerabilities, and thus propose changes to continue improving it.

For security reasons, it is important to mention that the model analyzed in this chapter was created only for research purposes, being a model that is not used in the company and that was simplified to focus the analysis only on an image classification model. Thus, data downloaded from the Internet and state-of-the-art architectures were used to apply AML techniques effectively.

8.1 Model's Context and Original Model Training Process

Although this problem may be similar to other image classifiers, such as those created with databases like ImageNet or COCO, additional considerations must be taken into account. When working with logos, it is not easy to find a wide variety of examples, so even though large volumes of data may be available, it is very likely that many of the images will be repeated or have minimal variations. Additionally, it should be considered that an image may not contain any logo, or it may have a different logo from the ones to be classified. The way to deal with these considerations will be analyzed in particular in the enhanced model, after presenting the dummy model used initially.

The first step in applying each of the AML techniques was to create **the original model, which is a well-elaborated base model to be used as a basis**. **This original model was trained following a traditional methodology, in which the aim was to achieve the highest possible performance (accuracy) on a in-distribution test set**. The different AML attacks and defenses will be tested on the original model to analyze the vulnerabilities it may have against adversarial attacks or even out-of-distribution cases.

In the process of creating the definitive original model, **a simple initial model was created to have an intuition about the vulnerabilities of the CNNs quickly**. Subsequently, **the enhanced model was created, which was trained in detail to achieve high performance and to have features that make it stronger for out-of-distribution cases according to state-of-the-art recommendations for logo classifiers**. These two models are explained next.

8.1.1 Initial Dummy Model

The initial dummy model was created as a proof of concept to determine the potential of a logo classifier to be attacked by an adversary. Thus, it is a model that was created quickly to test state-of-the-art AML attacks, and to use it as a basis to know its easy applicability, develop a pipeline to do so, and analyze improvements to be made in the next model, the enhanced model.

This model was created following the best practices considered in section 5, regarding the adaptations to be made to logo classifiers. The model was trained with images downloaded from the Internet for each of the classes, adding an additional class for the background or for any logo or image that does not correspond to any of the 3 logos searched by the classifier. This, as explained in section 5, is done to allow another previous model to conservatively propose all the crops of the image that may have a logo. In this way, it is important that the classifier can know if none of the logos of interest is in the image, and that it is not forced to choose between them, allowing the output scores of the soft-max layer to attribute more weight to another class.

Additionally, fine tuning was made from a pre-trained CNN. For training, only one hyper-parameter combination was tested for a quick creation. Adam optimizer was used and the learning rate was adjusted according to the batch-size allowed by the graphics card to be trained using GPUs, optimizing to minimize a Cross-entropy loss function. Input size of 256x256x3 (RGB) pixels was used, with a ResNet50 architecture pre-trained with ImageNet without including the dense layers, and adding two dense layers of 128 neurons. Finally, the soft-max layer was added to give output scores for each of the classes, which in total add up to 1.

Additionally, **early stopping** was used to avoid overfitting after 5 epochs without improving the loss function of the validation set. A **learning rate reduction factor** of 0.4 was also applied, to be applied after 2 epochs without improvement in the validation loss, to allow the network to find with smaller steps new directions that help it to get out of a possible plateau.

This model on an in-distribution test set did not achieve a high performance, with accuracy values of around 80%, reasons that will be explained in more detail below, together with the adversary attacks performed on it.

8.1.2 Enhanced Model

The enhanced model was created to create the best possible model based on the performance of an in-distribution test set. All available techniques are used to achieve its effectiveness and robustness without using AML defenses. In this way, this model will serve as a solid base for creating adversarial attacks and testing AML defenses. **Ensuring its high performance will avoid attributing its failures to adversarial attacks to poor design in its creation.**

Input size of 256x256x3 (RGB) pixels was used, with a ResNet50 architecture pre-trained with ImageNet without including the dense layers. Two dense layers of 128 neurons each were added to this ResNet50, testing models with and without Batch Normalization and

Dropout of 50% to avoid overfitting (having the best model only Batch Normalization without Dropout). Finally, the soft-max layer was added to give output scores for each of the classes, which in total add up to 1.

Internet data was used, trying to use logos with good definition, with enough variability between them, and finally leaving a **background/other class** with images that help to differentiate the logos. For example, in this last class, other brands logos with similar colors, fonts, and patterns to the logos of the other classes (Nestlé, Nespresso, and Purina) were added, so that the model had to differentiate between similar patterns to really ensure that it focuses on the relevant features of the logos from the three classes of interest.

Following the best practices explained in section 5 based mainly on Deep Learning for Logo Recognition (reference) and understanding the difficulty of dealing with logos, the following considerations were taken into account. For each of the classes, only 30 images were used after **duplicates removal**, considering that more images would be repeating the same pattern. To compensate for the **lack of intraclass variability, data augmentation was used**. Up to 30 degrees rotations were used,as larger rotations were considered improbable at the business level since they were rotations that would be noticed by a user, and considering that a larger rotation could be solved by passing the crops in various orientations to the model. This way, no additional difficulty is added to the model, focusing it on learning logos in a considerable margin of rotations.

Additionally, shear was used, it was allowed to have a horizontal and vertical shift, zoom, and it was not allowed to use any flipping, since it would be evident to a user that the logo is mirrored since most logos have their name written.

An important aspect added to improve the performance of the model, which was not used in the initial dummy model, was to **apply padding to the images to respect their aspect ratio.** This also makes sense at a business level, considering that for an adversary to use a logo, it must maintain its general proportions in order to go unnoticed. In order to avoid Overinterpretation of the background, the images were duplicated to have **black and white padding.**

The model was created after logically iterating the hyperparameters to guarantee a correct learning curve, adapting the learning rate to allow the model to progress in a smooth progressive way, and testing different SGD and ADAM with momentum possibilities. Cross-entropy loss minimization was used to train the model. Additionally, **early stopping** was used to avoid overfitting after 5 epochs without improving the loss function of the validation set. A **learning rate reduction factor** of 0.4 was also applied, to be applied after 2 epochs without improvement in the validation loss, to allow the network to find with smaller steps new directions that help it to get out of a possible plateau.

With the model created, a performance of 95% was achieved in the test set, similar to the training and validation set, showing little overfitting. However, it is important to mention that the sets are similar despite the use of data augmentation due to the low intraclass variability available. With this analysis before applying AML techniques, it can already be expected that **the model may be doing Overinterpretation** as it is likely that it is not using the most appropriate features to find the patterns in the logos, when compared to the logic applied by a human. This can be exploited by an adversary and will be analyzed in the next section.

8.2 Attack Approaches

8.2.1 AML Attacks Over Initial Dummy Model

As explained before, an initial dummy model was created as a **proof of concept to determine the potential of a logo classifier to be attacked by an adversary.** Thus, it is a model that was created quickly to test state-of-the-art AML attacks and use it as a basis to know its easy applicability, develop a pipeline to do so, and analyze improvements to be made in the next model, the enhanced model.

Using a small set of logos, this model was used to apply 3 different white-box attacks using the ART library: **FGSM, Carlini & Wagner, and Adversarial Patch.** It was found that most attacks achieved their goal. However, it was useful to note the type of perturbations each attack performed, as shown with four examples in Figure 8.1

The FGSM method proposed very aggressive perturbations, which, even if they achieve their goal of being not correctly classified by the model, will probably be perceived by a user. On the other hand, the Carlini & Wagner attack made imperceptible changes at a human visual level, modifying certain pixel regions minimally. Finally, the Adversarial Patch iterated the location of a gray circle of predetermined size on the image and could not fool the model in all cases. Interestingly, the Adversarial Patch was able to fool the model in some cases by placing the patch in areas outside the logo in the image. This shows evidence of the **vulnerability of these models to suffer Overinterpretation** and be able to affect its prediction by modifying aspects that should be irrelevant, such as its background.

Additionally, it was identified that for several logos without any attack the model failed to perform the classification correctly. When analyzing the data used directly, it was found that the default model does not add padding to the images, so it could be changing their aspect ratio to adapt it to the square dimensions of 256x256 pixels. Thus, in image crops with highly irregular dimensions, the model was not able to make correct predictions. This helps as feedback to create the enhanced model, as explained above.

8.2.2 Hand-crafted, Random Noise Attacks, and Difficult OOD Cases over Enhanced Model

After learning from the weaknesses of the initial dummy model, the enhanced model was created following the considerations explained in section 8.1.2. With this high-performance model, an attempt was made to create hand-crafted attacks to simulate the case of an adversary who wants to find weaknesses in the model in an improvised way to analyze if there is any pattern that can be obtained. 23 different perturbations were tested in a generic Nestle logo image.

Subsequently, the robustness of the model was analyzed by perturbing the images with random noise, using the following techniques:

- **Random Gaussian Noise:** Making random changes over the pixels based on a Gaussian distribution with mean 0 and a specified standard deviation. Standard deviations with values of 2, 5, 10, 20, 50, 100, 200, 300 and 500 were tested over the RGB image represented with values between 0 and 255.

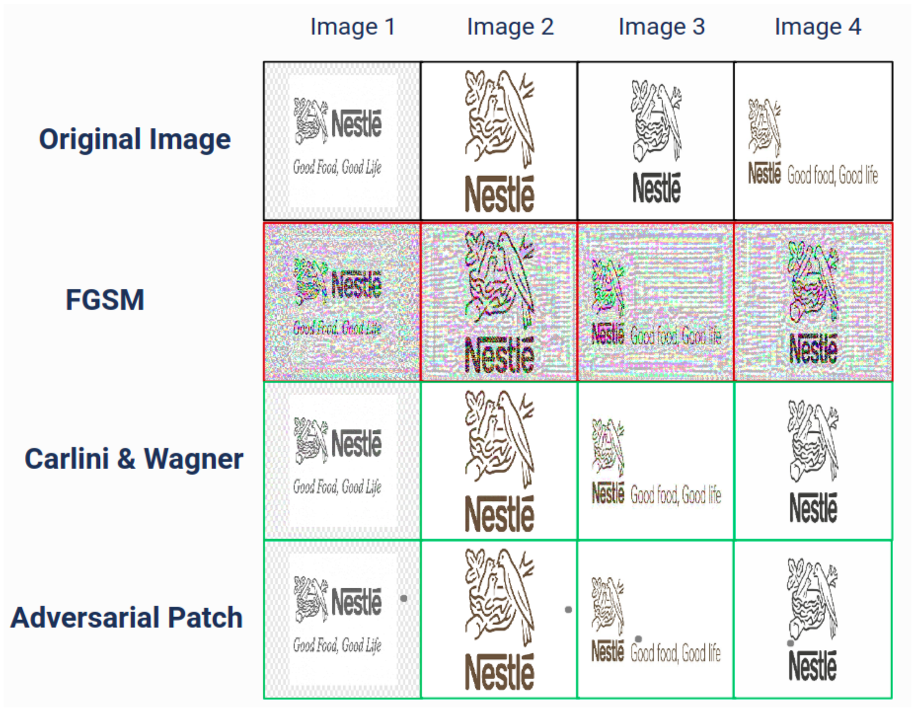


FIGURE 8.1: Initial Dummy Logo Classification Model Example Attacks. For four images the FGSM, Carlini & Wagner and Adversarial Patch attacks are shown. Attacks that are considered visually aggressive are marked in red. It is interesting to note the strong perturbations proposed by FGSM, the soft visual perturbations of Carlini & Wagner, and how the Adversarial Patch can propose patches in regions outside the logo region.

- **Salt & Pepper Noise:** Changing a proportion of random pixels into black (0 value for the three RGB channels) or white (255 for the three RGB values). Proportions of 0.05, 0.1, 0.2 and 0.3 were tested.
- **Speckle:** Multiplicative noise which by default has no variations using the OpenCV library.
- **Logo Color Pepper:** Changing a proportion of random pixels into the main color of the logo. For example for a blue Nestlé logo, the exact RGB color from the letters is used and randomly added to the image. Proportions of 0.05, 0.1, 0.2 and 0.3 were tested.

In addition, images with unusual Nestlé logos were searched and collected, finding logos with modified colors, adapted to advertisements (for example, a 3D Nestlé logo with chocolate texture and depth), etc. These images are used as an approximation to out-of-distribution cases, with which the robustness of the model to variations not contemplated in the training process can be analyzed, giving an idea of the features or representations created by the model.

8.2.2.1 Results and Conclusions

From the hand-crafted attack attempts, some simple cases that the model managed to classify correctly can be observed in Figure 8.2. It can be seen from these images that not just any modification can make the model behave erroneously, even covering letters that are considered important in the logo. On the other hand, Figure 8.3 shows cases that made the model misclassify the logo, which required much more aggressive perturbations.

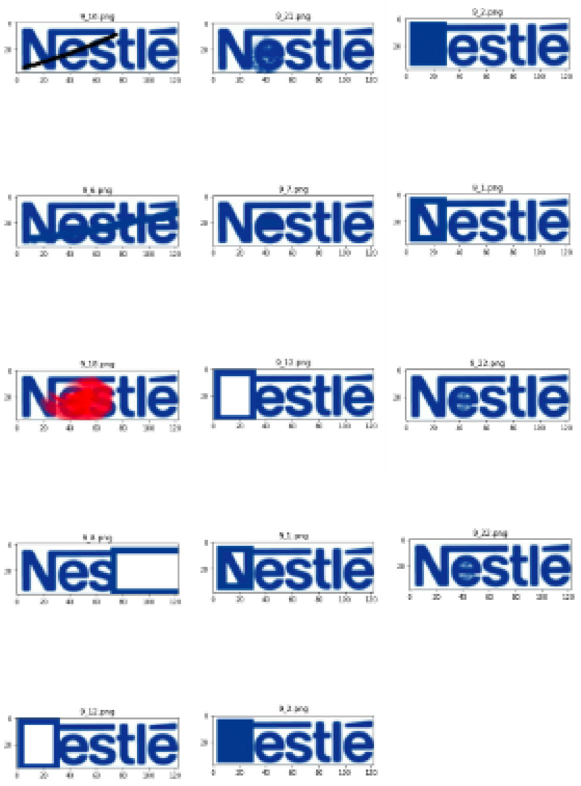


FIGURE 8.2: Hand-crafted attack attempts correctly classified by the enhanced model.

These hand-crafted attempts were **useful to make hypotheses about the most influential parts of the logo when making the correct prediction.** Additionally, it serves as example to show that **not any perturbation will cause the model misbehave, no matter how strong it may appear visually to a human**. This can give a **false sense of robustness** to the model when associated with human capabilities and demonstrates that **attacks must be performed at a numerical level optimized for the CNN used.**

In the random-noise case, each of the variations explained previously was executed 10 times with different pseudo-random numbers (seeds). This was done in order to make a more reliable analysis of the influence of each type of noise and at each different intensity, since, for example, it could happen that an attack is effective when selecting certain influential pixels, while another one with different pixels fails to affect the model. In this way, for each of the 10 attempts, the predicted class was recorded, but also the confidence given by the model. Additionally, this process was performed for the logo with different padding backgrounds, using a white background, a black background and another characteristic background. Statistics were created with the obtained results and presented with heatmaps that could be used to compare different images.

An example of the attacks performed on one of the images is shown in Figure 8.4 and Figure 8.5 . Additionally, an example of the type of data analyzed is shown in Figure 8.6. For each type of random noise, variation, and background, a statistical analysis of the predictions obtained by the model is performed, considering the predicted label and its confidence. Figure 8.6 shows an example of the model performance with a specific image. In this case, using a Gaussian noise with a standard deviation of 50 or less, the model is robust, but from 100

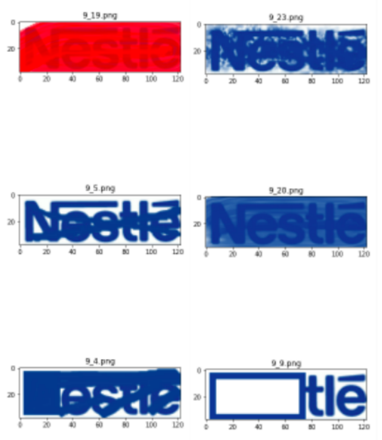


FIGURE 8.3: Hand-crafted attack attempts misclassified by the enhanced model.

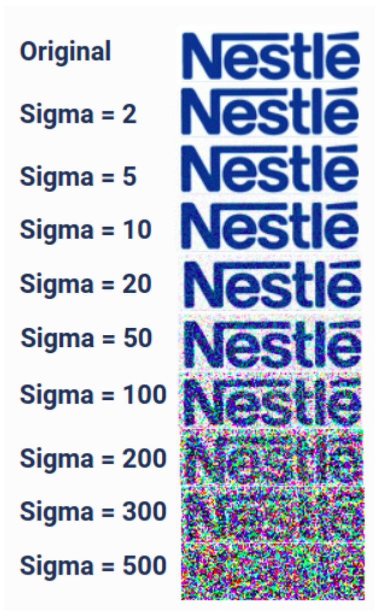


FIGURE 8.4: Gaussian Noise with different Standard Deviation Values. Values represent intensities for images with ranges between 0 and 256.

onwards it starts to fail in some cases and progressively gets worse with a higher standard deviation.

By performing this process with different images for each of the classes, **it was possible to determine which ones are more likely to be affected and by what type of random noise.** Although exact results are not disclosed for security reasons, it was found that some logos resisted high-intensity attacks, while others within the same class did not. Also, for example, the model seems to be more vulnerable to the proposed Logo Color Pepper than to Salt & Pepper, **proposing as hypothesis that Logo Color Pepper directly affects features related with the shape of the letters.**

8.2.3 AML Attacks over Enhanced Model

Similar to the first approach, in section 8.2.1, attacks were created with the methods FGSM, Carlini & Wagner, Adversarial Patch, and Spatial Transformation Attack only with rotation,

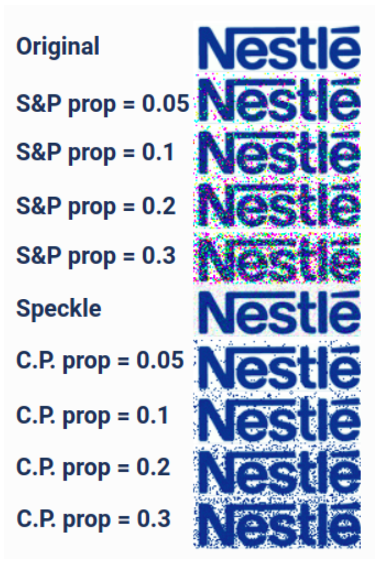
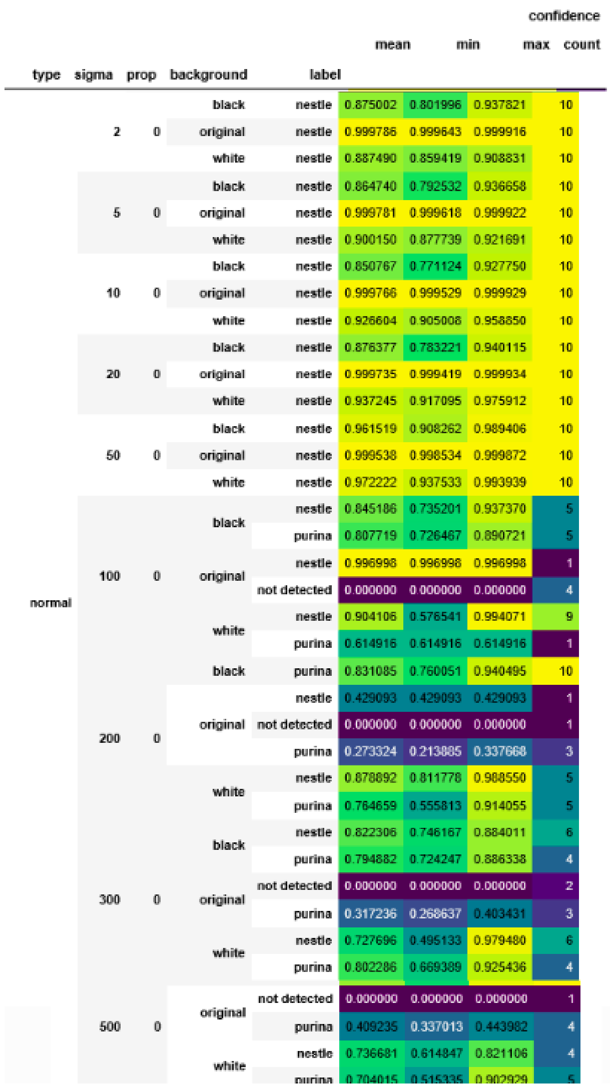


FIGURE 8.5: Salt & Pepper Noise, Speckle and Logo Color Pepper Noise. Values represent the proportion of affected pixels.



					confidence			
					mean	min	max	count
type	sigma	prop	background	label				
normal	2	0	black	nestle	0.875002	0.801996	0.937821	10
			original	nestle	0.999786	0.999643	0.999916	10
			white	nestle	0.887490	0.859419	0.908831	10
	5	0	black	nestle	0.864740	0.792532	0.936658	10
			original	nestle	0.999781	0.999618	0.999922	10
			white	nestle	0.900150	0.877739	0.921691	10
	10	0	black	nestle	0.850767	0.771124	0.927750	10
			original	nestle	0.999766	0.999529	0.999929	10
			white	nestle	0.926604	0.905008	0.958850	10
	20	0	black	nestle	0.876377	0.783221	0.940115	10
			original	nestle	0.999735	0.999419	0.999934	10
			white	nestle	0.937245	0.917095	0.975912	10
	50	0	black	nestle	0.961519	0.908262	0.989406	10
			original	nestle	0.999538	0.998534	0.999872	10
			white	nestle	0.972222	0.937533	0.993939	10
	100	0	black	nestle	0.845186	0.735201	0.937370	5
				purina	0.807719	0.726467	0.890721	5
			original	nestle	0.996998	0.996998	0.996998	1
				not detected	0.000000	0.000000	0.000000	4
			white	nestle	0.904106	0.576541	0.994071	9
				purina	0.614916	0.614916	0.614916	1
	200	0	black	purina	0.831085	0.760051	0.940495	10
			original	nestle	0.429093	0.429093	0.429093	1
				not detected	0.000000	0.000000	0.000000	1
				purina	0.273324	0.213885	0.337668	3
			white	nestle	0.878892	0.811778	0.988550	5
				purina	0.764659	0.555813	0.914055	5
	300	0	black	nestle	0.822306	0.746167	0.884011	6
				purina	0.794882	0.724247	0.886338	4
			original	not detected	0.000000	0.000000	0.000000	2
				purina	0.317236	0.268637	0.403431	3
			white	nestle	0.727696	0.495133	0.979480	6
				purina	0.802286	0.669389	0.925436	4
	500	0	original	not detected	0.000000	0.000000	0.000000	1
				purina	0.409235	0.337013	0.443982	4
			white	nestle	0.736681	0.614847	0.821106	4
				purina	0.704015	0.515335	0.902929	5

FIGURE 8.6: Gaussian Noise Results Example. For each random-noise technique, variation and background 10 different experiments were performed, analyzing the the confidence statistics for each label predicted.

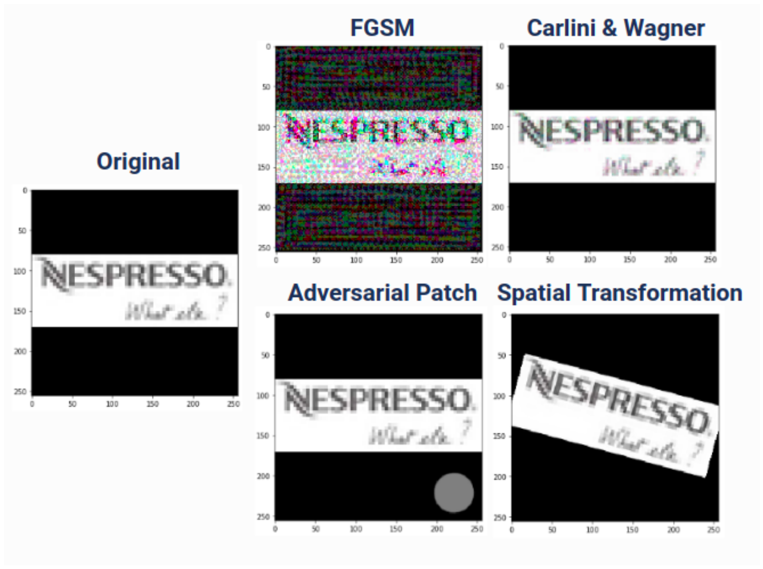


FIGURE 8.7: AML Attacks Example. FGSM, Carlini & Wagner, Adversarial Patch and Spatial Transformation attacks performed over original image.

but this time on the enhanced model. A test set was used as a basis for proposing adversarial examples and evaluating their effectiveness in fooling the model along with the required perturbation.

Each attack was tested for each image with one version using black padding and another white padding, and repeated to reduce noise in the results due to random aspects of the algorithms. An example can be seen in Figure 8.7.

8.2.3.1 Results and Conclusions

Each attack was tested for each image with one version using black padding and another white padding, and repeated to reduce noise in the results due to random aspects of the algorithms.

It was found that **FGSM** modified the pixels of the images by an average of 16% (considering a scale of 256), and 83% of the examples were classified incorrectly by the model. In the attack, an epsilon of 0.3 was used, parameter explained in section 5.4.1 and used in Equation 5.2).

On the other hand, **Carlini & Wagner**, with only 0.1% of pixel changes, made the model misclassify 38% of the cases. For these attacks, an alpha of 0.3 was used, with a maximum of 10 iterations, a learning rate of 0.01, and confidence of 0. The confidence in this case of the ART library implementation determines the gap that was considered as κ in section 5.4.3 in Equation 5.5.

In the case of **Adversarial Patch**, only 3% of the images were affected, despite using a relatively large patch of 0.2% of the image scale. Finally, **Spatial Transformation** using only rotations (no translations) affected 12% of the images, allowing it to iterate 10 different angles with maximum rotations in each direction of 45 degrees.

Each of these analyses was **repeated for each class independently**, and it was found that although the model has a good performance on each of the original classes, **it is susceptible to be attacked more in certain classes than in others.** Additionally, it was possible to analyze the distribution of the changes in the classification, among **which it was possible to detect which classes are more easily confused by the model, and also which classes are easier to convert into the other/background class, which is probably the worst scenario at the business level.**

Also, visually, it was interesting to find specific patterns repeated in the Adversarial Patch attacks covering the same area of certain logos, which could be analyzed as a possible critical point of a logo. There are still some cases in which the disturbance is made on the padding area, which could imply some kind of Overinterpretation. Additionally, certain images are selected to **visualize the proposed disturbance to search for more interesting patterns.**

8.2.4 Substitute Models Variability and Attack Transferability

Finally, a Grey-box scenario is analyzed in which an adversary does not have direct access to the model but can guess about the data and type of architecture used to train the CNN.

This setting is considered quite likely in many situations where public data is available that is closely aligned with the data in the model to be attacked. Additionally, for certain applications such as in images, similar state-of-the-art architectures are likely used with the same frameworks, so it may be easy to create a substitute model that is close enough to the real model. As explained in part in section 3, (Su et al., 2018) analyzed the ease of transferability of attacks made with one architecture to another. For example, the ease of transferability between CNNs with similar architectures was demonstrated, among other empirically analyzed patterns.

For this reason, **tests are performed by training three different models (substitute models) but following the same methodology of the enhanced model explained previously.** Considering that the training process has a random component, each of these models will have different final parameters. Each of these models was analyzed at the performance level, finding that at a general level, they have similar performance, but each one seems to have different abilities to predict different classes correctly.

Training several models following the same process is interesting to be able to **analyze the variability that may exist in the performance of a model due to purely random aspects during the training process.** It was interesting first to estimate the **distribution of the substitute models predictions for each of the text examples.** With this distribution, it was first possible to analyze from the data point of view which cases seem challenging to be correctly categorized by the three models, which ones are correctly by the three models, and then the intermediate cases. From this, it is possible to know that a case that is always wrongly categorized represents a lack of capacity of the model given the used training data or even a lack of capacity from the architecture selected. The correctly classified images may provide confidence, and finally, the intermediate cases show the possible biases that may occur randomly in the training process, even if they all have the same bases.

After analyzing the different models, adversarial examples are created with the test set using the aforementioned methods: FGSM, Carlini & Wagner, Adversarial Patch, and Spatial Transformations. In this, it is possible having adversarial examples for a combination of substitute model, attack method, and test example. Thus, for the 3 substitute models with 4 types of attacks, there are 12 sets of adversarial examples of the size of the test data set. Each of these sets is tested on the 3 models to analyze the **transferability of the attacks, for which the ground truth of the original image, the classification of the base model with which it was built, and the classification of the analyzed substitute model are simultaneously contrasted.**

This analysis is shown in more detail in section 8.3 when looking at transferability between original substitute models, but also between models with defenses.

8.3 Attacks and Defenses Effectiveness Methodology

Having already analyzed the model in terms of attacks, and having found prospects for improvement, it was decided to apply a defense specifically designed to make the model more robust in the face of adversarial attacks. After analyzing in detail all the state-of-the-art possibilities to reinforce evasion attacks in section 3, it was decided to start with a technique with great potential by applying Adversarial Training by means of Adversarial Regularization. It is recommended to read in detail section 3.3.3 to understand the motivation behind this technique.

Although it is relatively easy to apply this defense on the model having the necessary tools, it is very important to follow a proper methodology to measure its impact on the model in terms of performance and adversarial robustness. Consequently, the developed methodology is explained below, which seeks to mitigate different biases found in state-of-the-art, allowing isolating the effects and measuring the impact of the defense on the model in a complete way. These biases in the metrics are explained theoretically in section 3.4, which are recommended to be read in order to understand the basis of the considerations of this methodology.

This methodology will be explained step by step, describing how defenses are usually evaluated in many state-of-the-art papers, explaining their disadvantages and biases, and finally explaining the methodology used to mitigate these problems. his is presented descriptively and graphically to facilitate the understanding of the procedure, the relationship between each of the objects created, and finally, to understand their parameters and dependencies.

8.3.1 Conventions

This section begins by presenting some of the conventions used in order to facilitate the descriptive and graphic explanations. Figure 8.8 shows the conventions used for each of the icons of the visual representations. These are explained in the order presented below:

- **Standard Training Process:** Standard Training Process minimizing the loss function for high performance (conventional Cross-entropy loss function for classification models). Requires a set of training data (X) and the ground truth labels (Y), having as output a Standard Model.
- **Standard Model:** Model created with a Standard Training Process.

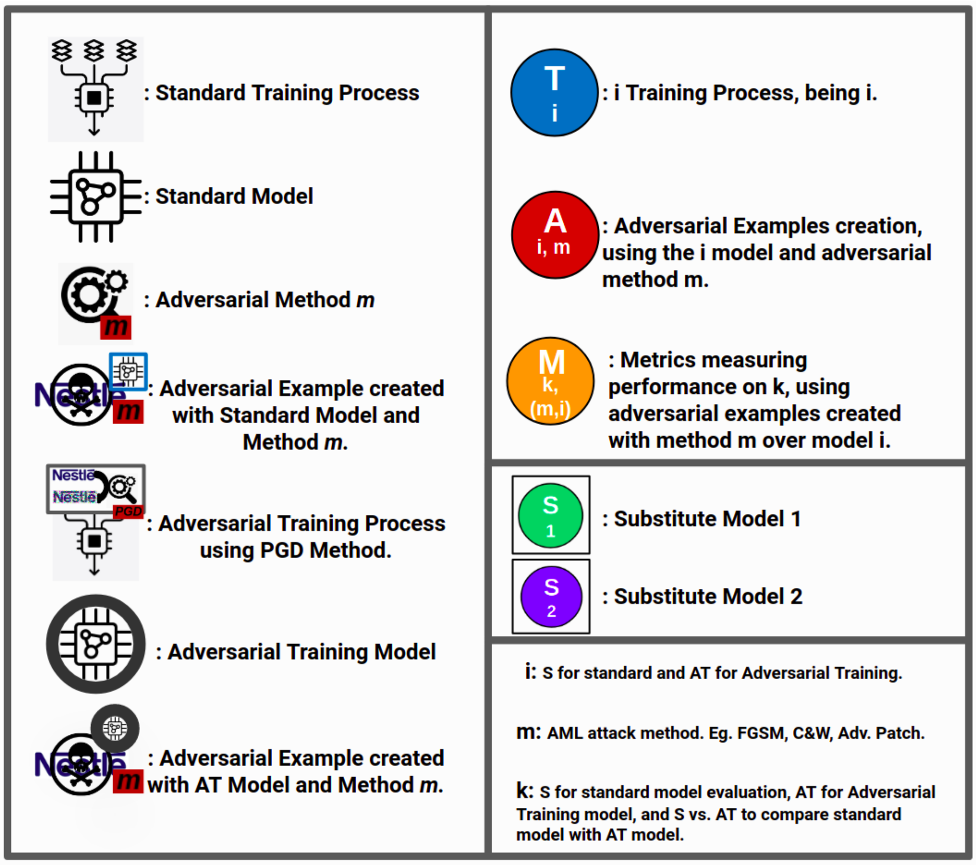

: Standard Training Process
: Standard Model
: Adversarial Method *m*
m
: Adversarial Example created with Standard Model and Method *m*.
: Adversarial Training Process using PGD Method.
PGD
: Adversarial Training Model
: Adversarial Example created with AT Model and Method *m*.
T i
: i Training Process, being i.
A i, m
: Adversarial Examples creation, using the i model and adversarial method m.
M k, (m,i)
: Metrics measuring performance on k, using adversarial examples created with method m over model i.
S 1
: Substitute Model 1
S 2
: Substitute Model 2
i: S for standard and AT for Adversarial Training.
m: AML attack method. Eg. FGSM, C&W, Adv. Patch.
k: S for standard model evaluation, AT for Adversarial Training model, and S vs. AT to compare standard model with AT model.

FIGURE 8.8: Representations Icon Convention.

- **Adversarial Method:** Adversarial examples creation process, using the adversarial method m and applying it to a test data set by optimizing the attacks with a base model. The test data set and model are used as input.
- **Adversarial Example created with Standard Model and Attack Method m:** It is important to notice that each adversarial example should be represented considering the original data example, the attack method m but also the model used, which in this case is the standard model, represented with the blue box in the upper right corner from the icon.
- **Adversarial Training Process using PGD method:** Process to retrain (continue training) a standard model using adversarial regularization, using the traditional PGD attack method to solve the inner maximization problem. Haves as input the standard model and the training data set.
- **Adversarial Training Model**: Model created as output of the Adversarial Training Process.
- **Adversarial Example created with AT Model and Attack Method m:** It is important to notice that each adversarial example should be represented considering the original data example, the attack method m but also the model used, which in this case is the AT model, represented with the AT model icon in the upper right corner from the icon.

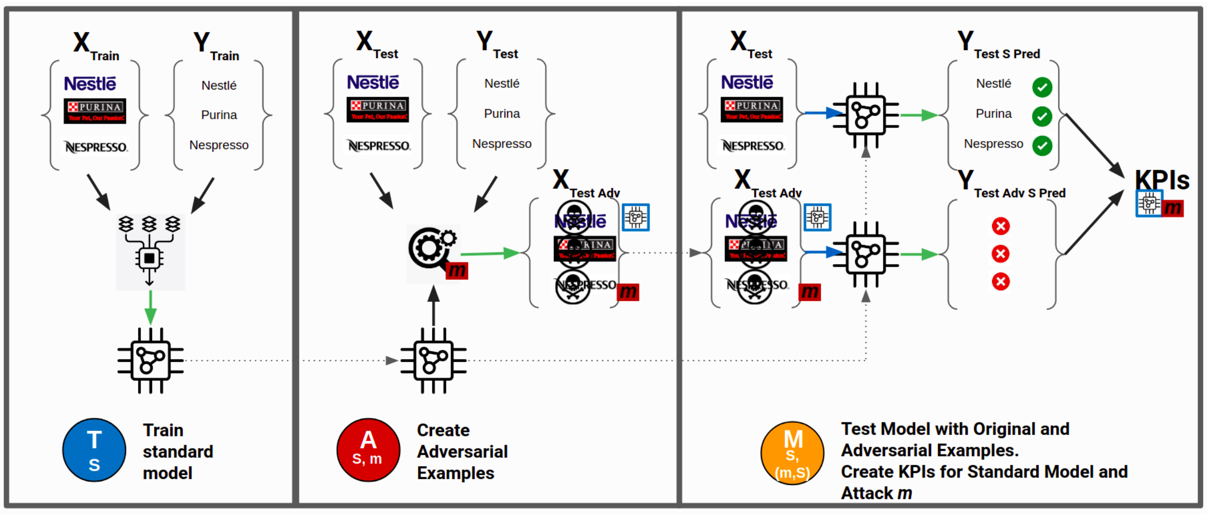


FIGURE 8.9: Standard Attack Effectiveness Measurement Process. After training the model, the same model is used to create adversarial examples based on a test set. Comparing predictions between the test set, and the adversarial test set the attack effectiveness over the model can be measured.

Now considering the processes, the training processes are shown in blue, which may consider Standard or Adversarial Training processes. The red circle is used in processes creating adversarial examples using the adversarial method *m* and optimizing with a model i (standard or Adversarial Training model). Finally, in orange are presented processes for the creation of KPIs, being possible to create metrics to measure the impact of certain adversarial examples (created with the adversarial method *m* and optimizing with model *i*) over a model compared to the ground truth, or to compare the performance of two models (standard and AT model).

Finally, in order to further explain Figure 8.11, there are two symbols to represent the substitute model 1 and 2. This mark is used to differentiate different substitute models appearing in the same flowchart.

8.3.2 Traditional Attacks and Defenses Effectiveness Methodology

In order to present the proposed methodology, it is important first to recall some traditional processes used to test models against adversarial attacks and measure the improvements achieved with a defense method.

Figure 8.9 shows how to measure the robustness of a model to adversarial evasion attacks, as was done in the previous section. It can be seen that after training the model with process T_S, this model is used together with method m and a test data set to create the adversarial test data set in $A_{S,\, m}$. Finally, in $M_{S,\, (m,\, S)}$ how the predictions of the model change with attacks created based on itself are analyzed. It is important to note that the adversarial examples have a model and a method associated with them. Also, the KPIs measure the adversarial robustness of the standard model against the attack method m, optimized with the same standard model.

For evaluating a defense, it is usual to find methodologies following the process presented in Figure 8.10. Starting with T_{AT}, the standard model is taken, and the defense (in this case, Adversarial Training) is applied to it using the same training data. Here it is important

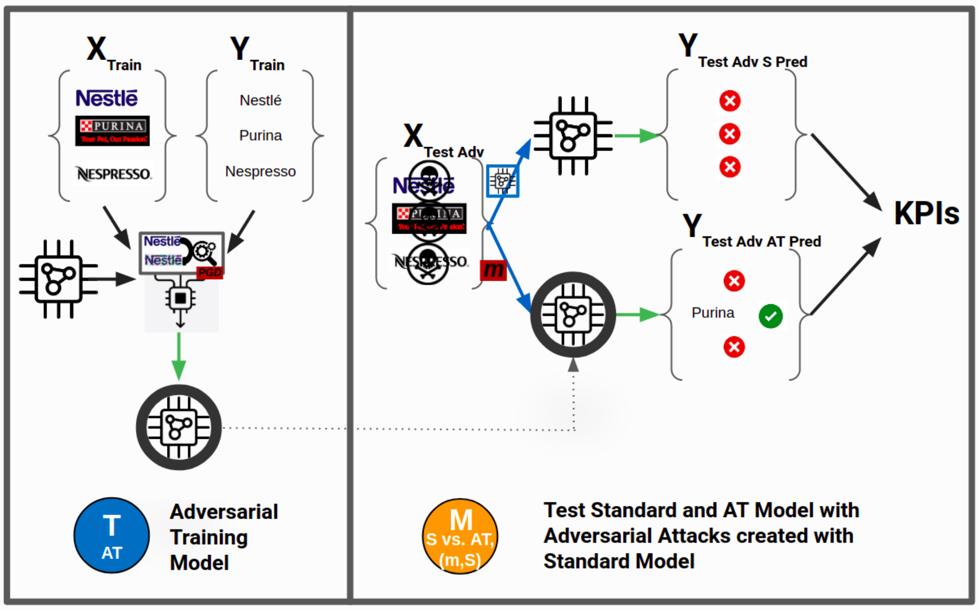


FIGURE 8.10: Standard Defense Effectiveness Measurement Process. After training the standard model and the Adversarial Training model based on it, adversarial examples are created using the standard model and a test data set. Comparing predictions between the standard model and the Adversarial Training model using the created adversarial examples, the effectiveness of the Defense can be measured.

to consider that the training approach, as shown in the representation of the Adversarial Training Process icon (Figure 8.8, automatically creates the adversarial examples used in Adversarial Regularization in the process, using PGD.

In the evaluation of the defense, the previously created attacks using an m-method and standard model are used, as it was done in $A_{S,\, m}$ in Figure 8.9, and evaluated on the standard and AT model in $M_{S \text{ vs. AT, } (m,\, S)}$ in Figure 8.10. In this case, it is clear that a White-box attack designed to attack the standard model is used as a basis for comparing two models, among which the same standard model is included. **This is a biased measurement since the attacks will tend to affect the standard model more by definition of the adversarial test set used. The other model will tend to have the advantage only for not sharing the same parameters**, as it was also explained in **section 3.4.1.2** (Bias to Attacks Designed Using the Original Model). For this reason, a **methodology using adversarial test data sets that are impartial to both models evaluated should be considered in order not to favor either of the two models.**

To the previous bias must also be added a bias in the metrics used to measure the Standard model versus the AT model, which are explained in **section 3.4.1.3 (Bias to Only Originally Effective Attacks)**, and will be understood in greater detail below.

8.3.3 Proposed Defense Effectiveness and Attack Transferability Methodology

The purpose of this methodology is to create KPIs to measure different aspects of the effectiveness of attacks (from an attacker's point of view) and the effectiveness of using Adversarial Training as a defense against them.

8.3.3.1 Considerations

Having identified the biases of other methodologies, the most relevant considerations are presented:

1. To reduce the **bias to only originally effective attacks** (explained in section 3.4.1.3, and in 8.3.2), several models trained with the same hyperparameters and training methodology (**substitute models**) were created, in order to create adversarial examples. Using adversarial examples created with a substitute model (external model) makes more sense to compare the standard model with the model with defense.

2. The predictions of the original data for the standard model and the model with defense should be analyzed beforehand, because before testing an adversarial example, both predictions may differ. For example, it may happen that the model with defense incorrectly predicts an original image and therefore incorrectly predicts it with an adversarial example created with the same image. It is necessary to analyze when having incorrect classification if it is **attributable to the attack method or the model's performance** to any variation from the same image.

3. Not only the vulnerabilities of the original model should be considered as a basis to see if the model with defense solves them, but also analyze if the model with defense has **new vulnerabilities that the original model does not have.** To achieve the above, **all candidate adversarial examples should be considered to compare the original model and model with defense.** This will allow to evaluate not only which attacks that affect the original model also affect the model with defense, but also to identify attacks that did not affect the original model but did affect the model with defense.

8.3.3.2 Methodology and Results Collection

The methodology created can be analyzed in a more user-friendly visual form in Figure 8.11, which shows in a simplified form how to create adversarial examples and measure defenses with two substitute models.

It can be seen that two substitute models are trained, one with a green circle and the other with a purple circle representing them. For each substitute model, an original model is first created in $\mathrm{T_s}$, followed by the Adversarial Training model in $\mathrm{T_{AT}}$, following the same logic as shown in Figure 8.9 and Figure 8.10. Then for each type of attack, adversarial examples are created using a test set and the original model. As a result of the previous 3 processes for each substitute model, an original model, a model with defense, and a set of adversarial examples for each attack method m are created.

Subsequently, the adversarial examples are exchanged between the two surrogate models to use an external and unbiased adversarial test set (showed with thicker color arrows). In $\mathrm{M_{S\ vs.\ AT,\ m}}$ compares the robustness to adversarial examples of each original substitute model with its corresponding model with defense for each adversarial test set of each attack method m.

This process is applied with 3 substitute models and with 5 types of attacks following the next steps:

1. 3 standard models are trained with the same training process of the enhanced model explained previously. Creating models M_{S1}, M_{S2}, M_{S3}.

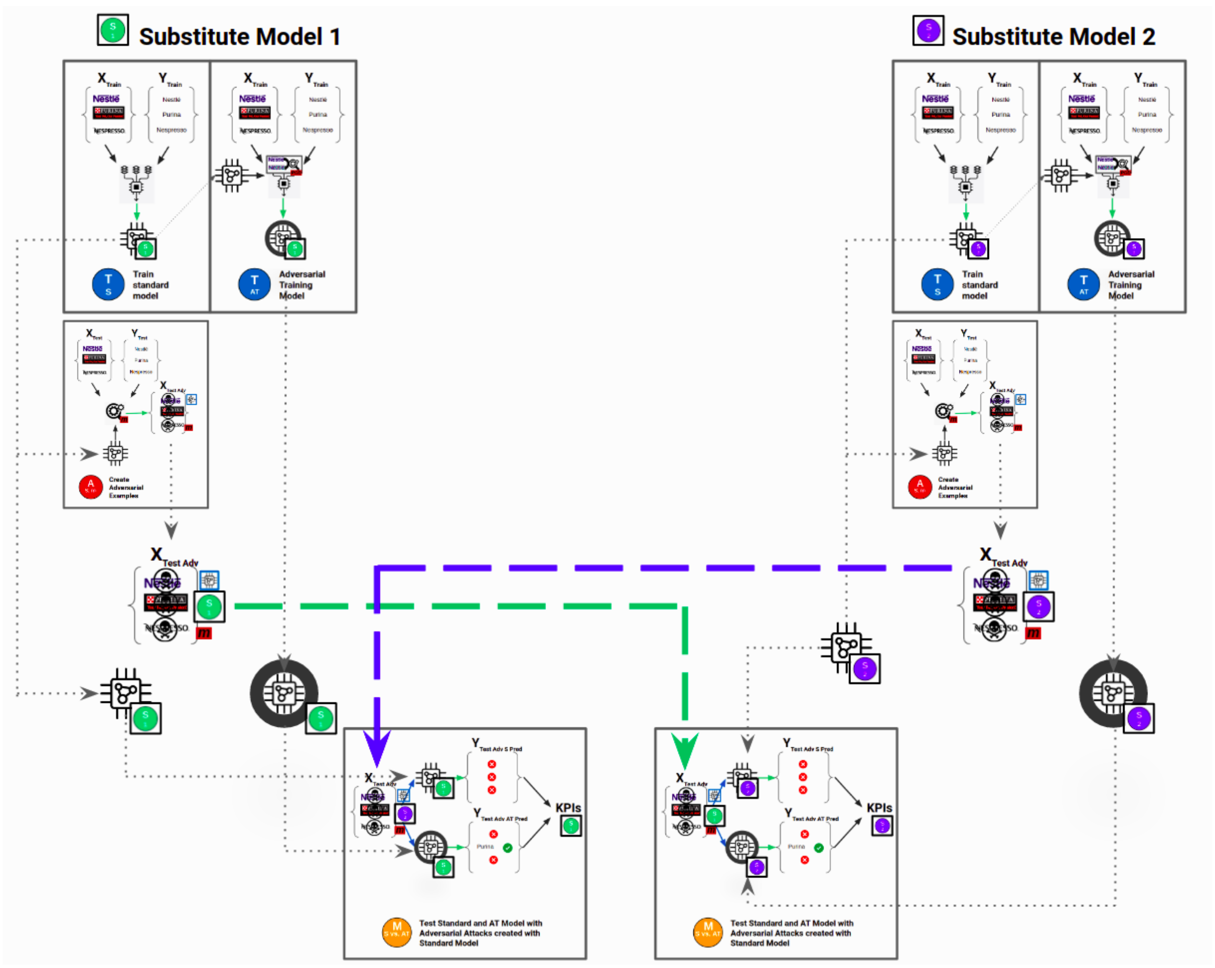

Substitute Model 1
Substitute Model 2
Train standard model
Adversarial Training Model
Create Adversarial Examples
Test Standard and AT Model with Adversarial Attacks created with Standard Model
KPIs

FIGURE 8.11: Proposed Defense Effectiveness Measurement Process, based on Substitute models. Substitute models are trained, together with their relative Adversarial Training models. Adversarial examples are created using the standard model and a test data set. Comparing predictions between the standard model and the Adversarial Training model using the created adversarial examples, the effectiveness of the Defense can be measured.

2. For each of the models re-training is done with the defense (Adversarial Training), in order to create 3 models with defense: M_{AT1}, M_{AT2}, M_{AT3}.
3. Classification is made on the original test dataset with each of the 6 models, in order to create 6 prediction vectors: $y_{original,S1}$, $y_{original,S1}$,..., $y_{original,AT3}$.
4. For each of the 6 models (the 3 standard models and their respective 3 with AT) adversarial test sets are created with the 5 attacks: FGSM, PGD, Carlini & Wagner, Adversarial Patch and Spatial Transformation. This produces 30 adversarial test data sets: $X_{S1,FGSM}$, $X_{S1,PGD}$,..., $X_{AT3,AP}$, $X_{AT3,ST}$, being the first sub-index the model used for its creation and the second sub-index the attack method.
5. Classification is performed with each of the 6 models on the 30 adversarial test data sets created, for a total of 180 prediction sets, creating vectors as for example: $y_{S1,FGSM,AT2}$, being the first two sub-indexes the same from the adversarial test data set used, and the third sub-index the model used to make the predictions.

8.3.3.3 Evaluation Methodology (KPIs)

Already having the results of the experiments, different KPIs are estimated for different purposes, among which are:

1. **Original test set accuracy KPIs:** By creating traditional performance statistics for each of the substitute models, both the original and the models with defense. The confusion matrix for each of the 6 models can be compared to identify biases in each model, and a similar analysis can be done for easy and challenging cases as was done in Section 8.2.4. In this case it is interesting to see how the performance of each substitute model changes with its own re-training with Adversarial Training.
2. **Attack creation effectiveness:** Similar to section X, it is analyzed which attacks were effective on which examples and classes at the time of their creation with their base model. Here each attack is still measured with its source model to see as a baseline whether the attacks achieved their optimization goal and also to be able to measure the perturbations necessary to achieve it. Different original models can be compared to see if there is variability in robustness between substitute models. It is also **useful to assess whether it is more difficult for the attack methods to create adversarial examples that achieve their goal with the models with defense when compared to the original models.** This can be done by comparing the rate of effective adversarial examples per class, but also the required perturbations. This analysis can be complemented with the visual representation technique explained later.
3. **Adversarial examples transferability:** The effectiveness of transferring adversarial examples from one substitute model to another can be analyzed, analyzing not only the cases that are effective in their base model, but also those that are not. With this information, incremental analyses can be made to see cases in which an attack originally ineffective in its base model is effective in another model, and also among the effective cases with its base model, which are transferred and which are not. **It is also important in this case to consider the possible errors in the model classification with the original test dataset, since in these cases an adversarial example is not necessary to make the model perform incorrectly.**

4. **Defense robustness:** For each substitute model, the predictions of the standard model and the model with defense are compared, as explained previously with Figure 8.11. In this analysis it is again important to consider not only the ground truth, but also the predictions made by the model on the original dataset. This process is similar to the previous one, so that an incremental analysis can be performed, which will be explained in more detail in the results.

8.3.4 Adversarial Regularization

The previous methodology is applied using Adversarial Training as a defense by means of Adversarial Regularization. The Neural Structured Learning library (NSL) is used with the Adversarial Regularization approach explained in section 5.5.1, using the same default parameters, with L_2 norm with a multiplicative factor of 0.2 for the adversarial loss, and Cross-entropy loss for the performance loss. All other training hyperparameters are the same as those used in the enhanced model, in order to simply use AT to continue training the original model to create better representations.

8.3.4.1 Results and Conclusions

For security reasons, to avoid going into more detail on the exact results obtained, each of the considerations made for each of the KPIs in section 8.3.3.3 are explained below.

8.3.4.1.1 Original Test Set Accuracy KPIs

For the Original test set accuracy KPIs (item 1), confusion matrices were created for each model, and the accuracy, recall, and F1-score were compared for each class. Additionally, from the errors, it was analyzed in which class the errors were categorized, and the images corresponding to the errors were visualized for each of the models.

In this process, the performance of the original substitute models versus the models with defense was compared. Changes in performance metrics were analyzed, and interesting patterns were found. In general, it was found that the models with defense have slightly higher accuracy, but when analyzing by classes, it was found that this is achieved by increasing the performance more in certain classes than others, or even by decreasing it in one class.

Another conclusion showed that the models with defense, although not all of them improved their accuracy considerably with the *other/background* class, this class had a precision of 100% in all cases. This implies that although the model has errors, it **has no false negatives by never classifying the logos of interest in the no logo class.** From a business point of view, this is positive, being **more conservative** without having false negatives (which are the worst-case scenario), without affecting or even in certain scenarios reducing false positives.

Additionally, analyzing the 3 original models versus the 3 models with defense using aggregate statistics, finding, in general, better performance in the models with defense. Analyzing the variability of the models with defense, it was found that the number of images that have errors in the 3 models is reduced when compared to the original models, which **may suggest that if used together, they could further reduce errors.**

8.3.4.1.2 Attack Creation Effectiveness

For the second type of KPIs, the effectiveness of each adversarial example was measured in the same model with which it was created. In addition, the perturbations required to do so were analyzed. Doing so shows whether there are different difficulties in creating attacks in the standard models versus the models with defense. **Assuming that it will be more difficult for a more robust model to create adversarial examples, either failing or requiring stronger perturbations.**

It was found that it was more difficult for all methods to create adversarial examples in the AT models, as the number of effective cases was reduced. By opening the analysis by class to avoid Simpson's paradox, it was found that not all classes improve in the same way, which may help to think of ideas to **make certain classes more robust or even use different thresholds depending on the class.** It was interesting to note that larger perturbations were required for certain classes, which should be appended to the previous analysis.

Finally, it was interesting to analyze certain cases visually, to get an idea of how aggressive the perturbations are and how easily a user can identify them.

8.3.4.1.3 Adversarial Examples Transferability and Defense Robustness

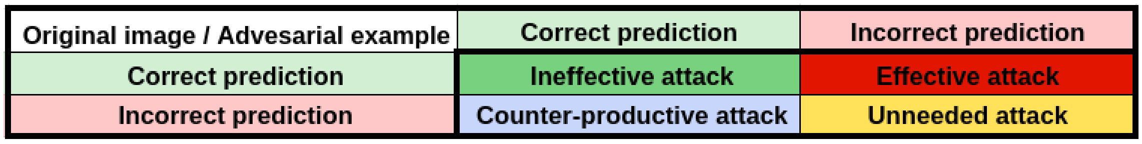
As explained previously, it is possible to analyze which attacks are transferred from one model to another, and it is important to analyze all scenarios. It should be analyzed if an effective attack in the base model is transferred to another model, but also if an ineffective attack is transferred as well, in order to evaluate new weaknesses and not only confirm old weaknesses.

In this process, the robustness of the defense is directly measured by comparing with external adversarial examples the performance of a standard model and its version with AT as in Figure 8.11. For each of the comparisons made, which vary according to the method and the external substitute model, all possible scenarios were analyzed, making a count of each case, and creating a matrix to analyze what this represents.

First, in Figure 8.12, a matrix comparing the predictions of the model on the original image (rows) versus the image with adversarial attack (columns) can be seen. The first row shows the original examples correctly classified by model. The top left position is the case in which the model correctly predicts the original example and also the adversarial example created, which represents an ineffective attack (in green color). The case on the top right is the case in which the model correctly predicts the original example but not the adversarial example, which is an effective attack (in red color).

On the other side, less intuitive but important for analyzing new weaknesses, is the second row. The bottom left position is the case of a misclassified original example, but correctly classified when the adversarial example is performed, being a counter-productive attack (in gray). Although this case is very uncommon, it does occur on certain occasions and is usually ignored in the state of the art. Finally, there is the case at the bottom right in which the model cannot correctly classify the original image nor its adversarial attack, which makes sense in most cases, being called unneeded attack (in yellow).

Having classified each case in one of the 4 categories, the original model and the model with AT results are crossed to evaluate the implication of each scenario in Figure 8.13. As can be



Original image / Advesarial example	Correct prediction	Incorrect prediction
Correct prediction	Ineffective attack	Effective attack
Incorrect prediction	Counter-productive attack	Unneeded attack

FIGURE 8.12: Original Example vs. Adversarial Example Predictions Scenarios Matrix

seen in this table, the first column contains the case for the standard model and the second column for the model with defense, having a total of 16 rows, one for each combination of the 4 cases of both models.

The purpose of this table is to summarize for each scenario (row) whether or not it is better to use the model with defense compared to the standard model, presented in the last column. To do this, **each scenario analyzes whether there is improvement in the natural performance in the fourth column, and whether there is improvement in the adversarial performance in the penultimate column**. Each value compares whether there is improvement in the model with defense with respect to the standard model, so it can have values of: reduction, improvement, similar or unknown.

Additionally, there is a column with a short description of the scenario in the third column. It can be seen that the conclusion in the last column depends on the previous two columns. When it is evident that the model with defense is better than the standard model, there is a Yes in the last column, when it is evident that it is worse a No, when they have the same behavior it is Similar (both models will have the same effect). Finally there are Unknown cases in which one model is better in one aspect while the other in the opposite, or in which it is not possible to conclude for any of the two properties which one is better.

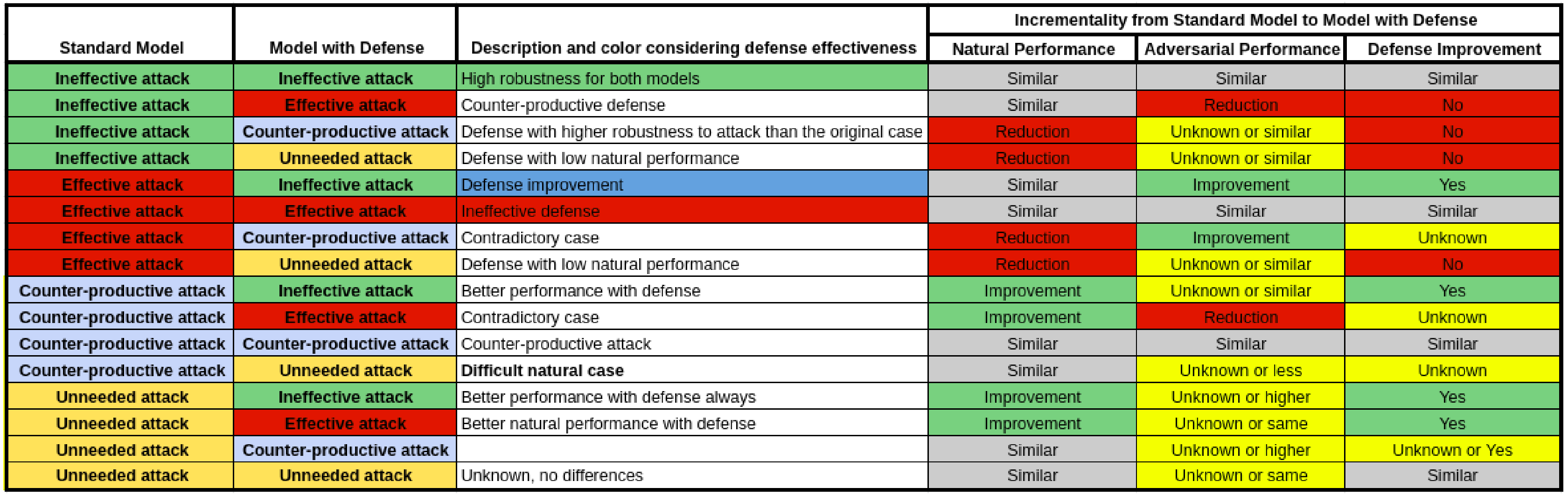
State of the art commonly only uses rows 5 and 6 to analyze the cases that improve in defense, since it takes the adversarial examples that were effective in the standard model and then analyzes which of them are transferred to the model with defense. Although many of the cases fall into one of these scenarios, it is important to consider all of them for analysis. To do this analysis, it is possible to count the cases that fall into each of the 16 scenarios, and contrast their conclusion from the last column.

This procedure was done for each of the attacks and each of the substitute models, and very interesting results were found. For FGSM, for example, it was found that few effective attacks are transferred from the standard to the defense, but there are also cases that did not affect the standard model but now affect the model with defense, which in some cases adds up to more negative than positive cases.

In the trained models it was found that in most attack methods, **there is an advantage of the model with defense, being favorable its use to have a higher adversarial robustness than the standard model.**

8.3.4.2 Representations Visualization and Analysis

Finally, to complement the previous approaches, an analysis of the representation of the examples by each of the models is made by visualizing the 128-dimensional embedding created by them as output of the last dense layer of the CNN. The objective is to see the effect



Standard Model	Model with Defense	Description and color considering defense effectiveness	Incrementality from Standard Model to Model with Defense		
			Natural Performance	Adversarial Performance	Defense Improvement
Ineffective attack	**Ineffective attack**	High robustness for both models	Similar	Similar	Similar
Ineffective attack	**Effective attack**	Counter-productive defense	Similar	Reduction	No
Ineffective attack	**Counter-productive attack**	Defense with higher robustness to attack than the original case	Reduction	Unknown or similar	No
Ineffective attack	**Unneeded attack**	Defense with low natural performance	Reduction	Unknown or similar	No
Effective attack	**Ineffective attack**	Defense improvement	Similar	Improvement	Yes
Effective attack	**Effective attack**	Ineffective defense	Similar	Similar	Similar
Effective attack	**Counter-productive attack**	Contradictory case	Reduction	Improvement	Unknown
Effective attack	**Unneeded attack**	Defense with low natural performance	Reduction	Unknown or similar	No
Counter-productive attack	**Ineffective attack**	Better performance with defense	Improvement	Unknown or similar	Yes
Counter-productive attack	**Effective attack**	Contradictory case	Improvement	Reduction	Unknown
Counter-productive attack	**Counter-productive attack**	Counter-productive attack	Similar	Similar	Similar
Counter-productive attack	**Unneeded attack**	**Difficult natural case**	Similar	Unknown or less	Unknown
Unneeded attack	**Ineffective attack**	Better performance with defense always	Improvement	Unknown or higher	Yes
Unneeded attack	**Effective attack**	Better natural performance with defense	Improvement	Unknown or same	Yes
Unneeded attack	**Counter-productive attack**		Similar	Unknown or higher	Unknown or Yes
Unneeded attack	**Unneeded attack**	Unknown, no differences	Similar	Unknown or same	Similar

FIGURE 8.13: Original Example vs. Adversarial Example Predictions Scenarios Matrix

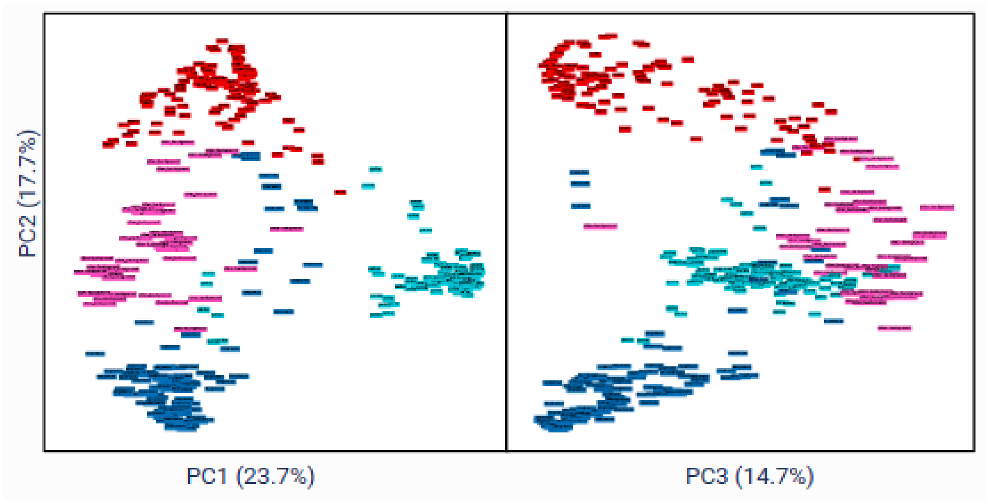


FIGURE 8.14: Standard Model Representation Visualization using PCA.

of using Adversarial Regularization to create better representations, which can be distinguished more smoothly and clearly.

For these visualizations, the embedding created by the neural network before the soft-max layer was extracted and used as input data of 128 dimensions in the TensorBoard platform to apply dimensionality reduction techniques and get an idea of how the examples are distributed. This is explained in section 3.4.2.

Figure 8.14 shows the representation of the standard model, with the projection of the examples in the principal component 1 and 2 on the left, and the principal component 2 and 3 on the right. For each of the components, the percentage of the variance of the data that is explained can be seen on the axis. For each of the components, the percentage of the data variance explained can be seen on the axis. Finally, it can be seen in different colors each of the classes of the logos to be classified, being in red Nestlé, dark blue Nespresso, pink Purina, and light blue other/background.

In the standard model representation, the classes are generally separable, although certain cases are slightly mixed at the boundaries. On the other hand, in Figure 8.15, we can see the same graph but for the model with Adversarial Regularization, in which it can be observed that the classes seem to separate in a more organized way.

It is interesting to note that for the model with Adversarial Regularization (Figure 8.15) **an elongated distribution is created for each of the classes under study (Nestlé, Nespresso and Purina), which all converge in the pink region in the other/background class.** This may make sense since the model groups the difficult images with higher variability in this class, but correctly separates the more structured patterns from the other classes under study. **This may give an idea of the great ability of the Adversarial Regularization model to create better representations than the standard model.**

Also, in Figure 8.16, the location of the adversarial examples created in the representation space can be observed. In this Figure, first, the original examples are shown in black and the adversarial examples in red, in the second graph only the original examples, in the third graph only the adversarial examples, and in the fourth graph, the adversarial examples are separated into ineffective cases (in black) and effective cases (in red).

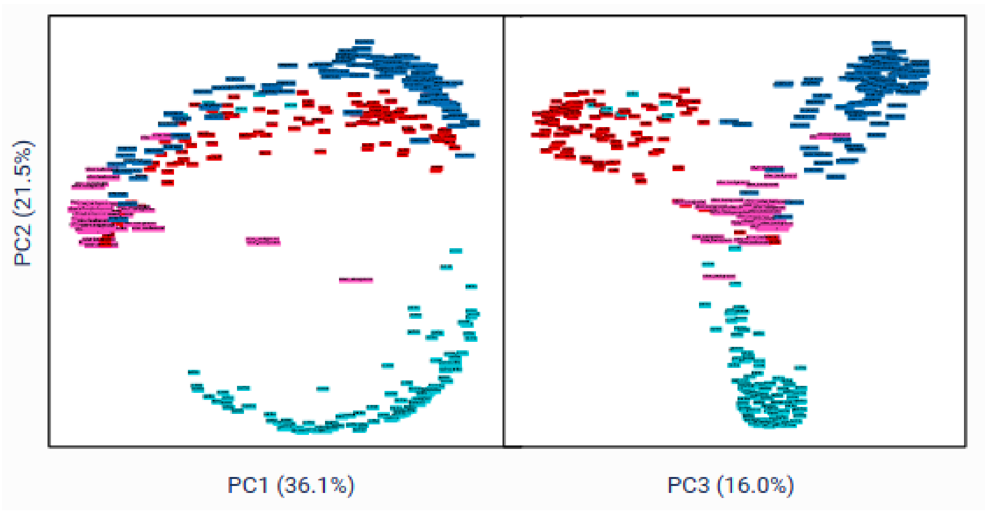


FIGURE 8.15: Adversarial Training Model Representation Visualization using PCA.

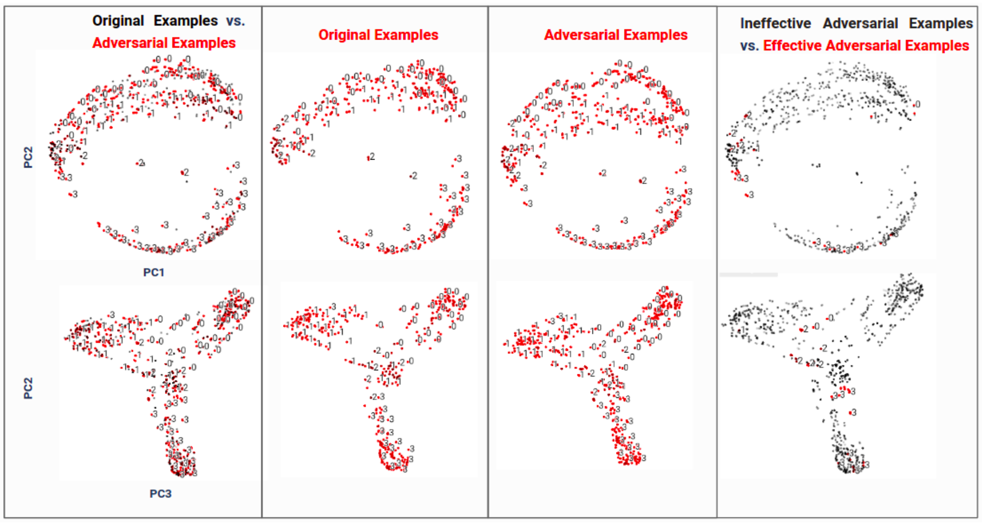


FIGURE 8.16: Adversarial Training Model Representation Visualization using PCA. Showing original examples, and effective and ineffective adversarial examples created with the Carlini & Wagner method.

This is useful to understand how the attacks affect the representation of an example and to see if it makes sense to use techniques to detect possible adversarial examples through anomaly analysis, as explained in section 3.3.2. This should be analyzed since, as explained at the beginning of section 3.1, the property that adversarial examples usually exhibit of a different distribution from the real cases, is not always fulfilled in Computer Vision (Xi, 2020).

It can also be observed in Figure 8.16 in the last graph the location of the effective attacks in red that in certain cases are in the other/background class zone, or concentrated in other locations, which helps to make a more detailed analysis of this type of images with higher weaknesses.

Finally, these visualizations are useful to directly observe the change in the representation of an example when being perturbed with an attack, as shown in Figure 8.17. In this, it can

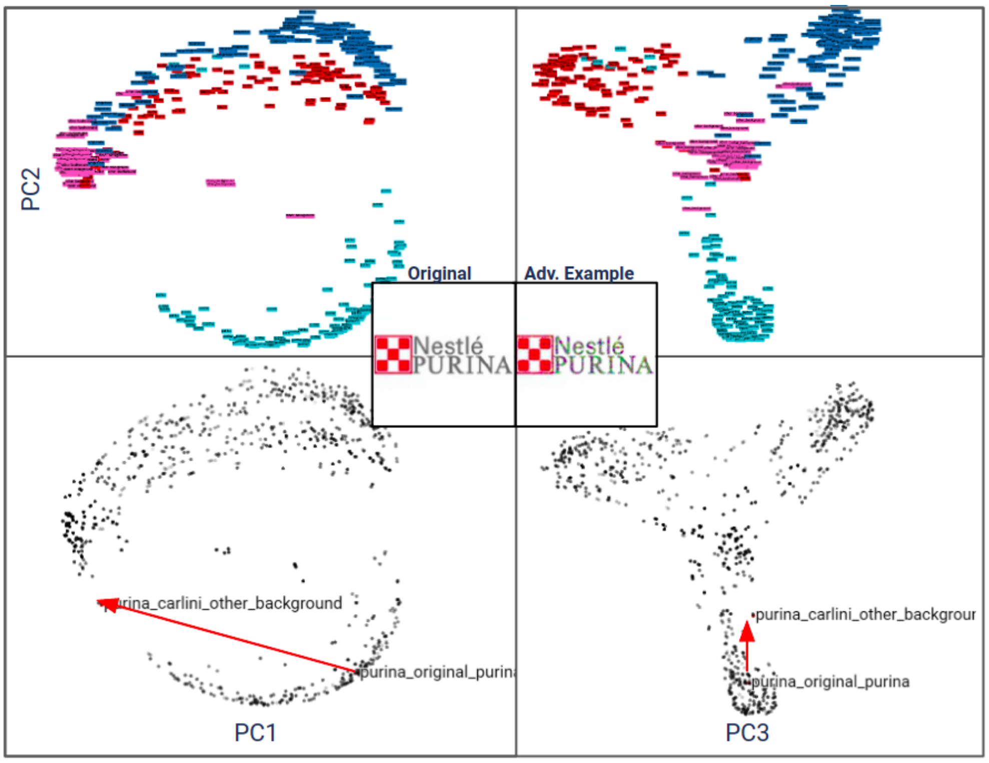


FIGURE 8.17: Example Movement over the Representation Manifold when Perturbed using Carlini & Wagner Method.

be seen that the image was originally classified as Purina (in light blue), after the hardly noticeable perturbation with Carlini & Wagner manages to create an adversarial example that moves into the other/background zone (pink). This is an example of an **effective evasion case on the model used**, although it is notable that **the adversarial example is located in a zone outside the real data manifold**, having a different distribution. This demonstrates a case where it would have **potential to use some anomaly detection technique** to detect it even though the model with Adversarial Training fails to do it.

8.3.5 Dummy Adversarial Regularization Ensemble

An ensemble model is tested by creating an ensemble model from the trained models. This is done to complement different models with different strengths in order to correctly determine the class of an example under analysis, and to see the potential of the 3 substitute models to have fewer examples with shared errors than the 3 standard models (as explained in section 8.3.4.1).

At a general level, the results are slightly better in the analysis, considering voting of the 3 models to determine the class of an example. However, it is not significant enough and depends on how conservative this analysis is to be done, which could sacrifice false positives for false negatives.

To try to have more considerable results, it is recommended to train more models to evaluate the effect in the presence of more predictors with different strengths and biases. Additionally, as explained in section 3.3.3.3, it is recommended to analyze how to implement *Ensemble*

Diversity (Pang et al., 2019) to directly guarantee in the joint training of the different models accurate predictions but with variability of the remaining values of the output of the soft-max layer.

8.4 Case Study 2 General Conclusions and Recommendations

8.4.1 General Conclusions

This case study showed the **importance of considering all ML best practices** to create a model with good performance and robustness **before applying AML techniques.** In this way, it is possible to really add value with AML techniques, isolating the incremental effect and thus trying to avoid mixing adversarial robustness with poor base performance or in-distribution robustness. It is also important to **previously choose architectures proven to have a good performance-robustness** ratio in state of the art. Being relevant to consider those with lower attack **transferability of attacks from other architectures.**

Additionally, it was important to show that a model's **difficulty in making correct predictions is not necessarily aligned with the difficulties experienced by a human being.** Two observations are created from the previous statement. First, even if very aggressive perturbations to the human eye are made random or hand-crafted, it cannot be guaranteed that they are difficult to predict for the model. Using such cases as an example can lead to erroneous conclusions and a **false sense of adversarial robustness.** Second, it is observed that it is not easy to perform effective adversarial attacks by hand using only small perturbations. This implies that **it is necessary to use optimized AML methods** in order to minimize the perturbations needed to achieve an attack.

Before using defenses, it is important to analyze the adversarial robustness of the base model by performing different attacks with different difficulties and approximations to **detect weaknesses and strengths to compare when applying a defense.** Varying the type of attacks can allow discovering different insights of the model, including attacks that may be more related to overinterpretation, such as rotations made with Spatial Transformation and Adversarial Patch.

Related to the previous point, it should be considered that not only in a White-box scenario, an adversary would be able to create efficient attacks. **High attack transferability can be achieved by creating a Grey-box scenario where the attacker can access similar data and create a substitute model using a similar architecture to the original model.**

To analyze the adversarial robustness of a model, analyses with **different approaches are proposed.** First, it is proposed to analyze the **difficulty of each of the attack methods to create adversarial examples** for each model. This is under the assumption that a model with higher adversarial robustness will reduce the number of effective attacks of an attack method or force it to use more aggressive perturbations. Also, the **embedding** created by the DNN can be analyzed to visualize in a lower dimensionality an approximation of its **representation.** This allows identifying how the classes are separated to **visualize the discrimination potential** and analyzing the location of the examples with efficient adversarial attacks and their **location concerning the real data manifold to consider the potential of using anomaly detection techniques.**

In addition, **a methodology was proposed to allow performing unbiased tests to evaluate the effectiveness of a defense.** Additionally, a complete way of analyzing the results was proposed, which avoids biases explained in section 3.4.1. It avoids using only one type of attack to **avoid the bias of the model strength against certain attacks used in its training process with Adversarial Regularization.** It also avoids the bias to evaluate only originally effective cases in the standard model, to **also evaluate new weaknesses** and make an analysis contemplating all scenarios regarding ground-truth, prediction of the original example, prediction of the adversarial example, and making the combinatorial of these cases between the standard model and the model with defense. This **allows making an incremental analysis to know if there is an improvement in natural performance and adversarial robustness. With both effects and with different attacks it can be decided whether or not to use the defense.**

8.4.2 Adversarial Training Model Conclusions

Performing all the analyses to evaluate the effectiveness of the model defended with Adversarial Regularization based on the enhanced model, it was found that **it has advantages when analyzing it from different perspectives and for different types of attacks.** Additionally, the results were carried out by training the model several times, with several substitute models to analyze the variability of the method.

It was found that the model has a **natural performance that is equal or better than the standard model**, and additionally **reduces false negatives** by having 100% precision for the class where no logo of interest occurs (other/background) in the test set, which is a **conservative case at the business level.**

The attack methods were less effective against the model with Adversarial Training, using the same perturbation intensity, which demonstrates **higher robustness to White-box attacks.** It is important to mention that not for all attacks the improvement was equally considerable.

In the analysis of the robustness of the defense against attacks with substitute models with different attack methods, there were **favorable results when analyzing all the scenarios of differences between natural performance and adversarial robustness.**

Finally, when analyzing the **visualizations of the representations** created by the standard models and the models with Adversarial Training (Adversarial Regularization), it was possible to observe **higher order, greater cohesion between cases of the same class, and a better separation of the different classes.** Additionally, its shape allows to perform anomaly analysis in a simpler way, which can be useful as a complement to find cases outside the real data manifold. These cases can be considered as possible attempts of adversarial attacks, or at least out-of-distribution cases.

8.4.3 Future Recommendations

It is also recommended to evaluate how to make a more robust model by **combining different techniques**, from simple to more complex ones, and considering each stage of the model. Starting with simple improvements, it is proposed to train the model with more **variability of training examples** by applying different filters, random noise, colors, and backgrounds to create a more robust base model. It is also recommended to try a model that uses contrast normalization and perhaps grayscale to be color independent (unless the color is considered a favorable factor for filtering out disturbed logos). Also, to test the model by **the logos in more classes, to differentiate variations of the same logo that are visually very different**, in order to focus the model's efforts to focus on more constant patterns and not group different cases in the same class.

It is proposed to evaluate other defenses with great potential from state of the art, such as the Input Transformation Techniques explained in detail in section 3.3.3.4, in particular ***Barrage of Random Transforms for Adversarially Robust Defense*** (Raff et al., 2019), with some of the recommended 25 transformations applied randomly with the suggested parameters. Finally, it is considered useful to apply **anomaly detection techniques for the representation created by the model with Adversarial Training**, to further reduce the cases of adversarial examples that fall out of the real data manifold.

Chapter 9

Conclusions and Future Work

9.1 General Conclusions

Adversarial Machine Learning is a research branch that is very important to consider in the development of Machine Learning models, in particular in those problems where there is motivation and risk for an attacker to try to breach them. In the particular case of Black-box ML models such as DNNs and certain tree ensemble methods, it is very important to correctly study how to make them as robust as possible to **achieve generalization (standard in-distribution robustness), but also adversarial and out-of-distribution robustness.**

AML is a branch of research that requires considerable dedication at the state-of-the-art research level to effectively apply it in practice and to remain up-to-date. AML is a fairly recent and booming branch, which is on its early stages of research.

The state of the art in AML has been based on a battle between attacks and defenses, with a great variety of effective attacks and low effective defenses, being far from creating defenses that manage to protect models in a generic way. However, this should not be considered a failure, since like any other type of security system, it is difficult to foresee all types of future attacks, of different types and magnitudes, and the defenses created are useful to mitigate the problem and shield the model from a wide variety of attacks.

Additionally, there are no standard metrics or methodologies that allow to directly compare each of the defensive methods that are created, and the defenses are usually very dependent on the specific details of the ML model, the data, the target and the assumed attacks. Therefore, it is difficult to choose the defenses to be implemented and it is of foremost importance to stay up-to-date with the latest research trends on the topic. This additionally makes it **quite critical to understand the ML at hand in depth.**

Therefore, a methodology was followed to consider in detail the difficulties that a ML model may present at a general level, including its susceptibility to adversarial attacks for technical and business reasons, but also relating it directly to problems of overinterpretation or low robustness to previously unseen cases. This allowed to qualitatively define how to focus the improvements to the models, trying to improve the root problems, giving priority to the simplest improvements with a good relationship between effort and impact. It was also useful to choose which models to prioritize and how to approach tests with adversarial attacks.

Additionally, different ways of perceiving the robustness of a model were presented, while creating exploratory analyses and KPIs to allow the comparison of different models. In case study 1, KPIs were designed to measure the difficulty of proposing adversarial attacks with a specific method, evaluating the intensity and features involved and their effectiveness. This was additionally useful to evaluate different attack scenarios and assumptions and help find an optimal decision threshold. These metrics can be automated with different attacks to create reports providing some intuitive insight into the adversarial robustness of a model and for comparison with other models.

The second case study explained the importance of creating less biased analyses and metrics in relation to the type of attack used in the defense (in the case of Adversarial Training) by testing more attack methods variations. Also, to avoid favoring one of the models by using external attacks. Finally, it showed the relevance of considering all possible ground truth and performance scenarios over natural and adversarial examples to avoid biasing the analysis to only initially effective cases.

9.2 Technical Acquired Knowledge

Most state-of-the-art evasion attacks are designed for computer vision applications, which means that the input are typically raw examples, and that DNNs (CNNs in this case) are usually used. This has two implications: on the one hand, when the algorithm is optimized to minimize perturbations, equal importance is given to all input features (pixels) to measure perturbations, which in another scenario, represents biases by **not considering the importance of each variable at the business level, its potential to be modified by the adversary (leverage variable) and its feasibility by not considering its range and the dependence between variables at a logical level.**

On the other hand, this implies that most effective methods are based on using gradients, considering that DNNs are based on derivatives (gradient descent), seeking to go against the loss function by optimizing attacks. This means that in other methods, such as ensemble-tree models, gradients cannot be directly applied since they are based on discrete decisions and not differentiable functions. Even with Black-box methods such as ZOO, which approximate the derivatives, there are problems with integer variables (**not being able to propose perturbations modifying these features**) as there is no variability in data around the point to be evaluated, which happens no matter how much the ZOO hyperparameters are modified.

Therefore, it is necessary to use other types of methods specifically designed for the type of model used, as was done in case study 1. This implies that it is difficult to create methods that directly propose an attack from the raw data that is logical, measures the perturbation, and is efficient. On the other hand, it makes necessary to be creative in order to use other more **exploratory approaches, testing changes, analyzing the importance of the variables in different ways, and using counterfactuals to guarantee analysis with real and feasible cases.**

Furthermore, it was learned about the **importance of considering the transferability** of attacks created with other models. This avoids thinking that only in a purely white-box attack scenario (knowing everything about the model) can AML techniques be used by an adversary to affect the real model. This highlights the importance of selecting the model to reduce its intrinsic transferability based on state-of-the-art, considering how easy it is for an adversary to imitate the model, and testing with substitute models.

Finally, the advantages of using Adversarial Training to improve the model's adversarial robustness and its representation were observed. This technique showed how by means of small perturbations around the natural example and looking for a similar representation, a model is obtained that is more difficult to attack while maintaining or improving the natural performance. In fact, it is a **useful technique to create a better representation** that makes it easier to visualize the data distribution with dimensionality reduction techniques. This allows in a more organized way to identify the clusters of each class and thus evaluate their closeness to other points of different classes. This enables the identification of certain types of adversarial attacks that are atypical for the data manifold present in the model's representation space.

9.3 Future Work

A wide variety of interesting research can be derived from this thesis to continue to make cyber security ML models more reliable and with greater capabilities. Therefore, it is proposed to implement some good practices to make the models technically more private, evaluate other ML techniques to complement the current models, test other interesting state of the art defenses to extend the case studies, and finally investigate and test other ML areas aligned with the AML objectives.

9.3.1 Privacy Good Practices

It is recommended to consider the implementation of some simple good practices to improve the privacy of the models and thus prevent them from being intuited by an adversary. Many of these are currently used, but it is considered relevant as future work to guarantee them in each of the models to be developed.

Starting with the simplest practices, it is recommended to block requests that do not make sense according to the data type, its range, and business sense. It is also recommended to add some randomness in the response times to make it difficult to guess about the type of model and architecture used. It is also possible to codify the output somehow, starting by rounding it to avoid allowing precision in a Black-box adversarial attack and even modifying it in cases of suspicious origin requests.

Models can also be made to analyze the origin of the requests, seeking to create anomaly analysis of the consumption pattern of the models. These models made to protect the other models, being based on patterns, will be more difficult to deceive by an adversary as they are not easily influenced. In addition to measuring the behavior of requests, it is possible to apply anomaly detection techniques such as those explained in *Deep Learning for Anomaly Detection: A Survey* (Chalapathy and Chawla, 2019), focusing on detecting inputs that do not resemble the normally used data and thus analyzing them to decide whether to block them.

Finally, improvements can be analyzed at the deployment level that allows multistage computation so that the processes are isolated so that if an adversary manages to access the model, it is only partially. The data preparation stage can be completely separated from the inference stage and complemented with the techniques explained in the previous paragraph.

9.3.2 Evasion Promising Defenses

It is proposed to continue testing defenses as was done in case study 2. Among the most promising ones analyzed in chapter 3 in this work, it is recommended to test Curriculum-based Adversarial Training. This defense will be an adaptation of the training process already performed in case study 2, doing it step-wise by progressively increasing the intensity of the attacks (see section 3.3.3.1.2).

Also, after the simple final test done in case study 2, it is proposed to test the creation of model ensembles promoting diversity as explained in section 3.3.3.1.2. This technique may require more effort due to the dependency that must be created between models at the time of training, but it is considered that it may have great potential.

Finally, it is especially recommended to test *Barrage of Random Transforms for Adversarially Robust Defense explained in section 3.3.3.4*, because of the apparent potential it has and the approach that relies more on creating high diversity data to force the model to learn the really relevant patterns, and having such a high random component that makes it difficult for an adversary to guarantee that an adversarial example is effective with small perturbations. This method can be analyzed together with the data augmentation commented in section 3.3.3.1.4 (Zantedeschi, Nicolae, and Rawat, 2017) to improve the robustness without compromising clean images performance.

9.3.3 Potential ML Techniques for Cyber Security

Some techniques that are considered to have a high potential to be applied to cyber security problems are recommended below. In the case of analyzing the behavior of requests at a time series level to detect anomalies that may represent suspicions of iteration attempts by an adversary, it is recommended to explore the techniques in *DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series* (Munir et al., 2019a).

It is also recommended to apply techniques such as *Hardening Random Forest Cyber Detectors Against Adversarial Attacks* (Apruzzese et al., 2020), in which they claim to improve the robustness of models such as a Random Forest (probably generalizable to more types of models). The proposed technique is interesting because it does not specifically apply a defense to the model but is based on creating two models in series. One by creating a traditional binary classifier, with which probability labels are created for the hard classes in order to know not only the class but also its confidence in the model (Condenser), and then training a regressor with these continuous labels (Receiver). This allows having a distilled model that indirectly gives different weights to each training example according to its difficulty to be binary classified. In this way, a better representation is created, and it is possible to have a model in which the output represents in a better way the probability that a case belongs to either class.

When training models it is also recommended to use Curriculum Learning to progressively train the model starting by modeling the easier cases and then the more difficult ones to give better performance and overall robustness (Soviany et al., 2021). Finally, in the case of binary detectors, it is recommended to consider one-class classification methods, considering it is very likely that it only makes sense to model one of the classes, being the other class the absence of the first class. This is important as it does not make sense to attempt to model patterns for all the cases that do not agree with a pattern. For this purpose, multiple techniques for images such as Deep auto-encoders for images and one-class variations of models such as Support-Vector Machine and Random Forest could be tested in practice to see their performance and robustness. It is recommended to use *One-Class Classification: A Survey* (Perera, Oza, and Patel, 2021) as a basis to learn about the taxonomy, the requirements of each technique, and their advantages and disadvantages.

9.3.4 AML Related Research Topics

Additionally, some ideas of AML or related branches to investigate and apply for cyber security are mentioned. Considering that most of the AML literature focuses on image problems, it is recommended further to investigate other areas such as Natural Language Processing. It is also recommended to analyze further the susceptibility of models to other types of attacks. For example, when creating datasets containing examples created by an adversary (e.g. phishing), explore the possibility of poisoning attacks as explained in section 3.2.2.1, and examine how to detect them as much as possible.

On the other hand, the case studies can be complemented with techniques coming directly from the field of Explainable AI to understand the features in which the model is fixed with and without any defense (section 4.2.2). It is also recommended to use Sufficient Input Subsets (SIS) (Carter et al., 2020)to understand Overinterpretation in case study 2, a technique with which it is possible to know which are the minimum pixels of an image that are sufficient to give a correct classification, and thus detect spurious patterns to avoid with data augmentation in this feature in the training dataset (section 4.2.3).

Finally, it is recommended to consider branches of research that have not been analyzed much but are of great importance in the operation of an ML model, such as Active Learning. This area consists of analyzing how to retrain an ML model to keep it updated with recent data, knowing how to filter it correctly, how often to update it, and how to optimize the process to detect when the performance of the model has declined.

Bibliography

Apruzzese, Giovanni et al. (2020). "Hardening random forest cyber detectors against adversarial attacks". In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 4.4, pp. 427–439. ISSN: 2471285X. DOI: 10.1109/TETCI.2019.2961157. arXiv: 1912.03790. URL: http://arxiv.org/abs/1912.03790http://dx.doi.org/10.1109/TETCI.2019.2961157.

Athalye, Anish, Nicholas Carlini, and David Wagner (2018). "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples". In: *35th International Conference on Machine Learning, ICML 2018* 1, pp. 436–448. arXiv: 1802.00420. URL: https://arxiv.org/abs/1802.00420v4.

Bai, Tao et al. (2021). "Recent Advances in Adversarial Training for Adversarial Robustness". In: pp. 4312–4321. DOI: 10.24963/ijcai.2021/591. arXiv: 2102.01356. URL: https://arxiv.org/abs/2102.01356v5.

Balaji, Yogesh, Tom Goldstein, and Judy Hoffman (2019). "Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets". In: ISSN: 2331-8422. arXiv: 1910.08051. URL: http://arxiv.org/abs/1910.08051.

Berman, Daniel S. et al. (2019). "A survey of deep learning methods for cyber security". In: *Information (Switzerland)* 10.4, p. 122. ISSN: 20782489. DOI: 10.3390/info10040122. URL: https://www.mdpi.com/2078-2489/10/4/122/htmhttps://www.mdpi.com/2078-2489/10/4/122.

Bianco, Simone et al. (2015). "Logo recognition using CNN features". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9280, pp. 438–448. ISSN: 16113349. DOI: 10.1007/978-3-319-23234-8_41.

– (2017). "Deep learning for logo recognition". In: *Neurocomputing* 245, pp. 23–30. ISSN: 18728286. DOI: 10.1016/j.neucom.2017.03.051. arXiv: 1701.02620.

Biggio, Battista et al. (2011). "Bagging classifiers for fighting poisoning attacks in adversarial classification tasks". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6713 LNCS, pp. 350–359. ISSN: 03029743. DOI: 10.1007/978-3-642-21557-5_37. URL: https://www.researchgate.net/publication/221094150_Bagging_Classifiers_for_Fighting_Poisoning_Attacks_in_Adversarial_Classification_Tasks.

Brown, Tom B. et al. (2017). "Adversarial Patch". In: DOI: 10.48550/arxiv.1712.09665. arXiv: 1712.09665. URL: https://arxiv.org/abs/1712.09665v2.

Brundage, Miles et al. (2018). "The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation". In: ISSN: 2331-8422. DOI: 10.17863/CAM.22520. arXiv: 1802.07228. URL: http://arxiv.org/abs/1802.07228.

Cai, Qi Zhi, Chang Liu, and Dawn Song (2018). "Curriculum adversarial training". In: *IJCAI International Joint Conference on Artificial Intelligence* 2018-July, pp. 3740–3747. ISSN:

10450823. DOI: 10.24963/ijcai.2018/520. arXiv: 1805.04807. URL: https://arxiv.org/abs/1805.04807v1.

Cao, Xiaoyu and Neil Zhenqiang Gong (2017). "Mitigating evasion attacks to deep neural networks via region-based classification". In: *ACM International Conference Proceeding Series* Part F1325, pp. 278–287. DOI: 10.1145/3134600.3134606. arXiv: 1709.05583. URL: https://arxiv.org/abs/1709.05583v4.

Carlini, Nicholas and David Wagner (2017). "Towards Evaluating the Robustness of Neural Networks". In: *Proceedings - IEEE Symposium on Security and Privacy*, pp. 39–57. ISSN: 10816011. DOI: 10.1109/SP.2017.49. arXiv: 1608.04644. URL: https://arxiv.org/abs/1608.04644v2.

Carter, Brandon et al. (2020). "Overinterpretation reveals image classification model pathologies". In: ISSN: 2331-8422. arXiv: 2003.08907. URL: http://arxiv.org/abs/2003.08907.

Chalapathy, Raghavendra and Sanjay Chawla (2019). "Deep Learning for Anomaly Detection: A Survey". In: arXiv: 1901.03407. URL: http://arxiv.org/abs/1901.03407.

Chen, Bryant et al. (2019). "Detecting backdoor attacks on deep neural networks by activation clustering". In: *CEUR Workshop Proceedings* 2301. ISSN: 16130073. arXiv: 1811.03728. URL: https://arxiv.org/abs/1811.03728v1.

Chen, Pin Yu et al. (2017). "ZOO: Zeroth order optimization based black-box atacks to deep neural networks without training substitute models". In: *AISec 2017 - Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, co-located with CCS 2017* 17, pp. 15–26. DOI: 10.1145/3128572.3140448. arXiv: 1708.03999. URL: http://arxiv.org/abs/1708.03999http://dx.doi.org/10.1145/3128572.3140448.

Chen, Tianqi and Carlos Guestrin (2016). "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 13-17-August-2016, pp. 785–794. DOI: 10.1145/2939672.2939785. arXiv: 1603.02754. URL: https://arxiv.org/abs/1603.02754v3.

Dhurandhar, Amit et al. (2018). "Explanations based on the Missing: Towards contrastive explanations with pertinent negatives". In: *Advances in Neural Information Processing Systems* 2018-Decem, pp. 592–603. ISSN: 10495258. arXiv: 1802.07623. URL: https://arxiv.org/abs/1802.07623v2.

Ding, Gavin Weiguang et al. (2018). "MMA Training: Direct Input Space Margin Maximization through Adversarial Training". In: ISSN: 2331-8422. arXiv: 1812.02637. URL: http://arxiv.org/abs/1812.02637.

Engstrom, Logan et al. (2017). "Exploring the Landscape of Spatial Robustness". In: *36th International Conference on Machine Learning, ICML 2019* 2019-June, pp. 3218–3238. DOI: 10.48550/arxiv.1712.02779. arXiv: 1712.02779. URL: https://arxiv.org/abs/1712.02779v4.

Gao, Yansong et al. (2019). "StriP: A defence against trojan attacks on deep neural networks". In: *ACM International Conference Proceeding Series*, pp. 113–125. ISSN: 21531633. DOI: 10.1145/3359789.3359790. arXiv: 1902.06531.

Gilmer, Justin et al. (2018). "Motivating the Rules of the Game for Adversarial Example Research". In: ISSN: 2331-8422. arXiv: 1807.06732. URL: http://arxiv.org/abs/1807.06732.

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy (2015). "Explaining and harnessing adversarial examples". In: *3rd International Conference on Learning Representations, ICLR*

2015 - Conference Track Proceedings. arXiv: 1412.6572. URL: https://arxiv.org/abs/1412.6572v3.

Gopalan, Arjun et al. (2021). "Neural Structured Learning: Training Neural Networks with Structured Signals". In: *WSDM 2021 - Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 1150–1153. DOI: 10.1145/3437963.3441666. URL: https://doi.org/10.1145/3437963.3441666.

Gutierrez, Pierre and Jean-Yves Gérardy (2016). "Causal Inference and Uplift Modeling: A Review of the Literature". In: *Jmlr* 67, pp. 1–13.

He, Warren et al. (2017). "Adversarial example defenses: Ensembles of weak defenses are not strong". In: *11th USENIX Workshop on Offensive Technologies, WOOT 2017, co-located with USENIX Security 2017*. arXiv: 1706.04701.

Huang, Ruitong et al. (2015). "Learning with a Strong Adversary". In: arXiv: 1511.03034. URL: http://arxiv.org/abs/1511.03034.

Juuti, Mika et al. (2019). "PRADA: Protecting against DNN model stealing attacks". In: *Proceedings - 4th IEEE European Symposium on Security and Privacy, EURO S and P 2019*, pp. 512–527. DOI: 10.1109/EuroSP.2019.00044. arXiv: 1805.02628. URL: https://arxiv.org/abs/1805.02628v5.

Karimi, Mohammad and Alireza Behrad (2019). "Logo Recognition by Combining Deep Convolutional Models in a Parallel Structure". In: *4th International Conference on Pattern Recognition and Image Analysis, IPRIA 2019*, pp. 216–221. DOI: 10.1109/PRIA.2019.8786032.

Kariyappa, Sanjay and Moinuddin K. Qureshi (2019). "Improving Adversarial Robustness of Ensembles with Diversity Training". In: arXiv: 1901.09981. URL: http://arxiv.org/abs/1901.09981.

Kasirzadeh, Atoosa and Andrew Smart (2021). "The Use and Misuse of Counterfactuals in Ethical Machine Learning". In: *FAccT 2021 - Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pp. 228–236. DOI: 10.48550/arxiv.2102.05085. arXiv: 2102.05085. URL: https://arxiv.org/abs/2102.05085v1.

Kurakin, Alexey et al. (2018). "Adversarial Attacks and Defences Competition". In: *undefined*, pp. 195–231. ISSN: 2331-8422. DOI: 10.1007/978-3-319-94042-7_11. arXiv: 1804.00097.

Liao, Fangzhou et al. (2018). "Defense Against Adversarial Attacks Using High-Level Representation Guided Denoiser". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1778–1787. ISSN: 10636919. DOI: 10.1109/CVPR.2018.00191. arXiv: 1712.02976. URL: https://arxiv.org/abs/1712.02976v2.

Liu, Wei et al. (2015). "SSD: Single Shot MultiBox Detector". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9905 LNCS, pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2. arXiv: 1512.02325v5. URL: http://arxiv.org/abs/1512.02325http://dx.doi.org/10.1007/978-3-319-46448-0_2.

Liu, Yanpei et al. (2017). "Delving into transferable adversarial examples and black-box attacks". In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*. arXiv: 1611.02770. URL: https://arxiv.org/abs/1611.02770v3.

Madry, Aleksander et al. (2018). "Towards deep learning models resistant to adversarial attacks". In: *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. arXiv: 1706.06083. URL: https://arxiv.org/abs/1706.06083v4.

Mahmood, Kaleel et al. (2021a). "Beware the black-box: On the robustness of recent defenses to adversarial examples". In: *Entropy* 23.10. ISSN: 10994300. DOI: 10.3390/e23101359. arXiv: 2006.10876. URL: http://arxiv.org/abs/2006.10876http://dx.doi.org/10.3390/e23101359.

– (2021b). "Beware the black-box: On the robustness of recent defenses to adversarial examples". In: *Entropy* 23.10, p. 1359. ISSN: 10994300. DOI: 10.3390/e23101359. arXiv: 2006.10876. URL: http://arxiv.org/abs/2006.10876http://dx.doi.org/10.3390/e23101359.

Mahony, Niall O' et al. (2019). "Deep Learning vs. Traditional Computer Vision". In: Advances in Intelligent Systems and Computing 943. Ed. by Kohei Arai and Supriya Kapoor. DOI: 10.1007/978-3-030-17795-9. arXiv: 1910.13796v1. URL: http://arxiv.org/abs/1910.13796http://dx.doi.org/10.1007/978-3-030-17795-9.

Malatras, Apostolos, Ioannis Agrafiotis, and Monika Adamczyk (2021). *Securing Machine Learning Algorithms Report Title*. Tech. rep. European Union Agency for Cybersecurity, p. 70. URL: https://www.enisa.europa.eu/publications/securing-machine-learning-algorithms.

Mao, Chengzhi et al. (2019). "Metric learning for adversarial robustness". In: *Advances in Neural Information Processing Systems* 32. ISSN: 10495258. arXiv: 1909.00900. URL: https://arxiv.org/abs/1909.00900v2.

Metzen, Jan Hendrik et al. (2017). "On detecting adversarial perturbations". In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*. arXiv: 1702.04267. URL: https://arxiv.org/abs/1702.04267v2.

Moosavi-Dezfooli, Seyed Mohsen, Alhussein Fawzi, and Pascal Frossard (2016). "DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-Decem, pp. 2574–2582. ISSN: 10636919. DOI: 10.1109/CVPR.2016.282. arXiv: 1511.04599. URL: https://arxiv.org/abs/1511.04599v3.

Munir, Mohsin et al. (2019a). "DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series". In: *IEEE Access* 7, pp. 1991–2005. ISSN: 21693536. DOI: 10.1109/ACCESS.2018.2886457.

– (2019b). "DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series". In: *IEEE Access* 7, pp. 1991–2005. ISSN: 21693536. DOI: 10.1109/ACCESS.2018.2886457.

Pang, Tianyu et al. (2019). "Improving adversarial robustness via promoting ensemble diversity". In: *36th International Conference on Machine Learning, ICML 2019* 2019-June, pp. 8759–8771. arXiv: 1901.08846. URL: https://arxiv.org/abs/1901.08846v3.

Papernot, Nicolas and Patrick McDaniel (2016). "On the Effectiveness of Defensive Distillation". In: arXiv: 1607.05113. URL: http://arxiv.org/abs/1607.05113.

Papernot, Nicolas et al. (2016). "Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks". In: *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016*, pp. 582–597. ISSN: 2375-1207. DOI: 10.1109/SP.2016.41. arXiv: 1511.04508. URL: https://arxiv.org/abs/1511.04508v2.

Papernot, Nicolas et al. (2017). "Practical black-box attacks against machine learning". In: *ASIA CCS 2017 - Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security*, pp. 506–519. DOI: 10.1145/3052973.3053009. arXiv: 1602.02697. URL: https://arxiv.org/abs/1602.02697v4.

Perera, Pramuditha, Poojan Oza, and Vishal M. Patel (2021). "One-Class Classification: A Survey". In: DOI: 10.48550/arxiv.2101.03064. arXiv: 2101.03064. URL: https://arxiv.org/abs/2101.03064v1.

Raff, Edward et al. (2019). "Barrage of random transforms for adversarially robust defense". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2019-June, pp. 6521–6530. ISSN: 10636919. DOI: 10.1109/CVPR.2019.00669.

Ravi, Sujith (n.d.). *Introducing Neural Structured Learning in TensorFLow*. URL: https://blog.tensorflow.org/2019/09/introducing-neural-structured-learning.html.

Redmon, Joseph et al. (2015). "You Only Look Once: Unified, Real-Time Object Detection". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-Decem, pp. 779–788. ISSN: 10636919. DOI: 10.1109/CVPR.2016.91. arXiv: 1506.02640. URL: https://arxiv.org/abs/1506.02640v5.

Rosenberg, Ishai et al. (2021). "Adversarial Machine Learning Attacks and Defense Methods in the Cyber Security Domain". In: *ACM Computing Surveys* 54.5. ISSN: 15577341. DOI: 10.1145/3453158. arXiv: 2007.02407. URL: https://arxiv.org/abs/2007.02407v3.

Song, Yang et al. (2018). "PixelDefend: Leveraging generative models to understand and defend against adversarial examples". In: *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. arXiv: 1710.10766. URL: https://arxiv.org/abs/1710.10766v3.

Soviany, Petru et al. (2021). "Curriculum Learning: A Survey". In: DOI: 10.48550/arxiv.2101.10382. arXiv: 2101.10382. URL: https://arxiv.org/abs/2101.10382v1.

Stutz, David, Matthias Hein, and Bernt Schiele (2019). "Disentangling adversarial robustness and generalization". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2019-June, pp. 6969–6980. ISSN: 10636919. DOI: 10.1109/CVPR.2019.00714. arXiv: 1812.00740. URL: https://arxiv.org/abs/1812.00740v2.

Su, Dong et al. (2018). "Is robustness the cost of accuracy? – A comprehensive study on the robustness of 18 deep image classification models". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11216 LNCS, pp. 644–661. ISSN: 16113349. DOI: 10.1007/978-3-030-01258-8_39. arXiv: 1808.01688. URL: https://arxiv.org/abs/1808.01688v2.

Verma, Sahil et al. (2021). "Counterfactual Explanations for Machine Learning: Challenges Revisited". In: DOI: 10.48550/arxiv.2106.07756. arXiv: 2106.07756. URL: https://arxiv.org/abs/2106.07756v1.

Weng, Tsui Wei et al. (2018). "Evaluating the robustness of neural networks: An extreme value theory approach". In: *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. arXiv: 1801.10578. URL: https://arxiv.org/abs/1801.10578v1.

Xi, Bowei (2020). "Adversarial machine learning for cybersecurity and computer vision: Current developments and challenges". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 12.5. ISSN: 19390068. DOI: 10.1002/wics.1511. arXiv: 2107.02894. URL: https://arxiv.org/abs/2107.02894v1.

Xie, Cihang et al. (2018). "Mitigating adversarial effects through randomization". In: *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. arXiv: 1711.01991. URL: https://arxiv.org/abs/1711.01991v3.

Xie, Cihang et al. (2019). “Improving transferability of adversarial examples with input diversity”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2019-June, pp. 2725–2734. ISSN: 10636919. DOI: 10.1109/CVPR.2019.00284. arXiv: 1803.06978. URL: https://arxiv.org/abs/1803.06978v4.

Xie, Cihang et al. (2020). “Adversarial examples improve image recognition”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 816–825. ISSN: 10636919. DOI: 10.1109/CVPR42600.2020.00090. arXiv: 1911.09665. URL: https://arxiv.org/abs/1911.09665v2.

Xu, Weilin, David Evans, and Yanjun Qi (2017). “Feature Squeezing Mitigates and Detects Carlini/Wagner Adversarial Examples”. In: ISSN: 2331-8422. arXiv: 1705.10686. URL: http://arxiv.org/abs/1705.10686.

– (2018). “Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks”. In: DOI: 10.14722/ndss.2018.23198. arXiv: 1704.01155. URL: http://arxiv.org/abs/1704.01155http://dx.doi.org/10.14722/ndss.2018.23198.

Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, Quanquan Gu (2020). “Improving Adversarial Robustness Requires Revisiting Misclassified Examples”. In: *Iclr* 29.9, pp. 1–17. arXiv: arXiv:1809.03008v2. URL: https://openreview.net/pdf?id=rklOg6EFwS.

Zantedeschi, Valentina, Maria Irina Nicolae, and Ambrish Rawat (2017). “Efficient defenses against adversarial atacks”. In: *AISec 2017 - Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, co-located with CCS 2017*, pp. 39–49. DOI: 10.1145/3128572.3140449. arXiv: 1707.06728. URL: https://arxiv.org/abs/1707.06728v2.

Zhang, Chong, Huan Zhang, and Cho Jui Hsieh (2020). “An efficient adversarial attack for tree ensembles”. In: *Advances in Neural Information Processing Systems* 2020-Decem. ISSN: 10495258. arXiv: 2010.11598. URL: https://arxiv.org/abs/2010.11598v1.

Zhang, Yu et al. (2021). “A Survey on Neural Network Interpretability”. In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 5.5, pp. 726–742. ISSN: 2471285X. DOI: 10.1109/TETCI.2021.3100641. arXiv: 2012.14261. URL: http://arxiv.org/abs/2012.14261http://dx.doi.org/10.1109/TETCI.2021.3100641.

Zheng, Stephan et al. (2016). “Improving the robustness of deep neural networks via stability training”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-Decem, pp. 4480–4488. ISSN: 10636919. DOI: 10.1109/CVPR.2016.485. arXiv: 1604.04326. URL: https://arxiv.org/abs/1604.04326v1.