ECOLE POLYTECHNIQUE DE BRUXELLES

HELLENIC REPUBLIC
National and Kapodistrian University of Athens
EST. 1837

Toward Robust Credit Card Fraud Detection: A Domain-Specific Study of Adversarial Machine Learning

Thesis presented by Daniele LUNGHI
in fulfilment of the requirements of the PhD Degree in Engineering and Technology
("Doctorat en Sciences de l'ingénieur et technologie")
Academic Year 2025-2026

Supervisor:
Professor Gianluca BONTEMPI
Co-supervisors:
Professor Dimitris SACHARIDIS
Doctor Alkis SIMITSIS
Professor Yannis IOANNIDIS

DE DS

Jury members

Chairman

Prof. Daniele BONATTO
Université Libre De Bruxelles
Belgium

Secretary

Prof. Dimitris SACHARIDIS
Université Libre De Bruxelles
Belgium

Promoter

Prof. Gianluca BONTEMPI
Université Libre De Bruxelles
Belgium

Co-Promoters

Dr. Alkis SIMITSIS
Athena Research Center
Greece

Prof. Yannis IOANNIDIS
National and Kapodistrian University of Athens
Greece

External Members

Prof. Minos GAROFALAKIS
Athena Research Center
Greece

Dr. Maura PINTOR
University of Cagliari
Italy

Prof. Gauthier LAFRUIT
Université libre de Bruxelles
Belgium

Prof. Dimitrios GUNOPULOS
National and Kapodistrian Univeristy of Athens
Greece

Prof. Erini NTOUTSI
Bundeswehr University Munich
Germany

Acknowledgements

This doctorate has been a long and whining road, full of challenges and hard moments. It has also been one of the greatest adventures of my life, an adventure that I am happy to have undertaken and carried through to this day. As this chapter of my life is closing, I want to acknowledge the people who made all of this possible. It is going to be a long list.

My supervisor, Gianluca. Throughout these years, you have guided me. You helped point my research in the right direction, left me the freedom to explore my intuitions, but you also forced me to be precise and methodical. You have given me invaluable feedback, insightful hints, and shown a talent, professionalism, and work ethic that I find inspiring. Thank you for all of the support.

The jury members for kindly accepting to review my thesis, attending my defense and for having offered precious suggestions to improve this work.

All the people involved in DEDS and the European Union's Horizon 2020. Thanks to you, I had the rare opportunity to live in three different countries, collaborate with researchers from diverse backgrounds, expand my horizons, and engage with so many great researchers from the ULB, the ARC, and Worldline. I learned a lot from all of you.

Alkis and Olivier. Alkis, you have been a mentor to me these years, and I thank you for that. You always pushed me to achieve more and showed enthusiasm for my work. It meant a lot. Olivier, your passion and enthusiasm for research are contagious. I hope to remain as passionate a researcher as you are.

Marie, Véronique, Maryka, and all the secretaries at the faculties of Sciences and Polytechnique. You really helped me a lot navigating my PH.D., thank you very much.

My co-authors: Gianluca, Olivier, Gian Marco, Yannick and Tom. This thesis would not be here without your efforts. More importantly, you made research fun.

My colleagues and friends at the ULB: Marco, Gian Marco, Valerie, Elias, Davide, Yannick, Elladio, Pascal, Apurva, Inas, Axel, Anirban… you made this office feel like home. The coffee breaks, lunch times, interesting discussions, fun schemas on the board and so many other things have been bright moments for me, and I cherish them deeply.

Professor Boracchi from Polimi, who trusted me when I just started, was a great first introduction to the academic world that I now love so much.

My friends Ilan, Sebastiano, Erika, Alessandra, Alessio, and Sara. I have many reasons to thank all of you personally, but you all made my life much better. I want to thank you for all these years.

My girlfriend Val. This last year has been wonderful. I cherish the time we spend together, I am grateful for the support you have always given me, I admire all the things you do, and I love you. You are a wonderful person, and I am so excited to share this moment with you. Even more, I am excited for all that is yet to come.

My family. Mum, thank you for all the times you still help me when I call you with all sorts of problems, for all the things you have done for me, and for being the greatest mum I could hope for. Anna, my best friend, and future author of the Lunghi-Lunghi paper. My grandpa, who is now in Milano, and all those who are not with us anymore. My dad, whom I still miss to this day and as long as I will live. Per finire: nonna, non mi hai visto laurearmi, ma avresti visto la difesa del dottorato. I wouldn't be here without you. Vi voglio bene.

Roads go ever ever on,
Over rock and under tree,
By caves where never sun has shone,
By streams that never find the sea;
Over snow by winter sown,
And through the merry flowers of June,
Over grass and over stone,
And under mountains in the moon.

Roads go ever ever on
Under cloud and under star,
Yet feet that wandering have gone
Turn at last to home afar.
Eyes that fire and sword have seen
And horror in the halls of stone
Look at last on meadows green
And trees and hills they long have known.

The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way
Where many paths and errands meet.
And whither then? I cannot say.

The Road goes ever on and on
Out from the door where it began.
Now far ahead the Road has gone,
Let others follow it who can!
Let them a journey new begin,
But I at last with weary feet
Will turn towards the lighted inn,
My evening-rest and sleep to meet.

J. R. R. Tolkien

Data-driven fraud detection utilizes machine learning to automatically detect fraudulent activities. Due to its scalability and strong generalization capabilities, it plays a crucial role in securing online payment systems. Traditional fraud detection approaches, however, do not account for the possibility that fraudsters may adapt their behavior to bypass the fraud detection system, which is a limitation they may exploit. The study of adaptive attackers and of the defenses to employ against them is called adversarial machine learning. In computer vision, adversarial attacks have proven to be highly effective, leading to an arms race between attackers and defenders. The proposed solutions, however, are generally application-specific, and their applicability to the domain of credit card fraud detection is dubious, with only a handful of works trying to apply adversarial techniques in this context. This thesis aims to bridge the gap between credit card fraud detection and adversarial machine learning through three main contributions:

- A systematic assessment of adversarial attacks through the lens of their applicability to the context of credit card fraud detection. We discuss the impact of transaction aggregations, concept drift, and attackers' capabilities in securing fraud detection systems. We complement this analysis with an empirical assessment of some of the most common attacks in adversarial machine learning literature, showing how they struggle to adapt to the fraud detection domain due to a fundamental misalignment between the threat models of the domain and those for which the algorithms were designed.
- The first quantitative model of fraudsters in the fraud detection literature. This analysis, backed by experiments performed on real industrial data, models fraudsters' actions as a function of the information they may have access to, such as the card they stole and the previous transactions performed with it. The resulting model, showing the limitations of fraudsters' knowledge about the cards they steal, led to the development of two novel oversampling algorithms: MIMO ADV-O and TimeGan ADV-O, capable of performing in line with other state-of-the-art resampling algorithms. The ADV-O framework became the backbone of this thesis model of fraudsters' capabilities and knowledge.
- The design of a new adversarial attack for credit card fraud detection, called FRAUD-RLA. FRAUD-RLA employs reinforcement learning to generate fraudulent transactions efficiently. The algorithm takes into account the specificities of fraud detection to operate under limited knowledge and capabilities, thereby

maximizing the collected reward over time. FRAUD-RLA is designed to provide an example of a family of RL-based adversarial attacks we should be aware of, and be an assessment tool for the security of fraud detection systems.

Our analysis reveals that credit card fraud detection exhibits several peculiarities that render traditional adversarial attacks ineffective due to the limited attackers' knowledge and other domain-specific constraints. However, we also show how it is possible to design domain-specific attacks using reinforcement learning as a tool. Overall, this thesis highlights the necessity of further studying the adversarial security of fraud detection engines and provides a framework for modeling, understanding, and assessing the threat.

Résumé

La détection des fraudes basée sur les données utilise l'apprentissage automatique pour détecter les activités frauduleuses. Grâce à son évolutivité et à ses solides capacités de généralisation, elle joue un rôle crucial dans la sécurisation des systèmes de paiement en ligne. Les approches traditionnelles de détection des fraudes ne tiennent toutefois pas compte de la possibilité que les fraudeurs adaptent leur comportement pour contourner le système de détection des fraudes, ce qui constitue une limite dont ceux-ci peuvent tirer parti. L'étude des attaquants adaptatifs et des défenses à mettre en œuvre contre eux est appelée *adversarial machine learning.* Dans le domaine de la vision par ordinateur, les attaques adversariales se sont révélées très efficaces, entraînant une course entre les attaquants et les défenseurs. Les solutions proposées sont toutefois généralement spécifiques à une application donnée et leur applicabilité au domaine de la détection des fraudes à la carte de crédit est discutable, seules quelques études ont jusqu'alors tenté d'appliquer des techniques adversaires dans ce contexte. Cette thèse vise à combler l'écart entre la détection des fraudes bancaires et l'apprentissage automatique antagoniste grâce à trois contributions principales :

– Une évaluation systématique des attaques adversariales sous l'angle de leur applicabilité au contexte de la détection des fraudes bancaires. Nous discuterons de l'impact des agrégations de transactions, du glissement conceptuel et des capacités des attaquants sur la sécurité des systèmes de détection des fraudes. Nous compléterons cette analyse par une évaluation empirique de certaines des attaques les plus courantes dans la littérature sur l'apprentissage automatique antagoniste, montrant comment elles peinent à s'adapter au domaine de la détection des fraudes en raison d'un décalage fondamental entre les *threat models* de ce domaine et ceux pour lesquels les algorithmes ont été conçus.

– Le premier modèle quantitatif des fraudeurs dans la littérature sur la détection des fraudes. Cette analyse, étayée par des expériences menées sur des données industrielles réelles, modélise les actions des fraudeurs en fonction des informations auxquelles ils peuvent avoir accès: carte de crédit volée ou transactions précédemment effectuées avec celle-ci. Le modèle obtenu, lequel montre les limites des connaissances des fraudeurs sur les cartes qu'ils volent, a conduit au développement de deux nouveaux algorithmes de suréchantillonnage : MIMO ADV-O et TimeGan ADV-O, tous deux capables de fonctionner à l'instar d'autres algorithmes de rééchantillonnage de pointe. Le cadre ADV-O est devenu la colonne vertébrale de la modélisation des capacités et connaissances des fraudeurs utilisée dans cette thèse.

- Conception d'une nouvelle attaque adversariale pour la détection des fraudes bancaires, appelée FRAUD-RLA. FRAUD-RLA utilise *reinforcement learning* pour générer efficacement des transactions frauduleuses. L'algorithme tient compte des spécificités du problème de la détection des fraudes pour fonctionner avec des connaissances et des capacités limitées, maximisant ainsi la récompense collectée au fil du temps. FRAUD-RLA est conçu pour fournir un exemple d'une famille d'attaques adversariales basées sur l'apprentissage par renforcement dont il nous faut tenir compte , et pour servir d'outil d'évaluation de la sécurité des systèmes de détection des fraudes.

Notre analyse révèle que la détection des fraudes à la carte de crédit présente plusieurs particularités qui rendent les attaques adversariales traditionnelles inefficaces, de part les connaissances limitées des attaquants ou d'autres contraintes spécifiques à ce domaine. Cependant, nous montrons également comment il est possible de concevoir des attaques spécifiques au domaine en utilisant les outils offerts par l'apprentissage par renforcement. Dans l'ensemble, cette thèse souligne la nécessité d'étudier plus en profondeur la sécurité adversariale des moteurs de détection des fraudes et fournit un cadre pour modéliser, comprendre et évaluer les menaces auxquelles fait face la sécurité bancaire.

Part I

Part I

Chapter 1

Introduction

The increased digitalization of our economy has led to a progressive shift from cash to online and card payments, with cash accounting for only 39% of retail purchases in Belgium in 2024, down from nearly 45% in 2022 [1]. In the first half of 2022, credit card payments in the euro area accounted for over 36 trillion euros [2]. Digital payments offer numerous advantages, facilitating convenient money transfers and contributing to the rise of the modern data economy. The growth of digital payments, however, has raised worries over the risks posed by financial fraud. Unauthorized credit card payments, also known as frauds, are particularly concerning. The impact of payment fraud can be profound, affecting multiple levels of the financial system. Banks face rising costs associated with fraud prevention, reimbursement, and security upgrades. Consumers may lose trust in digital payments, leading to hesitation in adopting new technologies. Additionally, regulatory bodies must continuously update and enforce expensive security measures to protect both businesses and individuals from emerging threats. Furthermore, online transactions and e-commerce have made it easier for criminals to exploit vulnerabilities, as many online payments require only card details rather than the physical card itself. Hence, scenarios where a physical card is not used, known as Card-not-present (CNP) [92], have become the dominant form of payment fraud, posing a significant risk to online transactions [12] and accounting for over a billion euro loss in the EU in 2018, almost 80% of the total value of fraud (see Figure 1.1).

The need to mitigate the economic impact of financial fraud has prompted researchers and companies to develop solutions that combat this phenomenon. At a high level, we can identify two main strategies [43]: *fraud prevention*, which focuses on designing safe-by-design payment systems, and *fraud detection*, where fraudulent transactions are identified when attempted. Fraud prevention includes measures such as the Card Verification Method (CVM), which verifies cardholder details and ensures legitimacy, as well as Personal Identification Numbers (PINs) and 3D Secure (3DS), which require users to verify their identity through a one-time password or biometric authentication before completing online payments.

Despite the effectiveness of preventive measures, fraud cannot always be stopped before it happens. Fraud detection systems are, therefore, essential for ensuring the security of payment systems. A straightforward approach to detecting fraudulent transactions is through human inspection; investigators can observe a transaction, investigate its

[1] Cash payments continue to fall, frustration with ATM availability remains high in Belgium
[2] Payments statistics: first half of 2023

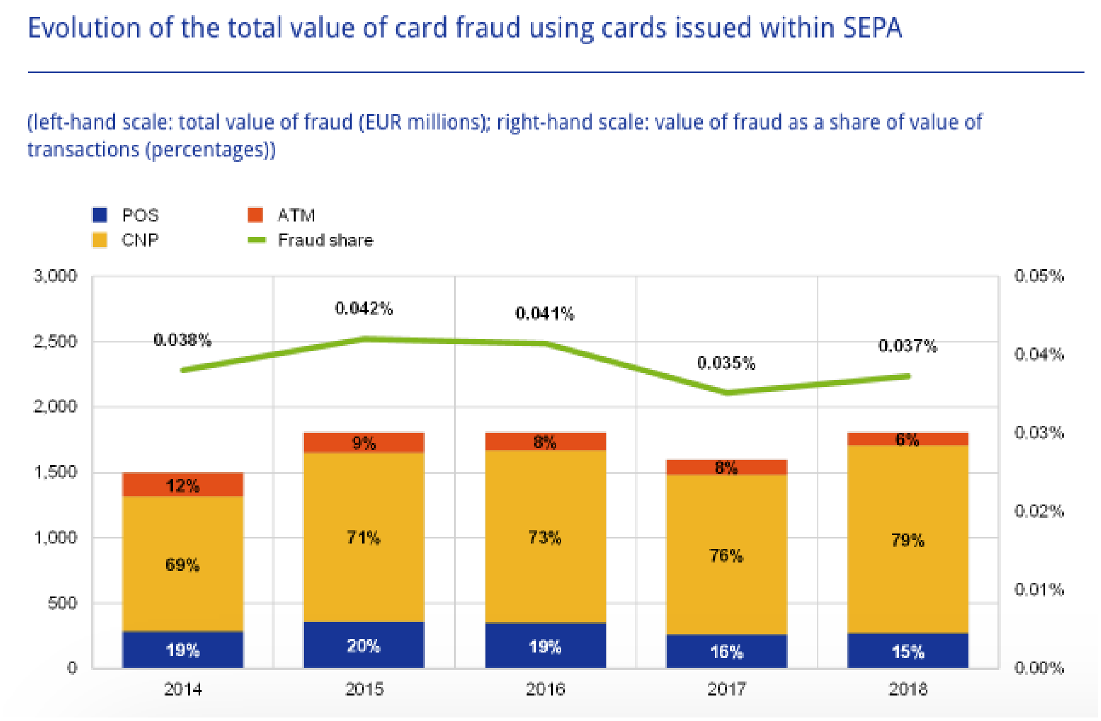


Figure 1.1: Evolution of the total value of card fraud using cards issued within SEPA from 2014 to 2018. Card-not-present frauds account for the vast majority of reported frauds. Image taken from the ECB 6th report on card fraud.[3]

origin, and determine whether it is genuine or fraudulent. Such an approach, however, is impossible to implement at scale: the sheer volume of transactions processed every day far exceeds the number of investigators companies can reasonably hire for the job. A second approach involves designing a set of expert rules, which can be automatically applied to check every transaction cost-effectively. These rules can be simple, such as blocking all transactions involving a known malicious entity, or complex and statistically based, such as blocking transactions whose amount is a statistical outlier. This rule-based approach is cost-effective and straightforward, but it struggles with detecting complex patterns, making more advanced solutions a necessity [92].

1.1 Machine Learning and fraud detection

In rule-based fraud detection, experts establish a set of rules that should automatically trigger an alarm and apply them to each transaction entering the system. This type of work relies on human-crafted algorithms and works well when the rules are clear and relatively stable. In some cases, however, explicit programming is impractical or insufficient, and we need the program to learn the optimal behavior from data automatically. This approach is known as *machine learning* (ML), a subfield of computer science that intersects closely with statistics and probability theory. In machine learning, the programmer defines the task, the evaluation metric, and a learning approach. The goal of the machine learning algorithm is to improve its performance on the task through some training experience rather than relying solely on hard-coded rules [112].

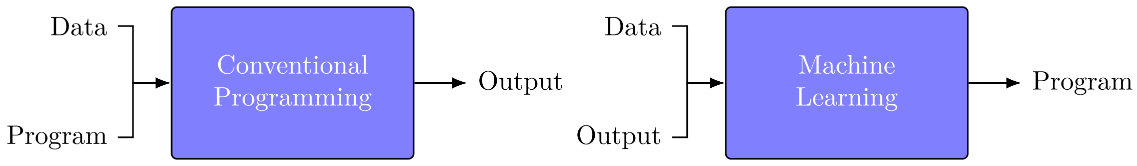


Figure 1.2: Conventional programming (left) and supervised machine learning (right). In conventional programming, rules (the program) and data are combined to produce outputs, whereas in machine learning, data and outputs are used to learn the program (model).

Machine learning can be categorized into three main subtypes: supervised, unsupervised, and reinforcement learning. Supervised learning is defined as estimating the unknown model that maps known inputs to unknown outputs. In turn, supervised approaches can be divided into classification and regression. The former learn to associate an input with a class automatically, the latter learn the relationship between the input and a continuous variable output [21]. In many real-world scenarios, supervised learning is not a viable solution. In these cases, we can still infer valuable information directly from the data itself. Even without explicit supervision, patterns and structures in the data can be identified and exploited to gain insight. Examples of unsupervised learning applications include anomaly detection, where observations are monitored to identify any anomalous patterns [35], and synthetic data generation, where machine learning algorithms learn to produce new data from scratch with similar characteristics to those they have been trained on [68]. Finally, in applications such as robotics and gaming, the task is not to learn from a dataset, but to optimize a strategy learning from experience. In these cases, reinforcement learning (RL) involves optimizing the decision-making process of an agent who interacts with an environment. The RL algorithm, also known as an agent, learns through trial and error how to select the best policy that maximizes its reward over time [150]. Examples of RL problems are teaching an agent to optimize navigation in a maze [154] and bandit algorithms, where agents learn the best action among a pool of fixed-reward actions, each associated with a stochastic label [91].

Fraud detection primarily relies on supervised learning methods, where models are trained using historical transaction data labeled as either fraudulent or legitimate. Specifically, it is an example of a classification problem, where the goal is to automatically label a transaction as genuine or fraudulent [30, 43, 92, 26]. The system is provided with historical data about credit card transactions, each labeled as either fraudulent or legitimate. The model learns patterns associated with fraudulent behavior and can subsequently flag new, unlabeled transactions that exhibit similar characteristics [92].

At a high level, online payments work as follows [92, 28, 45]. Cardholders own one or more cards and can use a card they own to contact a terminal and perform a payment. Sometimes, fraudsters can clone or steal a card and then use it to connect to a set of vulnerable terminals and perform fraudulent payments. Certain terminals may be more vulnerable due to relaxed security measures, outdated hardware, or their presence in high-risk locations. Both genuine and fraudulent payments comprise an amount, a timestamp, other transaction-specific features, and terminal and card identifiers. Cards and terminals are also characterized by additional features, such as the country in which

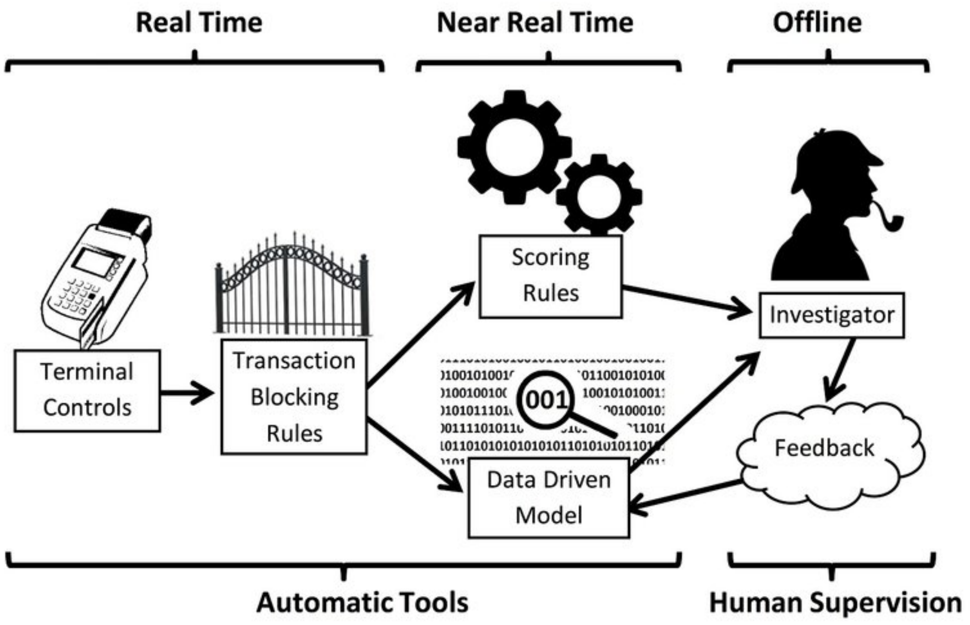


Figure 1.3: Fraud detection system, image taken from [28]. Transactions are analyzed through a set of simple statistical rules and machine learning algorithms. Human investigators then analyze the alerts raised by the previous parts of the pipeline.

the terminal is located or the type of card. Importantly, because terminals and cards are assigned unique identifiers, fraud detection systems can leverage aggregated data such as the average transaction amount at a specific terminal over the past week to identify suspicious activity.

Payment processing companies collect transactions over time, enrich them with contextual information such as aggregations, and use them to build a training set of transaction-label pairs. The training dataset is then used to train a previously selected machine learning algorithm to automatically identify the patterns that distinguish genuine and fraudulent transactions.

Despite its effectiveness, fraud detection is not limited to machine learning. Instead, it comprises different layers: combining human-made rules to identify known patterns, machine learning, and a careful use of human investigators. First, simple human-made rules are used in real-time to block transactions that match known fraudulent patterns [46]. Afterward, both machine learning and rule-based scoring strategies are combined to flag suspicious activity retroactively. The combination of rule-based and machine-learning solutions is designed to leverage the strengths of both approaches. Rule-based approaches are simple to implement, interpretable, and are built on expert knowledge, but they struggle with generalization. Machine learning, on the other hand, can detect previously unseen fraudulent patterns, model complex data distributions, and scale more easily to high-dimensional data. For this reason, data-driven methods have remained a cornerstone of credit card fraud detection for over a decade. Finally, human investigators are used to analyze the transactions highlighted by the previous layers of the pipeline. By focusing on a limited number of transactions, investigators provide human insight and verify the validity of the alerts raised by the earlier layers.

Over the years, research on fraud detection has continually evolved to address the numerous challenges that the fraud detection domain poses to practitioners. From the challenge of training and evaluating machine learning algorithms on heavily unbalanced datasets where genuine transactions far outnumber the fraudulent ones [28, 47] to working in an ever-changing environment [122, 44], the numerous challenges have led to continuous improvement in the machine learning components of fraud detection.

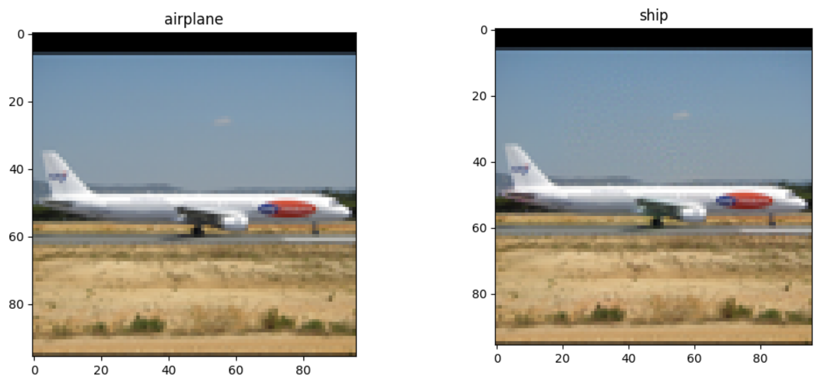
Research on fraud detection, however, has traditionally considered fraudsters' behavior as a statistical phenomenon, independent from the fraudulent system itself. This can be a severe limitation: fraud involves malicious agents (the fraudsters/adversaries) who actively adapt their behavior in response to the detection mechanisms they encounter. When a strategy becomes ineffective, they can modify it. More critically, fraudsters may design their attacks to explicitly seek weaknesses in the target model. The issue of modeling the attacking patterns of an intelligent agent that adversarially adapts to the deployed fraud detection systems remains a widely open problem.

1.2 Adversarial Machine Learning

As an increasing part of our life has moved online, the need to protect us from malicious agents has led to an arms race between defenders and attackers in fields like spam [42], malware [96], and intrusion detection [157]. Challenges that were once handled through human analysis and rule-based discrimination are increasingly incapable of reaching the required scalability our increasingly interconnected world requires. The necessity of operating machine learning models in adversarial environments, where an adversary actively works to cause the implemented model to behave differently from its intended behavior, led to the creation of a new research field called Adversarial Machine Learning (AML). Over the last two decades, adversarial machine learning has become a research topic of increasing interest, particularly due to the significant initial results obtained in the field of image recognition [125, 144].

At a very high level, adversarial machine learning can be defined as the study of how machine learning algorithms can be attacked and defended [82]. Adversarial attacks can be generally grouped into two main categories: poisoning attacks, which alter the training data to corrupt the learning process and cause the model to misbehave, and evasion attacks, which focus on manipulating inputs at test time to fool the trained classifier into making incorrect predictions [133]. A classic example of evasion attacks comes from the image recognition domain, where researchers have discovered that slight, often unnoticeable modifications in an image can lead the classifier to predict the wrong class for an image it would otherwise correctly predict [152, 29, 39] (see Figure 1.4).

Despite the remarkable results in computer vision, applying AML techniques to other domains has proven particularly challenging. Studying adversarial attacks requires a careful understanding of the threat posed by the attacker, where their goals, knowledge level, and capabilities must be precisely defined before designing the attacks. Such a



(a) Original Image **(b)** Perturbed Misclassified Image

Figure 1.4: Example of an adversarial attack on image recognition. The attack has been generated using the implementation of the C&W attack [29]. On the left, the original image and the label predicted by the classifier, on the right, the adversarially modified image and the predicted label.

process, referred to as threat modeling [29, 33] from the equivalent concept in computer security [147], has proven to be extremely application-specific, making domain adaptation of attacks (and therefore defenses) extremely challenging. For instance, some attacks rotate or translate images [59]. The idea behind it is generalizable: such attacks maintain the meaning of the original observation while performing successful attacks. The attack, however, relies on an image-specific property: humans treat rotated and translated images as the same object. Equivalents in applications such as fraud and intrusion detection are not always immediately apparent.

Despite the best efforts of researchers, most applications remain relatively understudied. In recent years, significant efforts have been made to extend the adversarial machine learning literature to address the challenges of malware [78, 132] and intrusion detection [50], where researchers have developed new techniques to account for the distinct nature of data and threat models. Despite this recent trend, however, credit card fraud detection has received very little attention in the AML community, leaving the field widely unexplored.

1.3 Motivations and aims

This work aims to bridge the gap between credit card systems and fraud detection by presenting the first systematic exploration of their intersection in the literature. We organize our approach around four main subgoals: (1) developing a new extensive threat model for adversarial attacks in fraud detection, (2) providing a clear theoretical and quantitative evaluation of how existing methods perform under this model, (3) developing new adversarial attacks to test and improve the robustness of fraud detection systems, and (4) discussing possible strategies to enhance fraud detection based on the vulnerabilities uncovered.

We identify two primary benefits of conducting this analysis: enhancing the security of credit card fraud detection systems and expanding our broader understanding of adversarial machine learning. From a fraud detection perspective, the importance is evident: fraud detection engines operate in an intrinsically adversarial environment, where intelligent attackers/fraudsters have the incentive to develop adaptive strategies; analysing their adversarial robustness is a practical necessity for building resilient systems.

Regarding adversarial machine learning, it is worth noting that the majority of research has been conducted on a limited number of applications, primarily in a laboratory setting, with few real-world case studies. This severely limits our understanding of these threats in real-world applications, where machine learning models interact with complex data pipelines. As such, their effectiveness in the real world remains an open question. While this thesis has no pretense of providing a comprehensive guide on how to study adversarial machine learning in real-world security applications, evaluating and enhancing the robustness of realistic fraud detection use cases, as we do in this thesis, can be an example for researchers facing similar issues in other fields, and can ultimately be a small step towards transforming AML from a study on machine learning models' properties to a functioning security tool.

1.4 Thesis Contributions

Understanding the threat posed by adversarial attacks to fraud detection requires multiple steps: from threat modeling and assessing the effectiveness of existing approaches to designing new attacks to test ideas that seem dangerous. We organize our work into three main contributions, each presented in a paper and in a Section of Part II of this work.

Contribution 1 The first major contribution of this thesis, presented in Chapter 5, is a thorough analysis of the feasibility of existing adversarial approaches in the domain of fraud detection. The analysis is composed of two main parts: the theoretical and the empirical. In the theoretical analysis, we discuss the primary challenges that fraudsters encounter when targeting fraud detection systems and compare them to the assumptions underlying most existing approaches. In the empirical analysis, we focus on a specific advantage fraudsters enjoy in the context of fraud detection, namely the fact that, unlike in computer vision, most adversarial attacks do not need to fool the human eye, given that investigators analyze only a small minority of the transactions. We show experimentally that such attacks are not well-suited for the domain of credit card fraud detection, highlighting a fundamental misalignment between existing methods and the specific requirements of financial fraud scenarios. This contribution highlights the primary differences between credit card fraud detection and previous domains, discusses how existing approaches may perform in this setting, and provides a baseline for the remainder of the thesis.

Contibution 2 In our second contribution, presented in Chapter 6, we partnered with an industrial partner, Worldline SA, to study the behavior of fraudsters using real historical transactions. Specifically, we developed the first quantitative model of fraudsters in the literature, called ADV-O, where we model fraudsters' temporal behavior and how much it is influenced by previous transactions performed with the card they steal. The analysis led to two main results: first, we showed how fraudsters employ time-dependent strategies, but are generally not influenced by the card's history, supporting our hypothesis that most fraudsters do not have access to this information. Second, we used our model to develop two new algorithms for enriching and balancing the training set by generating new synthetic fraudulent transactions. The first, MIMO ADV-O, captures the dependencies between consecutive fraudulent transactions on the same card by training a multi-target regression model to predict transaction characteristics based on previous events, thus enabling the generation of realistic synthetic fraud data. The second, TimeGAN ADV-O, builds on the TimeGAN framework to model chains of fraudulent transactions as time series. These algorithms form an oversampling strategy with performance comparable to that of state-of-the-art methods.

Contribution 3 : Our final major contribution, presented in Chapter 7, proposes a new adversarial attack for credit card fraud detection. We build upon the previous two contributions to design a new threat model that aligns with fraudsters' goals, knowledge, and capabilities. We also design a new adversarial attack based on reinforcement learning that utilizes this approach to bypass classifiers. This attack, called FRAUD-RLA, is designed to maximize the attacker's reward by optimizing the exploration-exploitation trade-off and requiring significantly less knowledge than competitors. Our experiments, conducted against a state-of-the-art fraud detection system, indicate that FRAUD-RLA is effective, even considering the severe constraints imposed by our threat model.

1.5 Activities Summary

This thesis was conducted under Grant 955895 of the HORIZON EUROPE Marie Sklodowska-Curie Actions, and it involved a collaboration among four institutions: the Université libre de Bruxelles (ULB), the Athena Research Center (ARC), the National and Kapodistrian University of Athens (NKUA), and Worldine SA. As such, during this period, I spent one year working at the ARC and was a research intern at Worldline for three months in 2024, hosted by the company's R&D department.

I attended multiple seminars at the ULB and participated in the four Schools organized by the DEDS, as well as the 5th ACM Europe Summer School on Data Science. In 2022, I collaborated in teaching the "Big Data: Distributed Data Management and Scalable Analytics" course at the ULB. Between 2024 and 2025, I supervised a master's student in collaboration with Politecnico di Milano. Throughout my PH.D., I peer-reviewed several papers for IEEE Access.

I have also supervised a Master's student in collaboration with Politecnico di Milano, leading to the development of a new adversarial attack against unsupervised federated learning, which we intend to submit to the International Conference on Machine Learning (ICML) in January 2026. I participated in the "AI Cybersecurity Deephack: Advance European Defence Capabilities" in April 2025. Our team proposed a solution that included evasion attacks on phishing detectors and won second prize. More details on both contributions can be found in Chapter 8.

Finally, in collaboration with the DEDS consortium, I wrote a chapter for the forthcoming book, which comprises the research of more than ten Ph.D. students [103].

Published papers

- Lunghi, D., Paldino, G. M., Caelen, O., & Bontempi, G. (2023). An adversary model of fraudsters' behavior to improve oversampling in credit card fraud detection. IEEE access, 11, 136666-136679 [101].
- Lunghi, D., Simitsis, A., Caelen, O., & Bontempi, G. (2023, June). Adversarial learning in real-world fraud detection: Challenges and perspectives. In Proceedings of the Second ACM Data Economy Workshop (pp. 27-33) [104].
- Lunghi, D., Simitsis, A., & Bontempi, G. (2024, July). Assessing adversarial attacks in real-world fraud detection. In 2024 IEEE International Conference on Web Services (ICWS) (pp. 27-34). IEEE [102].

Submitted papers

- Lunghi, D., Molinghen, Y., Simitsis, A., Lenaerts, T., & Bontempi, G. (2025). FRAUD-RLA: A new reinforcement learning adversarial attack against credit card fraud detection. arXiv preprint arXiv:2502.02290. [100]. Paper submitted to the IEEE Transactions on Dependable and Secure Computing.

Pre-prints

- Lunghi, D., Simitsis, A., & Bontempi, G. (Forthcoming). Adversarial Learning for Fraud Detection. In G. Dejaegere, A. Abello, K. Torp and A. Simitsis (Eds.), Data Engineering for Data Science. Springer. [103]

1.6 Presentations

The content of this work has been presented at the following venues:

- DEDS Winter School on "Ethical and Legal Aspects of Data" 2022, Athens, Greece
- DEDS Summer School on "Big Data Analytics and Management" 2022, Cesena, Italy
- DEDS Winter School on "Entrepreneurship and Innovation" 2023, Aalborg, Denmark
- Second ACM Data Economy Workshop 2023, Seattle, USA
- Summer School on "Applied Statistics" 2024 Barcelona, Spain

– IEEE International Conference on Web Services (ICWS) 2024 Online

Internal dissemination at the ULB has also been an important part of this thesis, with different aspects of this work presented to other researchers at the 1st IF@ULB workshop, to students at the SmartMonday [4], and to University researchers at the ULB Cybersecurity Seminar.

1.7 Code availability

In compliance with open science practices, the code of the contributions of this thesis is published on the online platform GitHub. Whenever possible, the code is released with instructions on how to generate or download the datasets used:

- For the paper "Assessing adversarial attacks in real-world fraud detection", Chapter 5: Repository 1
- For the paper "An adversary model of fraudsters' behavior to improve oversampling in credit card fraud detection" Chapter 6: Repository 2
- For the paper "FRAUD-RLA: A new reinforcement learning adversarial attack against credit card fraud detection" Chapter 7: Repository 3

1.8 Financial support

This thesis was supported by the HORIZON EUROPE Marie Sklodowska-Curie Actions under Grant 955895.

1.9 Thesis Outline

We structured this work into three parts. Part I introduces the background of this research. We first present the machine learning foundations of this work in Chapter 2. We then discuss, in Chapter 3, the problem of credit card fraud detection, and, in Chapter 4, that of adversarial machine learning. We also describe the literature on adversarial attacks in credit card fraud detection.

Part II contains our contributions, divided into three chapters. Chapter 5, which presents the work of Contribution 1, begins by outlining the main challenges fraudsters face in credit card fraud detection and identifies the key questions that remain unaddressed in previous literature. Chapter 6 and Chapter 7 introduce our second and third contributions, respectively, describing the problems they address, the algorithms developed, and the corresponding empirical results.

Finally, Part III concludes the thesis in Chapter 8 by summarizing the results of this work, outlining their broader impact on adversarial machine learning, and discussing promising future research directions to advance the adversarial robustness of fraud detection systems.

[4] Event Link

1.10 Index of Abbreviations

Abbreviation	Description
AML	Adversarial Machine Learning
AMT	Amount
BB	Black Box
BRF	Balanced Random Forest
C&W	Carlini & Wagner
CSR	Constrained Success Rate
ER	Empirical Robustness
FD	Fraud Detection
FDS	Fraud Detection System
FGSM	Fast Gradient Sign Method
FN	False Negatives
FP	False Positives
GAN	Generative Adversarial Network
GRU	Gated Recurrent Unit
MAE	Mean Absolute Error
MDP	Markov Decision Process
MIMO	Multi-Input-Multi-Output
MISO	Multi-Input-Single-Output
ML	Machine Learning
MSE	Mean Squared Error
(D)NN	(Deep) Neural Network
PCA	Principal Component Analysis
PGD	Projected Gradient Descent
PO-MDP	Partially Observable Markov Decision Process
PPO	Proximal Policy Optimization
PR-AUC	Area under the Precision-Recall Curve
RF	Random Forest
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SMOTE	Synthetic Minority Oversampling Technique
SR	Success Rate
SMDP	Semi-Markov Decision Process
TN	True Negatives
TP	True Positives
TP@K	Precision at Top K
TRX	Transaction
VAE	Variational Autoencoder
WB	White Box

1.11 Index of Notations

Symbol	Description
n	Size of input space
X	Input space
m	Size of output space
Y	Output space
x	Model input
y	General output
D	Dataset
D_{TR}	Training set
D_{TS}	Test set
N_D	Size of dataset D
t	Time
f	Classifier
f_{in}	Probability component of the classifier
f_{out}	argmax function
f_{logit}	Neural network classifier before normalization
f_{prob}	Softmax function
L	Loss function
f_θ	Classifier with parameters θ
θ	Parameters
$\circ$	Function composition operator

Table 1.2: Notation Table: Supervised Machine Learning

Symbol	Description
G	Generator
$\mathcal{D}$	Discriminator
$\mathcal{R}$	Reconstructor
E	Embedder
L_G	Generator loss
L_D	Discriminator loss

Table 1.3: Notation Table: Generative Adversarial Networks (GANs)

Symbol	Description
S	State space
A	Action space
R	Reward function
T	Transition function
O	Observation space
Ω	Observation function
a_t	Action executed at time t
s_t	State at time t
π_θ	Policy with parameters θ

Table 1.4: Notation Table: Reinforcement Learning

Symbol	Description
c_i	Card of index i
m_j	Terminal of index j
$x_{i,j,t}$	Transaction performed with card c_i on terminal m_j at time t
$\bar{C}$	Number of customers in the system
$\bar{M}$	Number of terminals/merchants in the system
$\psi_c(i,t)$	Function returning past transactions of a customer
$\psi_m(j,t)$	Function returning past transactions of a terminal
F_i	Customer's or terminal's features
$x'(i,j,t)$	Controllable features of transaction $x_{i,j,t}$
ϕ	Feature engineering process

Table 1.5: Notation Table: Fraud Detection

Symbol	Description
t_{init}	Earliest possible attack time
t_{final}	Latest possible attack time
g	Adversarial perturbation function
x_{adv}	Adversarial attack
d	Distance metric
$\mathcal{S}$	Attack success indicator
B	Attack perturbation budget
C_{atk}	List of compromised cards
M_{atk}	List of compromised merchants
k	Constraints satisfaction function
$\bar{y}$	Target class
$\hat{f}$	Substitute model
Υ	Set of valid transformations for problem space attacks

Table 1.6: Notation Table: Adversarial Machine Learning

Chapter 2

Machine Learning

2.1 Supervised machine learning

Supervised machine learning involves training an algorithm to automatically associate inputs with corresponding outputs. This is achieved through a process called *training*, where the algorithm is presented with a set of input-output pairs collectively known as the training set. By learning from these examples, the algorithm builds a mapping from inputs to outputs that allows it to make accurate predictions on new, unseen data [20].

Formally, let us assume that there are an input space $X \in \mathbb{R}^n$, where n is the size of the input space, and an output space Y, where $Y \in \mathbb{R}$ for regression problems and $Y \in [0,1]^m$ for classification problems, where there are m classes. For classification, let us assume that any output $y \in Y$ can be represented using a one-hot encoding. This means y is a vector in $\{0,1\}^m$ such that:

$$y[j] = \begin{cases} 1, & \text{if the input } x \text{ belongs to class } j, \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

Supervised learning assumes there exists an unknown function $f' : X \to Y$ which maps each input $x \in X$ to its corresponding output $y \in Y$.

A standard assumption in machine learning is that the data are generated in a stationary and stochastic manner, meaning each example (x, y) is drawn independently from a fixed joint distribution:

$$p(x, y) = p(x)p(y \mid x) \tag{2.2}$$

We further assume that each target value can be expressed through its expected value $E[y|x] = f'(x)$ and its variance ε, where $f' : X \to Y$ is the true (discriminative) function, and ε is a noise term that captures the variability or uncertainty in the observations. The goal of supervised learning is to find a function $f \in H$, where H is the hypothesis space of all possible functions from which f can be selected, that approximates f', such that the predictions $f(x)$ are close to the true outputs y according to some loss function L. In classification, the learned function f is often structured in two stages:

- $f_{in} : X \to [0,1]^m$, which maps inputs to a probability distribution over classes,
- $f_{out} : [0,1]^m \to Y$, which selects the predicted class label, often using `argmax`.

Suppose we are given a training dataset:

$$D = (x_i, y_i)_{i=1}^{N_D} \sim p(x, y), \tag{2.3}$$

where each (x_i, y_i) is drawn from the underlying distribution $p(x, y)$ and N_D denotes the size of D. To select f, a procedure called training is started. Specifically, the following steps should be taken:

– Decide hypothesis space H from which f can be selected.

– Decide a loss function $L(\cdot, \cdot)$, measuring the distance between the value y and the predicted output $f(x)$.

– Find the function f_θ of parameters θ that minimizes the loss.

The ideal goal is to minimize the structural loss, i.e.,

$$\theta^* = \arg\min_\theta \int\int L(y, f_\theta(x))dp(y|x)d(x) \tag{2.4}$$

The true distributions $p(y|x)$ and $p(x)$ are unknown, making the direct optimization infeasible. A proxy that can be used is the *empirical loss*, which, for a dataset D, means:

$$\theta^* = \arg\min_\theta \frac{\sum_{i=1}^{N_D} L(y_i, f_\theta(x_i))}{N_D} \tag{2.5}$$

where the denominator does not change the optimization process but makes the loss computation independent from the size N_D of D. In fact, the loss is generally computed as the average loss, i.e., $L = \sum_{i=1}^{N_D} L(y_i, f(x_i))$ We also refer to $f(x)$ as $\hat{y}$.

2.1.1 Metrics

To evaluate the quality of a classifier f, we need appropriate evaluation metrics. These metrics are necessary understand how well the model is performing, both during and after training. Some metrics are used during the training process to guide the optimization of the model's parameters. Others are typically used after training to assess the performance of the final classifier on a separate validation or test set. These evaluation metrics help evaluating the model's generalization ability and help in comparing different classifiers.

Regression Regression evaluation is done in terms of distance, where the goal is to minimize the distance (or error) between the predicted output $f(x)$ and the real outputs y. The two main approaches are the Mean Squared Error (MSE) and the Mean Absolute Error (MAE), defined in Equations 2.6 and 2.7 respectively.

$$MSE(D, f) = \sum_{i=1}^{N_D} \frac{(y_i - f(x_i))^2}{N_D} \tag{2.6}$$

$$MAE(D, f) = \sum_{i=1}^{N_D} \frac{|y_i - f(x_i)|}{N_D} \tag{2.7}$$

Classification Classification metrics measure how well the classifier predicts the target class. For balanced problems, i.e., problems where all classes are broadly equally represented, the most common metric is the prediction accuracy, which measures the fraction of the data correctly predicted. We formally define the accuracy as:

$$ACC(D, f) = \sum_{i=1}^{N_D} \sum_{j=1}^{m} \frac{y_i[j] \cdot f(x_i)[j]}{N_D} \tag{2.8}$$

Note that for each class j $y_i[j] = 1$ if j is the corrected class for x_i and $y_i[j] = 0$ otherwise, and $\hat{y}_i[j] = 1$ if j is the predicted class and $\hat{y}_i[j] = 0$ otherwise, meaning that, for each couple (x_i, y_i), the numerator will increase by one only if $y_i = f(x_i)$.

Accuracy measures the quality of the final prediction $f(x)$, but, due to its discrete nature, it cannot be optimized directly Therefore, it is often useful to also assess the intermediate prediction $f_{\text{in}}(x)$, which provides a more granular view of classifier quality and, thanks to its continuous nature, can be more suitable for facilitating the training process. For this reason, the Cross-Entropy (CE) is often used to evaluate the quality of the probability outputs. The CE is defined as:

$$CE(D, f) = -\frac{1}{N_D} \sum_{i=1}^{N_D} \sum_{j=1}^{m} y_i[j] \cdot \log\Big(f_{\text{in}}(x_i)[j]\Big), \tag{2.9}$$

where N_D is the number of data points and m is the number of classes.

Another natural way to evaluate classifiers is through the use of a confusion matrix. his matrix compares the predicted values against the actual values in a dataset, where each row represents the predicted class and each column the real class. In this case, for a dataset D the value of each cell of row j and column i represents the number of inputs $x \in D$ that have class j and have been classifier as i, i.e. for which $y[j] = 1$ and $f(x)[i] = 1$.

Binary classification Binary classifiers are often evaluated through the use of a confusion matrix, where values are typically labeled as $y = 1$ (positive class) or $y = 0$ (negative class). The confusion matrix in this case categorizes predictions into four groups: True Positives (TP), where the model correctly predicts $f(x) = 1$, True Negatives (TN), where the model correctly predicts $f(x) = 0$, False Positives (FP), where the model incorrectly predicts $f(x) = 1$, and False Negatives (FN), where the model incorrectly predicts $f(x) = 0$.

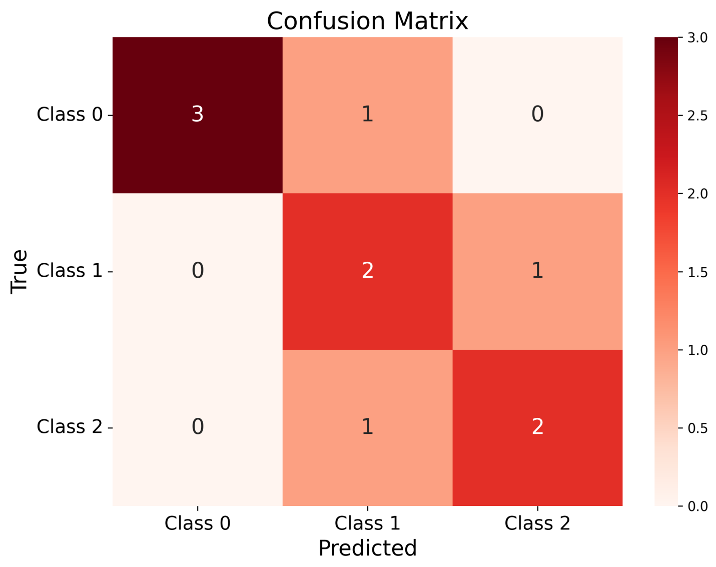


Figure 2.1: Confusion Matrix for a three-class classification problem. The values of the main diagonal represent the inputs correctly classified by the classifier.

$$\begin{aligned}
TP(D,f) &= \sum_{i=1}^{N_D} \frac{y_i \cdot f(x_i)}{N_D} \\
TN(D,f) &= \sum_{i=1}^{N_D} \frac{(1-y_i)\cdot(1-f(x_i))}{N_D} \\
FP(D,f) &= \sum_{i=1}^{N_D} \frac{(1-y_i)\cdot f(x_i)}{N_D} \\
FN(D,f) &= \sum_{i=1}^{N_D} \frac{y_i \cdot (1-f(x_i))}{N_D}
\end{aligned} \tag{2.10}$$

Starting from the confusion matrix, other important metrics can be defined: the Recall, which measures the fraction of samples from the positive class that are detected, the Precision, which measures the fraction of correctly raised alarms, and the F1 Score, which combines the two metrics. The three equations are defined as:

$$\begin{aligned}
\mathrm{PR}(D,f) &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \\
\mathrm{RC}(D,f) &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \\
F_1(D,f) &= 2\cdot\frac{\mathrm{PR}\cdot\mathrm{RC}}{\mathrm{PR}+\mathrm{RC}}
\end{aligned} \tag{2.11}$$

Finally, the cross entropy (Equation 2.9, can be redefined for binary classification as the Binary Cross Entropy (BCE), the formula of which is:

$$BCE(D,f) = -\sum_{i=1}^{N_D} \frac{y_i \cdot log(f_{in}(x_i)) + (1-y_i)\cdot log(1-f_{in}(x_i))}{N_D} \tag{2.12}$$

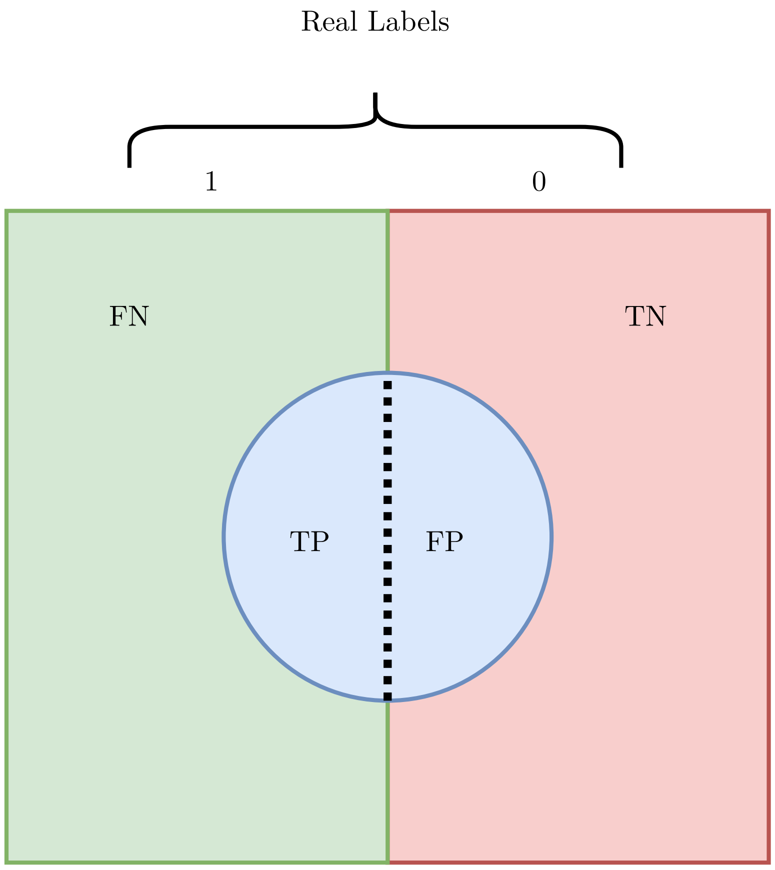


Figure 2.2: Confusion Matrix elements for binary classification. The circle represents the model's prediction, with values inside the circle being predicted as positive (i.e., $f(x) = 1$).

2.1.2 Algorithms

Linear and logistic regression Linear regression assumes a linear relationship between the input and output. The resulting equation represents the output $f(x)$ as:

$$f(x) = w \cdot x + b \tag{2.13}$$

w is called weight vector, and b is the 5. The optimal values w^* and b^* are computed minimizing the MSE over the training set D, i.e., as

$$w^*, b^* = \arg\min_{w', b'} MSE(D, w \cdot x + b) \tag{2.14}$$

Logistic regression is the extension of linear regression to classification. The algorithm works as follows: first, the function f is represented by taking a linear function of the input vector, so that

$$f(x) = \sigma(w \cdot x + b) \tag{2.15}$$

where σ is the sigmoid function, defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.16}$$

Finally, both algorithms can be regularized by applying a penalization factor to the magnitude of the weights.

$$w^* = \arg\min_w MSE(x) + \lambda |w|_K^k \tag{2.17}$$

where k is the regularization factor and $|w|_K^k$ is the L_k norm, expressed as:

$$|w|_K^k = \sum_{i \in N_w} = (\sum_{i=1}^{n} |w_i|^k)^{\frac{1}{k}} \tag{2.18}$$

Lasso regularization (also called $L1$, $k = 1$) and Ridge ($L2$, $k = 2$) are the most common.

Decision Tree Decision trees are non-parametric models that work by recursively splitting the data space, identifying the optimal split values in the data. For classification, the algorithm proceeds as follows. First, the initial dataset is identified as the root of the tree, and its entropy over a dataset $D = \{x_i, y_i\}_{i=1}^{N_D}$ is computed according to:

$$ENT(D) = -\sum_{j=1}^{m} \sum_{i=1}^{N_D} \frac{y_i[j]}{N_D} * \frac{log_2 \sum_{i=1}^{N_D} y_i[j]}{N_D} \tag{2.19}$$

where m is the number of classes.

Then, the space is cut across a feature to maximize the information gain, i.e., the difference between the original entropy and the average entropy of the two resulting nodes, which represent the two sides of the cut. Since entropy is a measure of the data's disorder, the higher the entropy, the more mixed the classes are. Decreasing the entropy helps split data among leaves that tend to be mono-class, effectively, measuring how much better the new cut divides the data compared to the previous situation. Continuous variables are split by considering thresholds between sorted unique values (i.e., splits occur at boundaries between different-class samples), while categorical variables are split by partitioning categories. The algorithm then recursively applies the same mechanism to the nodes until a splitting condition is met, such as when all samples in a node share the same label or a predefined maximum depth is reached.

Decision trees can also be employed for regression. The main difference lies in the measurements to minimize; in classification problems, the algorithm minimizes the entropy, whereas in regression problems, it uses the MSE between the value of the leaf (expressed as the average) and the value of all inputs inside the leaf.

Random forest Random forest classifiers combine multiple decision trees to improve the prediction quality. First, a set of trees is generated using random parts of the data, and observations are then classified using majority voting among the trees, or, for regression, model votes or averaging. To ensure diversity among the trees, they cannot search for the most important features when splitting a node; instead, they are limited to searching for the best feature among a randomly selected subset of features.

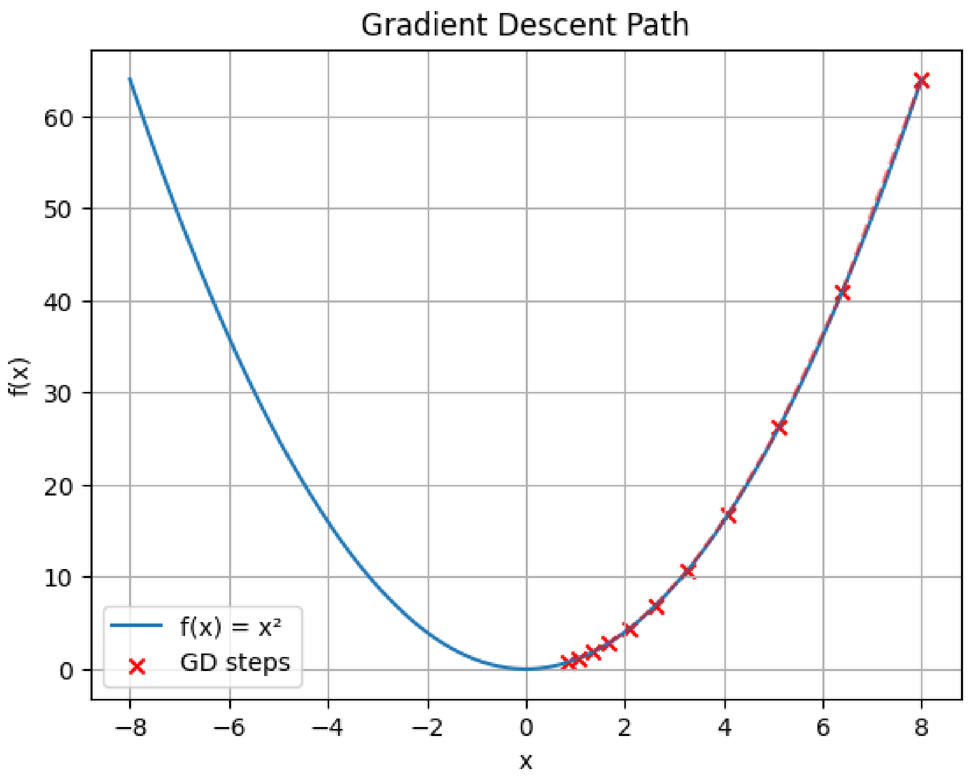


Figure 2.3: Gradient descent with fixed learning rate. At each step, the algorithm computes the function's gradient and moves in the direction of its negative. As the derivative decreases and the points approach the local minimum, the step sizes shrink accordingly.

Neural Network Neural Networks (NN), or Deep Neural Networks (DNN), are extremely powerful parametric machine learning algorithms composed of artificial neurons. On an abstract level, we can define a neural network as a topology, a loss, and an optimizer, whose combination gives rise to the desired behaviors of the network. Neural networks are composed of a set of atomic elements, called artificial neurons or neurons, typically defined by the equation $y_h = h(w \cdot x_h + b)$, where w represents the input, w is the weight vector, b is the bias term, h is the activation function and x_h the input of the neuron. Common choices for the activation function are the sigmoid function and the ReLU, where the sigmoid follows Equation 2.16 where the ReLU is defined as

$$ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.20}$$

The most straightforward architecture, shown in Figure 2.4, is the fully connected feed-forward neural network, where neurons are organized in layers, each connected to the next. The layers between the input and output layers are referred to as hidden layers, representing intermediate representations of the input data. The input data runs through the network until it reaches the last layer.

In this case, the hypothesis space H is the space of all possible weight combinations. For classification problems, the most common loss is the binary cross-entropy (Eq. 2.12), while for regression, the Mean squared error (Eq. 2.6) is used. Training is typi-

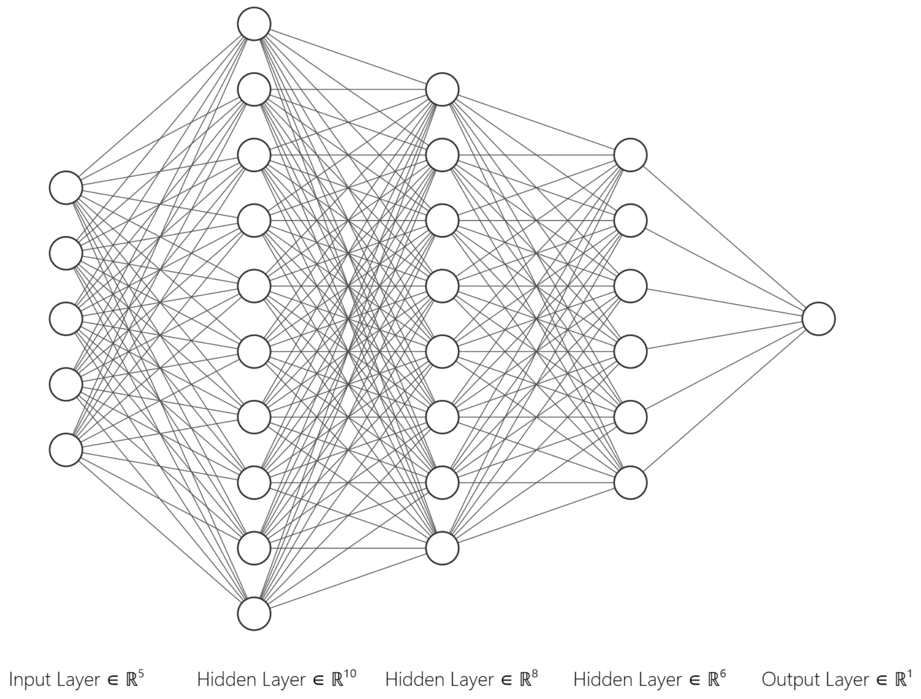

Input Layer ∈ $\mathbb{R}^5$
Hidden Layer ∈ $\mathbb{R}^{10}$
Hidden Layer ∈ $\mathbb{R}^8$
Hidden Layer ∈ $\mathbb{R}^6$
Output Layer ∈ $\mathbb{R}^1$

Figure 2.4: Neural network architecture. The blue and red edges represent positive and negative weights, respectively, and the opacity represents the magnitude of each weight.

cally performed using gradient-based optimization algorithms, most notably Stochastic Gradient Descent (SGD) and its variants, such as Adam [86]. These algorithms rely on the gradient descent principle: at each step, the gradient of the loss function with respect to the network's parameters is computed, and the algorithm takes a step towards the negative of its negative. The weights updates are then computed through the backpropagation algorithm [138], which efficiently propagates the gradients of the loss function by applying the chain rule in reverse through the network layers.

Finally, when using neural networks for classification, we can further decompose the f_{in} into two elements: $f_{logits} : \mathbb{R}^n \to \mathbb{R}^m$, which maps the input x to its representation $z = z(x)$ on the last layer of the neural network, and $f_{prob} : \mathbb{R}^m \to [0, 1]^m$ which normalizes z to obtain a probability distribution over the classes.

Recurrent Neural Networks Traditional neural networks process data one observation at a time, with no inherent notion of order or temporal dependency between inputs. However, NNs can be adapted to work in time series data, where each observation depends not only on its features but also on past events, and models need to incorporate a sense of sequence and memory. Recurrent Neural Networks (RNNs) address this problem by utilizing hidden states, which capture information about previous time steps and are employed to maintain information. During training, RNNs are typically unrolled across time, creating a sequence of computations that correspond to each time step in the input data. This structure enables backpropagation through time, a variant of the backpropagation algorithm that computes gradients across the entire sequence by applying the chain rule at all time steps. After processing the sequential data through the RNN layers, the resulting output can be further passed through one or more fully connected layers to obtain the desired output.

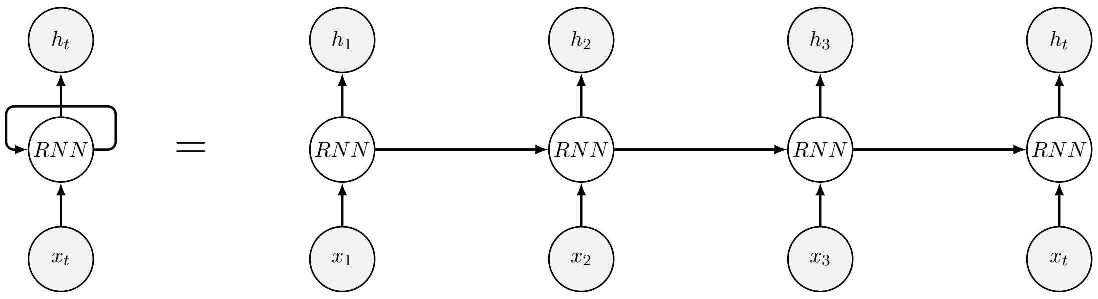


Figure 2.5: Vanilla Recurrent Neural Network architecture. At each iteration, the hidden state is passed to the next iteration, hence passing the signal from observations over time and continuing the chain of derivation over time.

2.2 Unsupervised learning

For many tasks, labeled data is not necessary or not available. For example, unlabeled data can be organized through clustering techniques to uncover hidden structures or patterns [70], or dimensionality reduction techniques such as Principal Component Analysis (PCA) [164] can be used to facilitate learning from high-dimensional datasets. However, we focus here specifically on two unsupervised applications that have significant relevance and impact in the context of this thesis. We focus on anomaly detection because it can be used as an unsupervised solution to enrich fraud detection algorithms [26] and for synthetic data generation, due to its application in generating Mimicry attacks (see Subsubsection 4.2.3.2).

2.2.1 Anomaly detection

In many applications, data follow expected patterns, and deviations from these patterns can indicate issues such as system faults, fraud, or rare events, making the detection of such anomalies a crucial goal. Unlike supervised learning, anomaly detection can also be performed in the absence of labels, as algorithms learn to model the system's normal behavior and identify any deviations as anomalies.

In unsupervised anomaly detection, this often translates to assessing the probability of an observation $x \in X$ under the data distribution $p(x)$. Specifically, anomalies are observations for which $p(x) < \tau$, with τ being a pre-defined threshold. Note that in practice, $p(x)$ is unknown, meaning that approaches may either attempt to estimate a proxy distribution $\hat{p}(x)$ or employ heuristics to identify anomalies. In practice, this can be viewed as a binary classification algorithm, where we refer to the class $y = 1$ as the class of anomalous data and $y = 0$ as the class of other observations.

2.2.1.1 Algorithms

Statistical Tests A straightforward approach to anomaly detection involves explicitly modeling the probability distribution $p(x)$ of the data and flagging points in low-probability regions as anomalies, often by thresholding based on quantiles [146]. This approach tends to struggle in high-dimensional or sparse data environments, where estimating quantiles becomes significantly challenging, severely reducing the methods' effectiveness.

Isolation Forest Isolation Forest is an ensemble-based method designed for anomaly detection [95]. It constructs multiple isolation trees, where each tree is built by recursively partitioning the data using randomly selected features and split values. The idea behind the algorithm is that anomalies should, on average, be less likely to be close to clusters of normal data, thereby making them simpler to isolate. Thus, they tend to appear closer to the root of the tree, requiring fewer splits to isolate. The anomaly score is computed as the average path length over all trees; shorter average paths indicate more anomalous instances. By leveraging randomness and ensembling, Isolation Forest reduces variance and performs well even on high-dimensional datasets.

Autoencoders Autoencoders are neural networks trained to learn and reproduce their input data by encoding and then decoding it. They consist of two main components: an encoder that compresses the input into a low-dimensional latent space and a decoder that reconstructs the original input from this compressed representation. The network is trained by minimizing the reconstruction error, i.e., the mean square error (see Equation 2.6) between the original input and the network's output. The intuition behind autoencoders is that if the reconstruction error is low, the low-dimensional layer captures an efficient representation of the original data in a lower dimension.

Since the autoencoder has been trained primarily on non-anomalous data, its representation has been optimized to capture genuine behavior. Therefore, anomalous data points are likely to have higher reconstruction errors since they deviate from the patterns learned during training on normal data. This property can be exploited to detect anomalies in the data [92].

2.2.2 Synthetic data generation

The goal of generative modeling is to produce new samples that follow the same underlying distribution as the observed data. Following Equation 2.2, we aim to estimate the probability $p(x)$ or $p(x, y)$, depending on whether we also wish to capture class-conditional structure. However, on many occasions, directly estimating high-dimensional and complex distributions is exceptionally challenging. As a result, a variety of heuristics and approximate methods have been developed to address these tasks. In recent years, deep learning tools have emerged as particularly powerful for synthetic data generation, thanks to the capability of neural networks to approximate intricate data distributions without explicitly computing densities. Specifically, we discuss here Generative Adversarial Networks (GANs) [68] and Variational Autoencoders (VAEs) [87].

2.2.2.1 Metrics

Evaluating the quality of synthetic data is a complex task, and many different metrics capture specific aspects of data quality. First, if data are designed for a specific task (e.g., classification), one can compare the performance of different classifiers on real and synthetic data. If the results are similar, synthetic data can serve as a proxy for studying how algorithms perform on real data. Second, distinguishability trains a binary classifier to label records as real or synthetic; low classifier performance (near random with high variance) indicates high fidelity, whereas high distinguishability suggests overfitting or repetition [58]. Third, univariate similarity tests (e.g., Kolmogorov–Smirnov and Total Variation [166]) ensure that each feature's marginal distribution matches between real and synthetic data, while multivariate similarity tests examine joint trends—such as pairwise correlations—to verify that inter-feature relationships are preserved. Finally, the subjective assessment of generated data is always important. This is particularly evident in image generation, where the ultimate goal is to produce images that humans perceive as authentic; however, it is also a crucial sanity check in fraud detection and other applications involving tabular data.

2.2.2.2 Algorithms

Variational Autoencoders Variational Autoencoders (VAEs) are a specialized type of autoencoder designed for generating synthetic data. VAEs are composed of three main components:

– **Encoder** Which takes as input the observation x and learns its efficient representation $\phi(x)$, expressed as a mean μ and a standard deviation.

– **Latent Space**: The input is encoded through mean and standard deviation.

– **Decoder**: Which takes the sample from the latent space and tries to reconstruct the original input.

Variational autoencoders use KL-divergence as part of their loss function. The goal is to minimize the difference between the supposed distribution and the original distribution of the dataset. The loss is therefore computed as the sum of the reconstruction loss and the KL divergence. To guarantee the differentiability over the sampling operation, VAEs apply a parametrization where sampling is expressed as

$$x = \mu + e^{0.5 \cdot log(\sigma^2)} \cdot \epsilon \tag{2.21}$$

where ϵ is a fixed stochastic node, and μ and σ are learnt and are therefore part of the backpropagation rule.

Adversarial Neural Networks Generative adversarial networks employ two neural networks, referred to as the Generator and Discriminator, respectively, which interact to learn how to generate realistic synthetic data. The idea is straightforward: the generator learns to generate synthetic data that resembles the original data, while the discriminator learns to distinguish between real and synthetic data.

The generator optimizes the generator loss L_G, expressed as:

$$L_G = -\frac{1}{m}\sum_{1}^{m} log\mathcal{D}(G(z_i)) \tag{2.22}$$

where $G(z_i)$ is the synthetic observation outputted by the generator using as input a random noise vector z_i and $\mathcal{D}(G(z_i))$ is the discriminator's estimated probability that the generated sample is real.

Conversely, the discriminator's loss is expressed as

$$L_D = -\frac{1}{m}\sum_{1}^{m} log\mathcal{D}(x_i)) - \frac{1}{m}\sum_{1}^{m} log(1 - \mathcal{D}(G(z_i)) \tag{2.23}$$

where $D(x_i)$ is the discriminator's predicted probability that a real observation x_i is real. The discriminator learns to predict real data as real and synthetic data as synthetic, and is penalized for wrong predictions in either sense.

TimeGAN TimeGAN [167] is a variation of GAN designed for generating synthetic time series, i.e., ordered sequences of data $x_1, x_2, \ldots$. Compared to traditional GANS, TimeGAN needs to model the sequential nature of the data, i.e., maintain for each possible index i the conditional distribution $p(x_i|x_{1:i-1})$.

The resulting approach, illustrated in Figure 2.6, combines four neural networks:

- The embedder E, which takes as input the original networks and projects them into a lower-dimensional latent space.
- The reconstructor $\mathcal{R}$, which maps the embedding space back to the original data space.
- The generator G, which takes as inputs noise vectors in the data space and outputs synthetic data in the latent space.
- The discriminator $\mathcal{D}$, which is tasked with distinguishing embeddings of the real data from the noise vectors generated by the generator.

The three networks are trained by minimizing three different losses. Note that the generator is autoregressive, i.e., generates all observations after the first by taking its own outputs as inputs.

- The reconstruction loss, which measures the distance between the original data and the reconstruction of their embeddings.
- The unsupervised loss, which measures the capability of the generator to discriminate between the embedding of the real data and the synthetic data output by the generator, and is analogous to the traditional generator and discriminator losses in the traditional GANs.
- The supervised loss, where the generator receives sequences of embeddings of actual data and is trained to generate the embedding of the next data in the time series, being evaluated for its prediction against the real value, i.e., on the distance between x_t and $G(x_{[i=1\ldots t]})$.

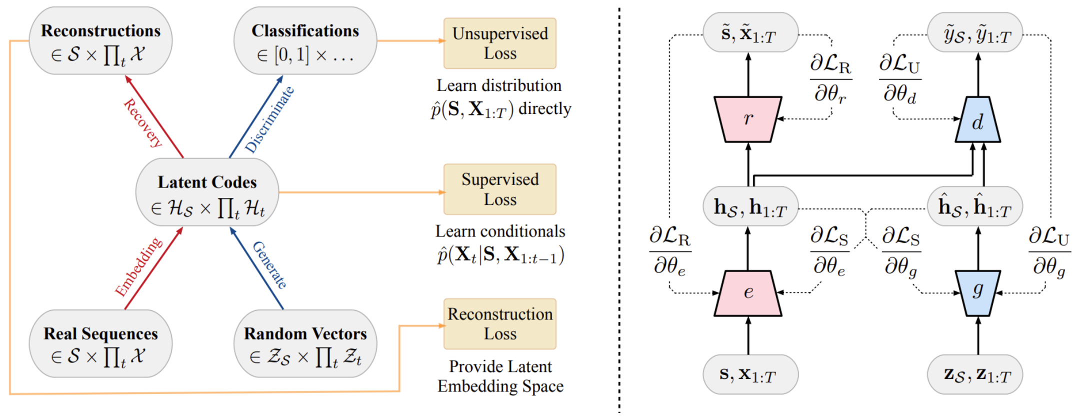

Reconstructions $\in \mathcal{S} \times \prod_t \mathcal{X}$
Classifications $\in [0,1] \times \ldots$
Unsupervised Loss
Learn distribution $\hat{p}(\mathbf{S}, \mathbf{X}_{1:T})$ directly
Recovery
Discriminate
Latent Codes $\in \mathcal{H}_\mathcal{S} \times \prod_t \mathcal{H}_t$
Supervised Loss
Learn conditionals $\hat{p}(\mathbf{X}_t | \mathbf{S}, \mathbf{X}_{1:t-1})$
Embedding
Generate
Real Sequences $\in \mathcal{S} \times \prod_t \mathcal{X}$
Random Vectors $\in \mathcal{Z}_\mathcal{S} \times \prod_t \mathcal{Z}_t$
Reconstruction Loss
Provide Latent Embedding Space
$\tilde{\mathbf{s}}, \tilde{\mathbf{x}}_{1:T}$
$\tilde{y}_\mathcal{S}, \tilde{y}_{1:T}$
$\frac{\partial \mathcal{L}_\text{R}}{\partial \theta_r}$
$\frac{\partial \mathcal{L}_\text{U}}{\partial \theta_d}$
r
d
$\mathbf{h}_\mathcal{S}, \mathbf{h}_{1:T}$
$\hat{\mathbf{h}}_\mathcal{S}, \hat{\mathbf{h}}_{1:T}$
$\frac{\partial \mathcal{L}_\text{R}}{\partial \theta_e}$
$\frac{\partial \mathcal{L}_\text{S}}{\partial \theta_e}$
$\frac{\partial \mathcal{L}_\text{S}}{\partial \theta_g}$
$\frac{\partial \mathcal{L}_\text{U}}{\partial \theta_g}$
e
g
$\mathbf{s}, \mathbf{x}_{1:T}$
$\mathbf{z}_\mathcal{S}, \mathbf{z}_{1:T}$

Figure 2.6: TimeGAN structure (on the left) and training schema (on the right). The image is taken from the original TimeGAN paper [167].

New synthetic data are generated by combining the generator, which generates synthetic data in the latent space, and the reconstructor, which maps them to time series in the data space.

2.3 Reinforcement learning

At its core, reinforcement learning (RL) is the study of how agents can make a sequence of decisions under uncertainty in order to maximize some notion of cumulative discounted reward [151]. A powerful framework to express most RL problems is the Markov Decision Process [15, MDP]. MDPs express the dynamics of a stochastic environment through a combination of states, actions, outcomes, and rewards. Formally, an MDP is expressed as $MDP = \{S, A, R, T\}$

where:

- S represents the state space, i.e., the space of all the states s in which the system can be
- A represents the action space, i.e., the space of all the actions a that the agent can take.
- $R : A \times S \times A \to \mathbb{R}$ is the reward function mapping the combination of starting and arriving states and the action to a continuous reward.
- $T : S \times A \to [0, 1]$ the transaction function, which for each state-action combination returns a probability distribution over the arrival states.

The agent interacts with the environment in discrete time steps: at each time step t, the agent observes the current state $s_t \in S$, selects an action $a_t \in A$, receives a reward $r_t = R(s_t, a_t, s_{t+1})$, and transitions to a new state s_{t+1} according to the transition distribution $T(s_{t+1} \mid s_t, a_t)$. The MDP model assumes the Markov Property, which states that the future can be determined solely from the present state, which encapsulates all the necessary information from the past.

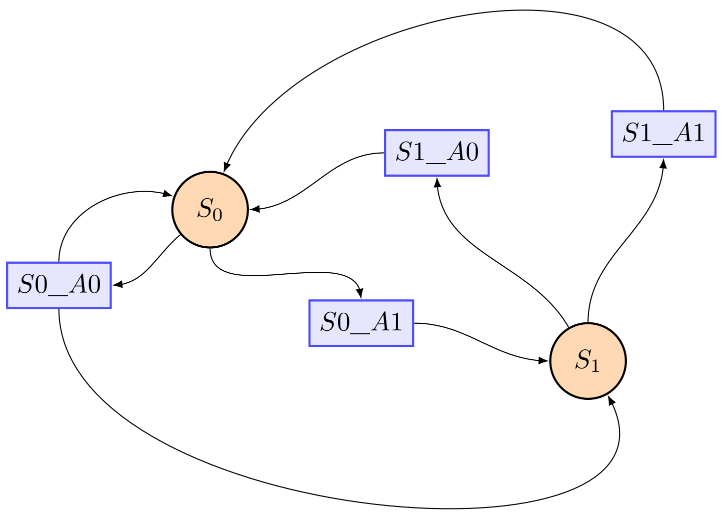


Figure 2.7: Markov decision process (MDP) example. There are two States ($S0$ and $S1$) and two actions ($A0$ and $A1$) that can be taken from each state. We represent the state-action pair as a state-action node (e.g. $S0\_A0$). Each state-action node is associated with a probability distribution over the edges starting from it. Each transition (expressed as an edge exiting a state-action node) is associated with a specific reward.

The goal of an RL algorithm is to find a policy that maximizes the expected reward. In formal terms, we define a policy π as a function that, for every state, returns a probability distribution over the action space A. We also define a discount factor γ, which measures the relative importance of rewards in the short term compared to the long term. We then define the agent's goal as finding the optimal policy π^* that maximizes $V^\pi(s)\ \forall s \in S$, i.e., the expected discounted sum of rewards under a given policy π, as shown in Equation 2.24.

$$V^\pi(s) = \mathbb{E}\left[\sum_t \gamma^t R(s_t, a_t, s_{t+1}) \mid s_0 = s\right] \tag{2.24}$$

where the action a_t is sampled from the policy π, i.e. $a_t \sim \pi$.

Note that in many cases, the exact state of the world may be unknown to the agent, who only receive a partial observation of the state. In this case, we fall in the case of Partially Observable Markov Decision Processes [121, POMDP], which are defined as $POMDP = \{S, A, R, T, \Omega, O\}$

where

- O represents the observation space.
- $\Omega : S \to O$ represents the observation function mapping the state of the world to the observation perceived by the agent.

Note that in this case, the policy takes as input the observation instead of the state.

2.3.1 Reinforcement Learning approaches

To identify the optimal policy in a reinforcement-learning problem, one can broadly follow two paradigms: value-based methods and policy-based methods. Value-based methods learn an estimate of the expected return for each state–action pair, known as the Q-value [161]. Once these Q-values are learned, a policy is derived, for instance by choosing the action that maximizes Q in each state, or by sampling from a categorical distribution issued from a softmax over Q-values [150]. Value-based algorithms are generally stable and sample-efficient in discrete action spaces, but are not readily applicable to continuous action spaces due to the categorical nature of the distribution induced by the Q-values. Extending Q-learning to continuous action spaces requires adaptations such as discretizing the action space [120]. Policy-based methods directly parameterize the policy $\pi_\theta(a|s)$ and optimize its parameters θ by gradient ascent on the expected return [163]. These methods directly output a probability distribution (categorical, normal, ...) and can therefore handle stochastic policies and continuous controls, though they may exhibit higher variance and require careful tuning.

Proximal Policy Optimization Proximal Policy Optimization (PPO) [143] utilizes an actor-critic architecture [14] comprising two networks: the actor, which learns the optimal policy $\pi = p(a|s)$, and the critic, which learns the value function for each state s. Note that the outcome of π is not necessarily an action, but rather a probability distribution over the action space.

PPO introduces a conservative policy update mechanism that prevents significant, destabilizing changes to the policy during training. Specifically, it introduces a clipped surrogate objective function, which constrains changes to small updates by using a clipping mechanism parameterized by ϵ. This ensures that each policy update remains within a safe range, improving the stability of the training process.

Formally, expressing the policy π as $\pi(\theta)$, with θ being the parameters that define policy, PPO defines the loss shown in Equation 2.25.

$$\begin{aligned} L^{\text{CLIP}}(\theta) &= -\,\mathbb{E}_t\Big[\min\Big(r_t(\theta)\,\hat{A}_t,\ \text{clip}\Big(r_t(\theta), 1-\epsilon, 1+\epsilon\Big)\,\hat{A}_t\Big)\Big], \\ r_t(\theta) &= \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}. \end{aligned} \tag{2.25}$$

The advantage estimate $\hat{A}_t$ measures the advantage of taking the action a compared to the value of the state s before taking the action and can be computed through Monte Carlo sampling of returns [114] and $r_t(\theta)$ is the probability ratio between the new policy and the old policy at time step t computed for each action $a \in A$.

The critic learns to estimate the value of each state s_t by minimizing the supervised value-function loss

$$L^{\mathrm{VF}}(\phi) = \mathbb{E}_t\Big[\Big(V_\phi(s_t) - G_t\Big)^2\Big], \\ G_t = \sum_{l=0}^{T-t-1} \gamma^l \, r_{t+l}. \tag{2.26}$$

where ϕ are the internal parameters of the algorithm, $V_\phi(s_t)$ the expected value of the state according to the critic network, and G_t the empirical return.

Finally, both networks are trained jointly by minimizing the total loss

$$L(\theta, \phi) = L^{\mathrm{CLIP}}(\theta) + c_1 \, L^{\mathrm{VF}}(\phi) - c_2 \, \mathbb{E}_t\Big[\mathcal{H}(\pi_\theta(\cdot \mid s_t))\Big], \tag{2.27}$$

where $\mathcal{G}$ is an entropy bonus to foster exploration and c_1, c_2 are hyperparameters balancing the terms.

Overall, PPO is known to require little hyperparameter tuning compared to other Deep RL methods and has been proven to perform well in a wide variety of tasks [143, 168].

Chapter 3

Credit card fraud detection

In this section, we will use the words "merchant" and "terminal" interchangeably, as well as the words "customer," "card," and "cardholder." While this is a simplification, the definition suffices to capture the main features of the domain's behavior.

Online payments can be viewed as a group of cardholders, each associated with a card, that connect to a set of terminals to perform some payments. In formal terms, we assume that a payment system comprises $\bar{C}$ cards/customers $\{c_1, c_2, \ldots, c_{\bar{C}}\}$ and $\bar{M}$ merchants/terminals $\{m_1, m_2, \ldots, m_{\bar{M}}\}$. We identify a transaction as $x_{i,j,t}$, where i represents the cardholder's index, j represents the merchant's index, and t represents the time. We then define two functions $\psi_c(i,t)$ and $\psi_m(j,t)$ that take as inputs the time and a customer and a terminal, respectively, and return all the transactions performed with that card or terminal before time t. We therefore define the transaction $x_{i,j,t}$ (or x) as:

$$x_{i,j,t} = \{x'_{i,j,t}, t, F_i, \psi_c(i,t), F_j, \psi_m(m,t)\} \tag{3.1}$$

where $x'_{i,j,t}$ are transaction's features such as the amount, t is the time, F_i are the card's features like the type of the card, F_j are the terminal's features like the country, and $\psi_c(i,t)$ and $\psi_m(j,t)$ return all the transactions before time t of the card c_i and the terminal m_j respectively.

We define a feature engineering process $\phi(x_{i,j,t})$, which takes as input the transaction and returns a set of aggregations. Specifically, it aggregates transactions over the customer [30] and terminal [101]. Consequently, credit card fraud detection systems divide each transaction into three groups of features [1, 159]:

- Transactions features, such as the amount and timestamp;
- Card features. This group includes both card characteristics, such as the card brand and card-based aggregations.
- Terminal features. This group includes terminal characteristics such as the country brand and terminal-based aggregations.

Since fraud detection is a binary classification problem [92], each transaction x is associated with a label $y(x_i) = \{0, 1\}$, where $y(x_i) = 1$ if the transaction is a fraud, and $y(x_i) = 0$ otherwise. The goal, coherently with the definition of classification reported in Section 2.1, is to train a classification function f mapping the input x to the output $y(x)$.

Ignoring, for simplicity, the human element of fraud detection, we can group the fraud detection classifier into three elements: the machine learning classifier f_{ML}, the statistical classifier f_{ST}, and the rule-based F_{RL} classifier. The feature engineering process ϕ is itself split into three elements: ϕ_{ML}, ϕ_{ST} and ϕ_{RL}, one for each of the classifier. We then say that the FDS raises an alarm if any of the three classifiers raises an alert, i.e.:

$$f(x) = 1 \Leftrightarrow f_{ML}(\phi_{ST}(x)) \vee f_{ML}(\phi_{ST}(x)) \vee f_{RL}(\phi_{RL}(x)) = 1 \tag{3.2}$$

Compared to the formulation of supervised learning presented in Section 2.1, credit card fraud detection presents multiple challenges that make the task exceptionally challenging. First, the assumption that each observation is independent of the others does not hold here: customers and terminals interact and influence each other, and transactions performed with the same card or on the same terminals are linked.

Genuine transactions far outnumber fraudulent ones, leading to an issue known as *class imbalance* [47]. This has a significant impact on the effectiveness of algorithms and the validity of metrics, making adaptations to imbalanced learning a necessity for fraud detection systems. Another significant challenge is *concept drift*, defined as a change in the data generation process that renders previously learned patterns obsolete over time, severely hindering the performance of machine learning algorithms [172]. Both challenges require significant machine adaptations in techniques and learning patterns, with substantial changes needed in the preprocessing, training, evaluation, and monitoring phases.

3.1 Class imbalance

Imbalanced learning, i.e., classification on datasets where one class severely outnumbers the others, plays an important role in multiple domains, such as rare events prediction [162], intrusion detection [11], churn prediction [25] and fraud detection [49].

When data are highly imbalanced, the minority class may be significantly more important than the others. In fraud detection, not detecting a fraud is generally worse than raising a false alarm. As such, some of the metrics discussed in Section 2.1 cannot be applied [92]. In a field where genuine transactions comprise more than 99% of the data, a model that always predicts the genuine class would be completely useless, yet still achieve over 99% accuracy. Two primary metrics should be used to evaluate fraud detection algorithms: the Precision-Recall Curve (PRAUC) [43, 61] and the precision top K TP@K [43].

The Precision-Recall curve (PR curve) is obtained by plotting the precision against the recall for all the different classification thresholds $\tau_1, \tau_2 \ldots$. The area under the curve (AUC) is then computed as the weighted mean of precisions achieved at each threshold, using the increase in recall from the previous threshold as weight [23]. The resulting formula can be expressed as

$$PRAUC = \sum_{\tau}((RC_\tau - RC_{\tau-1}) * PR_\tau) \tag{3.3}$$

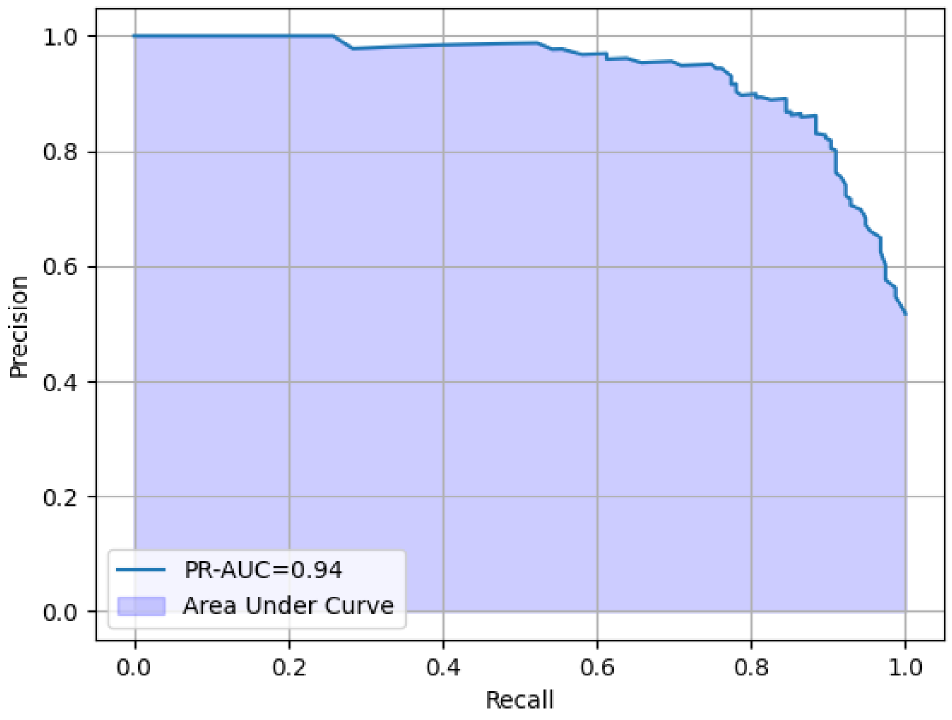


Figure 3.1: Example of a precision-Recall AUC. As the thresholds become progressively lower, the recall decreases and the precision increases.

where RC_τ and PR_τ are the precision and recall at threshold τ respectively. Compared to other metrics such as the $F1$ score, the PR-AUC has the main advantage of allowing for a comparison of the classifiers that is independent of the chosen threshold.

The precision top K $TP@K$ is expressed as $\frac{TP_K}{K_K}$ indicates the number of true positives in the K top returned alerts. The idea behind the metric is that in a field where humans need to supervise flagged transactions, the maximum throughput they can process can be roughly measured as K transactions per unit of time, where K and the time depend on the number of investigators, the time required to process an alarm, and other company-specific characteristics.

3.1.1 Algorithms

We can categorize the approaches used to address class imbalance into two main families: *algorithm-level* techniques and *data-level* techniques [88]. Algorithm-level techniques address the problem by modifying the classifiers to ensure robust accuracy in the face of data imbalance. This includes adjusting the cost matrix to increase the importance of misclassified samples from the minority class [153] and utilizing ensemble learning techniques, which train multiple learners on different parts of the training set to artificially increase the representation of observations from the minority class [98].

Data-level approaches balance the training set before classification. The two most common in the literature are *undersampling*, which removes a certain number of majority class observations from the training set, and *oversampling*, which artificially adds minority class samples to the dataset.

3.1.1.1 Undersampling

At a high level, undersampling techniques are based on the assumption that, when data is severely unbalanced, even a relatively small part of the majority class observation may be sufficient to correctly model the class distribution. The primary challenge for undersampling methods is selecting the observations to use. The simplest approach, also called Random Undersampling [111], involves randomly selecting items from the majority class. This solution is simple, efficient, and may be effective, but it carries a high risk of removing observations that are crucial for training the algorithm [63]. Other approaches, such as [109], select examples based on the distance of majority class examples to minority class examples. The main idea is that the majority samples close to minority ones are more likely to be close to the class boundary, and are therefore more informative than observations surrounded by others of the same class.

Undersampling tends to work better for large datasets [47] where the likelihood that the maintained subsample carries enough signal is higher. The main issues with this approach concern the potential loss of useful information given by the removal of observations from the training set and the potentially damaging change in the prior of the majority class [135]. An effective solution to the mentioned problem is EasyEnsemble [98], which exploits an ensemble of classifiers trained on different balanced subsamples of the original data. Each subsample is obtained through a random undersampling of the majority class, reducing the risk of losing information. The ensemble nature of a Random Forest algorithm makes it particularly suitable for the implementation of an EasyEnsemble technique: the model obtained is called Balanced Random Forest, and each of its Decision Trees is trained on a different data subsample [27]. The role of undersampling has been extensively studied in previous works [47, 26], showing good results in the fraud detection scenario.

3.1.1.2 Oversampling

A simple oversampling is achieved by randomly sampling and duplicating the observations of the minority class. This method increases the proportion of minority samples but is prone to overfitting. More complex oversampling methods, such as SMOTE [36], use data locality to generate new minority samples through interpolation. For instance, SMOTE randomly selects close samples in the feature space and interpolates between them to generate a new artificial sample. SMOTE has achieved considerable success in the oversampling literature, and numerous variants have been proposed over the years. Some of the most cited and renowned techniques try to limit the application of SMOTE to some defined subspaces. For example, ADASYN [74] uses a weighted distribution for different minority class examples, where more samples are generated for classes harder to learn, while Borderline-SMOTE [73] samples from all classes (and thus not only from the minority class) only next to the border between different classes, as samples in those areas are more likely to be informative for the classifier.

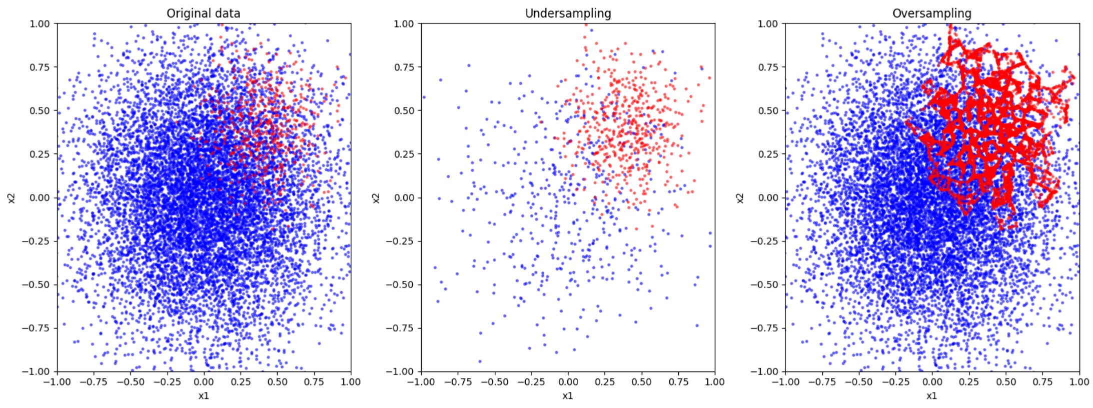


Figure 3.2: Resampling example with original data on the left, random undersampling in the middle, and oversampling with SMOTE on the right

A more recent version is K-Means SMOTE [90], which employs the k-means clustering algorithm, together with SMOTE, to rebalance skewed datasets. Such a combination aims to mitigate the impact of noise and addresses both inter-class and intra-class imbalances. The method consists of three steps: clustering, filtering, and oversampling. The first step consists in clustering the input space into k groups. The second step selects the clusters with a high proportion of minority class samples and assigns more synthetic samples to generate clusters where minority samples are sparsely distributed. The third step applies SMOTE in each selected cluster. Unfortunately, most SMOTE implementations do not allow maintaining the structure in the data, as the interpolation between admissible observations may not be an admissible observation (for instance, you cannot have a non-integer cardholder age). More generally, such methods heavily rely on the data locality assumption and may be detrimental in cases where the minority class has a complex distribution.

Generative methods for oversampling are based on the assumption that samples from the minority class follow a specific distribution [80]. The idea, common to Generative Adversarial Networks (GAN) [68] and Variational Auto Encoders (VAE) [87], is to model the distribution from historical data and use the estimated distribution to sample as many artificial minority observations as required.

The adoption of GAN, explained in Subsection 2.2.2, is considered a promising approach for fraud detection problems [64]. An interesting example is CT-GAN [165], a GAN-based method that models the distribution of tabular data and samples rows from this distribution. CT-GAN authors propose a model-specific normalization method to handle complex distributions, including non-Gaussian and multimodal ones. Furthermore, a conditional generator is proposed to address the imbalanced discrete columns that are not adequately handled by traditional GANs.

The existence of multiple techniques for dealing with imbalanced learning in fraud detection led to the necessity of analyzing and comparing them. A recent survey [3] shows that SMOTE is the most used resampling technique in fraud detection, but suggests that no single algorithm consistently outperforms all the alternatives. An empirical comparison of three resampling techniques in the context of fraud detection is performed in [140]. The authors compare Random Undersampling, SMOTE, and a variant of SMOTE called SMOTE Tomek [81], together with multiple feature engineering techniques. Interestingly, this work shows that Random Undersampling achieves the best performance among all algorithms, with SMOTE being the second-best performer. However, all performances are similar and significantly improved compared to the baseline, further showcasing the importance of resampling techniques for fraud detection.

3.2 Concept drift

A common assumption in machine learning is that the data are independent and identically distributed (i.i.d.), meaning they are drawn from the same distribution. The traditional assumption is that training and test data follow the same distribution $p(x, y)$ that the model sees at production time. In many applications, however, this is not the case. In fraud detection, in particular, customers' habits change over time, necessitating that the detection system adapt or become obsolete.

Concept drift, in turn, may be virtual if it only affects the data distribution $p(x)$ but not the conditional distribution $p(y|x)$ or real if $p(y|x)$ is also affected. Of the two, real shifts are the most dangerous; most machine classification algorithms are trained to build estimators of the conditional probability, and a shift in this relationship can severely degrade model performance.

Concept drift adaptation techniques can be grouped into two main families: active adaptation, where models attempt to detect changes in the data that trigger a retraining procedure, and passive approaches, where the system continuously retrained to incorporate new knowledge into the model. Active approaches, which monitor the classification error or the data directly [22, 5], tend to be suited when the model remains generally stable between abrupt changes.

More commonly used in fraud detection [45], passive approaches continuously adapt to the changing streams. This involves continuous retraining of the algorithm, either by simply adding the new data to the training set, or by using sliding windows, where only the most recent part of the training set is used by the algorithm: i.e.,

$$TR = \{\{x_i, y_i\} \quad \forall\{x_i, y_i\} \in D \qquad s.t. \quad t - x[t] < \tau_\Delta\} \tag{3.4}$$

where t is the current time and τ_Δ is the size of the sliding window. Alternatively, ensemble methods could be used, training different models on different time intervals and combining them to maximize the performance on recent incoming data [122].

3.3 Datasets availability

Financial data is notoriously hard to access due to its economic value and the need to comply with privacy regulations. This severely limits researchers' access to real datasets, making it exceptionally challenging to test techniques and share results. For fraud detection, two main approaches have been attempted to mitigate this challenge: data anonymization and synthetic data generation [8].

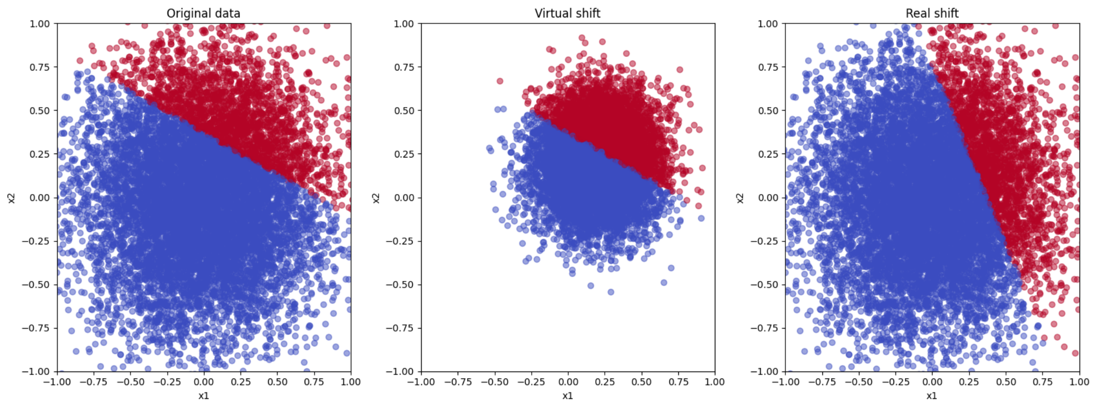


Figure 3.3: Example of concept drift in the data with original data (on the left), real change (in the middle), and virtual change (on the right).

Concerning data anonymization, Dal Pozzolo, Caelen, Johnson, and Bontempi [48] proposed the Kaggle ULB machine learning dataset, which contains over 250K real transactions performed in September 2013 by European cardholders, anonymized through a Principal Component Analysis transformation [105, PCA]. Due to the use of PCA to anonymize them, it contains only numerical data and presents no missing values. Despite its relative simplicity, the extreme imbalances in the data (492 frauds out of over 200,000 transactions) make it one of the most widely used datasets on Kaggle and, by far, the most frequently referenced when working on fraud detection. Similarly, Howard, Bouchon-Meunier, CIS, inversion, Lei, Lynn@Vesta, Marcus2010, and Abbass [76] made the IEEE-CIS payment transaction dataset available, which is widely used in the community for benchmarking fraud detection methods. The IEEE-CIS dataset contains both direct and aggregated features, which are anonymized to avoid privacy leaks.

In comparison with anonymised data, generators offer the significant advantage of more fine-grained access, as customer identities are known. This allows researchers to group transactions by customer or by terminal and to choose the aggregation level. It is important to acknowledge that any approach requiring customer-level operations can only be implemented in this manner. Two notable examples of such generators are the MLG generator [92],and the Federal Reserve CardSIM generator [6], both of which operate through simulation. Customers interact with terminals to create transactions according to probabilistic rules. Where the two approaches differ is in the way fraud is introduced; the MLG generator employs predetermined patterns, while CardSym uses a Bayesian approach to differentiate between genuine and fraudulent transactions. Despite the relative recency of works in the field, generators have been increasingly utilized in the publication of fraud detection research [123] to ensure the reproducibility of published results.

Chapter 4

Adversarial Machine Learning

Adversarial machine learning is firmly based on an accurate understanding of plausible threats. Poor risk modeling can have catastrophic consequences, ranging from a false sense of security to designing expensive defenses that address the wrong risks. In computer security, risk modeling is typically achieved through a careful design of the *threat model* [155], which involves using abstract models to identify and address security problems [147].

As such, this chapter starts a taxonomy of adversarial machine learning through threat modeling, grouping attacks based on the assumed attackers' *goal*, *knowledge*, and *capability* [18] and connecting it with the parts of the machine learning lifecycle that are most susceptible to each attack. Later, we will focus on the type of attacks that pose the most significant threat to fraud detection, presenting the main metrics used to evaluate the threat they pose, and the solutions proposed in the literature.

4.1 Taxonomy

Attack Time Malicious agents can attack the model at various stages of the process, depending on their goals and capabilities. First, they can interfere with the data collection phase by modifying or inserting some points in the training set. Such attacks, called *poisoning* [17, 144], aim to change the model's decision boundaries by targeting its training data [18]. Attackers can also target different parts of the machine-learning pipeline. For instance, they target the learning process when it is outsourced to untrusted sources [71, 118]. If attackers have no control over the algorithm's training process, they can still exploit it at runtime to deceive the classifier and expose existing errors in its decision boundaries. These so-called *evasion attacks* require weaker assumptions about the attackers' capabilities and pose a more realistic threat in fields where the test set can be safely assumed to be protected [104].

Goal In a system-centered view, the first question when designing threat models is which property of our system is at risk, i.e., the likely security violations we can face [13]. A helpful acronym is that of CIA: Confidentiality, Integrity, and Availability [141]. Systems must not leak information to unauthorized users (Confidentiality), must not allow unauthorized modification of information (Integrity), and must be able to serve genuine users without interruptions (Availability). The same principles can be applied to machine learning systems and are the first step in defining most threat models [13, 77, 18, 41, 33]. Specifically, we can categorize the attackers based on their intended security violations. For security applications, where the target model is often a detector (for malware, intrusions, frauds . . .), compromising the system's integrity means having some or all malicious points constantly bypass the system. Availability attacks may instead play on the false positive, i.e., the fraction of observations incorrectly depicted as fraudulent.

Another common distinction is that between *untargeted* attacks, which aim to reduce the model's accuracy in general, and *targeted* attacks, which aim to reduce the accuracy specifically for a particular class.

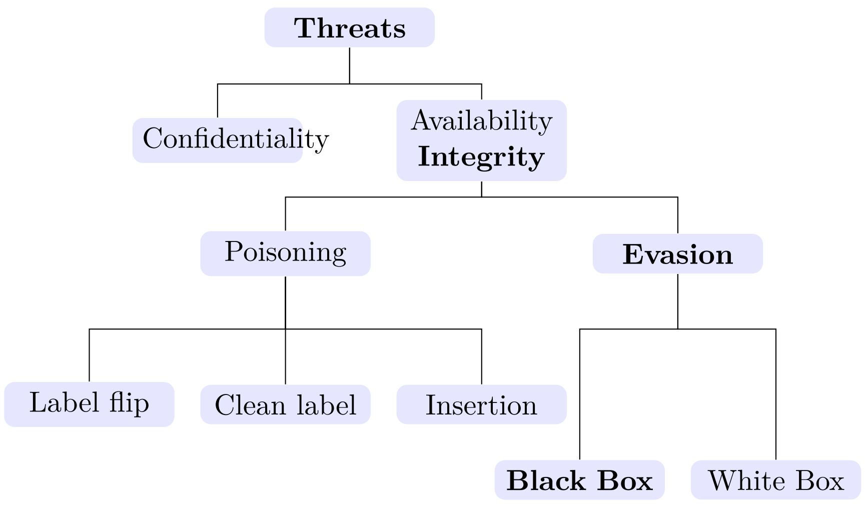


Figure 4.1: Hierarchical representation of threats posed by adversarial attacks in machine learning. In bold, the threats relevant to fraud detection.

Knowledge A key factor in threat modeling is the attacker's level of knowledge [16, 77], typically categorized as complete knowledge (White Box), partial knowledge (Grey Box), or zero knowledge (Black Box) [33, 50]. In white-box attacks, attackers have knowledge of both the structure and the function of the target classifier f_θ. Grey-box attacks, instead, do not know the parameters, but may have access to other information, such as the hypothesis space H from which f is selected, the training procedures used to train the target classifier. In Black Box, they have no internal knowledge and rely solely on output labels. Finally, under certain threat models, attackers can be assumed to have access to a training set TR' drawn from the same distribution as the training set TR on which f has been trained [52].

Capabilities The last axis to group adversarial attacks concerns attackers' capabilities, i.e., how they can interact with the system. For poisoning, attacks can happen by *insertion* of malicious data or *update* of existing data. Updates can occur at both data collection and inference times. For data collection, update attacks mean that the attacker can only modify observations already present in the training set. To do so, they can either modify the label of specific points (*label flip* attacks) [144, 72], or they can adjust their features without changing the label (*clean label* attacks) [130]. A label flip attack, for instance, would be modifying the label of certain transactions in the training set from fraud to genuine, hence moving the classification boundaries and making similar transactions more likely to be classified as genuine in the test set. A clean label attack, instead, would be modifying the features of a fraud in the training set. As the transaction features change, so do the boundaries in the data space.

Evasion attacks, instead, can be grouped by the way they craft their attacks. First, attacks can be generated by modifying existing observations with the goal of making the target classifier misclassify them, or they can create new observations from scratch. In attacks against image recognition systems, for example, attackers can either modify an image to force the system to misclassify it, or they can create a new image from scratch that, despite belonging to one class, is classified as belonging to a different one. In the overwhelming majority of attacks, the primary threat model concerns update attacks, where attackers modify an image to bypass a classifier while remaining undetected by the classifier [69, 152].

Notably, update attacks assume that the updated observation must be somewhat "similar" to the original one. Similarity can be assessed using some distance in the feature space, through indistinguishability for human eyes or application-specific losses, such as preserving important semantical characteristics of the original data [59]. Since most attacks work by solving optimization problems, they introduce the distance measure in the loss or use it to impose a constraint on the maximum perturbation allows by the attacker. For these purposes, the L_0, L_1, L_2, and L_∞ norms are the most used.

4.2 Evasion attacks

4.2.1 Problem formulation

Let us define a training set D_{TR}, and let us assume that an attacker wants to target a classifier f of parameters θ, which has been trained on a training set D_{TR} according to the training procedure defined in Chapter 3. Let us also call L the loss of the classifier f. Without loss of generality, let us assume that L is the cross-entropy loss, as defined in Equation 2.9.

We assume that an attacker has access to a test set $TS = \{x_i, y_i\}_{i=1}^{N_{TS}}$, and that it can modify the inputs $x_1, x_2, \ldots$ through a function $g : X \to X$ that takes as input an input x and returns an adversarial observation $x_{adv} = x + g(x)$. Let us call $TS_{adv} = \{g(x_i), y_i\}_{i=1}^{N_{TS}}$ the test set on which the attacker has modified every observation.

Assuming for simplicity that we only consider inputs on which the classifier predicts the correct class, we can say that an untargeted attack is successful on an input x if and only if $f(x + g(x)) \neq f(x)$. Formally, for an input x and a function g, we define a success indicator $\mathcal{S}$, which, for untargeted problems is:

$$\mathcal{S}(x, f, g) = \begin{cases} 1, & \text{if } f(x + g(x)) \neq f(x) \\ 0, & \text{if } f(x + g(x)) = f(x) \end{cases} \tag{4.1}$$

For targeted attacks, instead, we say that the attack is successful if the predicted class is equal to a desired label $\bar{y}$, i.e., if $f(x + g(x)) = \bar{y}$, making the computation of the success indicator:

$$\mathcal{S}(x, f, g, \bar{y}) = \begin{cases} 1, & \text{if } f(x + g(x)) = \bar{y} \\ 0, & \text{if } f(x + g(x)) \neq \bar{y} \end{cases} \tag{4.2}$$

Let us also define, for now, a set of constraints $k(x)$ such that each adversarially modified input must lie within the constraints, i.e., it must be true that $x+g(x) \in k(x)$. In practice, this generally takes the form of two main constraints:

– Domain constraints, which enforce that the adversarial input x_{adv} lies inside the domain X. In case of images, this translates into the *box constraints*, that enforce that each dimension of the adversarial attacks falls into the 0-1 range, i.e., $X \in [0, 1]^n$, where the value 1 has been computed as the normalization of 255, which is the maximum value each pixel can reach in image representation [39]. We call this type of constraint box constraints.

– Distance constraints: attacks need to be within a certain distance from the original input. This can be expressed as $d(x, x_{adv}) < B$, where $d : X \times X \to \mathbb{R}$ is a symmetric distance metric, and B is the perturbation budget, i.e., the max distance allowed. The most common measurement for the distance is the L_p norm, defined as in Equation 2.18.

We can now define the constraints formulation as:

$$x_{adv} \in k(x) \Leftrightarrow \begin{cases} x_{adv} \in X \\ d(x, x_{adv}) < B \end{cases} \tag{4.3}$$

Note that some attacks do not enforce the distance constraints, but rather aim at defining the successful attack within the closest distance from the original input [29, 38]. The domain constraints, however, are generally always enforced. Concerning the choice of the maximum allowed iteration B, the classical assumption is that B should be small enough to make the attack unnoticeable. As an example, in computer vision, this could mean generating a perturbation that humans cannot (easily) notice, but the specific value of the constant, as well as the evaluation, is highly empirical.

Black-box attacks introduce an additional layer of complexity compared to white-box settings: the attacker does not have direct access to the internal parameters or architecture of the target model f. Consequently, the attacker must first acquire knowledge about f through indirect means. We can identify two main families of strategies:

– **Surrogate modeling:** The attacker constructs a substitute model $\hat{f}$ that approximates f. This can be done either by collecting a dataset [29, 52] or by querying f [126], i.e., conducting preliminary exploratory attacks aimed solely at understanding the behavior of f. In both cases, once trained, $\hat{f}$ can be used to craft attacks which are then used against the original classifier f.

– **Local estimation:** Alternatively, the attacker may focus on estimating local properties of f, such as decision boundaries or gradients, in the neighborhood of a specific input [39, 24]. With a slight abuse of notation, we call this local approximation as $\hat{f}_{\text{local}}$. In this case, attacks for each input can be performed using the local approximation $\hat{f}_{\text{local}}$ as a local proxy of f.

Among black-box attacks, we call *decision attacks* those attacks that, when querying a model, require to observe only the final class prediction $f(x)$, as opposed to scoring-based attacks, that require to observe $f_{in}(x)$ or even $f_{logits}(x)$ [24].

4.2.2 Metrics

A simple metric for both untargeted and targeted attacks is the Success Rate (SR), i.e., the percentage of successful attacks among the attempted ones [89]. Recalling the attack success indicators defined in Equation 4.1 we can define the SR for an untargeted attack as:

$$SR(TS, f, g) = \sum_{i=1}^{N_{TS}} \frac{\mathcal{S}(x_i, f, g)}{N_{TS}} \tag{4.4}$$

From Equation 4.2, we then define it for targeted attacks as:

$$SR(TS, f, g, \bar{y}) = \sum_{i=1}^{N_{TS}} \frac{\mathcal{S}(x_i, f, g, \bar{y})}{N_{TS}} \tag{4.5}$$

Let us now define a more practical implementation of the success rate, called constrained success rate (CSR), where we enforce the constraints defined in Equation 4.3. We can now define the CSR for untargeted attacks as:

$$\begin{aligned} &CSR(TS, f, g) = SR(TS, f, g) \\ &\qquad\qquad \text{subject to} \;\; x_i + g(x_i) \in k(x_i), \quad \forall i \in \{1, \dots, N_{TS}\}. \end{aligned} \tag{4.6}$$

where the computation for targeted attacks comes from applying equation 4.5 instead of 4.4 inside equation 4.6.

Finally, other works measure the minimal perturbation size to craft successful attacks. The resulting metric, Empirical Robustness (ER) [115], is measured as follows:

$$\begin{aligned} ER(TS, f, g) &= \sum_{i=1}^{N_{TS}} \frac{\min_g d(x_i, g(x_i))}{N_{TS}} \\ \text{subject to} &\begin{cases} \mathcal{S}(x_i, f, g) = 1 \\ \qquad x_i + g(x_i) \in X \quad \forall x_i \in D_{TS} \end{cases} \end{aligned} \tag{4.7}$$

In the case of binary classification, the two indicators of success (targeted and untargeted) generally overlap. Taking fields like malware detection as an example, attackers aim to modify elements from the malicious classt that are correctly classified as such ($y = 1$, $f(x) = 1$). In this case, the target class is the genuine class, i.e., $\bar{y} = 0$. At the same time, untargeted attacks can only flip the prediction from fraud to genuine, i.e., $\mathcal{S}(x, f, g) = 1 \Leftrightarrow \mathcal{S}(x, f, g, \bar{y}) = 1$.

Finally, black-box attacks assume no prior knowledge of the target classifier and need to estimate the function f to craft successful frauds. We need to define a set of metrics to measure the cost of obtaining the required information. For surrogate models, this can be generally translated into the properties of the dataset TR' used to train the surrogate $\hat{f}$, with questions such as whether TR' belongs to TR, whether its drawn from the same distribution $p(x,y)$ or a similar why and its size being important metrics to define the applicability of the attack. For query-based attacks, we typically use the total number of queries [38], i.e., the average number of queries performed per input. Note that since query-based attacks build a local approximation of the model, the number of queries required is approximately the same for each input, meaning it grows linearly with the size of TS.

4.2.3 Algorithms

We present here the principal adversarial attacks in the literature. Most attacks have been designed in the context of computer vision, where each input x is a one or three-channel tensor representing an image [29].

We will first discuss White-Box attacks, then Black-box attacks, and finally, in Section 4.3, we will discuss adversarial attacks in the domain of credit card fraud detection. Finally, unless explicitly stated otherwise, we assume here that the target classifier f is a neural network.

4.2.3.1 White-Box

L-BFGS Arguably the first adversarial attack to have eve been designed, L-BFGS [152] is a targeted attack designed to minimize the L_2 loss. For each input x, the attack starts by defining the adversarial attack problem as:

$$\begin{aligned} g^*(x) &= \arg\min_g |x, x+g(x)|_2^2 \\ \text{subject to} &\quad \begin{cases} \mathcal{S}(x_i, f, g, \hat{y}) = 1 \\ x_i + g(x_i) \in X \end{cases} \end{aligned} \tag{4.8}$$

where $g : X \to X$ is the perturbation function that takes as input a transaction x and returns the perturbation vector $g(x)$ such that the adversarial attack is $x + g(x)$.

Rather than optimizing the problem directly, which is made harder by the high non-linearity of the constraint $\mathcal{S}(x_i, f, g, \hat{y}) = 1$, the authors propose to reformulate it as:

$$\begin{aligned} &\min_C C|g(x)|_2^2 + L(x+g(x)f, \hat{y}) \\ &\quad s.t. \quad x + g(x) \in X \end{aligned} \tag{4.9}$$

where the loss element L is the Cross Entropy defined in Equation 2.9. Since large values of C may lead to underweighting the loss element and not leading to a successful attack, the optimal (minimum) value of C is found through binary search. The optimization problem is then solved through the use of Limited-memory BFGS (L-BFGS), a quasi-Newton optimization algorithm using both first and second order derivatives to guide optimization [94]. Since quasi-Newton optimization algorithms use information about the gradient for their optimization, the target algorithm must be gradient-based.

Fast Gradient Sign Method Fast Gradient Sign Method (FGSM) [69] is one of the first adversarial attacks designed, and one of the simplest. Originally designed to minimize the L_∞ distance, it can also be adapted to minimize the L_2 distance.

The main idea of the algorithm is simple. Let us take a trained neural network classifier f and let us use the cross entropy loss (See Subsection 2.1.1) as its loss L. The gradient ∇L models the direction of maximum growth of the loss. Therefore, if we take a small step inside X in the direction of the gradient, we maximize the loss. By doing so, we maximize the distance between $f(x)$ and y, which may lead to a label flip and a misclassification.

Therefore, FGSM simply takes for each pixel a single step in the gradient direction. In the original L_∞ formulation, the magnitude for each pixel is set at a value ϵ, which represents the maximum perturbation C following the general formulation of Equation 4.10. Formally, the equation behind FGSM is the following:

$$g^*(x) = x + \epsilon \cdot sign(\nabla L(f(x)) \tag{4.10}$$

Due to the gradient-ascent procedure, FGSM required the algorithm to be gradient-based.

Projected Gradient Descent Projected Gradient Descent (PGD) [107] is an iterative extension through for the L_2 norm of FGSM, where the amount of the perturbation iteratively increases to cause the misclassification. Starting from $x_{adv}(0) = x$, each iteration is computed as:

$$x_{adv}(t+1) = Proj[x_{adv}(t) + \epsilon \cdot sign(\nabla L(\hat{f}(x_{adv}(t)), y(x))] \tag{4.11}$$

where we define as $x_{adv}(t)$ the value of the adversarial attack at time t, and the projection function $Proj$ projects adversarial examples into the $B - ball$ of acceptable changes. The attack stops when a successful attack has been found or after a predetermined number of iterations. Overall, PGD is a slower but more effective variant of FGSM.

DeepFool Let us consider a binary classifier f. Assuming the classifier linearity, the perturbation $g(x)$ maximizing the attack's effectiveness with the smallest perturbation square norm can be expressed as:

$$g(x) = -\frac{f(x)}{\|w\|_2^2} \cdot w \tag{4.12}$$

where f is the target classifier, and w is the parameter vector of the linear classifier. From this consideration, researchers developed Deepfool [115], an untargeted attack designed to be effective mainly against Neural Networks. The main idea is that neural networks can be locally approximated by linear models. Hence, the perturbation in 4.12 can be decomposed into a set of local linear attacks, where at each step t, w can be approximated by $\nabla\hat{f}(x(t))$. The resulting formulation is the following:

$$g(x_{adv}(t)) = -\frac{f(x_{adv}(t))}{|\nabla L(f(x_{adv}(t)), y)|_2^2} \cdot \nabla L(f(x_{adv}(t)), y) \tag{4.13}$$

where $x_{adv}(0) = x$, $\hat{f}(x)$ is the decision function, and $\nabla L(f(x_{adv}(t)), y)$ the gradient of the classifier's loss with respect to the input $x_{adv}(t)$. The attack can also be extended to multiclass classifiers and different norms.

Carlini & Wagner Carlini & Wagner [29] is one of the most effective attacks in White Box settings. The attack can optimize the L_0, L_2, and L_∞ norms. When minimizing the L_2 norm, C&W starts from the same formulation as Equation 4.8. The authors reformulate the problem as:

$$\begin{aligned} \min_{g} \, & d(x, x + g(x)) + C \cdot \tilde{l}(x + g(x), f, \bar{y}) \\ & s.t. \quad x + g(x) \in X \end{aligned} \tag{4.14}$$

where $\tilde{l}(x + g(x))$ is an arbitrary function that must respect the condition that if it is smaller or equal than 0, then the attack is successful, i.e., $f(x + g(x)) = \bar{y} \Leftrightarrow \tilde{l}(x + g(x), f, \bar{y}) \leq 0$, and C is a constant obtained using binary search. Recalling how f can be represented as $f = f_{out} \circ f_{in}$, and that $f_{in} = f_{prob} \circ f_{logit}$, where with $f_1 \circ f_2$ we mean that f_1 takes as input the output of f_2, the following functions can be used to solve the problem:

$$\begin{cases} \tilde{l}_1 = -L(f, x, t) + 1 \\ \tilde{l}_2 = \left(\max\limits_{i \neq t} f_{\text{in}}(x)[i] - f_{\text{in}}(x)[\bar{y}] \right)^+ \\ \tilde{l}_3 = SF\left(\max\limits_{i \neq t} f_{\text{in}}(x)[i] - f_{\text{in}}(x)[\bar{y}] \right) - \log_2 \\ \tilde{l}_4 = (0.5 - f_{\text{in}}(x)[\bar{y}])^+ \\ \tilde{l}_5 = -\log_2 \left(2 f_{\text{in}}(x)[\bar{y}] - 2 \right) \\ \tilde{l}_6 = \left(\max\limits_{i \neq t} f_{\text{logit}}(x)[i] - f_{\text{logit}}(x)[\bar{y}] \right) - \log_2 \\ \tilde{l}_7 = SF\left(\max\limits_{i \neq t} f_{\text{logit}}(x)[i] - f_{\text{logit}}(x)[\bar{y}] \right) - \log_2 \end{cases} \tag{4.15}$$

where $f_{\text{logit}}(x)[i]$ si the outout of the logit layer for a generic class i, $\bar{y}$ is the attacker's target class, SF is the softplus function defined as $SF(x) = log(1+e^x)$, x^+ is defined as $x^+ = \max(x, 0)$, and the $L(f, x, t)$ is the Cross Entropy, defined in Equation 2.9. The reason behind these functions is that the attack is successful only when the value of these functions is equal or smaller than 0, meaning that we can change the problem expressed in Equation 4.14 as an unconstrained optimization process, allowing to express the constrained optimization problem expressed in Equation 4.8 as in Equation 4.14, which presents one highly nonlinear constraint less.

The choice of using the logits in some losses is justified by the fact that f_{prob} generally flattens the gradients when the logits values are negative, hence making optimization significantly harder.

Equation 4.14 still requires that the adversarial attack respects the box constraints. Rather than solving the problem with PGD (see Equation 4.11) or clipped gradient ascent, i.e., incorporating the clipping into the function to be minimized by replacing $\tilde{l}(x)$ with $\tilde{l}(\min(\max(x + g(x), 0), 1)$, the authors propose a change of variables that automatically results into valid attacks- Specifically, for each dimension $i \in n$ it formulates the perturbation $g(x)$ on that dimension i (also expressed as $g(x)[i]$ with:

$$g(x)[i] = \frac{1}{2}(tanh(w[i]) + 1) - x[i] \tag{4.16}$$

where w is the substitute variable, and the fact that $-1 \leq \tanh w[i] \leq 1$ allows the perturbation to automatically satisfy the box constraints and results in a smoother version of the clipped descent algorithm, where the attack is less likely to be stuck on an extreme region of the box constraints.

Carlini & Wagner propose three different attacks, designed to optimize for the L_0, L_2, and L_∞ respectively. First the L_2 attack is define starting from Equations 4.14 and 4.16 as:

$$\arg\min_{w} |\frac{1}{2}\tanh(w) + 1)|_2^2 + C \cdot \tilde{l}(\frac{1}{2}\tanh(w) + 1) \tag{4.17}$$

with $\tilde{l}(x)$ defined as $\tilde{l}_6$ from Equation 4.15. The optimization is done through the Adam optimizer [85], and the value for the constant C is found through binary search.

The L_0 loss is not differentiable, meaning the approach of Equation 4.17, which requires gradient minimization, cannot be directly applied. Rather, the algorithm works iteratively through a variation of a greedy approach: at each round, the L_2 algorithm is run, and the features with the lowest change are selected, marked as unchangeable, and then updated. The algorithm is then run on all features except those marked unchanged. Each time the value of C is selected by starting with a small one and then, when the attack is not successful, doubling it until a threshold τ is reached or the attack is successful. The iterative process terminates when a successful attack is no longer possible for any value of C, indicating we have reached the minimum number of features for a successful attack. The last successful L_2 attack is used as the final L_0 attack.

For the L_∞ attack, the L_2 term in Equation 4.17 is replaced by the use of an objective function penalizing each term exceeding a threshold τ. The resulting algorithm can be expressed as:

$$\arg\min_{g} C \cdot \tilde{l}(x + g(x)) + \sum_{i=1}^{m}[(g(x)[i] - \tau)^{+}] \tag{4.18}$$

At each iteration, the algorithm reduces the value of τ if $g(x)[i] < \tau \forall \quad i$, otherwise it stops the search.

4.2.3.2 Black-box attacks

ZOO Zeroth-Order Optimization Based Black-box Attack (ZOO) [39] is a black-box attack designed against neural networks, which relies on the confidence scores of the model to estimate its behavior. The main idea of the attack is to use the Zeroth Order Optimization [97] to locally estimate the gradient of the classifier from the confidence scores and combine it with the Carlini & Wagner attack to obtain a powerful black box attack. They employ dimension reduction, hierarchical attack, and importance sampling techniques to limit the number of queries.

The first step in ZOO is the definition of a surrogate loss $\tilde{l}$, inspired by the C&W attack and defined as:

$$\tilde{l}(x, \bar{y}) = \max(\max_{i \neq \bar{y}}(log(f_{in}(x)[i])) - log(f_{in}(x)[\bar{y}]), -k) \tag{4.19}$$

where $k > 0$ and $log(0)$ is defined as $-\infty$. Contrary to WB settings, the function $\tilde{l}$ cannot be optimized through gradient ascent techniques in a black-box environment, as the gradients of $\frac{\delta \tilde{l}(x)}{\delta x_j}$ are unknown to the attackers. To solve this issue, the authors of ZOO propose to use Zeroth order optimization to estimate the gradient. Specifically, for each dimension $j \in m$, they compute the perturbation as a small perturbation in the direction of the gradient estimated gradient as:

$$\frac{\delta \tilde{l}(x)}{\delta x_j} \approx \frac{\tilde{l}(x + h \cdot e_j) - \tilde{l}(x - h \cdot e_j)}{2h} \tag{4.20}$$

where h is a small constant and e_j is the basis vector with only the j-th component set as 1.

Having defined the proxy for the gradient, ZOO optimizes the algorithm through stochastic gradient descent, where at each round the algorithm randomly picks one coordinate j, computes the gradient for the coordinate according to Equation 4.20, and takes a small step in the gradient direction. The optimization is done through ADAM. Since the attack complexity grows with the size n of the input space, the authors propose creating a mapping between X and a lower-dimensional projection, on which the attacks can be computed and then brought back to the original space. The size of the projection space can be determined through hierarchical attacks, where attacks are attempted first on smaller dimensions and, if no attack is found, are moved to a larger projection space until an attack is successful or no projection is made.

Boundary BoundaryAttack [24] is a decision-based adversarial attack, designed to minimize the L_2 loss of successful attacks. Boundary attack works by first finding an adversarial input $x_{adv}(0)$ such that $\mathcal{S}(x, f, x_{adv}(0))) = 1$ and the box constraints are matched. Note that $x_{adv}(0)$ does not need to be close to x, as the attack will later move towards the original input x. To bring the attack closer to the original input (minimizing the distance between the two), it iteratively modifies it by taking steps that maintain the attack within the adversarial region while reducing the distance from x.

At each iteration k, the attack is designed to: *i)* respect the box constraints, *ii)* apply a perturbation of fixed size, and *iii)* reduce the distance $d(x, x_{\text{adv}}^{(k)}) = \|x - x_{\text{adv}}^{(k)}\|_2^2$ between the adversarial example and the original input.

Algorithm 1 Boundary Attack

```
Require: Original input $x$, model $f$, target class label $\bar{y}$, max number of iterations $K_{\max}$
Ensure: Adversarial example $x_{\text{adv}}$ such that $\mathcal{S}(x_{\text{adv}}, f, \bar{y}) = 1$ and $d(x, x_{\text{adv}}) = \|x - x_{\text{adv}}\|_2^2$ is minimized
 1: Initialize $k \leftarrow 0$
 2: Sample $x_{\text{adv}}^{(0)} \sim \mathcal{U}(0,1)$ such that $x_{\text{adv}}^{(0)}$ is adversarial
 3: while $k < K_{\max}$ do
 4:     Draw random perturbation $\eta_k \sim \tilde{P}(x_{\text{adv}}^{(k)})$
 5:     if $\mathcal{S}(x_{\text{adv}}^{(k)} + \eta_k, f, \bar{y}) = 1$ then
 6:         $x_{\text{adv}}^{(k+1)} \leftarrow x_{\text{adv}}^{(k)} + \eta_k$
 7:     else
 8:         $x_{\text{adv}}^{(k+1)} \leftarrow x_{\text{adv}}^{(k)}$
 9:     end if
10:     $k \leftarrow k + 1$
11: end while
```

The procedure at each step k works as follows:

1. Sample a perturbation $\eta_k \sim \tilde{P}(x_{\text{adv}}^{(k-1)})$, typically from a Normal distribution. Rescale η_k to enforce conditions *i* and *ii*.

2. Project $x_{\text{adv}}^{(k-1)} + \eta_k$ onto the sphere centered at x with radius

$$r = \|x - x_{\text{adv}}^{(k-1)}\|_2^2, \tag{4.21}$$

ensuring the distance from the original input is preserved:

$$d(x, x_{\text{adv}}^{(k-1)} + \eta_k) = d(x, x_{\text{adv}}^{(k-1)}). \tag{4.22}$$

3. Take a small step from $x_{\text{adv}}^{(k-1)}$ toward x, guided by the projected perturbation, to gradually minimize the distance to the original input while maintaining adversariality.

Since Boundary does not rely on gradient information, it is suitable for attacking any classifier, including gradient-free models such as Random Forests.

HopSkipJump HopSkipJump [38] is another decision-based black box attack combining some Boundary and ZOO principles. The main idea is the same as Boundary, i.e., move across the decision boundary to get closer to the original observation. The main difference is that instead of performing the orthogonal step of Boundary, it exploits the local linearity of neural networks to estimate the gradient using Zeroth Order Approximation. It then makes a small step toward the gradient and reaches the decision border again.

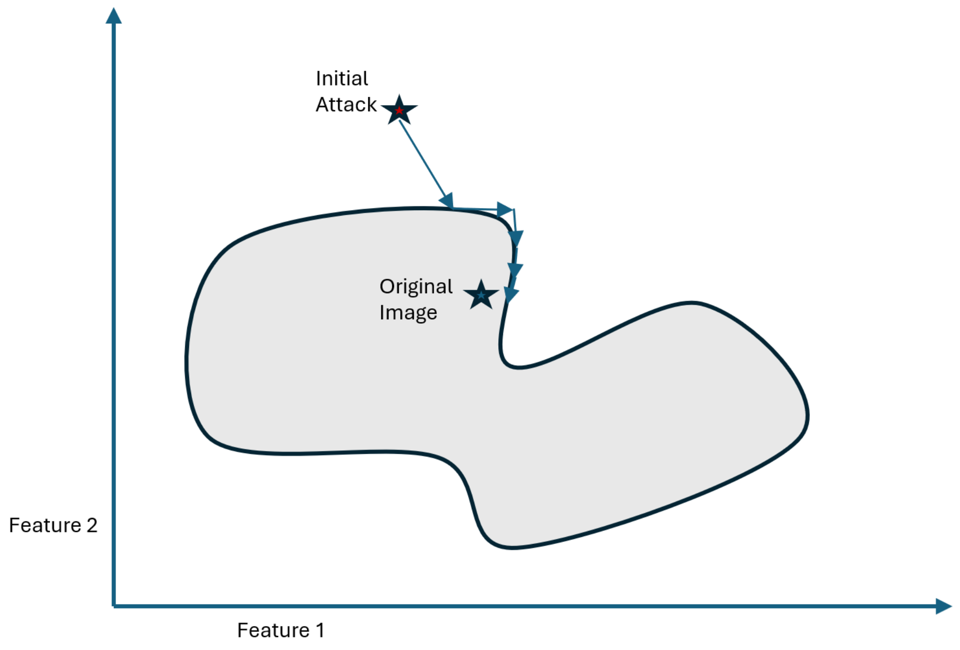
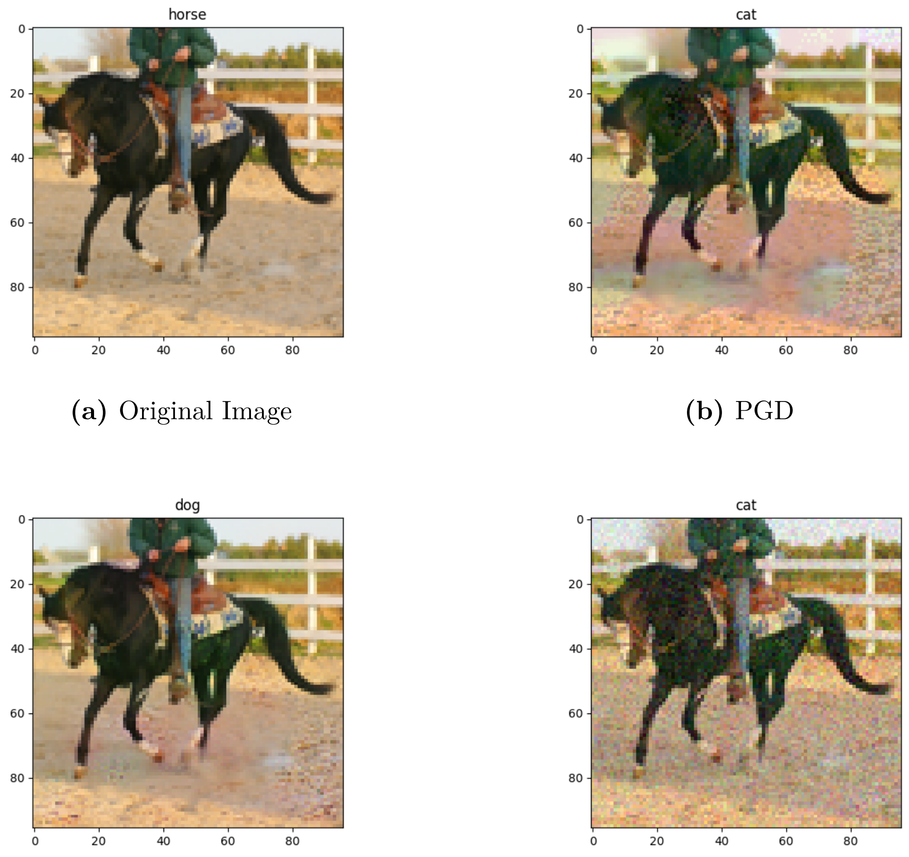


Figure 4.2: Boundary Attack. First, a random point is chosen as the initial attack. Then, the first perturbations move the input towards the original input. The attack later proceeds along the boundaries, closing the distance from the target.

Surrogate Model An interesting property of adversarial attacks is transferability. Namely, it has been shown that attacks created against a surrogate model are effective against a different, unknown target model [16]. Attacks have been shown to transfer across different datasets and models [124], exploiting the alignments among model gradients [52] and, more generally, the similarities among the decision boundaries of classifiers trained on the same task. Empirically, complex models have proved particularly vulnerable to such attacks [52].

Attackers can use this property to craft attacks using surrogate models [125, 52, 50]. The idea is as follows: since querying the target classifier may not always be feasible or efficient, if the attacker can access a labeled training set, they can train a surrogate model whose properties mimic those of the target. Attackers then test their attack on the resulting classifier, also known as a surrogate, and utilize attack transferability [52] to break the target classifier.

Mimicry The idea behind Mimicry attacks is to mimic the behavior of genuine users with malicious intent [160]. Depending on the considered threat model, this can go from simply reproducing the statistical properties of genuine data to mimicking the behavior of a particular user while satisfying a set of domain constraints [32]. An important part of intrusion and malware detection systems is signature-based classifiers, which identify patterns in network traffic and code that match patterns present in known malicious malware and attacks. In this case, detection is based on recognizing known deterministic patterns; the problem becomes, therefore, generating a valid attack that does not contain the signature. For machine learning systems, however, the problem is generally more complex, as various definitions and metrics for imitating genuine users' behavior can be employed.



(a) Original Image **(b)** PGD

(c) Carlini & Wagner **(d)** Boundary

Figure 4.3: Example of adversarial attacks on images. These attacks have been run with very little hyperparameters tuning to show how attacks can easily generate realistic images.

4.2.3.3 Problem space attacks

The attacks defined so far are designed to work in the feature space, i.e., they assume that either there is no feature engineering process, or if there is a feature engineering process the attackers can insert observations in the feature space [148]. Formally, given the adversarial perturbation g, the target class $\bar{y}$ and the feature engineering process ϕ, the adversarial attack can be defined as $x_{adv} = g(\phi(x))$ s.t. $f(x_{adv}) = \bar{y}$.

In domains like malware detection, where features are hand-crafted, operating directly in the feature space may be infeasible [132]. In this case, attacks need to operate in the data space, creating an adversarial attack $x_{adv} = g(x)$ such that $f(\phi(x_{adv})) = \bar{y}$.

A possible formalization for the problem is the following [132]: attackers can iterate a set of transformations $g_1, g_2 \ldots \in \Upsilon$, where Υ is the set of the available transformation, and the final attack can be expressed as $g = g_1 \circ g_2 \circ \ldots$. The final attack should preserve the semantics of the original input, i.e., it should maintain a set of desirable properties in the feature space, and it should be robust to preprocessing while respecting the domain constraints in the data space.

To define the set of transformations, a common approach is to use reinforcement learning. Solutions based on RL have been successfully applied in the domain of malware detection [78, 132, 158]. As an example, [78] formulates the problem as a reinforcement learning task, where the attacker has a malware they want to cloak without damaging its functionalities. The attack is expressed as an RL problem, where the actions are viable transformations, such as manipulating existing section names and creating new unused sections. The state is the program (pre-processed to facilitate the ML task), and the reward indicates whether the attack evades the engine. These attacks can work both in a white-box and black-box manner, where only the final, hard-label decision is disclosed [158].

4.2.4 Defenses

It has been advocated that the choice of features may increase the vulnerability of a model to adversarial attacks [79]. In particular, a set of highly predictive features that is incomprehensible to humans, and may hence be potentially modified without the humans noticing it. Specifically, researchers have divided features based on their usefulness, i.e., how well they correlate with the true label, and robustness, i.e., how well they correlate with the true label even when the input is adversarially modified. Notably, while non-robust features suffice for classification, they reduce the robustness of classifiers trained on them, effectively creating a vulnerability. This is in line with the conjecture, which uses the lean of causality theory and speculates that robustness issues come from learning spurious correlations rather than meaningful causal relations in the data [66] One approach to address this issue would be to create a more robust dataset that excludes non-robust features.

An approach designed for neural networks proposes a defense mechanism based on regularization [136]. This is inspired by the classic regularization employed in training, which can be seen as a weight regularization, to implement a new technique called input regularization. The key idea is that by reducing the effect that small changes in the data space may have on the classifier's decisions, the impact of small changes is automatically limited, and adversaries require higher leverage on the data to breach the model.

Another defense mechanism for Deep Neural Networks (DNN) is distillation, a technique to train a neural network using knowledge transfer from a different DNN. [127] proposes a distillation version using the knowledge extracted from a DNN to improve its resilience to adversarial samples. This knowledge is then used to reduce variation around the inputs, utilizing distillation to enhance the model's generalization capabilities and, consequently, its robustness against adversarial attacks.

A widespread defense against adversarial attacks is adversarial training; i.e., the use of adversarial samples in the training of a machine learning model. A form of regularization [69], adversarial training significantly enhances the model's robustness against attacks used during the training process. However, recent works show that training a model against multiple attacks may be cumbersome [128], and training against a specific type of perturbation typically does not guarantee protection against different types of attacks [145]. While some techniques that defend against multiple perturbations exist [156], adversarial training is still a highly incomplete defense. Moreover, the well-known trade-off between robustness and accuracy [169] implies that all the proposed defenses have a cost, and employing them when a threat is not present may result in an unjustifiable loss of accuracy for the classifier.

To date, most defenses have been successfully broken. For example, the original distillation defense was defeated by [29]. Moreover, even defenses that initially appeared robust against existing attacks often provide a misleading sense of security. They can cause gradient-based attacks to fail, yet are later broken under more rigorous evaluations. This includes defenses relying on obfuscated gradients, which can generally be bypassed once attacks are properly adapted, as well as cases where errors in attack optimization or implementation such as issues with attacks convergence and issues with the loss function [133].

4.2.5 Adversarial learning implementation

So far, we have introduced the theoretical principles behind adversarial attacks. We now discuss their practical implementation, with a focus on two key aspects: the libraries commonly used to reproduce existing attacks and the selection of appropriate hyperparameters.

Libraries Originally, the only way to use published attacks was to reimplement or reuse the code from the original paper. However, as adversarial machine learning has become increasingly studied and the number of papers on the topic has exploded, more and more libraries have been developed to implement the most common attacks and defenses in the literature. In this thesis, we rely on the Adversarial Robustness Toolbox (ART [119]), a Python library originally developed by IBM, which contains the most commonly used attacks and is compatible with both TensorFlow/Keras and PyTorch classifiers. Note that some attacks, such as the ones specifically developed for fraud detection, are not available in any library. For this reason, when necessary, we will implement them based on the paper description and the required adaptations to operate under the defined threat model.

Hyperparameters tuning Choosing the right hyperparameters is crucial for the effectiveness of adversarial attacks. For example, the number of iterations in algorithms like PGD and Boundary and the value of C in C&W can significantly influence the strength of the attack. Similarly, parameters like ϵ and Bin PGD and FGSM are crucial factors to determine the overall success rate of these attacks.

Ultimately, hyperparameter selection is largely empirical. It often relies on two strategies: (1) repeating the choices made in previous papers, especially for experimental parameters like B (2) performing empirical assessments, for instance, observing when the perturbed image begins to blur. The common practice in the literature is to assume the worst-case scenario, tuning hyperparameters to maximize the effectiveness of attacks. Notably, in black-box attacks, this comes with a tradeoff. Since the process involves repeatedly attacking the same classifier to adjust the attacks' hyperparameters and each attempt queries the model, multiple attacks against the same target resemble evaluating multiple ML algorithms on the same test set, in a process that partially resembles overfitting.

4.3 Fraud detection and adversarial machine learning

In this section, we examine the solutions proposed in the literature specifically for the context of credit card fraud detection. Although these solutions are relatively few, we highlight how they differ from previously discussed techniques in both their approach and objectives. Each method attempts to tackle the unique challenges posed by adversarial scenarios. We present only attacks, as, to the best of our knowledge, no specific defense has been proposed.

To our knowledge, only two main solutions have been proposed. Since they were both designed against the same fraud detection system, called BankSealer [31], we begin this section by introducing the main concepts of the engine, which differ partially from the formulation given in Chapter 3, which we follow in the thesis. We will then present the two main solutions, naming them after the approach they follow (Mimicry [32] and Substitute [33] respectively).

Target classifier BankSealer relies on three different customers' profilings. The local profiling characterizes individuals' spending patterns by modeling users' transactions to build a histogram. Continuous features are discretized through binning, and categorical ones are divided based on occurrence counting. The anomaly score of each transaction is measured against the customer's histogram through a modified Histogram Based Outlier Score (HBOS) [67]. The global profiling groups customers into classes by clustering simplified versions of the local profiles, and giving to each cluster a global anomaly score. Finally, temporal profiling avoids the repetition of genuine-looking transactions by monitoring the mean and variance of aggregated features such as the average amount spent by the customer over a certain time, and monitoring changes in the aggregations, effectively creating a threshold on the number of daily and monthly transactions.

Mimicry The goal of the first attack is to generate transactions that match the genuine users' profile, hence performing a Mimicry attack [32]. The attack assumes to have full knowledge of the target BankSealer classifier f, hence allowing it to compute the risk factor RS computed for each transaction. They define each transaction $x_{i,j,t}$ through its amount, time t, IBAN (which is a concept analogous for money transfer to the terminal defined in Chapter 3, and is therefore equivalent to j), and the cardholder's IP address (which we can identify with the cardholder i).

They assume that both i and j are fixed, meaning the attacker can only modify the amount, which we call AMT, and the time t. They define the goal of the attacker as the maximization of the amount stolen over time, minimizing the $RISK$ that is the output of the local profiling and maintaining the features below the thresholds defined by the classifier. The resulting optimization can be expressed as:

$$\max \sum_{x \in X_{adv}} AMT(x) \\ \text{s.t.} \quad \begin{cases} RS(x) \leq \hat{RS}, \quad \forall t \in X_{adv} \\ 0 \leq \sum_{t \in M} AMT(x) \leq \tau_{AMT} \\ 0 \leq N_d \leq \tau_{N_d} \\ 0 \leq N_m \leq \tau_{N_m} \end{cases} \tag{4.23}$$

where X_{adv} is the set of adversarial transactions made by the attacker, N_d and N_m are their number of daily and monthly transactions, respectively, and τ_{N_d} and τ_{N_m} are the thresholds posed by the temporal profile on the two values. To avoid the interference from the original cardholder, the attack assumes that a Trojan blocked them from accessing their bank application, facilitating the optimization of equation 4.23.

Surrogate Model In the second work [33], the authors rely on the following three assumptions:

– The attacker has access to a dataset to train the surrogate model.

– The attacker can observe the last month of transactions performed by the customer and their funds availability.

– The attacker can execute transactions on behalf of the victims.

They then formalize the attackers' knowledge as the tuple:

$$\kappa = \{D, \phi, f, w, X_i\} \tag{4.24}$$

where D represents the training set knowledge, ϕ the feature engineering process used by the fraud detection engine, f_w the trained classifier of parameters w, and X_i all the previous transactions performed by the customer i. In the most challenging setting, the black-box one, the authors assume that the attacker has access to the last month of transactions performed by the customer and to a training set TR' that shares some similarities with the one used to train the classifier f, and that they can use TR' to train a surrogate classifier f'. Assuming they know the feature engineering process ϕ, they can then create transactions and bypass the oracle. The knowledge Θ_{WB} assumed for black-box scnearios can be expressed as:

$$\Theta_{BB} = \{D', \phi, f', w', X'_i\} \tag{4.25}$$

where X'_i are only the transactions performed in the last month.

As in the previous case, the attacker controls only the amount AMT and time t of the transactions. They generate raw transaction candidates and then use the surrogate model to filter only those that are unlikely to be detected by the target system. In essence, each candidate corresponds to an amount and a timestamp. For the former, the authors assume they follow a consistent strategy where they perform low, medium, and high amounts of fraud, respectively. The transaction time candidates are instead selected through a bucketization of the time. The algorithm is the following: for each transaction performed by the cardholder, all aggregations used by the classifier are considered. Specifically, for each window τ and transaction x, a candidate $x + \tau + \epsilon$ is generated, with ϵ being a very small time interval, and $\tau = 0$ is added to the aggregations pool.

Part II

Part II

CHAPTER 5

Assessing adversarial attacks in fraud detection

Adversarial Machine Learning has been extensively studied, with numerous attack and defense strategies having been developed over the years, particularly in the domain of image recognition. Recent works have explored AML in domains such as malware [78] and intrusion detection [51], providing theoretical insights into domain-specific challenges.

A notable gap remains in the analysis regarding fraud detection. Even the few works that have studied this problem have done so starting from a specific threat model, which, although plausible, fails to capture the full range of possible adversarial behaviors and scenarios. Overall, the literature lacks a systematic and rigorous examination of the threats posed by AML to fraud detection systems, which is necessary to design new attack strategies and critically evaluate the threat posed by existing ones under realistic conditions, both requirements for properly assessing the robustness of existing systems.

In this chapter, we examine the specific constraints and requirements that the credit card fraud detection domain imposes on potential adversarial attackers. These application-specific factors play a key role in shaping the feasibility and effectiveness of various attack approaches. We then analyze how well current attack strategies perform when subjected to these domain-specific constraints. This threat analysis, conducted in this chapter, provides a clearer picture of the current vulnerabilities in fraud detection systems and forms the foundation for the novel attack method introduced in Chapter 7.

The rest of the chapter is organized as follows: we define in Section 5.1 a high-level threat model specific to fraud detection. This formulation re-adapts the general framework previously introduced in Section 4.2 to suit the fraud detection problem described in Chapter 3, examining the primary challenges that adversaries face when attempting to attack fraud detection models. For each challenge, we discuss its practical implications and analyze how it impacts the effectiveness of the algorithms introduced in Chapter 4. In Section 5.2, we instead focus on what is possibly the main advantage fraudsters face when compared with computer vision, the quasi-total lack of human supervision in the detection process. In Section 5.3, we present experimental results designed to empirically showcase the significance of the issue raised in Section 5.2, comparing the performance of state-of-the-art adversarial techniques with that of a simple baseline attack explicitly designed to take advantage of the absence of human supervision. Finally, in Section 5.4 we discuss the results and their significance in the context of this thesis.

5.1 Threat modeling fraud detection

The preeminent threat to fraud detection mechanisms is evasion [33, 34, 104], as poisoning a fraud detection system necessitates gaining access to its training data. Since such data remain inaccessible to the general public, the only way to modify the training set is to engage in transactions that undergo initial scrutiny by the existing fraud detection system before they are utilized for knowledge updating. Consequently, introducing a misclassified data point would necessitate the prior resolution of the evasion problem of bypassing the existing classifier.

Specifically, we can formulate the problem as follows, using Equation 3.2 to define the target fraud detection system f. Attackers compromise a subset C_{atk} of cards and can use them to interact with a subset M_{atk} of merchants that are either compromised by the attacker (i.e., merchants through which the attacker can transfer or recover the transaction amount) or sell products that can be resold later. The fraudster cannot modify the merchants' characteristics, the only difference between the merchants in M_{atk} and the broader M is that they can only connect to the former to perform fruds. The attack happens in time, meaning that for each fraud $x'_{i,j,t}$ the attacker must choose the time t, respecting the constraints that transactions must be executed in the correct time order. We define the attack process as selecting a card from $C_i \in C_{atk}$, deciding an attack time t, choosing a merchant $m_j \in M_{atk}$, and then computing the transactions feature $x_{i,j,t}$. Whenever a transaction is blocked, the card C_i used in the process is removed from C_{atk}.

Algorithm 2 Fraud detection general attack formulation

```
Require: Card set $C_{atk}$, merchant set $M_{atk}$, classifier $f$, initial time $t_{init}$, final time $t_{final}$
 1: $t_{abs} \leftarrow t_{init}$
 2: while $C_{atk} \neq \emptyset$ and $t_{abs} \leq t_{final}$ do
 3:     Select a card $C_i \in C_{atk}$
 4:     Select a merchant $m_j \in M_{atk}$
 5:     Choose an attack time $t > t_{abs}$
 6:     $t_{abs} \leftarrow t$
 7:     Compute transaction features $x_{i,j,t}$
 8:     if $f(x_{i,j,t}) = 1$ then                    ▷ Transaction is blocked
 9:         Remove $C_i$ from $C_{atk}$
10:     end if
11: end while
```

The attacks should also be fully black-box as fraud detection systems are well-protected from public knowledge. In fact, the only possible interaction for attackers would be to perform frauds and see whether the fraud detection engine blocks the transactions, meaning only hard-label algorithms can be used. Finally, fraud detection is often performed using Random Forest classifiers [27]. RFs have a non-differentiable loss, meaning gradient-based attacks would need to employ transferability to work against it.

The attacks defined in Section 4.3 have solved a special case of the aforementioned problem, where C_{adv} includes only one card $\bar{i}$, the only merchant in M_{adv} is the fraudster's account $\bar{j}$, and the fraud detection engine does not perform terminal-level aggregations, meaning that $\psi_m(m,t) = \{\emptyset\}$.

In the following sections, we will focus on the primary challenges these issues pose to attackers. Specifically, we will focus on the challenges posed by the feature engineering process, the online nature of the attack strategy, the continuous repetition of the attack strategy, and the presence of multiple layers of classification.

5.1.1 Feature engineering

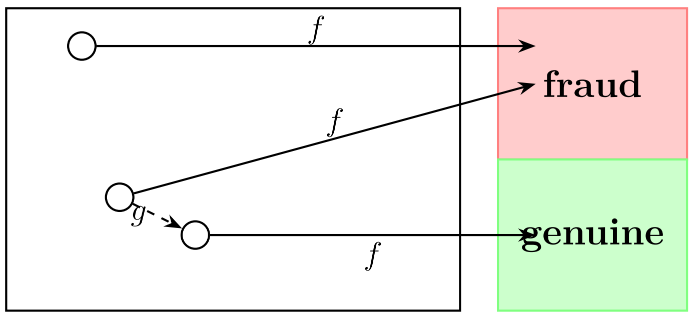
Fraud detection systems can utilize time-dependent features to work [92], where the past transactions of a card influence the probability of any transaction being accepted. Traditional adversarial attacks work at the aggregated feature level, assuming that the input is directly passed to the machine learning classifier f. Hence, to use these attacks, fraudsters must find transactions that, when processed together with past transactions, yield the same result as standard evasion attacks [33]. In general, such transactions are extremely hard or even impossible to find. Additionally, in real-world scenarios, several transactions' features are not observable or controllable by the attacker. For example, the average number of frauds on a terminal in the last few days, as used in [92], is generally unknown to users and fraudsters alike and cannot be considered in the evasion attack employed.

Take a simple case, shown in Figure 5.1, where we consider only two transactions. If the two inputs were independent, as is the case for images in computer vision, designing an attack for an input x would mean finding the input $g(x)$ that flips the predicted label $f(g(x))$ while maintaining some desired properties. If, however, we assume that in the middle, there is a feature engineering process that combines the two inputs x_1 and x_2, as is the case for the transactions aggregations defined in Chapter 3, finding a valid perturbation $g(x)$ means finding an attack that satisfied $f(\phi(x_1, g(x_2))) = 0$, where x_1 may be unknown to the attacker.

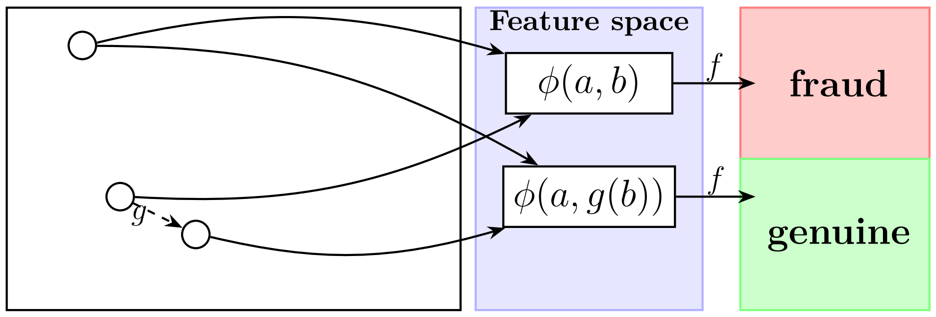
Past works on fraud detection (Section 4.3) have solved the problem by assuming complete or partial knowledge of the past transactions performed by the customer and of the feature engineering process ϕ, as well as access to a training set to train a surrogate model, allowing to compute $f(\phi(x_1, g(x_2))$ for each transaction x directly. This comes with a different set of constraints: first, obtaining access to surrogate training data is not necessarily easy in fraud detection. Second, obtaining past customers' transaction data requires infiltrating their devices with a Trojan or other costly methods. In Chapter 6 we will show that this seems to go beyond the capability of most fraudsters, and we will show that both constraints can be generally overcome in 7.

Other attacks would struggle even more. It would be difficult for white-box attacks to compute meaningful gradients, as aggregations like count or mean over a time window are not directly differentiable with respect to a single transaction's features. Decision-boundary attacks would instead struggle with changing the boundary, as all new transactions would face a slightly different boundary, even if they had the same controllable feature values.

We will further investigate the connection between fraudsters' behaviors and past transactions computed by the cardholder, as described in Chapter 6, where we aim to assess the dependency between the two factors using a real historical dataset.



(a) Computer vision



(b) Fraud detection

Figure 5.1: a) Computer vision. The rectangle on the right represents the input space, and the arrow f represents the classifier. Each image is directly processed by the classifier, meaning that when the attacker modifies an image, they do not need to consider any other images in the dataset. b) Fraud detection: the feature of a transaction may depend on the previous transactions. Hence, the optimal adversarial attack cannot be computed independently from all other transactions.

5.1.2 Attacking online systems

Online payments are continuously working and updating, either due to retraining as a concept drift adaptation solution (See Section 3.2), or because the addition of new transactions to the system modifies the feature engineering process, effectively changing the way attacks should be performed.

In general, credit card fraud detection is an example of a one-pass problem, meaning that the perturbation is added each time t without access to all the elements $x_{t'}$ in the series, where $t' > t$. This leads to a clear issue: old knowledge about the fraud detection system f may no longer be useful, as the decision boundary for transactions continuously changes over time.

Furthermore, this leads to a problem known as decision irrevocability [113]: in a streaming environment, the attacker must decide whether to launch an attack in the present moment, and once made, the decision is irrevocable. Past work has analyzed the prob-

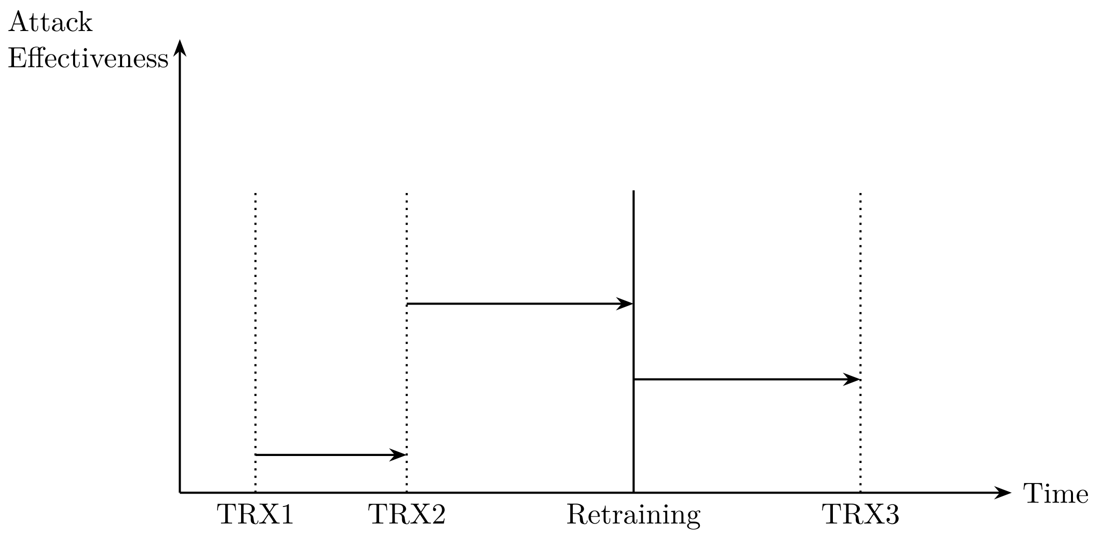


Figure 5.2: K-secretary problem in fraud detection. The effectiveness of an attack (y-axis) fluctuates over time as new transactions (TRX) arrive or the model is retrained. An attacker must make an irrevocable decision to act at a given point, mirroring the k-secretary problem.

lem of the attacks' irrevocability in a streaming environment through a theoretical approach, where a deterministic variant of the problem was studied. [113]. Specifically, they focus on a situation where the adversary must execute k successful attacks within discrete time composed of $|t|$ streaming data points, where $k << |t|$, and re-conduct it to the classic computer science problem, named k-secretary problem [65], where one must choose the k best candidates as secretaries from a randomly ordered set of n potential candidates.

In practice, this forces adversarial attacks to incorporate the attack time into their system. Treating the problem as static could lead to two problems. First, attacks need a tool to weigh knowledge based on their recency or validity, not unlike the concept drift problem we defined in Section 3.2. Second, there may be better and worse moments to attack; the choice may have a significant impact on the effectiveness of the attack.

Concept drift can be a challenge for Surrogate Attack and Mimicry too. The attacks defined in Section 6.2 for instance require access to a surrogate training set to define the attacks. This is generally challenging to obtain, due to the secrecy of fraud detection data. Even if they managed, the ever changing nature of the card payment process means that the utility of old training sets is further reduced, further complicating the fraudsters' job.

5.1.3 Repeated attacks and value maximization

In fraud detection, attackers have access to a limited number of payment cards and aim to maximize their profits before these cards are detected and blocked. Malware can be copied cost-free to target all devices protected by the breached malware detection tool. Therefore, given its inherent scalability, attackers are highly rewarded for any successful attack. Credit card fraudsters, instead, need to steal cards to perform their fraud and can only operate until they have exhausted their available cards, i.e., the cards they have compromised.

Furthermore, the number of attempted frauds an attacker can perform is limited by the number of cards they have stolen, as a suspicious number of attempted transactions will never be tolerated, meaning that black box algorithms need to be incredibly efficient in crafting new frauds. This represents a significant difference from the traditional threat model defined in computer vision, which implicitly assumes the same query budget for each attack image, and from malware detection, where successful attacks can be shared with other devices that share the same breached detectors.

The black-box attacks defined in Subsubsection 4.2.3.2 are poorly designed to perform multiple attacks; they estimate a local function f_{local} for each attack, and they lack a mechanism to reuse these estimations when crafting another attack from a different area of the data space, which is what happens in fraud detection. More generally, all attacks defined in Chapter 4 are computed independently for each observation. The optimization algorithm reported in Equation 4.20, for instance, must be independently computed for different x, making the scalability of these attacks at times questionable.

A less severe limitation concerns the goal: these attacks are designed to maximize the success rate, and they are designed to maximize the success rate of a single attack, nor the cumulative profit from a series of attacks under resource constraints.

5.1.4 Rule-based and statistical classification

Virtually all the attacks defined in the Chapter 4 treat the target system as a machine learning classifier. This is particularly true for gradient-based attacks, which need the model to be generally differentiable to find the optimal solution. However, fraud detection systems, as discussed in Chapter 3, are composed of multiple layers, with rule-based and statistical rules combining with data-driven models, as shown in Equation 3.2.

This has generally two main implications. First, the presence of clearly non-differentiable components like the rules makes gradient-based techniques generally unfeasible, as the decision function of the classifier remains fixed for large areas of the dataspace, effectively impeding any application of gradient descent. Second, some of the rules may focus on the frequency of such attacks: this severely limits the feasibility of a query-based approach. While query detectors can be bypassed by explicitly training attacks to evade detection, this is a cumbersome process, making attacks generally less scalable or efficient if these detectors are present.

It should be noted, however, that this issue is less relevant for techniques developed outside of the computer vision domains. The Mimicry and Surrogate model approach described in Section 4.3, are only marginally influenced by these factors, and RL-based attacks can be optimized to fool the fraud detection engine as a whole, without differencitating between single models and ensemble solutions.

5.2 Lack of human supervision

In this chapter, we have shown how traditional adversarial approaches encounter significant challenges when applied to fraud detection. However, all limitations could in principle be overcome. A similar process has been employed in loan application frauds, where researchers have identified the need for realism in attack generation, the requirement for custom norms for distance, and the presence of non-editable features as limitations for traditional approaches [34]. However, they also demonstrated that ZOO [39] can be modified to address these constraints, effectively making traditional solutions a suitable starting point for attacks designed for the loan application domain. The goal of this chapter is to assess whether a similar approach can be employed in credit card fraud detection.

In fraud detection, assuming a level of social engineering, attackers may have access to the feature engineering transformation, and we may imagine that the time-dependent nature of the problem could be solved by simultaneously stealing an extremely large number of cards and performing fraud with all cards considered. While these assumptions are highly pessimistic, they can be used as a worst-case scenario, a common approach in computer security literature. However, using attacks from other domains fails here because they are built on assumptions that are simultaneously too optimistic (e.g., full control over transaction features) and too pessimistic (e.g., the need to remain close to an 'original' fraud). Therefore, such a security analysis will fail, necessitating the implementation of alternative security measures.

Therefore, we will focus here on the main advantage attackers enjoy in fraud detection compared to computer vision: humans do not supervise all attacks. As such, fraudsters do not need to minimize the distance from the *original fraud*. The concept of original input originates from the computer vision domain, but it is not directly translatable to fraud detection. A possible comparison would be the transaction that the fraudster would have performed before employing adversarial techniques to cloak the attack. While this could be an essential advantage, we argue that current black box attacks cannot exploit it. In fact, most existing attacks are not feasible to work against fraud detection *because they make too strong assumptions.* Consider Boundary Attack as an example [24]. Boundary, shown in Figure 4.2, samples the data space to find a valid attack and then tries to get closer to the original observation by moving along the decision boundary of the model. It queries the model to evaluate whether the resulting attack is valid at each step, needing only the hard label as an answer. This is a reasonable assumption in image recognition, where the attacker aims at modifying an image label without making it look substantially different to human eyes, and generally when two similar images in the pixel space also look similar.

In fraud detection, the last step is unnecessary. Fraudsters can perform transactions as they wish, subject to certain constraints, and are not required to modify an observation so subtly that a human cannot recognize the change. Other constraints may still bind them, but there is no equivalent in credit card fraud detection of the original image to move towards. Once the closing steps are removed, the whole attack will be reduced to a random sample of the data space. If the problem could be tackled using a white box attack, techniques like Fast Gradient Sign Method (FGSM) [69] could still solve it by starting at a random point and moving along the gradient until they find a valid attack, eventually using optimization techniques to avoid local maxima. However, this is not possible in the black box hard-label scenario, nor will the discovered solution necessarily be optimal.

This severely affects the viability of using existing attacks as a benchmark for evaluating the robustness of existing classifiers. Analyzing what happens when the distance constraint becomes less relevant translates into minimizing Equation 4.6 for increasingly large values of the perturbation budget B.

We substantiate our hypothesis with a comprehensive experimental evaluation. First, we build two simplified fraud detection engines: a neural network and a random forest. We attack both models in a simple setting and show how both can be breached using traditional attacks, especially when we allow for significant attackers' capabilities. Next, we focus on situations where attackers do not need to be close to the original perturbation, particularly in more realistic black-box attacks. We show that while they can be effective given a limited number of queries, they are inferior to a highly naive attack we create for the occasion, which does not aim to minimize the distance. Naturally, our test attack is not designed to work in a real-world scenario. Therefore, intentionally, our experiments are performed in a simple scenario that does not consider issues such as complex feature engineering and the other constraints defined in the first part of the chapter.

Instead, we show that existing attacks are neither a viable choice nor can be used to assess the robustness of existing fraud detection engines. Robustness assessment for most applications should start from a correct threat design, and a critical evaluation of the viability of classic adversarial machine learning assumptions in the context of the target applications is required. This relies on the understanding of the specific challenges and opportunities the domain poses to attackers, as well as the development of further research on adversarial attacks and defenses, where the focus should be on the research generalization to avoid overfitting our security assessments on the properties of a specific domain problem.

5.2.1 Random Sampling Attack

To test our idea, we built a simple Random Sampling Attack. The attack is given a maximum distance inside which to search and a maximum number of queries. Centering the attack on the original fraud, it samples the dataspace within the allowed distance until it finds a valid attack or reaches the allowed number of queries. This straightforward algorithm mimics the initial steps of the boundary attack, with the only improvement being the inclusion of a distance constraint.

5.3 Experiments

Algorithm 3 Random Sampling Attack

```
Require: Classifier $f$, original frauds $\{x_1, x_2, \ldots, x_n\}$, maximum distance $B$, maximum number of queries $Q_{\max}$
Ensure: Adversarial mapping $g(x)$ for each fraud $x$
 1: $q \leftarrow 0$                                        ▷ Initialize query count
 2: for each original fraud $x \in \{x_1, x_2, \ldots, x_n\}$ do
 3:     while $q < Q_{\max}$ do
 4:         Sample $x'_{adv}$ such that $\|x'_{adv} - x\| \leq B$
 5:         $q \leftarrow q + 1$
 6:         if $f(x'_{adv}) = 0$ then                   ▷ Valid attack found
 7:             $g(x) \leftarrow x'_{adv}$
 8:             break
 9:         end if
10:     end while
11: end for
```

5.3.1 Problem Formulation

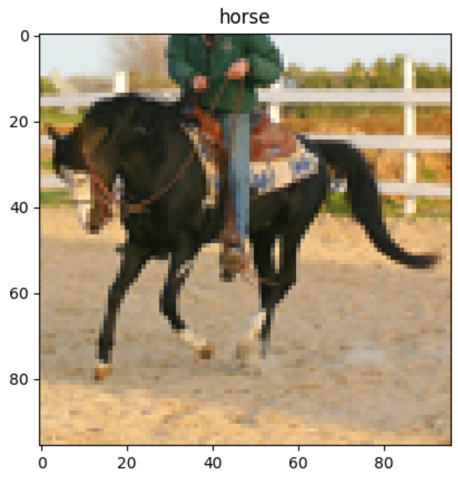
The attacks are conducted in the feature space and without considering the non-ML components of fraud detection, meaning that the problem formulation resembles that of Equation 4.6 more than the one defined in 2. This is consistent with designing attacks in a simplified setting, following the pessimistic assumption of this Section.

Formally, we consider fraud detection as a binary classification problem, where each fraud is characterized by a tuple $\{x, y\}$, where x is the feature vector and $y \in 0, 1$ is the label. We refer to genuine transactions as those for which $y = 0$, and to fraudulent transactions as those for which $y \neq 0$. We call a fraud detection engine a binary classifier f, which takes an observation x as input and tries to predict its label y. The classification is correct if $f(x) = y$ and wrong otherwise. For black-box attacks, we will call q the number of queries to the model that they require to craft an attack. In this case, we modify the notation to $g(x, \hat{f}_q)$ since the attacks are based on the classifier estimation obtained after q queries.

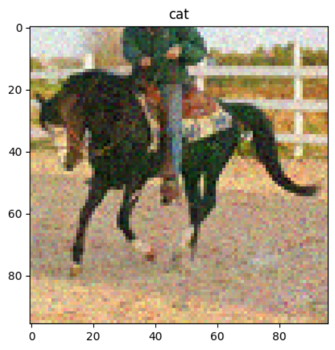
5.3.2 Metrics

We evaluate adversarial attacks in terms of their success rate, number of queries, and change magnitude. Concerning the attacks, we use the quadratic norm $\|g(x, \hat{f}_q)\|_2^2$, a distance metric for the change magnitude d. We then measure the success rate, defined as the percentage of modified transactions that changed the label (see Subsection 4.2.2), moving from being classified as fraudulent to genuine. As the success rate depends on the change magnitude, we build a curve that measures the attack success rate for each allowed change magnitude. Finally, we count the number of queries each attack uses.

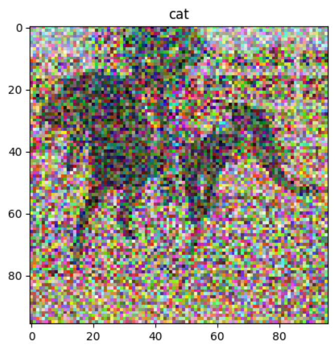
Regarding fraud detection classifiers, we evaluate their performance using the metrics outlined in Subsection 2.1.1.



(a) Original Image



(b) Boundary



(c) Random Sampler Attack

Figure 5.3: Example of Random Sampler Attack and Boundary compared on their effectiveness in image recognition, a domain where attacks' imperceptibility is extremely relevant. A visual inspection of the images shows how Random Sampler struggles to generate realistic attacks in this domain. The distance for the Random Sampler attack was selected to guarantee the attack's success, which is zero for small perturbation budgets.

5.3.3 Implementation Details

Fraud Detection Classifiers We rely on the classical implementation available on SKLearn [129] for the Random Forest classifier. Concerning the Neural Network, we built our model as a simple Sequential network on Keras instead. Regarding data imbalance, we employ standard processing techniques commonly used in the fraud detection literature [48].

Adversarial Attacks To implement the attacks, we rely on the Adversarial Robustness Toolbox (ART) [119]. Among the implemented algorithms, we focus on the evasion attacks defined in Chapter 4 that enable us to use the L_2 distance metric as a common measure.

Dataset For our experiments, we use the widely used Machine Learning Group Kaggle dataset [48] for fraud detection, discussed in Section 3.3. We normalize the data to enable the implementation of existing attacks on it using standard norms directly.

5.3.4 Experiments Settings

	Accuracy	Recall	Precision	F1	PR-AUC
Random Forest	0.9987	0.8875	0.5685	0.6921	0.7737
Neural Network	0.9989	0.8412	0.6263	0.7143	0.7301

Table 5.1: Fraud Detection classification performance metrics

Our experiments intend to corroborate the analysis performed in Chapter 3 by comparing Random Sampling attacks with existing attacks in standard settings and then when relaxing the distance constraint. As competitors, we employ white and black box approaches, with the former evaluated solely in terms of success rate and the latter also assessed based on the number of queries required for the attacks.

To do so, we divide our experiments into three phases. First, we split the data into a training and a test set. We then jointly train a Random Undersampler and two standard detection classifiers: a Random Forest and a Neural Network. We measure their performance in a non-adversarial environment. Then, we test all attacks against the classifier they can be employed against and measure their performance for different change budgets, which increase to an infinite value. We measure the effectiveness of the attacks against both classifiers.

A clear best practice is lacking in adversarial machine learning literature concerning the choice of the attacks' hyperparameters. Attacks are often optimized at hand to reach the best performance. However, this approach is both time-consuming and not scientifically sound, as hyperparameter tuning is typically performed on the same model one wants to attack, effectively leading to an information leakage that is unreasonable in a black box setting. Similarly, cross-validation would suffer from the same problem.

In this chapter, we rely on random sampling among the available hyperparameters, providing a lower bound on the effectiveness of the attack. Then, we analyze the number of employed queries for all attacks, measuring their impact on the success rate given a certain allowed change magnitude. This results in a scatterplot. Experiments are repeated multiple times to provide significance.

5.3.5 Results Evaluation

First, we show the results of the two classification algorithms in Table 5.1. Significant undersampling was employed to achieve a high recall score and effectively test the attacks. The results on a Neural Network are shown in Figure 5.4.

As expected, when considering the standard setting of attacking a neural network with a slight change budget, white box attacks are superior to all the others, given that they can exploit the model's knowledge. However, many queries can help black box models achieve similar performance. Crucially, random sampling is not a viable approach in

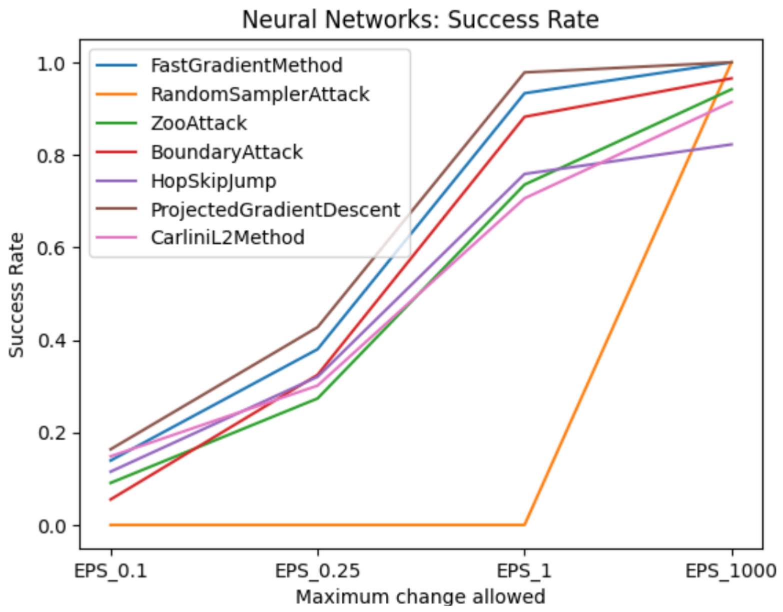


Figure 5.4: Attack success rate against a Neural Network. While white-box attacks are effective at small budgets, Random Sampling (orange line) only becomes effective at large budgets, where it matches or exceeds other black-box methods.

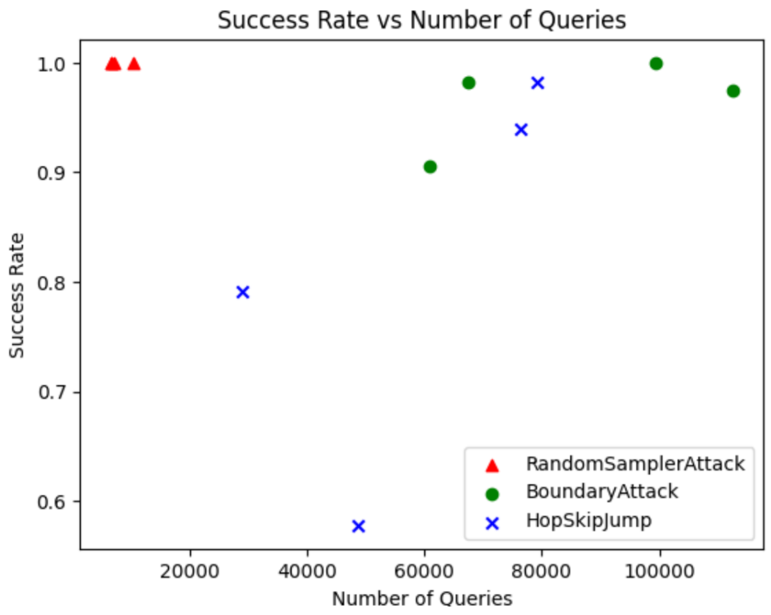


Figure 5.5: Queries Budget and Success Rate relationship against Neural Networks with four runs per method. The results show how the RandomSampler attack can reach a success rate in line with the best runs of Boundary and Hopskipjump while using a significantly lower number of queries.

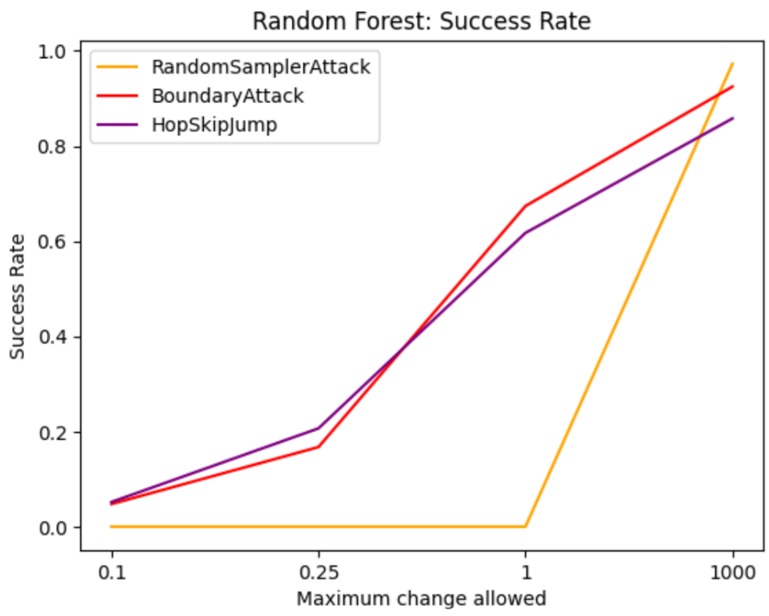


Figure 5.6: Attacks Success Rate against Random Forest Classifier, evaluated for different allowed budgets

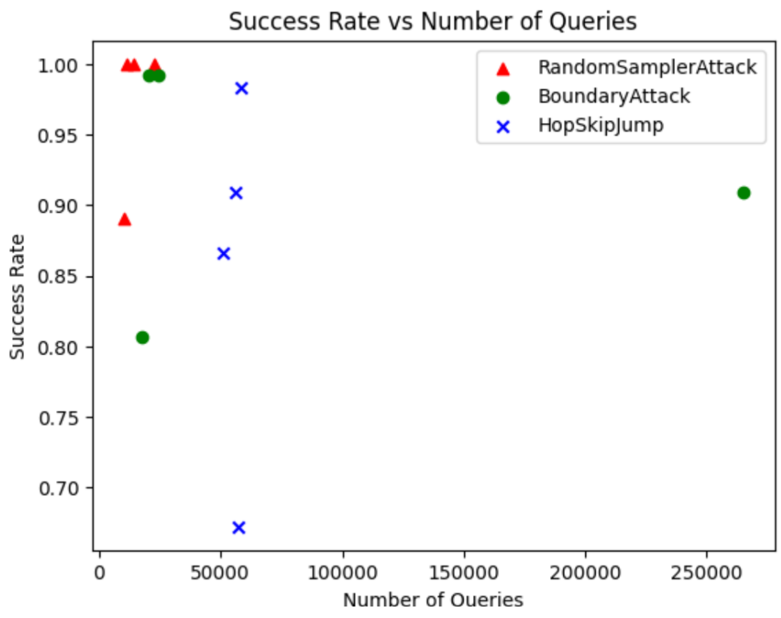


Figure 5.7: Queries Budget and Success Rate relationship against Random Forests with four runs per method.

these settings due to the lack of efficient mechanisms to simultaneously identify viable attacks and minimize the distance.

Even with the relatively large budget of 1, which allows attacks to reach a success rate far larger than 50%, the attack is still utterly incapable of achieving any success. However, when we consider a change budget large enough to make the constraint irrelevant, Random Sampling performs in line with the best attacks in the literature. More importantly, it does so in the most efficient way. As shown in Figure 5.5, no other attack is as efficient and effective regarding queries and success rate as the Random Sampling in this setting.

Figures 5.6 and 5.7 show how the results hold for Random Forest. For both target classifiers, the relative performance of Boundary and HopSkipJump varies slightly from distance to distance. This is due to the choice of hyperparameters and the number of queries employed, which vary accordingly. The second interesting observation is how Random Forests are significantly more robust to attacks than Neural Networks. Boundary and Hopskip are always less effective for small changes when facing a Random Forest Classifier. This observation, in line with previous findings from adversarial machine learning literature [55], is particularly relevant for credit card fraud detection, where random forests are widely used [49, 101].

Overall, our results confirm our expectations based on the analysis done in Chapter 3. First, when working under traditional adversarial machine learning assumptions, attacks are effective on credit card fraud detection data, indicating that they can also be employed on different datasets, such as those from image recognition. At the same time, even using a different classifier type can reduce the attacks' effectiveness. Moreover, successful attacks often employ multiple queries to achieve optimal performance. Real-world credit card fraud detection engines block the cards before they reach this number. Finally, attacks are based on the notion of minimizing the distance. When this limitation is ignored, they are inferior to the Random Sampler Attack.

5.4 Conclusions

In the information age, online applications and production are generating a large amount of data, enabling data-driven decisions to enhance the efficiency of most economic decisions. Increasingly more interconnected data-driven systems rely on machine learning, which is essential in the modern data economy. While machine learning is now central to securing financial systems, most methods used to assess its security have been developed for other domains, and the transferability of these techniques is dubious.

Studying the vulnerabilities of machine learning algorithms and systems is necessary to ensure their robustness and trustworthiness. Research on Adversarial Machine Learning has been addressing this issue for more than a decade [77]. However, the heavy focus on image recognition has significantly influenced how most techniques have been designed, and their assumptions can severely impact the applicability of adversarial attacks to different domains.

In this chapter, we focused on evasion attacks against fraud detection. We showed how even a straightforward attack can outperform existing techniques when some of the assumptions they are built on are no longer valid. Our attack could not be applied to a real fraud detection environment, where some variables are hidden from the attackers and cards are easily blocked; however, the same holds true for other existing attacks. Moreover, we showed how most attacks perform better against a neural network classifier than against other algorithms, meaning that the specific classifier used in an application impacts the application's robustness. Finally, we showed how the number of required queries for most attacks is incompatible with the credit card fraud detection environment, making them unfeasible.

Crucially, this does not prove that fraud detection is secure. The lack of a clear attack direction does not guarantee security. Instead, if we do not know the true vulnerabilities of our system, we risk attackers discovering them first, which can lead to zero-day attacks. As unknown vulnerabilities cannot be patched, this is an extremely dangerous scenario [19]. The same principle can be applied to different data-driven systems. Each application presents different attacking scenarios and assumptions, and the excessive focus on image recognition can lead us to misdirect our research and security threats assessments.

Chapter 6

An adversary model of fraudsters' behavior to improve oversampling in credit card fraud detection

6.1 Introduction

Ultimately, this thesis aims to improve the robustness of fraud detection engines *vis-à-vis* adaptive fraudsters, who constantly modify their strategies to evade existing controls. A significant step toward this objective is to understand what fraudsters already do. This will not directly answer any question concerning the strongest fraudsters' capabilities, but can provide a baseline about their behavior, what factors influence it, and ultimately, what kind of attacks we are likely to see in the future.

To achieve this goal, we construct here a model of fraudster behavior based on the analysis of real historical transactions provided by our industrial partner, Worline S.A. In particular, we investigate whether the actions of fraudsters follow regular patterns and whether they exhibit any systematic relationship with the transaction history of the cards they target. As discussed in Chapter 5, the latter is significant because it is closely related to the threat model assumptions made by previous works in the field.

Our initial analysis focused on modeling the dependency between the last genuine transaction and the subsequent first fraudulent transaction on a compromised card. Were this dependency statistically relevant, it would mean that fraudsters already use this information to design their transactions, following smart attack patterns and possibly performing some variations of the Mimicry attack defined in Section 4.3. However, despite extensive modeling efforts, we did not obtain results with sufficient accuracy to establish a reliable predictive pattern. Instead, we realized that the behavior of fraudsters does indeed exhibit time-dependent patterns. This behavior may not be adaptive, but it suggests they are following some kind of strategy, a necessary step towards adversarial behavior.

To leverage our modeling work, we propose a new resampling framework, named *Adversary-based Oversampling* (ADV-O), for oversampling credit card transactions that are fraudulent. In fact, credit card fraud detection is a heavily imbalanced problem, with genuine transactions far outnumbering the fraudulent ones. Despite numerous works addressing the issue through the use of algorithms [131, 123] and data-level techniques [117, 48], the existing methodologies currently fail to acknowledge that fraudsters may employ various strategies to circumvent card blocking and enhance the success rate of their fraudulent activities. ADV-O is the first work in fraud detection literature that takes this into account.

ADV-O relies on two different learning algorithms. The first algorithm, named MIMO ADV-O, models the dependency between consecutive fraudulent transactions executed with the same card, emulating the behavioral patterns exhibited by fraudsters. Specifically, we train a multi-target regression model to forecast certain characteristics of a fraudulent transaction based on its preceding one, and we utilize this model to generate a set of artificial fraudulent transactions. The second algorithm, called TimeGAN ADV-O, is based on TimeGAN[167] (see Subsection 2.2.2), a popular adaptation of Generative Networks for time series. This algorithm is used to model each chain of fraudulent transactions as a time series.

Both MIMO ADV-O and TimeGAN ADV-O differ from traditional oversampling strategies used in fraud detection. Conventional techniques like SMOTE and GAN treat each fraudulent transaction as an independent sample from the same distribution and do not consider any dependency on the card used for the transaction or its transaction history. ADV-O algorithms, instead, model transactions as complex time-dependent problems.

The primary contribution of this chapter is a novel framework for managing the imbalance in fraud detection data, which explicitly models the behavioral patterns of fraudsters and their time-dependent dynamics. In particular

- We propose the first quantitative model of fraudsters' behavior, where the fraudsters' actions depend on the card they have access to and their previous actions.
- We uncover the existence of a substantial dependency between consecutive fraudulent transactions through a comprehensive analysis of transactions history.
- We design a novel framework composed of two oversampling algorithms, both based on a study of fraudsters' behavior, called MIMO ADV-O and TimeGAN ADV-O, respectively.
- We conduct a comprehensive experimental evaluation of the proposed methodology using over 20 million real credit card transactions, comparing it with state-of-the-art oversampling techniques.
- To ensure the reproducibility of our results, we design a transactions simulator inspired by existing literature and replicate our experiments on synthetic, publicly available data.

The organization of the chapter is the following: Section 6.2 reviews the current literature, Section 6.3 provides a formal description of the problem, Section 6.4 describes the main contributions, while Section 6.5 assesses the effectiveness of our algorithm on both a large real dataset (20M+ transactions over two months) provided by our industrial partner and a synthetic dataset. We then explain in Section 6.6 the statistical tests used to evaluate the results of our approach. Finally, we discuss the results obtained and analyze the plausible next steps for research in Section 6.7.

6.2 State of the art

To the best of our knowledge, no quantitative model of fraudsters has ever been designed. The closest related work is synthetic transaction generators such as the MLG generator [92] and CardSIM [6]. We refer to the full explanation of both in Section 3.3 in the Background chapter. Both works, however, use simplified fraudulent patterns for attackers and are not built on models trained on real data.

Concerning the resampling algorithm, we refer to Section 3.1 for a general description of imbalance learning solutions. We focus here on the closely related domain of time series analysis and oversampling. In fact, when we consider the time dynamics of the frauds, we enter the domain of time series analysis. Cardholders exhibit specific spending patterns, creating time dependencies among their transactions.

In fraud detection literature, multiple works consider genuine transactions as time series to detect anomalies [116, 137]. Instead, [60] uses the time series nature of the transactions to train classifiers explicitly designed for time series. In line with other works that suggest the possible utility of deep learning in fraud detection [4], they use an ensemble of Long Short Term Memory recurrent networks [75], which work on data rebalanced with a new hybrid resampling method called SMOTE-ENN, which combines oversampling and undersampling algorithms to best resample the data.

Closer to our work, the authors of [99] model both genuine and fraudulent transactions as time-dependent and use this to craft new features. Interestingly, explicitly considering frauds as time series is much less common. The most related work [170] starts from the assumption that frauds are a time series and designs a variant of a GAN to exploit this fact in generating synthetic frauds, that are then used to perform oversampling. The resulting synthetic frauds are then analyzed to verify whether they are similar to the original frauds in the dataset. Their influence on a classifier is then compared to that of a VAE and a generic GAN using accuracy as the primary metric. The main limitations of the work lie in the lack of comparison with other oversampling algorithms, the lack of metrics more suited for imbalanced classification tasks, and the absence of an explicit model of fraudsters' behavior.

Time series can also be employed for oversampling, where synthetic data are generated in series instead of single data points. This is particularly relevant for our work, as considering the frauds as a time series allows us to compare such methods to classical approaches. Defining a good generative model for time series data is not trivial, as the generated data should resemble the original points in terms of *point-to-point* similarity (i.e., the statistical similarity between all points in a time series) and conditional dependence between sequential values. For our work, we opted to use TimeGAN [167] (See Subsection 2.2.2).

6.3 Problem Formulation and notation

We assume to have access to a training set D_{train} and a test set D_{test} of transactions related to a set $C_1, C_2 \dots C_{\bar{C}}$ of c cards. Each transaction is denoted by $z_{i,j}$ [1] where j stands for the transaction number and i for the associated card. A transaction $z_{i,j}$ is described by a feature vector $x_{i,j}$ of size n and a label $y_{i,j}$ indicating whether the transaction is fraudulent ($y_{i,j} = 1$) or genuine ($y_{i,j} = 0$). For the sake of conciseness, we will also use the notation $z^+_{i,j}$ to indicate that the transaction $z_{i,j}$ is a fraud ($y_{i,j} = 1$) and $z^-_{i,j}$ if $z_{i,j}$ is genuine ($y_{i,j} = 0$). The transaction features belong to three main categories: *i) the card's features*, e.g. the transaction history and the cardholder information, *ii) the terminal's features* e.g., its location, activity time, and transactions history, and *iii) the transaction features*, e.g., the amount and time of the transaction. We will refer to generic terminal features with TRM and to transaction features with TRX. The fraud detection problem is formulated as a binary classification problem [49] where a classifier associates to each feature vector $x_{i,j}$ an estimated probability $p^{i,j}_{in}$ of fraud. The assessment of the detection procedure involves training the classifier on D_{train} and evaluating its accuracy on D_{test} using a set of conventional metrics described in the experimental section.

6.4 Contributions

The main assumption of this chapter is that fraudsters exhibit a behavioral pattern that can be partially inferred from the transactions they perform. This pattern can be conceptually decomposed into two components:

– fraudsters adapt to the characteristics of the cards they have access to;

– they perform various transactions according to a certain plan, meaning we can see each chain of frauds as a series of highly correlated and time-dependent elements.

Our work aims to validate these assumptions, design a quantitative model of fraudsters' behavior from real data, and show that this model is predictive of fraudsters' actions. We use such a model in a new framework for oversampling in fraud detection, named "Adversary Based Oversampling" (ADV-O).

To model the fraudsters' behavior, two scenarios may be considered:

– *genuine-to-fraud* scenario: the hypothesis is that if a fraudster wants to cloak their frauds as genuine transactions, they will imitate the cardholder's behavior the first time they use the compromised card.

– *fraud-to-fraud* scenario: the hypothesis is that the fraudster uses a specific logic to generate a series of fraudulent transactions. Hence, we can model his behavior as a stochastic process, such as a Markov Chain, where each fraud is a state and the next state can be viewed as a stochastic function of the previous one.

[1]Bold letters, such as z, indicate vectors.

We primarily focus on the fraud-to-fraud scenario, as the experiments reported in Subsubsection 6.5.5.1 suggest that the genuine-to-fraud scenario is too challenging to tackle from a data-driven perspective. In what follows, we consider two implementations of the ADV-O strategy.

6.4.1 MIMO ADV-O

The first ADV-O algorithm, denoted *MIMO ADV-O* (pseudo-code in Algorithm 4), models the fraudster behavior with a Multi-Input Multi-Output (MIMO) regression model. Specifically, MIMO ADV-O applies machine learning regression to learn the set of multivariate dependencies g existing between the n features of $z^+_{i,j}$ and the ones of $z^+_{i,j+1}$. We decompose the learning problem into a set of n Multi-Input Single-Output (MISO) problems. This means that for each fraud $z^+_{i,j+1}$ we create n regression tasks where the output is the n^{th} transaction feature $x_{i,j+1}[\nu], n = 1 \dots N$ and the input is the feature set $[x^1_{i,j}, x^2_{i,j}, \dots, x^N_{i,j}]$ of $z^+_{i,j}$. For each of the n MISO tasks, we train a regressor $\mathcal{R}_n$, $n = 1, \dots, N$, which takes as input all the features of the previous fraud and outputs only the νth feature of the following one. The composition $\mathcal{E}$ of the n single-output regressors $\mathcal{E} = \{\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_N\}$ estimates the MIMO mapping $g : \mathbb{R}^N \to \mathbb{R}^N$ between the features of two consecutive frauds whose aim is to model the fraudsters' behavior in the fraud-to-fraud scenario. Since regression returns numerical variables, categorical features are converted into numerical ones before training (Subsection 6.5.1). In summary, for each fraud $z^+_{i,j} \in D_{train}$, MIMO ADV-O adds a new artificial fraud by applying all the regressors in $\mathcal{E}$, and adds the predicted fraud to the training set.

Algorithm 4 MIMO ADV-O Algorithm

```
Require: $D_{\text{train}}$                                   ▷ Training dataset
Require: $r$                                                  ▷ Desired oversampling ratio
Ensure: $D_{\text{train}}$ augmented with artificial frauds to achieve ratio $r$
 1: Initialize $\mathcal{E} = \{\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_N\}$     ▷ Initialize the set of regressors
 2: for $n = 1, \dots, N$ do                                 ▷ For each transaction feature
 3:     Train $\mathcal{R}_n$ using $D_{\text{train}}$ and the $\nu$-th feature as the target
 4: end for
 5: while the ratio of frauds in $D_{\text{train}}$ is less than $r$ do   ▷ While desired ratio is not
    reached
 6:     for each fraud $z^+_{i,j} \in D_{\text{train}}$ do            ▷ For each fraud in the training set
 7:         for $n = 1, \dots, n$ do                          ▷ For each transaction feature
 8:             Predict the $\nu$-th feature of the following fraud using $\mathcal{R}_n$
 9:             Add the predicted feature to $z^+_{\text{art}}$
10:         end for
11:         Add $z^+_{\text{art}}$ to $D_{\text{train}}$
12:     end for
13: end while
```

6.4.2 TimeGAN ADV-O

MIMO ADV-O is built on an implicit Markov Chain representation of fraudsters' behavior, which considers each fraudster's action as based solely on the previous fraud, ignoring longer time patterns.

In the second ADV-O approach, we consider each fraud chain as a time series. This has two advantages: *i)* it takes into consideration possible dependencies among non-consecutive frauds *ii)* It allows us to reuse oversampling algorithms from the time series literature. The second ADV-O algorithm is denoted *TimeGAN ADV-O* (pseudo-code in Algorithm 5), and employs TimeGAN [167], a commonly used algorithm for time series oversampling, and applies it to the fraud chains.

Algorithm 5 TimeGAN ADV-O Algorithm

```
Require: $D_{\text{train}}$                                   ▷ Training dataset
Require: $r$                                                  ▷ Desired oversampling ratio
Ensure: $D_{\text{train}}$ augmented with artificial fraud chains to achieve ratio $r$
 1: Initialize TimeGAN with Embedder $E$, Recovery function $R$, Generator $G$, Discriminator $D$, and Supervisor $S$
 2: Train the TimeGAN model on $D_{\text{train}}$
 3: while the ratio of fraud chains in $D_{\text{train}}$ is less than $r$ do      ▷ While desired ratio is not reached
 4:     for each fraud chain $x_i = \{x_{i,1}, x_{i,2}, ..., x_{i,T}\}$ in $D_{\text{train}}$ do
 5:         Generate a random noise series $\epsilon = \{\epsilon_1, \epsilon_2, ..., \epsilon_T\}$
 6:         Create synthetic time-series in the latent space: $z_i = G(\epsilon)$
 7:         Transform synthetic latent series into the original feature space: $z^+_{\text{art}}$
 8:         Add $z^+_{\text{art}}$ to $D_{\text{train}}$
 9:     end for
10: end while
```

First, we express the fraudulent transactions performed on the i_{th} card with $x_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,T}\}$, i.e., a time series of length T. We then employ TimeGAN to consider longer dependencies among the frauds.

Notably, this approach differs from classic generative models, as it learns the distribution of the fraud chains. Classical oversampling algorithms, instead, learn the distribution of single frauds, ignoring the dependency among the frauds within each chain.

6.4.3 MIMO vs TimeGAN ADV-O

MIMO ADV-O models the process leading to the generation of a fraud $z^+_{i,j+1}$ given the previous fraud $z^+_{i,j}$ through a discriminative model. The resulting function is then applied to all frauds in the dataset, emulating how a generic fraudster would perform the next fraud given the previous one. TimeGAN ADV-O instead models each chain of frauds as a time series, meaning it does not explicitly estimate the fraudsters' behavior. Moreover, TimeGAN ADV-O is a generative model, meaning that the artificial frauds are not explicitly built on the ones in the original training data. The two approaches differ, yet they share strong similarities, as both base their ability to emulate fraudsters'

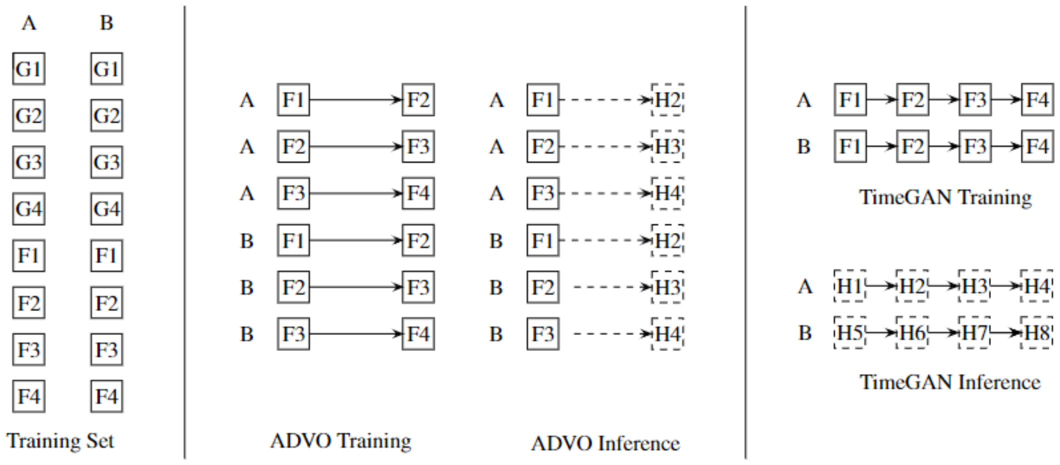


Figure 6.1: Schematic Representation of ADV-O and TimeGAN Training and Inference Phases, illustrated with a toy example. The figure delineates the transactional activities of two distinct cards, labeled as ’A’ and ’B’. The genuine transactions conducted with these cards are denoted by ’G1’, ’G2’, ’G3’, and ’G4’, while fraudulent activities are represented as ’F1’, ’F2’, ’F3’, and ’F4’. The inference phase generates artificial transactions, which are denoted as ’H1’, ’H2’, and so forth.

behavior on modeling the dependency among frauds performed with the same card, despite their differences.

Figure 6.1 illustrates the MIMO ADV-O and TimeGAN ADV-O training and inference phases with a toy example. The figure delineates the transactional activities of two distinct cards, labeled as ’A’ and ’B’. The genuine transactions conducted with these cards are denoted by ’G1’, ’G2’, ’G3’, and ’G4’, while fraudulent activities are represented as ’F1’, ’F2’, ’F3’, and ’F4’. The inference phase leads to generating artificial transactions, which are indicated as ’H1’, ’H2’, and so forth.

6.5 Experiments

This section describes the experimental assessment of the ADV-O strategy and is organized as follows. We describe in Subsection 6.5.1, Subsection 6.5.2 and Subsection 6.5.3 the real dataset, the metrics, and the statistical tests used for the experiments, respectively. Subsection 6.5.4 introduces the synthetic data generator. Finally, Subsection 6.5.5 shows and discusses the results of the experiments on the industrial dataset, and Subsection 6.5.6 reproduces the results on the synthetic dataset.

6.5.1 The real dataset

Real-world fraud detection systems are typically composed of five modules (detailed in [46]), where transactions are first checked (e.g., PIN) at the terminal level, then filtered by simple blocking rules. If the transaction is not discarded, a series of additional checks are performed using scoring expert-based rules and data-driven models. Finally, transactions that raise a fraud alert are reviewed by human investigators, who assess whether the transaction is indeed fraudulent. From this perspective, it is crucial to control the false alarm rate, as human investigators are a critical and limited resource.

The industrial partner Worldline S.A provided the transactional data used for the experimental assessment. The dataset refers to a period from 01/05/2018 to 30/09/2018 (DD / MM / YYYY) and includes over 60 Million online, international transactions (each associated with a card and a terminal). Our experiments are performed using a sliding window approach, where the data are divided into windows of two weeks. Each window is split into two 7-day smaller windows: the first is the training set, and the second is the test set. We report the average and standard deviation of the results across various windows, and we use these results to assess the statistical significance of the performance differences among the different classifiers.

The ratio between frauds and total transactions is significantly less than 1% (though its exact value cannot be disclosed), yielding a strongly unbalanced learning problem. The raw dataset comprises 46 features, including both categorical and numerical values. Though the exact nature of the original features is not disclosable for confidentiality reasons, it is important to remark that they contain information about the transaction, the card-holder, the terminal and that some of them result from a feature engineering process extracting meaningful aggregates (e.g., average expenditure in the last month). Given the presence of categorical variables in the raw dataset and the adoption of regression techniques in our strategies, special attention has to be devoted to categorical encoding. Several techniques for the encoding of categorical variables exist in literature [134] like *Integer Encoding* and *One Hot Encoding*. In our experiments, we adopt a technique for encoding categorical variables called "target encoding," already used in our previous works on fraud detection [43]. We replace each category with the empirical frequency with which a transaction belonging to such a category is fraudulent. This encoding maps each category to an informative numerical value representing the conditional a-priori probability that a transaction belonging to such a category is fraudulent. For this work, we decided to work on a significantly simplified version of the problem, where we selected only a small subset of features. We decided to focus on the amount (AMOUNT) and two other features, both categorical, that we deemed the most important for classification. For simplicity and to conceal their true names, we call them X_TERMINAL and Y_TERMINAL, as both features are linked to characteristics of the terminals used in the transactions. The goal is to show the results in a simplified yet realistic context to extend the work to the full dataset in future works. This also allows us to reproduce the results on simulated data without dealing with the complexities of emulating a high-dimensional dataset.

6.5.2 Metrics

The assessment of our proposed algorithm can be done at two different levels: the accuracy of the fraudster model (described in Subsubsection 6.5.2.1) and the quality of the oversampling algorithm (discussed in Subsubsection 6.5.2.2).

6.5.2.1 Accuracy metrics of the fraudster model

The fraudulent behavior is modeled with a set of n independent MISO regression tasks (Section 6.4) whose accuracy can be measured using conventional regression metrics, such as the coefficient of determination. The coefficient of determination

$$R_n^2 = 1 - \frac{\sum_{i,j}\left(x_{i,j}[\nu] - \hat{x}_{i,j}[\nu]\right)^2}{\sum_{i,j}\left(x_{i,j} - \bar{x}[\nu]\right)^2}, \qquad n = 1, \ldots, n \tag{6.1}$$

measures the proportion of the variation of the feature $x[\nu]$ that can be explained by the νth regression model, where $x_{i,j}[\nu]$ is the true value of the νth feature of the transaction $z^+_{i,j}$, $\hat{x}_{i,j}[\nu]$ is the predicted value and $\bar{x}[\nu]$ the average of $x_{i,j}[\nu]$ over all cards and transactions. In order to assess the overall accuracy of the MIMO task, we compute the average $R^2 = \frac{\sum_{n=1}^{n} R_n^2}{n}$ over all the n features.

Note that a R^2 score significantly larger than 0 suggests good predicting abilities, with 1 being the score of a perfect regressor. Any negative or null value implies no predictive power.

6.5.2.2 Oversampling quality metrics

We measure here the quality of oversampling through its impact on the performance of a classifier trained on the real data enhanced with the oversampling. The most common metrics to assess fraud detection

To assess the quality of the learning algorithms under this setting, we use the Area Under the Precision-Recall Curve, which is a well-reputed way to assess classification accuracy in strongly unbalanced settings [43, 62]. We also use the *Precision Top k*. Both metrics are explained in detail in Subsection 2.1.1.

6.5.3 Statistical Tests

Statistical tests play a crucial role in evaluating and comparing different algorithms, enabling researchers to determine the statistical significance of their results and draw meaningful conclusions. In the context of imbalanced learning (Section 3.1), where the performance of various oversampling techniques is assessed, it is essential to employ rigorous statistical tests to ensure that observed differences in performance are not due to chance or random variations in the data. In this study, we employ the Friedman test, followed by the Nemenyi post-hoc test, to conduct pairwise comparisons of the proposed Adversary-based oversampling technique with other state-of-the-art algorithms, as suggested by [53]. The Friedman test is a non-parametric test designed for comparing multiple treatments, while the Nemenyi test facilitates multiple comparison procedures and helps to identify significant differences between specific pairs of algorithms.

The results of these tests are presented using a Critical Distance (CD) plot, which provides a visually intuitive way to represent the relative performance of the algorithms and their statistical significance. The CD plot consists of horizontal lines with markers representing the average ranks of each algorithm, while a vertical line indicates the critical distance. If the markers for two algorithms are farther apart than the critical distance, it can be concluded that their performance differs significantly. This approach enables researchers to quickly and effectively assess the comparative performance of various oversampling techniques and identify the most effective ones in tackling the challenges of imbalanced learning in fraud detection. When presenting the performance of various oversampling algorithms in Tables 6.3a and 6.3b, we apply report in bold the methods that are not significantly worse than the best one according to the Friedman-Nemenyi test. For the results of each individual test, we refer to the Appendices.

6.5.4 A synthetic generator for reproducible results

Results on real data are an important part of our analysis, yet they are unfortunately not reproducible because of the evident confidentiality issues in financial-related domains [8]. Though some of the authors of this chapter contributed to releasing an encoded fraud detection dataset (the Kaggle credit card dataset in [135]), Such a dataset is not suitable here (missing card identifiers), and no other publicly available datasets exist to validate the proposed approach. To increase the reproducibility of the present work, this section presents then a simplistic simulator of transactional data, which may be used to assess our approach with non-confidential data. The generator, in spite of its simplistic nature, has some interesting characteristics:

– it models the terminal and the cardholder with a small number of features: the terminal is described by two geographical coordinates, while the cardholder is described by two geographical coordinates, and the spending behaviour

– it preserves the customer-terminal-transaction nature of the original dataset (like the ULB-MLG generator in [92]) which allows identifying pairs of consecutive frauds,

– it creates a statistical association between consecutive frauds, in concordance with the assumption of this chapter, which has been corroborated by the real data experiment.

The proposed generator's structure is the following: first, two populations (customer and terminal profiles) are created by sampling their low-dimensional distributions. As long as a cardholder is not hacked, they are repeatedly associated with a random terminal based on the geographic location, and their transactions are generated in an i.i.d. manner. Then, as simulation time goes by, a portion of cardholders switches from the genuine to the fraudster category. The behavior of fraudulent cardholders differs in terms of the association strategy to a terminal and the correlation existing between consecutive amounts. Though it is evident that such a simulator simplifies the real process for the sake of reproducibility, it preserves three important properties of the transaction process: the large imbalance ratio, a difference between fraudulent and genuine users in choosing the terminal, and above all, the existence of a temporal association (corresponding to the mapping g estimated in Subsubsection 6.5.5.1) in fraudulent sequences only. More details on the generator and related synthetic datasets are available at https://github.com/FaramirHurin/ADV-O.git

Algorithm 6 Transactions generator

```
Require: Terminal distribution $p_m(m)$, Customer distribution $p_c(c)$
Require: Behavior functions $g$, $g_1$, $g_2$
Require: Maximum days $max\_days$, compromission probability $k(c)$
 1: Generate customers $C \sim p_c(c)$
 2: Generate terminals $M \sim p_m(m)$
 3: for all $c \in C$ do
 4:     $Compromised[c] \leftarrow$ false
 5:     $FirstFraud[c] \leftarrow$ false
 6: end for
 7: for $day = 1$ to $max\_days$ do
 8:     for all $c \in C$ do
 9:         if $Compromised[c] =$ false then
10:             With probability $k(c)$: $Compromised[c] \leftarrow$ true
11:             Generate genuine transactions using $g(c, M, day)$
12:         else if $FirstFraud[c] =$ false then
13:             Generate fraudulent transaction using $g_1(c, M, day)$
14:             $FirstFraud[c] \leftarrow$ true
15:         else
16:             Generate fraudulent transactions using $g_2(c, M, day)$
17:         end if
18:     end for
19: end for
```

Table 6.1: Accuracy of ADV-O in Predicting Initial Fraud Features Following Genuine Transactions on Real Data.

	MLP		**Ridge**		**RF**		**Naive**	
	Mean	*Std*	*Mean*	*Std*	*Mean*	*Std*	*Mean*	*Variance*
X_TERMINAL	-0.90	0.50	-0.03	0.0011	-0.05	0.0018	-9.71e+07	6.41e+16
Y_TERMINAL	-0.02	0.00074	-0.01	0.00029	0.0084	0.00051	-7.64e+05	3.13e+12
TX_AMOUNT	-0.03	0.0017	-0.0082	0.00032	-0.0034	0.00066	-0.31	0.19

Table 6.2: Accuracy of ADV-O in Predicting Subsequent Fraud Features on Real Data.

	MLP		**Ridge**		**RF**		**Naive**	
	Mean	*Std*	*Mean*	*Std*	*Mean*	*Std*	*Mean*	*Variance*
X_TERMINAL	-4.40	78.59	0.19	0.017	0.21	0.008	-6.55e+08	8.92e+18
Y_TERMINAL	0.035	6.01	0.93	0.001	0.93	0.001	-3.28e+07	4.57e+16
TX_AMOUNT	0.12	0.025	0.13	0.023	0.085	0.076	-0.16	0.048

6.5.5 Results on real data

On the basis of the metrics discussed before, we first assess the accuracy of the models predicting the fraudsters' behavior (Subsubsection 6.5.5.1) and then the quality of the related oversampling algorithm (Subsubsection 6.5.5.2).

Table 6.3: Comparison of Accuracy Metrics for Different Models on Real and Synthetic Data: the table presents the mean and standard deviation values for various performance metrics such as Precision, Recall, F1 score, PRAUC, PRAUC_C, Pk50, Pk100, Pk200, Pk500, Pk1000, and Pk2000. The models evaluated include ADVO, Baseline, Baseline_balanced, CTGAN, KMeansSMOTE, Random, SMOTE, and TIMEGAN. The metrics provide insights into the effectiveness of these models for the task, with higher values indicating better performance. The best models for each metric, according to the statistical tests, are highlighted.

(a) Real Data

	Baseline		**Baseline_bal**		**CTGAN**		**KMSMOTE**		**Random**		**SMOTE**		**TimeGAN ADV-O**		**MIMO ADV-O**	
	mean	*std*	*mean*	*std*	*mean*	*std*	*mean*	*std*	*mean*	*std*	*mean*	*std*	*mean*	*std*	*mean*	*std*
Precision	**0.18**	0.05	0.01	0.0	0.06	0.01	0.01	0.0	0.01	0.0	0.08	0.02	**0.1**	0.02	**0.09**	0.02
Recall	0.07	0.02	**0.76**	0.03	0.26	0.05	**0.48**	0.04	**0.47**	0.04	0.25	0.05	0.24	0.05	0.25	0.05
F1	0.1	0.03	0.02	0.0	0.1	0.02	0.02	0.0	0.03	0.01	**0.12**	0.02	**0.14**	0.02	**0.13**	0.02
PRAUC	0.05	0.02	**0.11**	0.02	0.05	0.01	0.06	0.01	0.06	0.02	0.06	0.02	**0.07**	0.02	**0.07**	0.02
PRAUC_C	0.08	0.02	**0.21**	0.03	0.08	0.02	0.1	0.02	0.1	0.02	0.09	0.02	**0.11**	0.02	0.1	0.02
Pk50	**0.29**	0.13	0.08	0.04	0.18	0.08	0.15	0.07	0.15	0.08	**0.26**	0.1	**0.25**	0.09	**0.25**	0.12
Pk100	**0.29**	0.13	0.08	0.03	0.18	0.06	0.15	0.06	0.16	0.06	**0.23**	0.09	**0.25**	0.08	**0.25**	0.1
Pk200	**0.26**	0.1	0.08	0.03	0.19	0.06	0.15	0.05	0.17	0.06	**0.24**	0.09	**0.27**	0.1	**0.25**	0.1
Pk500	**0.24**	0.1	0.08	0.02	0.2	0.08	0.15	0.05	0.17	0.07	**0.23**	0.09	**0.27**	0.1	**0.26**	0.11
Pk1000	**0.22**	0.08	0.08	0.02	0.21	0.07	0.15	0.05	0.17	0.06	**0.23**	0.09	**0.25**	0.07	**0.24**	0.08
Pk2000	0.18	0.06	0.08	0.02	0.19	0.06	0.16	0.05	0.2	0.06	0.2	0.07	**0.23**	0.07	**0.22**	0.06

(b) Synthetic data

	Baseline		**Baseline_bal**		**CTGAN**		**KMSMOTE**		**Random**		**SMOTE**		**TimeGAN ADV-O**		**MIMO ADV-O**	
	mean	*std*	*mean*	*std*	*mean*	*std*	*mean*	*std*	*mean*	*std*	*mean*	*std*	*mean*	*std*	*mean*	*std*
Precision	**0.26**	0.09	0.05	0.03	0.13	0.08	0.11	0.07	0.09	0.07	0.13	0.08	0.14	0.09	0.14	0.09
Recall	0.18	0.11	**0.9**	0.02	0.51	0.15	**0.59**	0.13	**0.78**	0.05	**0.58**	0.12	0.51	0.15	0.51	0.15
F1	0.2	0.1	0.1	0.05	**0.2**	0.1	0.18	0.1	0.15	0.1	0.21	0.11	**0.21**	0.11	**0.21**	0.11
PRAUC	0.17	0.11	**0.3**	0.11	0.22	0.13	0.2	0.12	**0.23**	0.13	**0.24**	0.13	0.22	0.13	0.22	0.12
PRAUC_C	0.32	0.15	**0.48**	0.08	0.38	0.15	0.39	0.11	**0.45**	0.1	0.41	0.12	0.39	0.15	0.38	0.14
Pk50	**0.53**	0.14	0.08	0.07	**0.43**	0.14	0.18	0.15	0.15	0.13	0.28	0.17	**0.46**	0.13	**0.46**	0.14
Pk100	**0.52**	0.14	0.09	0.08	**0.43**	0.1	0.18	0.12	0.14	0.11	0.28	0.13	**0.46**	0.11	**0.46**	0.12
Pk200	**0.5**	0.14	0.11	0.07	**0.44**	0.1	0.21	0.11	0.14	0.09	0.28	0.12	**0.46**	0.11	**0.45**	0.11
Pk500	0.4	0.16	0.12	0.08	**0.42**	0.11	0.22	0.1	0.13	0.08	0.28	0.11	**0.42**	0.11	**0.42**	0.11
Pk1000	0.31	0.14	0.12	0.07	**0.36**	0.12	0.21	0.09	0.13	0.09	0.27	0.11	**0.36**	0.12	**0.36**	0.11
Pk2000	0.23	0.12	0.13	0.08	**0.28**	0.11	0.2	0.1	0.13	0.09	0.26	0.11	**0.28**	0.11	**0.28**	0.11

Table 6.4: Accuracy of ADV-O in Predicting Subsequent Fraud Features on Synthetic Data.

	MLP		**Ridge**		**RF**		**Naive**	
	Mean	*Std*	*Mean*	*Std*	*Mean*	*Std*	*Mean*	*Variance*
X_TERMINAL	0.938	0.00023	0.914	0.00023	0.948	0.00020	-150.87	46111.70
Y_TERMINAL	0.930	0.00022	0.899	0.00027	0.950	0.00017	-105.09	22282.78
TX_AMOUNT	0.858	0.00023	0.855	0.00023	0.825	0.00025	-0.35	0.71

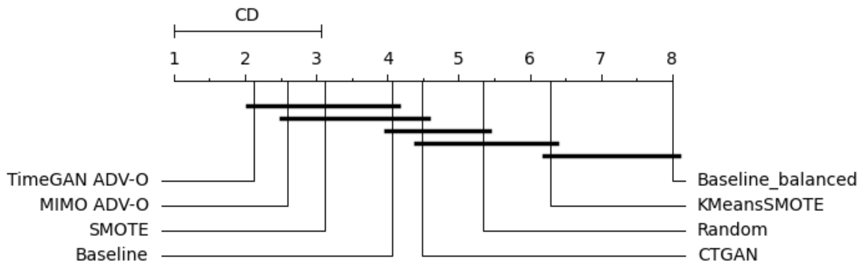


Figure 6.2: The critical difference (CD) plot shows the results of the Friedman-Nemenyi test for model comparison using the metric Pk1000 over real data.

6.5.5.1 Assessment of the fraudster model: results and discussion

This section considers four different regressors to predict the fraudster behavior in both genuine-to-fraud and fraud-to-fraud scenarios (Section 6.4): Random Forests, Feed Forward Neural Networks, Ridge Regressors, and a simple persistent model (called "Naive") which returns the latest observed value as a prediction. All algorithms (except Naive) are trained with a standard random grid cross-validation strategy for hyperparameter tuning[2] [129].

The preliminary experiment concerns the genuine-to-fraud scenario in its simplest setting by considering only continuous inputs. Table 6.1 shows the accuracy of ADV-O in predicting the features of the first fraud perpetrated on a hacked card: the table displays the regression results for the variables X_TERMINAL, Y_TERMINAL, and TX_AMOUNT, as predicted by MLP, RF, and Naive models on real data. Both the mean and standard deviation of the R2 metric for each variable predicted by the models are presented. The results suggest that it is particularly hard to have accurate predictions in this setting, though the Random Forest can predict some features with an R^2 somehow larger than 0 (and than the naive R^2). A possible explanation might be that many characteristics of a card, including its transaction history, may be impossible to retrieve for someone who has stolen or cloned the card. However, it must be noted this does not prove that fraudsters do not adapt their behavior to the card they have access to, but that we do not have any hint in this direction from our data.

We then consider the fraud-to-fraud scenario. We compare the accuracy of different regressors to a third approach, called naive, which simply reproduces the previous fraud. The results, presented in Table 6.2, reveal a significant difference between the

[2]Note that this may penalize algorithms like the Neural Network, which sometimes requires a more ad hoc training procedure than the other learners.

terminal features and the amount. The terminal's features exhibit certain regularities, which appear to be absent in the amount. Overall, the results support the assumption that learning a reliable model of the fraudster's behavior from historical data in a fraud-to-fraud scenario is possible and recommended. At the same time, there is still space for improvement. Given the performances shown in Table 6.2, we adopt Random Forest in the following.

6.5.5.2 Assessment of the oversampling approach: results and discussion

This section benchmarks MIMO ADV-O and TimeGAN ADV-O against two Baselines (no oversampling and undersampling through the use of a Balanced Random Forest) and four state-of-the-art oversampling strategies: random oversampling (RANDOM), SMOTE, K-Means SMOTE, and CT-GAN. We implement MIMO ADV-O using the regressor that, on average, performs better in Subsubsection 6.5.5.1, i.e., the Random Forest Regressor. We then apply the same oversampling factor (10%) to all oversampling algorithms.

Regarding SMOTE, we utilize the implementation provided by the Imbalanced-learn [93] library, using the default values for all parameters and setting the oversampling ratio to the one used in ADV-O. We adopt the default structure for CT-GAN: the generator is a two-layer fully-connected neural network with Batch-normalization and Relu activation functions, followed by a mix of activation functions to generate the synthetic row representation; the discriminator is also a two-layer fully-connected neural network with a LeakyReLU activation function and dropout for each hidden layer. For K-Means Smote, we tune the required proportion of minority class samples to filter a cluster: we set the `cluster balance threshold` to 0.1.

We assess here the impact of the oversampling technique on the final detection accuracy and requires the choice of a specific binary classifier.

We adopt a Random Forest classifier for the following reasons: (i) Random Forests have been shown to outperform other models in our previous studies [49, 27], (ii) they are beneficial to establish feature importance, which is highly valued by investigators to interpret the fraud detection process, and (iii) they provide a natural ensemble that can be easily exploited to address the imbalance, e.g., by feeding each decision tree with a balanced subset of the original data.

The assessment utilizes the Precision top K and the Area under the Precision-Recall curve, computed for the test set. Overall, Table 6.3a shows that both ADV-O algorithms outperform all other oversampling algorithms, with timeGAN ADV-O being slightly superior in most metrics. The comparison with the baselines is interesting. First, we can see that the balanced random forest outperforms all competitors in terms of PRAUC and PRAUC_C. Still, it pays a heavy price in terms of precision top K. Conversely, the baseline aligns with most other approaches in all metrics and is the best approach for low values of K. However, for large values of K, MIMO ADV-O and TimeGAN ADV-O are the best methods. Among traditional oversampling algorithms, SMOTE best holds the comparison with the ADV-O algorithms, especially concerning the precision top K. This is partially in line with previous results in fraud detection literature, which highlight how SMOTE has proven to generally be the most competitive oversampling algorithm [81]. Moreover, we performed our experiments on few dimensions. Since SMOTE is known to have issues with high dimensional data [108], this may help explain its good performance. Overall, the choice of approach heavily depends on the metric one wants to maximize. Still, the results show that oversampling can be beneficial when maximizing the precision top K, with time-based approaches being particularly helpful. Concerning the comparison between MIMO ADV-O and TimeGAN ADV-O, both algorithms achieved a remarkably close performance. We expected these results, as both algorithms rely on the same principle of modeling the time-dependent nature of the frauds. Finally, TimeGAN ADV-O performs slightly better in most metrics, although the difference is not statistically significant. More tests on different datasets may help us understand whether a difference exists and if modeling data as time series gives us a significant advantage compared to only considering the direct dependencies between consecutive frauds.

The result of the Friedman-Nemenyi test using the PK1000 over real data can be found in Figure 6.2. The plot displays the average rank of each model on the x-axis (the lower, the better), with models grouped into sets that are not significantly different from each other (marked by horizontal bars). CD indicates the minimum detectable difference (CD) at a 95% confidence level.

6.5.6 Results on synthetic data

The availability of a simulator allows for the extension of the experimental session to a new synthetic dataset. First, we show that in a simple environment, where the relationship between two consecutive frauds is relatively simple, MIMO ADV-O can predict the features of the following fraud with a significantly high R^2, as shown in Table 6.4.

Concerning the classic metrics for classification, we repeat here the same analysis performed on real data. We show the results in Table 6.3b. Similarly to what happens in real data, undersampling is the best approach in terms of PRAUC, but it severely damages the precision top K of the algorithm. Again, for low values of k the baseline is the best method, while with larger values, the ADV-O algorithms and CT-GAN are the best approaches. Finally, it should be noted that time-dependent methods perform on pair or better than all other oversampling methods, confirming the results obtained on the real data. The results of the Friedman-Nemenyi test using the metric Pk1000 on synthetic data be found in Figure 6.3.

6.6 Statistical tests

6.6.1 Real Data

We present here the results of the statistical tests performed on real data. The critical difference (CD) plots display the results of the Friedman-Nemenyi test for model comparison using various metrics on synthetic data. The choice of metrics is based on the fraud detection literature. The plots display the average rank of each model on

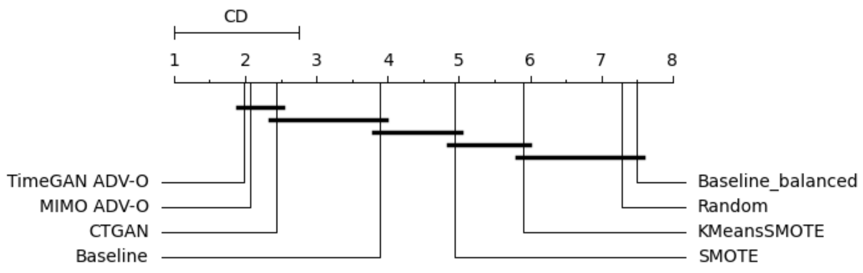


Figure 6.3: The critical difference (CD) plot shows the results of the Friedman-Nemenyi test for model comparison using the metric Pk1000 over synthetic data.

the x-axis (with lower values indicating better performance), with models grouped into sets that are not significantly different from each other (marked by horizontal bars). CD indicates the minimum detectable difference (CD) at a 95% confidence level.

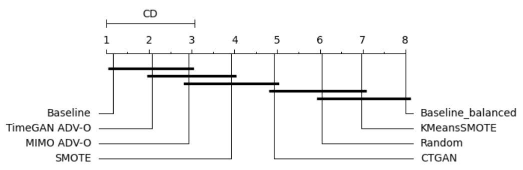


Figure 6.4: Critical difference plot using the Precision metric

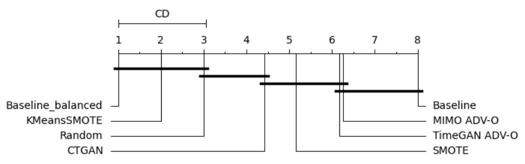


Figure 6.5: Critical difference plot using the Recall metric

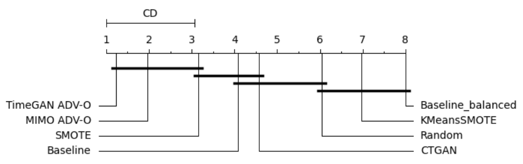


Figure 6.6: Critical difference plot using the F1 metric

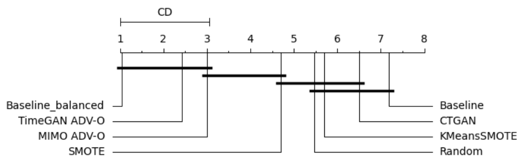


Figure 6.7: Critical difference plot using the PRAUC metric

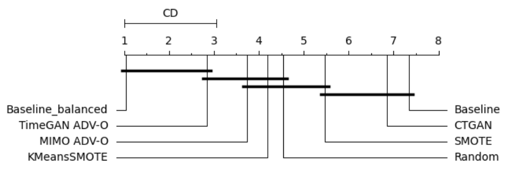


Figure 6.8: Critical difference plot using the PRAUC_C metric

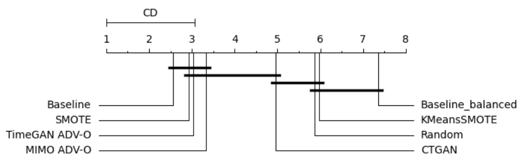


Figure 6.9: Critical difference plot using the Pk50 metric

6.6.2 Synthetic data

We present here the results of the statistical tests performed on synthetic data. The critical difference (CD) plots display the results of the Friedman-Nemenyi test for model comparison using various metrics on synthetic data. The choice of metrics is based on the fraud detection literature. The plots display the average rank of each model on

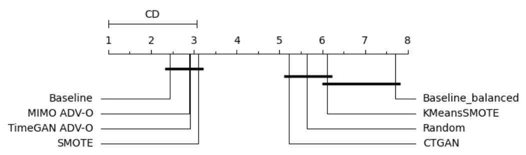


Figure 6.10: Critical difference plot using the Pk100 metric

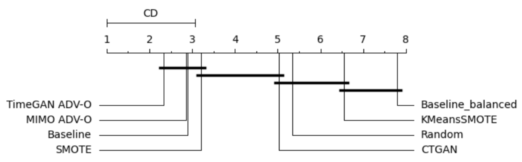


Figure 6.11: Critical difference plot using the Pk200 metric

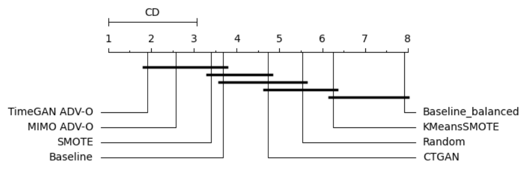


Figure 6.12: Critical difference plot using the Pk500 metric

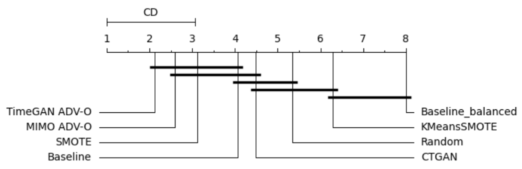


Figure 6.13: Critical difference plot using the Pk1000 metric

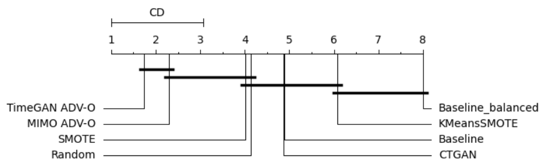


Figure 6.14: Critical difference plot using the Pk2000 metric

the x-axis (with lower values indicating better performance), with models grouped into sets that are not significantly different from each other (marked by horizontal bars). CD indicates the minimum detectable difference (CD) at a 95% confidence level.

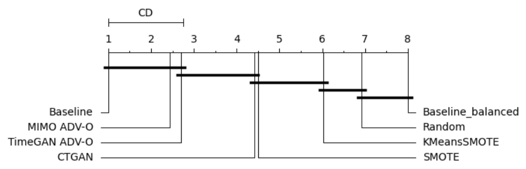


Figure 6.15: Critical difference plot using the Precision metric

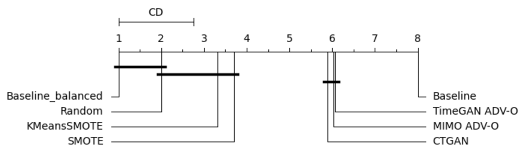


Figure 6.16: Critical difference plot using the Recall metric

6.7 Conclusions and Future Work

Imbalanced learning can severely limit the possibility of learning from the data if not adequately addressed. In this work, we propose a novel approach to address this problem, which considers the existence of behavioral patterns exhibited by fraudsters. We compose these patterns into two components: the adaptation to the card they steal and the correlation and dependency among the frauds they perform with each card. Our experiment shows that the latter is the most promising approach. To enhance the quality of oversampling algorithms for fraud detection, we propose a novel framework for oversampling in fraud detection that leverages the time-dependent nature of the fraud generation process through two distinct algorithms.

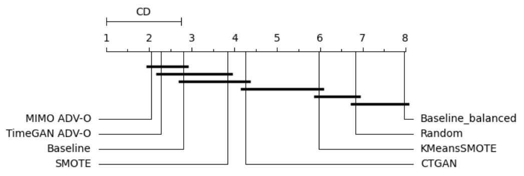


Figure 6.17: Critical difference plot using the F1 metric

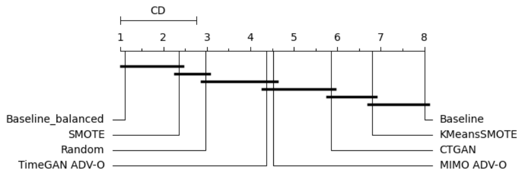


Figure 6.18: Critical difference plot using the PRAUC metric

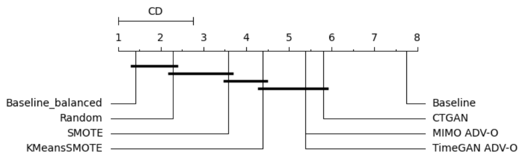


Figure 6.19: Critical difference plot using the PRAUC_C metric

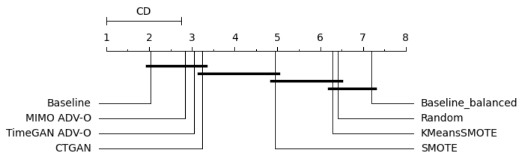


Figure 6.20: Critical difference plot using the Pk50 metric

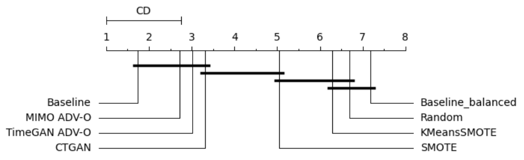


Figure 6.21: Critical difference plot using the Pk100 metric

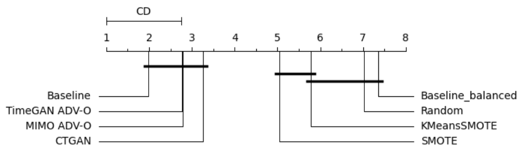


Figure 6.22: Critical difference plot using the Pk200 metric

First, we introduce MIMO ADV-O. This novel oversampling algorithm models the frauds generation process as the dependency of each fraud from the previous one performed with the same card. It uses it to craft plausible frauds for a compromised card. Then, the dependence of each fraud on the previous ones allows us to model each chain of frauds and use oversampling algorithms explicitly designed for time series generation. In particular, we propose TimeGAN ADV-O, an adaptation of the time series oversampling algorithm TimeGAN to the context of fraud detection. Our experiments show that the accuracy of a classifier trained on datasets oversampled with both ADV-O algorithms is comparable to or better than the best competitors.

Possible future extensions should concern the impact of feedback delay and concept drift, which have already been investigated in our previous works [46, 44, 27]. Other interesting research lines could concern the impact of feature-engineering and feature interpretability (e.g., by means of Shapley values [149]) on the accuracy of the fraudster model. Moreover, the work done in [60] shows that applying time-series classification techniques to fraud detection data can improve the classification performance of fraud detection engines, even when combined with static oversampling algorithms. Since we generate synthetic time series, feeding the resulting frauds to time series classifiers seems a natural extension of our work. Finally, considering frauds as time series opens the door to totally different approaches to fraud detection, where the compromission of a card can be seen as a *change* instead of an *anomaly*, and time-series classifiers can be used in addition or in assistance to existing methods.

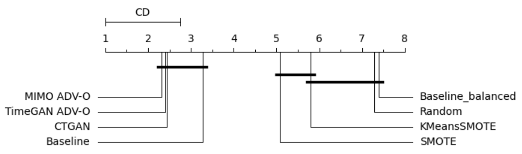


Figure 6.23: Critical difference plot using the Pk500 metric

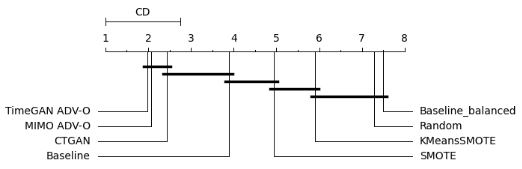


Figure 6.24: Critical difference plot using the Pk1000 metric

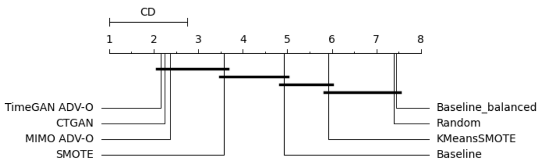


Figure 6.25: Critical difference plot using the Pk2000 metric

Chapter 7

FRAUD-RLA: A new reinforcement learning adversarial attack against credit card fraud detection

7.1 Introduction

In Chapter 5, we examined in detail how factors such as the feature engineering process, the online nature of the problem, the presence of multiple cards, and the overall complexity of the fraud detection pipeline pose substantial challenges to most attack strategies in the literature. Furthermore, we have observed how attacks designed initially for image recognition tasks are especially ill-suited to this domain. Such attacks aim at creating attacks that are imperceptible to the human eye, but these assumptions have no meaningful analogue in fraud detection.

Although the attacks defined in Section 4.3 remain possible and inherently risky, our analysis has shown that they generally require access to extensive datasets and knowledge of customers' historical transaction records. One might therefore be tempted to conclude that hiding or restricting access to this information would be sufficient to prevent meaningful attacks. The results presented in Chapter 6 strengthen this impression by showing that past transactions do not significantly influence fraudsters, possibly suggesting that they do not normally have access to this information.

Still, the security analysis remains incomplete. We must address a crucial open question: Is it possible to design effective attacks that do not rely on knowing datasets or past transactions? Answering this question is crucial to avoiding a false sense of security. In this work, we do it by introducing a new threat model specifically designed to take into account the main particularities of credit card fraud detection. Our threat model reduces the assumptions about attackers' capabilities by assuming no prior knowledge whatsoever, making the attacks easier to reproduce and thereby posing a more existential threat to credit card fraud detection systems. To work under these strict assumptions, we propose our main contribution, a novel adversarial attack against credit card fraud detection systems grounded in Reinforcement Learning [151, RL]. We model the attack as an RL problem, where the agent's objective is to maximize their revenue, which is computed as the total amount of transactions they can perform without being detected by the FDS. In that context, the agent is rewarded with the transaction's amount when it successfully fools the classifier. We identify Proximal Policy Optimization [143, PPO] as a suitable RL algorithm, and use it to create our attack named FRAUD-RLA (Reinforcement Learning Attack).

Our analysis and experiments show that FRAUD-RLA is the most effective attack against credit card fraud detection systems. Unlike previous works, FRAUD-RLA does

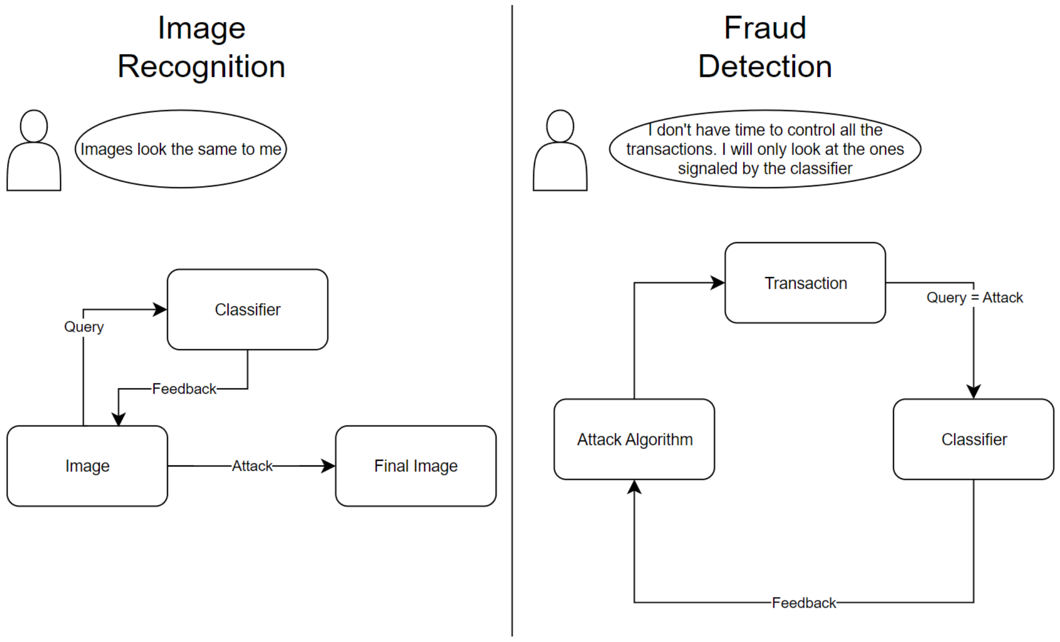


Figure 7.1: Visual representation of attacks in image recognition (left) and fraud detection (right). Traditional image attacks assume that multiple queries can be performed before the final attack is ready to be evaluated. Fraud detection instead must be performed as a continuous loop; each card is a transaction that helps collect money if undetected and can lead to the card being blocked otherwise.

not require the same degree of access to the model or the users' transaction history. Moreover, using RL allows us to explicitly model the trade-off between the exploration of the classifier boundaries and the exploitation of the best known attack pattern, widely known as the exploration-exploitation dilemma in the RL literature [106]. To the best of our knowledge, FRAUD-RLA is the only attack in the literature that addresses the domain-specific challenges of fraud detection while requiring no knowledge of customers' history or interaction with the fraud detection system beyond performing transactions. To demonstrate the effectiveness of our approach, we utilize FRAUD-RLA to attack a fraud detection system built according to the state-of-the-art literature [92, 123].

To facilitate comparison, we relax our threat model to enable a comparison with one of the few attacks in the literature specifically designed for credit card fraud detection, Mimicry [32]. The results, shown in Section 7.4, are mixed. Mimicry, which has access to the fraud detection system's training data and operates in an environment without concept drift or adversarial retraining, outperforms FRAUD-RLA in terms of the number and total amount of successful fraud. Still, FRAUD-RLA is capable of quickly improving its performance and stabilizing at over 200 dollars stolen per card.

Attacking fraud detection systems without prior knowledge is a complicated task, as shown by the relative effectiveness of Mimicry compared to our approach. Nevertheless, our work highlights that, even in this case, effective attacks are still feasible and should be taken into consideration when assessing the robustness of fraud detection. While further experimental validation on both synthetic and real-world datasets is needed to fully assess the risk, our results already illustrate the significant vulnerabilities exposed by adaptive, learning-based attackers.

Outline The rest of this Chapter is structured as follows. Section 7.2 introduces the threat model of our approach, defining the goals, knowledge, and capabilities of attackers. We discuss our approach in Section 7.3 and demonstrate its effectiveness through the experimental analysis presented in Section 7.4. Finally, Section 7.5 concludes the paper and discusses promising future research directions to improve the robustness of fraud detection systems.

7.2 Threat model

Goal In the context of credit card fraud detection, the objective of the attackers is to maximize the cumulative amount of transactions they can perform using a set of stolen cards. Such transactions must be conducted within a designated set of terminals that are either under the control of the attacker or that facilitate the sale of goods intended for subsequent sale on the black market, thereby effectively recycling the spent amount.

Fraudsters can steal cards at various times during the attack and use them to create fraudulent transactions. During the attack, customers can detect that their card has been stolen and block it, making the card unavailable. Alternatively, the bank can detect fraud and block the corresponding card at its own initiative to prevent further damage. Hence, fraudsters are presented with an exploration-exploitation dilemma [106] where they need to maximize their profit while simultaneously exploring new strategies to increase it in the future.

Knowledge In the computer security literature, the principle of "no security through obscurity" [142] is widely recognised. This principle encapsulates the fact that any concealed information could fall into the wrong hands, and thus, one should consider that the attackers have perfect knowledge of the system. Credit card fraud detection, however, presents a peculiar situation. While it is true that a talented attacker may, in principle, retrieve any information, for instance through social engineering [139], such attacks are unlikely to be scalable enough to pose a significant threat to fraud detection systems. For this reason, we assume that attackers may know the principles around which the system is built, like its feature engineering process, but not the weights of the trained classifier or the specific hard-coded rules, as both are difficult to obtain and generally change over time [28].

A second form of knowledge, specific to this application, is *data knowledge*, not to be confused with the *features knowledge* described by Suciu, Marginean, Kaya, Daume III, and Dumitras [148]. While *features knowledge* focuses on knowing the features engineering technique applied by the classifier, it still assumes that the attacker has complete knowledge of the input location in the original data space. In credit card fraud detection, however, each transaction is also evaluated based on aggregated features, as discussed in Chapter 3. Previous works Carminati, Santini, Polino, and Zanero [33] hypothesize that user devices could be infected with malware that can be used to observe previous card's transactions. This hypothesis, however, poses a strong requirement on the attackers' side and may limit the number of attacks they can perform. For this reason, this work focuses on the more general assumption that attackers know only the feature transformation and the general model characteristics. Previous fraud detection literature corroborates this hypothesis, as most fraudsters seem not to be influenced by the cardholder's behaviour before their first fraud with a new stolen card [101]. Finally, we assume that attackers have no access to the training set of the algorithm or similar data, forcing them to learn everything while performing the attacks.

Capabilities Attackers can directly control which terminal they want to connect to (among the ones under their control or intended for the black market), the transaction amount, and the date and time of the transaction. Cards' maximum spending is limited by the balance, card type, and bank rules. Attackers cannot extract value higher than the maximum spending capability of a card over the attack time. Furthermore, attackers can not modify fixed card features, such as the card holder, nor terminal features, such as its location. They may, in principle, find values that allow them to adjust as they want the aggregated features [33]. However, this would require a complete knowledge of the feature generation process and the previous transactions performed with the used card and terminal, which they do not generally have.

7.3 Fraud-RLA

In this Section, we present our main contribution, FRAUD-RLA, a new attack against credit card fraud detection systems based on Reinforcement Learning.

7.3.1 Problem formulation

In Fraud-RLA, we consider a bank system that essentially consists of a set of cards, C, and a set of terminals, τ. In this system, a series of transactions are performed chronologically from these cards to these terminals. The bank system has a trained classifier f at its disposal to determine whether a transaction is fraudulent or not. Note that when referring to the classifier f, we view it from the attacker's perspective; both human-made rules and machine learning classifiers are encompassed within f, which is therefore a highly non-linear and non-differentiable alogorithm.

We assume that the attacker has access to a pool $C_{atk} \in C$ of stolen cards and is able to perform fraudulent transactions with a set of accomplice terminals $M_{atk} \subset M$. At each instant, the decision-making problem of the attacker is fourfold. It must decide

1. which card from the pool to use;
2. in which terminal ($\in M_{atk}$) to perform the transaction;
3. what amount to spend;
4. when to perform the transaction.

When the attacker performs a transaction, the bank system uses f to classify it as genuine or fraudulent. As soon as a transaction is identified as fraudulent, the bank's system blocks it. Any subsequent transaction of the attacker with that card is rejected. Alternatively, cardholders can detect a fraudulent activity with their card and block it at their own initiative.

During the attack, we assume that the attacker is able to continuously replace any blocked card of the pool by a new one.

7.3.2 RL Environment

We model the decision-making problem as a single-agent partially observable Reinforcement Learning problem, where the environment is the bank system. Formally, the environment is a Partially Observable Markov Decision Process [151, 121, POMDP] defined by the tuple $\langle S, O, A, T, R, \Omega \rangle$ where S is the state space, O is the observation space, A is the action space, $T : S \times A \to S$ is the transition function, $R : S \times A \times S \to \mathbb{R}$ is the reward function, and $\Omega : S \to O$ is the observation function.

A state $s \in S$ of the environment encapsulates the complete state of the bank system at a specific date and time, i.e., the history of past transactions, the current card pool P available to the attacker, and the set of existing cards C and terminals τ.

In contrast, the agent only receives a partial observation of the state since the agent does not know the identity of the cardholder nor the card's transactions history, as discussed in Chapter 3. As such, at any given time, the agent can query the environment about a specific card $c \in P$ to receive an observation $o \in O$ of the current state $s \in S$. An observation contains only three pieces of information: the time elapsed since the card was stolen (i.e., added to the card pool), the current date and time, and whether the card is a credit or debit card.

An action $a \in A$ essentially represents a transaction to submit to the bank system. As such, an action consists of a multivariate vector comprising a strictly positive amount, a terminal location, and a positive time delta for when to perform the transaction with regard to the current timestamp. The reward $r_t = R(s_t, a_t, s_{t+1})$ is either zero if the transaction has been blocked (by the bank system or the cardholder) or if the amount exceeds the customer's balance. It is instead the transaction amount if the transaction is successful.

In FRAUD-RLA, we handle episodes on the card-level, independently of any other card. As a result, an episode is a sequence of transactions made with a single card, and an episode ends when the card is blocked.

7.3.3 Agent objective and training

In RL, the objective of the agent is to find the optimal policy π^* that maximizes $V^\pi(s) \forall s \in S$, i.e., the expected discounted sum of rewards under a given policy π (parametrized in θ, see Section 2.3), resulting in:

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0} \gamma^t R(s_t, a_t, s_{t+1}) | s_0 = s\right] \tag{7.1}$$

Since we are working in the scope of partial observability, the policy π takes as input an observation of the current state $o = \Omega(s)$, and optionally hidden states when working with recurrent networks, as discussed in Subsection 7.3.4.

It is important to acknowledge that the time is continuous in FRAUD-RLA. Moreover, since the agent chooses the timestamp of the next transaction, the time difference between two pairs of consecutive states is not constant and has to be accounted for during the optimization process. Concretely, the task designers have to determine "how much money is worth tomorrow in comparison to today" for the attacker and set the value of the distance factor γ accordingly (in this work, we use 0.99, but other choices are generally possible). The overall agent training loop is represented in Algorithm 7.

Algorithm 7 Fraud-RLA maximises the sum of transaction amounts over training time.

```
Require: Environment $M$, cards $C$, terminals $\tau$, classifier $f$, pool size $p$
 1: Initialize policy parameters $\theta$
 2: $action\_buffer \leftarrow$ PriorityQueue()
 3: for $c_i \in \{c_1, c_2, \ldots, c_p\} \subset C$ do                  ▷ Sample $p$ cards
 4:     $o \leftarrow$ M.observe($c_i$)                                  ▷ Retrieve observation
 5:     $a \leftarrow \pi(o)$                                            ▷ Compute action
 6:     $action\_buffer$.add($a$)                                        ▷ Buffer for later
 7: end for
 8: while $|M.buffered\_actions| > 0$ do
 9:     $card, trx \leftarrow M$.next_transaction()
10:     $M$.forward_until($trx$)
11:     $agg \leftarrow M$.aggregate($trx$)
12:     if $f(trx, agg) = 1$ then                                        ▷ If labelled as fraud
13:         $r_t \leftarrow 0$
14:         $C \leftarrow C \setminus \{card\}$                          ▷ Revoke card
15:         $card \sim U(C)$               ▷ Sample a new card using a uniform distribution
16:     else
17:         $r_t \leftarrow trx.amount$
18:     end if
19:     Update policy parameters $\theta$ according to $r_t$
20:     $o \leftarrow M$.observe($card$)
21:     $a \leftarrow \pi(o)$
22:     $action\_buffer$.add($a$)
23: end while
```

7.3.4 Agent policy

We identify Proximal Policy Optimization [143, PPO] as a suitable single-agent Deep Reinforcement Learning algorithm for FRAUD-RLA due to its ability to handle continuous action spaces. Additionally, PPO is known to require little hyper-parameter tuning compared to other Deep RL methods and has been proven to perform well in a wide variety of tasks [143, 168].

Internally, PPO uses an actor-critic architecture [14] (see Section 2.3). Since the environment is partially observable, we present two variants of PPO in this work: a first

Table 7.1: Actor network architecture of PPO.

Layer type	**Activation**	**Output size**
Input		*observation_size*
Linear	Tanh	32
Linear	Tanh	32
Linear		$n\_actions + n\_actions^2$

Table 7.2: Critic network architecture of PPO

Layer type	**Activation**	**Output size**
Input		*observation_size*
LayerNorm		
Linear	Tanh	32
Linear	Tanh	32
Linear		1

with linear layers and a second with recurrent ones (Gated Recurrent Units [40]). The critic network (detailed in Table 7.2) takes the agent observations as input and outputs the observations' values. The actor network (detailed in Table 7.1) takes as input the agent observation and outputs a probability distribution over the actions. In the following, we refer to the recurrent and non-recurrent versions as PPO and R-PPO (Recursive PPO), respectively.

Although the network architectures are similar to other works in the field of single-agent RL with continuous action spaces, we differ from most of them by learning both the means and the covariance matrix of the multivariate normal distribution from which the actions are sampled. In contrast, most works in the domain [143, 57] only learn the means of a normal distribution and either use a hand-crafted constant or an annealed value for the variance.

Our choice to learn the covariance matrix is motivated by two reasons. First, the covariance matrix represents the covariance of the features in the action space, such as the amount and the terminal location. These features are very likely to be correlated. For instance, it is reasonable to assume that the day of the week has an influence on the amount to spend, as spending habits are likely different on weekends than on weekdays.

Then, we assume that the attackers operate without prior knowledge of the data, forcing them to learn it throughout the training, and this includes not knowing the correlations among the features. The need to learn the covariance matrix is the reason why the Actor-Network has $n\_actions + n\_actions^2$ outputs, with $n\_actions$ values are for the means and $n\_actions^2$ values for the covariance matrix of the multivariate normal distribution from which actions are sampled. The covariance matrix output by the actor network is assured to be positive definite by multiplying it by its transpose.

A noticeable difference between PPO and R-PPO is that we train PPO on batches of transitions, whereas R-PPO is trained on whole episodes to account for the hidden states. Similarly to VAE and to the system classifier, we ran hyperparameter tuning with Optuna and report the final hyperparameters in Table 7.7.

7.3.5 FRAUD-RLA as a problem space attack

FRAUD-RLA shares many similarities with existing problem space attacks, especially from the field of malware detection. We align with the formulation of Pierazzi, Pendlebury, Cortellazzi, and Cavallaro [132], where attacks are defined as a set of transformations to apply in the data space to obtain a certain goal while preserving a set of properties both in the data and the feature space. In this case, we can interpret the policy of the agent as a set of transformations in the state space of the transaction time series, where we start from an empty series and progressively add transactions that respect a set of constraints (positive amount, reachability of the terminal, . . .). Notably, each change introduces some collateral effects, as the insertion of a transaction modifies the features and changes the way the following will be classified.

Compared with previous works, however, the problem faced by FRAUD-RLA presents significant differences, showcasing the need for a novel domain-specific attack.

7.4 Experiments

In this section, we present the experimental analysis of our work. We first describe our experimental setup, including the datasets, baselines, and methodology used. We then comment on the results, showing the effectiveness of the attacks. Finally, we provide a qualitative analysis of the frauds designed by the algorithms, using them to propose possible mitigation approaches.

7.4.1 Dataset and simulation

Anonymized real-world datasets are not suitable for this work, as the employed anonymization process does not enable the fraud detection system to process transactions through aggregated features and to impose limitations on the transaction frequencies for each card. Consequently, multiple attacks performed on the same card could not be detected by the fraud detection system, unrealistically simplifying the attacker's task.

Instead, we use CardSIM [6] to create a dataset of transactions suited for testing adversarial attacks on credit card fraud detection. CardSIM first creates the cardholders and terminal populations from two different distributions. Terminals and cardholders are both characterized by a location and ID. The simulator then generates transactions (both genuine and fraudulent) from cardholders to terminals. Cardholders exhibit behavioural features, including a propensity to pay online, the use of a credit card, the average distance from the terminals they use, transaction amounts, and transaction frequencies. Compared to the original generator, we increase the data complexity by creating two opposite correlations between the customer location and the amounts of genuine and fraudulent transactions. Since the customers' and terminal locations are correlated, this aims to establish a stronger dependency between the transaction amount, terminal location, and the likelihood of the transaction being fraudulent.

The world we generate simulates transactions between $20,000$ payers and $2,000$ terminals over a period of 2 years. The overall resulting dataset has a total of over 13 million transactions, where the fraudulent samples account for $\approx 1\%$ of the population. We split the transactions chronologically in three parts: a warm-up set of 30 days (521k transactions) to bootstrap the feature aggregation, a training set of 150 days (2.6M transactions) to train the classifier described in Subsection 7.4.2, and the rest of the dataset is left available to simulate the normal world behaviour during the attack.

7.4.2 Fraud Detection System

We build the Fraud Detection System according to the principles outlined in previous fraud detection literature, as described in Chapter 3. The objective of fraud detection systems is to discriminate genuine transactions from fraudulent ones. To achieve this goal, we build an FDS comprising three main components: a machine learning classifier (specifically, a Balanced Random Forest [37, BRF]), a statistical classifier that penalizes outliers on specific features, and a set of handcrafted rules for sanity checks. Specifically, we want our rules to detect spikes in the number of transactions performed with any card. If any of the three classifiers raises an alert, we block the transaction.

The BRF has been chosen due to its previous use in fraud detection learning literature [26] and to its capability to directly handle datasets with imbalanced classes. Furthermore, random forests (and, by extension, BRFs) are known in the adversarial machine learning literature to be generally more robust against adversarial attacks compared to neural networks [56], making them a strong candidate for testing attacks under conservative assumptions.

To train the BRF, we warm up the bank system by processing every warm-up transaction chronologically so that feature aggregation bootstraps properly. Then, we process the training transactions chronologically and compute the aggregated features for each one to create the feature dataset. Note that we use the true label of the transactions for the terminal-wise aggregated risk score. The BRF is then trained on this dataset of features to discriminate genuine transactions from fraudulent ones.

We fit the statistical classifier on the training data to label any transaction whose amount or aggregated daily risk score exceeds the percentiles shown in Table 7.3 as fraudulent. Finally, the handmade rules limit the number of transactions that can be performed on an hourly, daily, and weekly basis, respectively, as shown in Table 7.3. When a customer exceeds the hourly, daily, or weekly transaction limit, the rule-based classifier labels the transactions as fraudulent that exceed the limit. Note that the statistical classifier severely hinders the performance of our system by generating multiple false positives. While choosing to implement such a strict classifier may not be an optimal business choice, we decided to include it in our experiments to increase task complexity and reduce the opportunities for attackers to bypass the classifier by using extreme values.

We selected the FDS hyperparameters via hyperparameter tuning with Optuna [2], where we jointly train the three classifiers to maximize the F1 score of the FDS on an unseen test set. The metrics on a validation set is summarized in Tables 7.4 and 7.5.

The system's real-time functioning is as follows. Whenever a transaction enters the system, we build its aggregated features and evaluate it. We then return the predicted label, insert the transaction into the system, and use it to evaluate future transactions.

Table 7.3: Hyperparameters for the FDS.

BRF	
Number of trees	139
Balance factor	0.0647
Statistical classifier	
Amount quantiles	0.9976
Daily risk score quantiles	0.9999
Rule-based	
Hourly max	8
Daily max	19
Weekly max	32

Table 7.4: FDS metrics on the validation set

Metric	**Value**
Accuracy	0.98
Precision	0.27
Recall	0.55
F1-score	0.36

Table 7.5: Confusion Matrix of the Fraud Detection System

	Predicted	
	Fraud	**Not Fraud**
Fraud	2842	2286
Not Fraud	7744	509525

7.4.3 Baselines

Compared to previous works and to the best of our knowledge, FRAUD-RLA is the first attack to model the intricate feature engineering process of bank systems, and to assume no prior knowledge of past customers' transactions or access to any data prior to the attack time. As a result, no existing attack in the literature can be directly compared and we consider two baselines issued from the dataset and one from the literature to compare against FRAUD-RLA. The first baseline is directly computed from the simulated (validation) dataset and corresponds to the simulated fraudulent transactions. We refer to this baseline as *simulated frauds*. The second baseline, also issued form the dataset, corresponds to a clairvoyant attacker that is able to completely empty the accounts, leaving a balance of zero after the attack. We refer to this baseline as *clairvoyant*.

A more significant baseline is Mimicry [32], which we found to be the closest approach to ours, although it requires some adaptations to account for the complexity of our threat model. Since Mimicry is an adversarial model that produces adversarial samples by learning from the distribution of real data, we must first relax our threat model and assume that the attacker can retrieve a dataset of past transactions. Therefore, we assume that the attacker has infiltrated a subset of terminals and is therefore able to

Table 7.6: Mimicry (VAE) hyper-parameters.

Parameter	**Value**
Number latent dimensions	86
Number of hidden dimensions	106
Learning rate	4.96×10^{-4}
Training bath size	10
Number of epochs	2791
Quantile	0.9946
Number of transactions to generate	541
Number of infiltrated terminals	82
Weight of the KL divergence in the loss β	0.2539

collect a dataset of transactions on these compromised devices. This relaxation allows for the recollection of a dataset of transactions and the training of the generative model.

The second adaptation is related to the chronological nature of the problem that Mimicry originally overlooks. Since Mimicry only has access to transactions from the compromised terminals, it is not able to learn a representative transaction frequency for individual cardholders. As a result, we rather train Mimicry to mimic the hour of the day in which transactions occur and perform the generated transaction at the next occurrence of that hour of the day, i.e., in less than the next 24 hours. Finally, to balance realism, frequency, and financial gain, we apply for each fraud the following iterative approach:

1. Generate a batch of transactions
2. Select the transactions in the top k quantile, k being a hyper-parameter of the algorithm
3. Sort the remaining transactions chronologically and select the closest one in time

In this work, we implement Mimicry as a Variational Auto-Encoder [84, VAE] (see Subsection 2.2.2). We train the VAE to learn the distributions of amounts, terminals, and hours of attack. To do so, we first take the transactions in the training set of a number of selected infiltrated terminals, and keep only the amount, terminal coordinates, and attack time. We then train a VAE to generate transactions that follow the same distribution as the original ones. Attacks are then generated by sampling a batch of candidate transactions from the terminal and selecting the one closest to the original time. We fine-tune the model with Optuna [2], and show the hyperparameter values in Table 7.6. The balance between attack frequency and stealth is implicit in the batch size; a smaller batch results in a higher probability of a longer time distance between consecutive attacks.

7.4.4 Experimental setup

The goal of this phase is to show that FRAUD-RLA is capable of rapidly improving its effectiveness, stealing hundreds of dollars from each card. We compare it with three

Algorithm 8 Filtering synthetic transactions using VAE

```
Require: Training set TR = $\{X_{tr}, Y_{tr}\}$, threshold percentile $k$, trained variational autoencoder VAE, time $t_0$
Ensure: Normal, close in time, high-value synthetic transaction
1: Generate a batch $\hat{X}$ using VAE
2: Select top $k\%$ percentile by amount:
       $\hat{X}_{\text{top}} \leftarrow \{x \in \hat{X} \mid \text{amount}(x) \geq P_K(\hat{X})\}$
3: Sort $\hat{X}_{\text{top}}$ by time distance from reference time $t_0$
4: Select closest transaction:
       $x^* \leftarrow \arg\min_{x \in \hat{X}_{\text{top}}} \text{time\_distance}(x, t_0)$
5: return $x^*$
```

main baselines: the expected value that fraudsters can extract in the original dataset, the amount stolen by Mimicry, and the total balance stolen cards have, which is the cap for our algorithm. Following the threat model discussed in Section 7.2, we randomly select 10% of the terminals to be the only ones in which the attacker can perform transactions, and we test the performance of the attacks over 2000 cards, managing a pool of 50 cards at a time. A card is either blocked because a transaction has been classified as fraudulent by the FDS or because the cardholder has reported the fraud. We model the fact that a cardholder will eventually mention a suspicious activity by randomly selecting an expiration date from a normal distribution with a mean of 7 days and a standard deviation of 1.5 days. For the rest of the work, we will also use VAE to refer to Mimicry, and PPO and R-PPO to the traditional and recurrent versions of FRAUD-RLA.

Table 7.7: PPO and R-PPO hyperparameters

Parameter	**PPO**	**R-PPO**
Actor learning rate	9.56×10^{-4}	7.75×10^{-3}
Critic learning rate	6.67×10^{-3}	3.8×10^{-3}
c_1 start	0.92	0.41
c_1 end	0.28	0.33
c_1 annealing duration	2980 episodes	2728 episodes
c_2 start	0.16	0.19
c_2 end	0.08	0.03
c_2 annealing duration	2012 episodes	1699 episodes
Train interval	63 transitions	6 episodes
Number of training epochs	15	52
Minibatch size	47	5
Normalize rewards	True	False
Normalize advantages	True	False
Gradient norm clipping	None	2.39
GAE λ	0.95	0.95
Epsilon	0.2	0.2

Note that, unlike the baselines (Mimicry and Simulated), our method (PPO and R-PPO) has to be trained as it interacts with the environment. We train our agent (PPO and RPPO) with the two baselines (Simulated and VAE) with regard to these two metrics and discuss the results. During testing, simulated transactions are processed chronologically by the FDS alongside the fraudulent transactions of the agent. Each of these transactions is classified and incorporated into the dataset, which enables the model to build aggregated features on top of them and apply frequency-detection rules, which prevents attackers from inundating the system with transactions. Finally, once a card has been detected or it exceeds its allowed time, we block it, meaning no further transactions are allowed with it.

7.4.5 Results

7.4.5.1 Total amount compromised

We plot the average amount compromised with the 4000 cards by the three agents in Figure 7.2. As expected, the clairvoyant the simulated frauds respectively have the highest and the lowest performance. The first thing to note is that VAE is remarkably close to the clairvoyant agent. This is to be expected given that our dataset does not exhibit any concept drift or require algorithm retraining, and the compromised terminals are sufficient to provide a proper representation of the dataset.

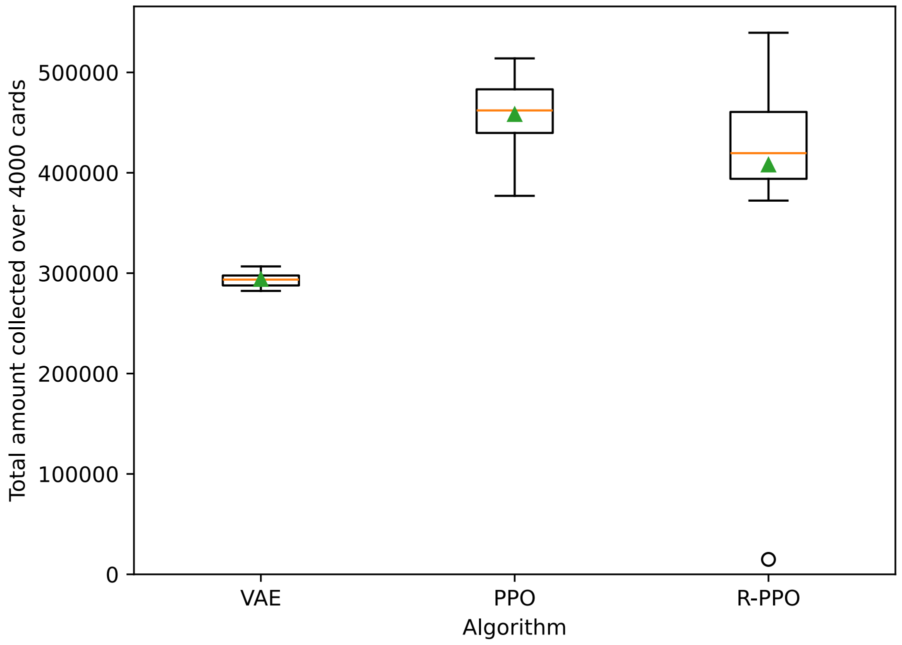


Figure 7.2: Total amount compromised with a budget of 4000 cards. These results are for 30 different seeds.

We can see that VAE compromises a total amount of 3.3M on average, while PPO and R-PPO, respectively, reach a total amount of 1.25M and 952k. In that regard, VAE is therefore three times 2.64× more effective than PPO and 3.45× more effective than R-PPO. It is however important to keep in mind that PPO and R-PPO are trained on-line and therefore take some time in the early stages of the attack in order to explore the classifier boundaries.

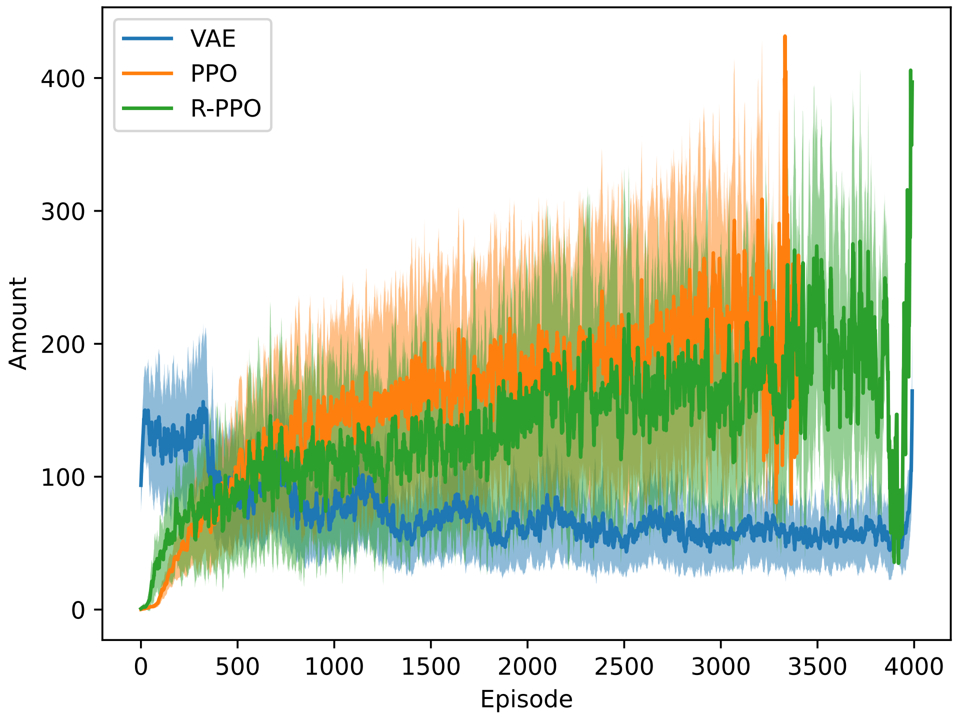


Figure 7.3: Average amount compromised per card over the course of the training. Results are averaged over 30 random seeds, shown with 95% confidence intervals and smoothed with a moving average with a window of size 20. Mimicry's performance decreases due to the presence of terminal aggregate features, which are initially irrelevant but become more important later. PPO attacks, on the other hand, tend to avoid detection and remain longer in the system; hence, the limited throughput used in these experiments causes PPO to only utilize a fraction of the available cards. This property highlights the importance of cards' availability when designing attacks.

7.4.5.2 Amount compromised per card

We show in Figure 7.3 the average amount collected per over the course of the 4000 cards of the training set. Note that multiple transactions can be performed with a single card and that this information is not represented in this plot.

We can see that PPO and R-PPO performance increases steeply with the first 1000 cards, then slightly increase between 1000 and 2000, and finally stabilize around their best policy, which is sensible with the c_2 hyperparameter annealing that encourages exploration fading around that value. By the end of the training, PPO and R-PPO are respectively able to compromise a total amount of 343 and 305 per card on average while VAE reaches an amount of 835 per card. In that regard, VAE is 2.43× more effective that PPO and 2.73× more effective than R-PPO.

7.4.5.3 Cause of detection

We show in Table 7.8 some insights on the causes of fraud detection for each algorithm, where we can see that VAE and PPO encounter very different challenges, with VAE being mostly blocked by the rule-based classifier while PPO and R-PPO are mostly blocked by the BRF.

Table 7.8: Causes of card blocking. The “Customer" row accounts for the cases where the card holder blocks its account after mentioning a fraudulent activity. Results are averaged over 30 random seeds and shown with standard deviation.

Blocked by	VAE	PPO	R-PPO
BRF	0.001 (±0.001)	0.673 (± 0.030)	0.559 (±0.061)
Statistical	0.000 (±0.000)	0.028 (± 0.003)	0.174 (±0.122)
Rules	0.226 (±0.021)	0.015 (± 0.002)	0.001 (±0.001)
Customer	0.772 (±0.021)	0.287 (±0.031)	0.265 (±0.067)

Table 7.8 shows the cause of detection for each card, with "Customer" meaning that the card was blocked by the customer after the interval was due, but without any transaction having been detected. The Table provides valuable insights on the attacks. First, the VAE, which has been trained on training data, produces data with statistical properties very close to those of genuine transactions and is therefore capable of systematically bypassing the Balanced Random Forest and the statistical classifier. Notably, the same does not always apply to rules, as excessive transaction frequencies are often detected by the FDS. PPO and R-PPO, on the other hand, quickly learn how to bypass the rules due to their adaptive nature. The BRF, with its more complex decision boundaries, constitutes a more significant challenge and is capable of blocking more than half of the cards used by both versions of the algorithm. The statistical classifier, and specifically the check on the amount, constitutes another challenge, as PPO is incentivized by its reward structure to output extremely high amounts. This provides us with an intuition on how to stop both attacks. First, the non-adaptive nature of VAE makes it vulnerable to carefully designed rules, especially if they concern aspects the algorithm cannot directly learn, such as the time delay between consecutive transactions from a customer. On the other hand, the statistical similarity between the frauds generated by VAE and genuine transactions makes it extremely challenging to stop them with traditional classification algorithms. On the contrary, stopping FRAUD-RLA seems to be mainly a question of making the tasks complex. Stopping it from making transactions whose value is too high and presenting it with complex classification algorithms both make the training process more challenging, limiting the total average amount the algorithm is capable of stealing.

7.4.6 Unsupervised detection and results

The main limitation in our experiments so far has been the limited impact of aggregated features on the system's decision. Since frauds in the dataset are inserted in an uncorrelated manner and do not present any temporal dynamics, the aggregated features play a limited role in the decisions taken by supervised classifiers trained on them.

To ensure the FDS is influenced by past transactions, we train an auxiliary unsupervised classifier with the goal of detecting anomaly patterns. Since we train the anomaly detector on all the features, it should be able to detect anomalous behaviors also in the aggregated features. As a classifier, we use the Isolation Forest (IFOR) classifier [95] due to its simplicity and robustness to outliers.

The resulting FDS has the same recall on genuine data, but due to the higher number of false positives, it has a slightly lower precision and F1 score (See Table 7.9). This aligns with previous fraud detection literature, which highlights the risk of false positives when using unsupervised approaches [26]. The positive change in terms of adversarial robustness, however, is dramatic. Mimicry, which lacks a concept of aggregated features, is severely affected, with the average stolen amount per card decreasing by over eight times (see Figure 7.4). Fraud-RLA is also affected; the amount stolen per card is also slightly lower, and PPO takes more time to perform frauds on each card, sometimes finishing after the deadline allowed in our experiments. Still, RL proves here significantly superior to Mimicry: the method adapts to the new, more complex scenario, and is capable of beating Mimicry both in terms of amounts per card (Table 7.4) and total stolen (Table 7.5.

The experimental results enable us to better understand which types of adversarial attacks are most effective under different settings. Specifically:

- **No dataset access** In situations where the attacker does not have access to any dataset to train Substitute models or Mimicry, Reinforcement Learning (RL)-based attacks seem to be the most viable option, representing a threat without any prior attackers' knwoedge.

- **Dataset available, minimal influence of side-effects**: When a dataset is available and side-effects are negligible (i.e., each attack is virtually independent and does not influence the detection of the following ones), mimicry-based attacks tend to outperform others.

- **Dataset available, high influence of side-effects, no access to past transaction history**: This is the scenario we observed when including the Isolation Forest classifier.Isolation Forest classifier. In such cases, FRAUD-RLA regains its advantage, as Mimicry tends to repeat the same attacks' patterns and lead to detection through aggregated features.

- **Dataset and access to historical stolen card transactions available** (as in Carminati et al. [32, 33]): While not directly tested in our experiments, prior literature suggests that when attackers have access to both datasets and detailed history of previously compromised accounts, Mimicry or substitute model attacks are particularly effective. These insights reinforce the idea that there is no universal attack strategy. Instead, approaches like FRAUD-RLA (and RL-based attacks more broadly) should be viewed as an essential component of attackers' broader possible attack methods.

Table 7.9: FDS metrics on the validation set after the inclusion of the Isolation Forest

Metric	Value
Accuracy	0.98
Precision	0.25
Recall	0.44
F1-score	0.32

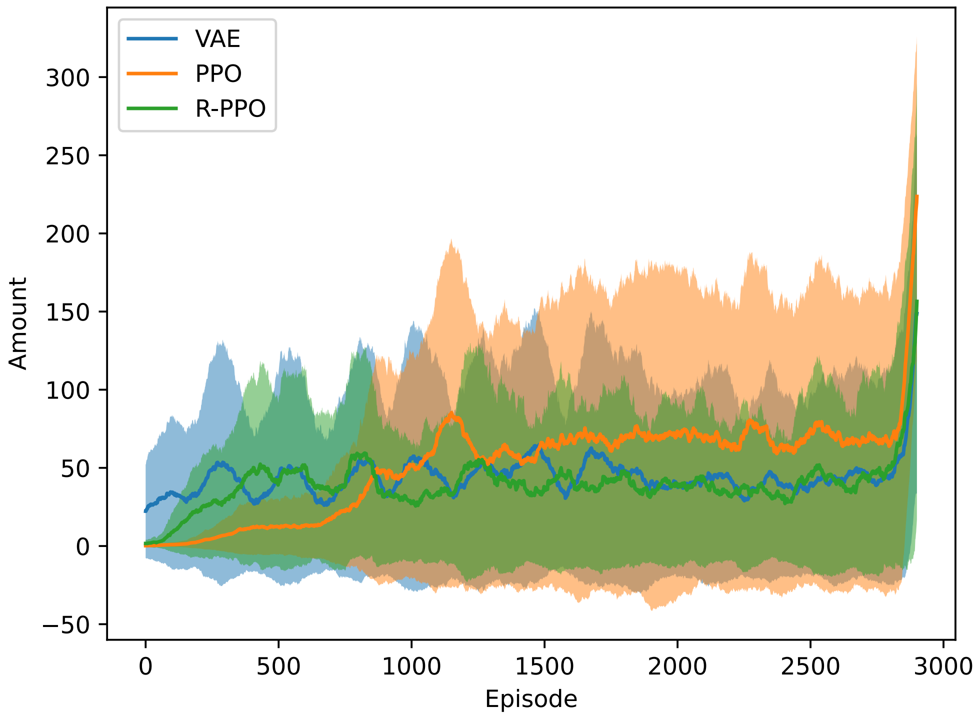


Figure 7.4: Average amount compromised per card over the course of the training. Results are averaged over 30 random seeds, shown with 95% confidence intervals and smoothed with a moving average with a window of size 20. The spike at the end is due to the way we record episodes; each episode is measured when it is completed, rather than when it enters the system. The last cards in the system are more likely to have survived multiple rounds than to have just been selected.

7.5 Conclusions and future works

Adversarial attacks are a growing threat, and credit card fraud detection systems are sensitive targets. The lack of research at the intersection of the two domains is a potential vulnerability for existing fraud detection engines. It increases the likelihood of catastrophic consequences should fraudsters identify and implement an effective attack strategy. This work aimed to mitigate this problem by analyzing the domain's main features, with a focus on the differences between it and more traditional domains, such as image recognition and malware detection.

Understanding the challenges attackers face is crucial when designing and modeling the threat of advanced machine learning for fraud detection. Therefore, we expanded exist-

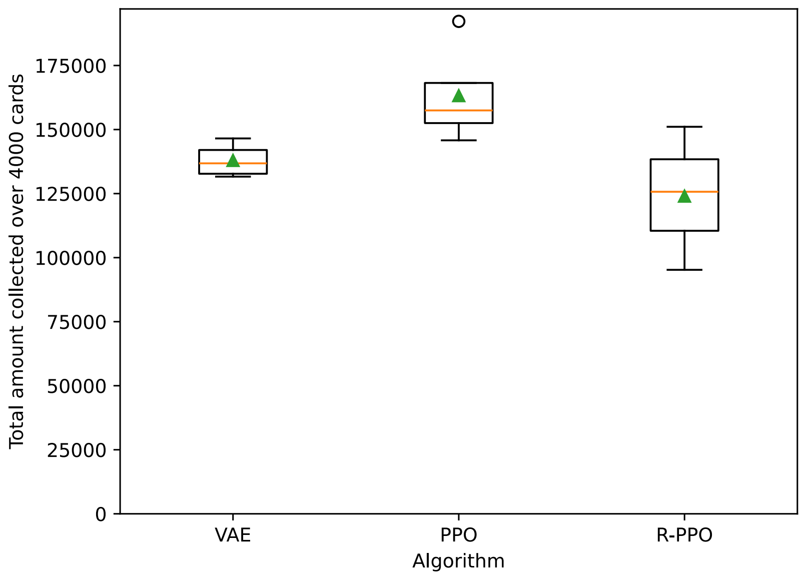


Figure 7.5: Total amount compromised with a budget of 4000 cards when including anomaly detection classification. These results are for 30 different seeds.

ing threat models to adapt them to credit card fraud detection. Specifically, we updated existing approaches to consider the lack of human supervision, the limited data knowledge and control, and the exploration-exploitation tradeoff. Using the defined threat model as a blueprint, we developed FRAUD-RLA, a novel attack designed explicitly to tackle the aforementioned issues. To do so, we modeled the problem of finding a successful fraudulent pattern in the shortest possible time as a Partially Observable Markov Decision Process, where the known fixed features represent the visible state and the action corresponds to selecting the optimal values for the controllable features. To optimize this problem, we employed Proximal Policy Optimization (PPO), a robust algorithm able to optimize continuous policies with little hyperparameter optimization.

To evaluate our approach, we developed a robust classifier utilizing state-of-the-art defenses to mitigate attacks. The resulting experiments, while highlighting the importance of having access to the classifier's training data, showcase the effectiveness of our approach in quickly learning an effective attack policy and bypassing the classifier. Future evaluation is required. First, the impact of concept drift was outside the scope of this work, but it may be significant for FRAUD-RLA and, even more so, for the compared baselines. Second, the use of adversarial training and other defenses was outside of the scope of this work, but should be carefully evaluated. More generally, further experimental validation on both synthetic and real-world datasets is needed to fully assess the risk. Still, our results already illustrate the significant vulnerabilities exposed by adaptive, learning-based attackers. Improving our understanding of adversarial security of fraud detection systems will be crucial in developing effective defenses. As security is frequently evaluated through the lens of red teaming [9], it is imperative to use the appropriate tools for comprehensive assessments and improvements. We designed FRAUD-RLA to fill that role eventually.

Future work may span several challenging directions. First, different reinforcement learning algorithms could be tried, such as contextual bandit algorithms [83, 171], that are dedicated to stateful single-step problems. Moreover, FRAUD-RLA could be extended to tackle other fraud detection aspects that we did not consider in this work, such as the presence of categorical variables and the possibility of delayed feedback caused by the presence of humans in the loop. More generally, assessing FRAUD-RLA in even more realistic environments is a necessary step to move beyond the lab-only evaluation phase [7] and transform it into a proper system to evaluate the robustness of the existing engines in production. Another promising direction would be to utilize FRAUD-RLA to develop robust defenses that can mitigate the effectiveness of attacks based on reinforcement learning in credit card fraud detection. Following this direction, we are currently researching the effectiveness of training a classifier to prioritize learning from features that are uncontrollable or unknown to the attacker to achieve *robust by design* credit card fraud detection.

7.6 Ethics Considerations and open science

Machine learning systems are employed in various applications, and research should always make them more secure. Unresponsable disclosure of vulnerabilities can give attackers a head start and severely limit the systems' security. Researching vulnerabilities, however, remains crucial to mitigate the risk posed by zero-day threats. Adversarial machine learning literature, for instance, is built on the assumption that published attacks improve our understanding of machine learning models' vulnerabilities.

In this contribution, we did not aim to exploit vulnerabilities of any specific system. Instead, we used open research as a baseline to define how fraud detection engines work. Our threat model and the attack we designed are not intended to directly apply to any system in production. As stated in Section 7.5, this would require updating the attack to address issues such as categorical variables, transaction frequency caps, and, in general, the multiple challenges that real-world systems present, which are not currently described in the literature. However, reinforcement learning does threaten fraud detection, as it has been shown to do in malware detection [78] and as we showed in this work. In reality, the lack of research on the topic is arguably one of the main vulnerabilities, as it increases the chance attackers may find successful strategies that employ reinforcement learning before the community has a solid understanding of the threat.

Therefore, the primary goal of this chapter is to provide a framework for studying these attacks and their potential to challenge the robustness of systems. First, this could be utilized by fraud detection practitioners to evaluate their systems, possibly extending and modifying FRAUD-RLA to use it for their own testing purposes. Ultimately, however, our work is intended to help us understand how to defend against RL-based attacks. Hopefully, FRAUD-RLA will help us and other researchers towards achieving this goal in the shortest possible time. Finally, all experiments were conducted on openly available data without targeting any in-production system. Finally, the threat model and the theoretical analysis were based on published works, and this chapter is designed to be fully open and reproducible.

Part III

Part III

Chapter 8

Conclusions

The increasingly ubiquitous role of data-driven applications has highlighted the need for trustworthy machine learning. Security, privacy, explainability, and fairness are not just desirable features; they are essential requirements. This thesis contributes to the broader discussion on trustworthiness by focusing specifically on the challenge of security within the context of fraud detection. Throughout this journey, we have worked to provide a systematic analysis of the threats posed by fraud detection systems, evaluating attacks and discussing how to mitigate their effectiveness. While this work has helped identify key vulnerabilities and explored new research directions through the development of algorithms like ADV-O and FRAUD-RLA, the field remains open. Fraudsters might develop new algorithms that employ entirely different approaches, thereby bypassing the defensive mechanisms that have limited the effectiveness of FRAUD-RLA. Similarly, while we have highlighted mechanisms that hinder fraudsters' capabilities, new defensive strategies could be developed to maximize their effectiveness.. Ultimately, this research should not be seen as a definitive endpoint. Rather, it is a step in an ongoing journey toward securing machine learning applications and, in particular, credit card fraud detection.

The rest of the chapter is organized as follows. In Section 8.1 we build a summary of the main contributions highlighted in the previous chapters, showing how they intertwine and the overall improvement they bring in fraud detection security. In Section 8.2 we reflect on the main lessons learnt throughout this journey. This chapter is a personal one in this thesis; it answers the question *"What do I wish I had known before starting this research?"*. Section 8.4 presents how this research has shaped other contributions I made to the field of security during my Ph.D. in areas such as phishing detection and federated learning. We also discuss here the main future directions we see.

8.1 Contributions

Credit card fraud detection offers a significantly different robustness landscape compared to more studied domains, such as image recognition. Limited card budget, time-dependent features, concept drift, and the lack of human supervision present the attacker with novel challenges and opportunities; the lack of understanding of how they affect adversarial attacks in the domain leaves room for skillful attackers to create new attacks, exploiting the gap in the research and the difficulties in assessing the severity of the threat.

This research asked two main questions: *Is credit card fraud detection robust against at-*

tackers? What are the main challenges to ensuring its robustness? In trying to answer this question, it quickly became clear how broad the scope of the research actually was: researching the security of an application requires a deep understanding of its functioning, accurate and comprehensive threat modeling, and deep penetration testing, which, in turn, requires combining existing and novel approaches to assess machine learning security. We noticed, however, that the question could be decomposed into more specific sub-questions, each addressing a different aspect of the broader challenge.

First, we needed to study the state of the art and clearly define the scope of the research. On the one hand, this proved extremely easy; a very limited number of works tackling this issue exist in the literature, and studying and understanding them required little time. Properly defining how attacks against fraud detection systems differ from other applications of adversarial machine learning, and identifying the main aspect we would focus on, took us more time. The main feature we found is the limited number of cards fraudsters have. Each card is distinct from the previous one and has a unique story, forcing attackers to merge the phases where the attacker learns from the model and the attack execution phase.

Having defined the foundations of this research, the first open question was whether existing methods from the adversarial machine learning literature could be reused in the fraud detection domain. A natural starting point was the work of Carminati et al. [33, 34], which was specifically developed for fraud detection scenarios and thus aligned well with our objectives. As discussed in Chapter 7, problem-space attacks also presented a viable approach. However, the vast majority of adversarial attacks have been developed in the context of image recognition. Therefore, determining whether such attacks could be effectively adapted for this new domain represented a critical turning point in shaping the direction of our research. In our first study, published in [102] and discussed in Chapter 5, we analyzed several prominent image recognition attacks to assess whether they could be modified to address the domain of fraud detection. Specifically, we focused on the constraints that most attacks pose on the magnitude of the attack, measured as the distance between the original and adversarial observations in the feature space. In fraud detection, where this is not a constraint, it becomes a limitation, making well-known attacks ill-suited for use.

The second research question focused specifically on the challenges and characteristics of detecting credit card fraud. This domain has been studied for decades, with numerous papers addressing issues such as extreme class imbalance, the presence of concept drift, and recurring fraudulent patterns, as well as constraints including delayed human feedback and the need to limit false positives due to the high cost of manual investigations. These studies have significantly advanced the field, yet they have largely approached fraud as a static classification problem, with limited consideration for the dynamic and strategic nature of fraudster behavior. In particular, no formal or quantitative model of the fraudster had ever been proposed. This left open critical questions; Do fraudsters know what happened to the cards they stole? How do their decisions evolve in response to system feedback or card status? Addressing these questions could serve two key purposes: it would enable the design of a more realistic and threat-aware fraud detection framework, and it could also lead to the development of a novel oversampling algorithm grounded in the behavior of attackers.

The result, detailed in Chapter 6 and published in [101], is ADV-O, a technique that captures and simulates fraudster decision-making patterns to enhance both robustness and detection accuracy. ADV-O attempts to model two distinct processes: the one leading to the first generation of fraud, given the previous genuine transactions performed with the card, and the generation of a fraudulent transaction as a function of these previous ones. Three main results were achieved in this paper. First, we noticed it was impossible to predict the features of the first fraud generated with a card given the previous genuine transactions, suggesting that fraudsters do not change their attack patterns due to the previous transactions, either because they do not want it or because, as we think, they cannot access this information. We were instead able to model the fraud generation process as a time series, where frauds with the same cards are strongly correlated. We used this to create a synthetic generation algorithm, which enriches existing datasets with new artificial frauds created using this model and achieves performance in line with other State-of-the-art oversampling algorithms.

The last step of this thesis was a realistic assessment of the threats posed to credit card fraud detection. Previous works [32, 33] have focused on a specific threat model, where fraudsters can observe a training set and the previous transactions performed with each card, and can therefore build a surrogate model on the aggregation and optimize attacks to bypass it. Our experience, however, suggests that the majority of fraudsters cannot observe either of the two. They might, however, draw inspiration from problem-space attacks in other applications [158, 132] and utilize them to generate new attacks. This led to the development of FRAUD-RLA (Chapter 7), an RL-based attack optimizing the exploration-exploitation tradeoff through PPO. The algorithm is capable of generating fraudulent transactions without any prior knowledge. Fortunately, we have also observed that incorporating unsupervised techniques into the fraud detection process can partially mitigate its effectiveness; however, further research on defenses is needed.

8.2 Lessons learned

This thesis is the result of a four-year exploration at the intersection of adversarial machine learning and credit card fraud detection. Throughout this work, we have outlined the contributions we made to the scientific literature, both in terms of methodological innovation and the development of new models and algorithms. This Section collects instead the main lessons I learnt during this process. These reflections are intended for those who wish to examine the robustness of fraud detection systems and, more broadly, for those interested in advancing the study of AML in areas and applications that have so far remained underexplored.

Let us start with the study of adversarial attacks. Specifically, as we attempted to apply the ideas from AML to credit card fraud detection, it became increasingly clear that a one-size-fits-all solution is unlikely to be feasible. First, no single attack can serve as the benchmark for evaluating robustness, as it all depends on the specific threat being evaluated. If we assume that most attackers won't have access to a bank dataset, then FRAUD-RLA is currently the most realistic threat, and we can consider RL-based attacks as the primary challenge to defend against. If we instead assume potent attackers, then Mimicry and Surrogate Model attacks become a significant danger. Attackers have a quiver full of different arrows, and they can choose which one to use to adapt to our model *adversarially*. We should always be aware of all of them. While this thesis has focused less on the defense part of the equation, we saw in Chapter 5 how simple changes like the implementation of a particular feature engineering system can stop various attacks from happening. Similarly, the experiments on FRAUD-RLA (Chapter 7) showed that different algorithms have significantly different robustness, and that aggregated features reduce the effectiveness of all attacks, even when they cannot prevent them completely. In this sense, this thesis showed that defending an application is a task that extends beyond the design of adversarial defenses, with the variety of solutions being a crucial component of a solid defense.

Let us now discuss which attacks should be chosen when assessing an FDS. If the problem is *easy* (as was the case of our simplified experiment in Chapter 5), even simple heuristics like the Random Sampler Attack suffice. The more we move towards traditional computer vision problems, where the imperceptibility of attacks is important, the more traditional attacks are effective and should be considered. When we consider feature sparsity and other domain-specific challenges, ad-hoc solutions should instead be preferred. Mimicry more easily bypassed the simpler version of the classifier than Fraud-RLA. It is also clearly not competitive in the image domain, where gradient ascent and approaches like the Boundary attack are superior. However, when the target is a system, RL has the clear advantage of directly optimizing for the task; it is more straightforward to learn the best policy to collect the maximum by passing from a multi-layered fraud detection system than finding an attack for each of the ML components of the system, combining them and adding a tool like operational research to optimze the amount. This makes it a vital security tool.

Finally, what percentage of our findings is specifically related to fraud detection? Many of the challenges and strategies observed in fraud detection are shared with other security applications, such as malware detection. For example, attacks in fraud detection need to:

- operate in a complex data space, where actions have numerous side effects and subtle consequences (see the effects on the aggregate features), making the design of effective attacks and defenses a complex problem.
- happen online, adapting to an ever-changing environment. The optimal timing of an attack becomes part of the adversary's strategy, turning adversarial machine learning into a streaming or online learning problem.
- not necessarily minimize perturbation or distance, as attackers can often freely choose certain input values within a plausible context, for instance, maximizing the stolen amount without violating basic consistency rules.

In this sense, it is not surprising that reinforcement learning, which has proven successful in other similar applications, has also proven to be a winning strategy in fraud detection.

To our knowledge, other features are instead specific to fraud detection. There are, however, some characteristics that we think are specific to fraud detection only and should not be considered when working in other applications. The two main examples are both linked to card payment dynamics: the complex feature engineering and the challenge of reproducing successful attacks. In fraud detection, each card has different characteristics and a different transaction history. As a result, attackers must often learn patterns on a per-card basis, while exploiting as much knowledge as possible about the target fraud detection engine. Furthermore, each attempt may lead to detection and the loss of access to a valuable asset. In contrast, in domains like software security, a successful attack can typically be replicated across thousands or millions of systems without the same limitations in frequency or quantity. This creates a much stronger exploration-exploitation tradeoff than in other adversarial domains, and shall be taken into consideration when modeling attackers in fraud detection.

8.3 Defending against adversarial attacks

Throughout this thesis, we have discussed the security of fraud detection systems by taking an attacker's point of view. By studying fraudsters and highlighting the challenges and opportunities they face, we have created a more comprehensive landscape of adversarial security in credit card fraud detection. Let us now conclude this thesis with a brief discussion on what this means to us when developing the defenses.

First, in Chapter 5, we have highlighted how distance-based measurements on performance attacks are not viable in fraud detection, where fraudsters can freely select the value of certain features. This gives us a valuable hint towards the development of defenses; defenses that are based on small quantities of noise or features robust against small changes are not a proper solution.

An interesting defense approach emerges from the results of Chapter 6: fraudsters may not know or be able to control certain features. While we did not investigate it in this thesis, a plausible conclusion is that if we are able to design classifiers that are more heavily influenced by features that are unknown or uncontrollable to the attacker without dramatic accuracy drops, this could lead to the development of a dramatically more robust FDS.

Finally, a promising direction comes from the experimental results of Chapter 7. There, we have shown how 1) the presence of multiple non-deterministic layers of detection is an obstacle to the effectiveness of RL-based classifiers, 2) the introduction of an unsupervised classifier that works on the aggregated feature space completely changes the attacks' effectiveness landscape, and generally affects all of the tested attacks. Investigating the effectiveness of creating complex loss landscapes for attackers is a promising direction, especially considering that, contrary to domains like image recognition, attackers have a severely limited number of queries, facing all the challenges discussed in Chapter 5.

8.4 Extensions and future works

The bulk of our research, as described throughout this thesis, has been heavily focused on evasion attacks against fraud detection systems. In broader terms, however, we were working on one application of machine learning security, and the lessons I learned in doing so helped me when tackling different challenges in the same broad domain.

The first parallel contribution originated from a collaboration with Politecnico di Milano. Working with a research group that was researching the use of unsupervised federated learning, we designed one of the first poisoning attacks against federated contrastive learning algorithms. The research was brought on by a master's student during his thesis, called "BadAvg: aggregation-aware backdoor attack against Federated Contrastive Learning" and co-supervised by me. We plan to finalize an article next month. The full description of the algorithm is outside of the scope of this thesis, but some of the intuitions behind it are inherently connected with this work.

First, most poisoning attacks in the literature assume that the training process of the model is performed on a secure platform, and attackers can only insert malicious inputs to cause a misclassification at test time. This makes sense in most contexts, but in federated learning, clients are notoriously not trustworthy, meaning we could assume full control of the training process. Dropping constraints that are unnecessary for the specific application is akin to what we did in Chapter 5 when we dropped the imperceptibility constraint of the attacks.

The second element is the combination of different approaches to design an effective attack. Previous literature has developed effective attacks against unsupervised learning algorithms, such as encoders [10], and has explored backdoor attacks against supervised federated learning algorithms that can resist the aggregation process. To our knowledge, no work has ever combined the two solutions, as we did in designing BadAvg, which builds on and extends the two approaches.

The second occasion on which I could apply the experience matured through this thesis was the "AI Cybersecurity DeepHack: Advance European Capabilities." The goal of the challenge was to propose an idea for an anti-phishing system usable by both civilian and military actors. The solution we propose, called "MORPH (Machine-learning for Observing Robust Phishing Hazards)," leveraged a federated learning setting to train a classifier and a simple adversarial training process to ensure its robustness. The idea was the following: starting from a detected phishing mail, we try to substitute one word at a time with a relevant word which is close enough in the embedding space (we used Word2Vec for testing it [110]), maintains a similar contextual embedding (for instance, using Bert [54]) and has a lower confidence score when classifying the mail as phishing. The process stops when the mail is not classified as phishing. The idea behind the attack is that we can use the contextual embedding to capture the meaning of a sentence, meaning that if the attack is close enough to the original mail, it will maintain a similar intent, hence preserving the message semantics. The resulting White-Box attacks are, in effect, an adaptation of the problem space attacks described in Chapter 4 to the NLP domain, and can serve as a starting point for developing adversarial learning loops. The proposed solution obtained second place in the competition, and we intend to research the topic further, possibly considering a commercial development of the idea.

In this thesis, we have tried to bridge the two fields of credit card fraud detection and adversarial machine learning, an intersection that has been largely ignored by the scientific literature so far. As one of the first works in the field, it has highlighted multiple research directions; however, further development is still needed to enhance both the theoretical and practical understanding of adversarial machine learning in the context of fraud detection.

First, having verified the effectiveness of RL-based adversarial attacks through the development of FRAUD-RLA, the next step is to design and implement defenses that can mitigate such threats. This may involve system-based protections, as well as model-based strategies, such as enhancing the weight of features that are uncontrollable or unknown to the attacker. In parallel, it is important to expand FRAUD-RLA to operate under delayed feedback scenarios, where human intervention and other factors create a delay between the execution of a fraudulent transaction and the block of the card used to generate it. This extension would be even more representative of real-world fraud detection systems and would not only help assess the threat that RL-based attacks pose under more realistic constraints but could also inform the design of defense mechanisms based on such mechanisms.

A second path would be to confirm the results obtained in ADV-O using different real-world industrial datasets. Specifically, it is necessary to verify that the behavioral patterns observed in our analysis, such as attackers not mimicking cardholder behavior, hold and are not specific to the dataset we used. In general, more collaborations with companies focused on adversarial aspects of fraud detection and cybersecurity should be pursued. These partnerships can provide access to richer datasets, insights into emerging threats, and opportunities to test proposed defenses. Close industry collaboration will also help ensure that academic work addresses the most pressing and realistic challenges.

Finally, it would be valuable to extend the findings of this thesis to other applications. We have already observed analogies while working on contrastive learning and phishing detection, where adversarial dynamics also play a significant role. More generally, We hope the analysis we did on fraud detection may inspire new approaches and perspectives for those working on machine learning security across different domains. In a world where cybersecurity is more crucial than ever, we cannot afford to be unprepared.

Bibliography

[1] Aderemi O Adewumi and Andronicus A Akinyelu. “A survey of machine-learning and nature-inspired based credit card fraud detection techniques”. In: *International Journal of System Assurance Engineering and Management* 8 (2017), pp. 937–953.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. “Optuna: A Next-generation Hyperparameter Optimization Framework”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019.

[3] Maram Alamri and Mourad Ykhlef. “Survey of Credit Card Anomaly and Fraud Detection Using Sampling Techniques”. In: *Electronics* 11.23 (2022), p. 4003.

[4] Fawaz Khaled Alarfaj, Iqra Malik, Hikmat Ullah Khan, Naif Almusallam, Muhammad Ramzan, and Muzamil Ahmed. “Credit card fraud detection using state-of-the-art machine learning and deep learning algorithms”. In: *Ieee Access* 10 (2022), pp. 39700–39715.

[5] Cesare Alippi, Giacomo Boracchi, and Manuel Roveri. “Just-in-time classifiers for recurrent concepts”. In: *IEEE transactions on neural networks and learning systems* 24.4 (2013), pp. 620–634.

[6] Jeffrey S. Allen. “CardSim: A Bayesian Simulator for Payment Card Fraud Detection Research”. In: *Finance and Economics Discussion Series* (2025). URL: `https://api.semanticscholar.org/CorpusID:276803699`.

[7] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. “Dos and don’ts of machine learning in computer security”. In: *31st USENIX Security Symposium (USENIX Security 22)*. 2022, pp. 3971–3988.

[8] Samuel A. Assefa, Danial Dervovic, Mahmoud Mahfouz, Robert E. Tillman, Prashant Reddy, and Manuela Veloso. “Generating Synthetic Data in Finance: Opportunities, Challenges and Pitfalls”. In: *Proceedings of the First ACM International Conference on AI in Finance*. 2020.

[9] Shahar Avin, Haydn Belfield, Miles Brundage, Gretchen Krueger, Jasmine Wang, Adrian Weller, Markus Anderljung, Igor Krawczuk, David Krueger, Jonathan Lebensold, et al. “Filling gaps in trustworthy development of AI”. In: *Science* 374.6573 (2021), pp. 1327–1329.

[10] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. “How to backdoor federated learning”. In: *International conference on artificial intelligence and statistics*. PMLR. 2020, pp. 2938–2948.

[11] Sikha Bagui and Kunqi Li. “Resampling imbalanced data for network intrusion detection datasets”. In: *Journal of Big Data* 8 (Jan. 2021).

[12] ’European Central Bank’. *European Central Bank. 6th report on card fraud.* 2020. URL: https://www.ecb.europa.eu/pub/cardfraud/html/ecb.cardfraudreport202008~521edb602b.en.html#toc2..

[13] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. “Can machine learning be secure?” In: *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. 2006, pp. 16–25.

[14] Andrew G Barto, Richard S Sutton, and Charles W Anderson. “Neuronlike adaptive elements that can solve difficult learning control problems”. In: *IEEE transactions on systems, man, and cybernetics* 5 (1983), pp. 834–846.

[15] Richard Bellman. *Dynamic programming*. Princeton, NJ: Princeton Univ. Pr, 1957. 339 pp. ISBN: 978-0-691-07951-6.

[16] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. “Evasion attacks against machine learning at test time”. In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*. Springer. 2013, pp. 387–402.

[17] Battista Biggio, Blaine Nelson, and Pavel Laskov. “Poisoning attacks against support vector machines”. In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. ICML’12. Edinburgh, Scotland: Omnipress, 2012, pp. 1467–1474. ISBN: 9781450312851.

[18] Battista Biggio and Fabio Roli. “Wild patterns: Ten years after the rise of adversarial machine learning”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 2154–2156.

[19] Leyla Bilge and Tudor Dumitraş. “Before we knew it: an empirical study of zero-day attacks in the real world”. In: *Proceedings of the 2012 ACM conference on Computer and communications security*. 2012, pp. 833–844.

[20] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[21] Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne. “Machine learning strategies for time series forecasting”. In: *European business intelligence summer school*. Springer. 2012, pp. 62–77.

[22] Giacomo Boracchi, Diego Carrera, Cristiano Cervellera, and Danilo Maccio. “QuantTree: Histograms for change detection in multivariate data streams”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 639–648.

[23] Kendrick Boyd, Kevin H Eng, and C David Page. "Area under the precision-recall curve: point estimates and confidence intervals". In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*. Springer. 2013, pp. 451–466.

[24] Wieland Brendel, Jonas Rauber, and Matthias Bethge. "Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models". In: *6th International Conference on Learning Representations (ICLR) 2018*. Vancouver, BC, Canada, Apr. 2018.

[25] J. Burez and D. Van den Poel. "Handling class imbalance in customer churn prediction". In: *Expert Systems with Applications* 36 (2009), from 4626 to 4636.

[26] Fabrizio Carcillo, Yann-Aël Le Borgne, Olivier Caelen, Yacine Kessaci, Frédéric Oblé, and Gianluca Bontempi. "Combining unsupervised and supervised learning in credit card fraud detection". In: *Inf. Sci.* 557 (2021), pp. 317–331. DOI: `10.1016/J.INS.2019.05.042`. URL: `https://doi.org/10.1016/j.ins.2019.05.042`.

[27] Fabrizio Carcillo, Andrea Dal Pozzolo, Yann-Aël Le Borgne, Olivier Caelen, Yannis Mazzer, and Gianluca Bontempi. "SCARFF : a Scalable Framework for Streaming Credit Card Fraud Detection with Spark". In: *Information Fusion* 41 (Sept. 2017). DOI: `10.1016/j.inffus.2017.09.005`.

[28] Fabrizio Carcillo, Andrea Dal Pozzolo, Yann-Aël Le Borgne, Olivier Caelen, Yannis Mazzer, and Gianluca Bontempi. "Scarff: a scalable framework for streaming credit card fraud detection with spark". In: *Information fusion* 41 (2018), pp. 182–194.

[29] Nicholas Carlini and David Wagner. "Towards evaluating the robustness of neural networks". In: *2017 ieee symposium on security and privacy (sp)*. Ieee. 2017, pp. 39–57.

[30] Michele Carminati, Roberto Caron, Federico Maggi, Ilenia Epifani, and Stefano Zanero. "BankSealer: A decision support system for online banking fraud analysis and investigation". In: *computers & security* 53 (2015), pp. 175–186.

[31] Michele Carminati, Roberto Caron, Federico Maggi, Stefano Zanero, and Ilenia Epifani. "BankSealer: A Decision Support System for Online Banking Fraud Analysis and Investigation". In: *Computers & Security* 53 (2015).

[32] Michele Carminati, Mario Polino, Andrea Continella, Andrea Lanzi, Federico Maggi, and Stefano Zanero. "Security evaluation of a banking fraud analysis system". In: *ACM Transactions on Privacy and Security (TOPS)* 21.3 (2018), pp. 1–31.

[33] Michele Carminati, Luca Santini, Mario Polino, and Stefano Zanero. "Evasion attacks against banking fraud detection systems". In: *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. 2020, pp. 285–300.

[34] Francesco Cartella, Orlando Anunciacao, Yuki Funabiki, Daisuke Yamaguchi, Toru Akishita, and Olivier Elshocht. “Adversarial Attacks for Tabular Data: Application to Fraud Detection and Imbalanced Data”. In: *Proceedings of the Workshop on Artificial Intelligence Safety 2021 (SafeAI 2021), co-located with the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2021).* 2021.

[35] Varun Chandola, Arindam Banerjee, and Vipin Kumar. “Anomaly detection: A survey”. In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.

[36] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. “SMOTE: synthetic minority over-sampling technique”. In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[37] Chao Chen, Andy Liaw, Leo Breiman, et al. “Using random forest to learn imbalanced data”. In: *University of California, Berkeley* 110.1-12 (2004), p. 24.

[38] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. “Hopskipjumpattack: A query-efficient decision-based attack”. In: *2020 ieee symposium on security and privacy (sp).* IEEE. 2020, pp. 1277–1294.

[39] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. “Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models”. In: *Proceedings of the 10th ACM workshop on artificial intelligence and security.* 2017, pp. 15–26.

[40] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.* Sept. 3, 2014. DOI: 10.48550/arXiv.1406.1078. arXiv: 1406.1078[cs]. URL: http://arxiv.org/abs/1406.1078 (visited on 05/30/2025).

[41] Antonio Emanuele Cinà, Kathrin Grosse, Ambra Demontis, Sebastiano Vascon, Werner Zellinger, Bernhard A Moser, Alina Oprea, Battista Biggio, Marcello Pelillo, and Fabio Roli. “Wild patterns reloaded: A survey of machine learning security against training data poisoning”. In: *ACM Computing Surveys* 55.13s (2023), pp. 1–39.

[42] Michael Crawford, Taghi M Khoshgoftaar, Joseph D Prusa, Aaron N Richter, and Hamzah Al Najada. “Survey of review spam detection using machine learning techniques”. In: *Journal of Big Data* 2 (2015), pp. 1–24.

[43] Andrea Dal Pozzolo. *Adaptive machine learning for credit card fraud detection.* 2015.

[44] Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. “Credit card fraud detection and concept-drift adaptation with delayed supervised information”. In: *2015 international joint conference on Neural networks (IJCNN).* IEEE. 2015, pp. 1–8.

[45] Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. “Credit card fraud detection: a realistic modeling and a novel learning strategy”. In: *IEEE transactions on neural networks and learning systems* 29.8 (2017), pp. 3784–3797.

[46] Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. "Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.8 (2018), pp. 3784–3797.

[47] Andrea Dal Pozzolo, Olivier Caelen, and Gianluca Bontempi. "When is Undersampling Effective in Unbalanced Classification Tasks?" In: *ECML PKDD 2015: Machine Learning and Knowledge Discovery in Databases* (2015). Ed. by Annalisa Appice, Pedro Pereira Rodrigues, Vítor Santos Costa, Carlos Soares, João Gama, and Alípio Jorge, pp. 200–215.

[48] Andrea Dal Pozzolo, Olivier Caelen, Reid A Johnson, and Gianluca Bontempi. "Calibrating probability with undersampling for unbalanced classification". In: *2015 IEEE Symposium Series on Computational Intelligence*. IEEE. 2015, pp. 159–166.

[49] Andrea Dal Pozzolo, Olivier Caelen, Yann-Aël Le Borgne, Serge Waterschoot, and Gianluca Bontempi. "Learned lessons in credit card fraud detection from a practitioner perspective". In: *Expert Systems with Applications* 41 (Aug. 2014), pp. 4915–4928.

[50] Islam Debicha, Benjamin Cochez, Tayeb Kenaza, Thibault Debatty, Jean-Michel Dricot, and Wim Mees. "Adv-Bot: Realistic adversarial botnet attacks against network intrusion detection systems". In: *Computers & Security* 129 (2023), p. 103176.

[51] Islam Debicha, Benjamin Cochez, Tayeb Kenaza, Thibault Debatty, Jean-Michel Dricot, and Wim Mees. "Review on the feasibility of adversarial evasion attacks and defenses for network intrusion detection systems". In: *arXiv preprint arXiv:2303.07003* (2023).

[52] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. "Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks". In: *28th USENIX security symposium (USENIX security 19)*. 2019, pp. 321–338.

[53] Janez Demšar. "Statistical comparisons of classifiers over multiple data sets". In: *The Journal of Machine learning research* 7 (2006), pp. 1–30.

[54] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *North American Chapter of the Association for Computational Linguistics*. 2019. URL: https://api.semanticscholar.org/CorpusID:52967399.

[55] Yifan Ding, Liqiang Wang, Huan Zhang, Jinfeng Yi, Deliang Fan, and Boqing Gong. " Defending Against Adversarial Attacks Using Random Forest ". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2019, pp. 105–114. DOI: 10.1109/CVPRW.2019.00019. URL: https://doi.ieeecomputersociety.org/10.1109/CVPRW.2019.00019.

[56] Yifan Ding, Liqiang Wang, Huan Zhang, Jinfeng Yi, Deliang Fan, and Boqing Gong. “ Defending Against Adversarial Attacks Using Random Forest ”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2019, pp. 105–114. DOI: 10.1109/CVPRW.2019.00019. URL: https://doi.ieeecomputersociety.org/10.1109/CVPRW.2019.00019.

[57] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. “Benchmarking deep reinforcement learning for continuous control”. In: *International conference on machine learning*. PMLR. 2016, pp. 1329–1338.

[58] Khaled El Emam, Lucy Mosquera, and Richard Hoptroff. *Practical synthetic data generation: balancing privacy and the broad availability of data*. O’Reilly Media, 2020.

[59] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. “A rotation and a translation suffice: Fooling cnns with simple transformations”. In: (2017).

[60] Ebenezer Esenogho, Ibomoiye Domor Mienye, Theo G Swart, Kehinde Aruleba, and George Obaido. “A neural network ensemble with feature engineering for improved credit card fraud detection”. In: *IEEE Access* 10 (2022), pp. 16400–16407.

[61] Tom Fawcett. “An introduction to ROC analysis”. In: *Pattern recognition letters* 27.8 (2006), pp. 861–874.

[62] Tom Fawcett. “An introduction to ROC analysis”. In: *Pattern Recognition Letters* 27.8 (2006). ROC Analysis in Pattern Recognition, pp. 861–874. ISSN: 0167-8655.

[63] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, Bartosz Krawczyk, and Francisco Herrera. *Learning from imbalanced data sets*. Vol. 10. 2018. Springer, 2018.

[64] Ugo Fiore, Alfredo De Santis, Francesca Perla, Paolo Zanetti, and Francesco Palmieri. “Using generative adversarial networks for improving classification effectiveness in credit card fraud detection”. In: *Information Sciences* 479 (2019), pp. 448–455.

[65] PR Freeman. “The secretary problem and its extensions: A review”. In: *International Statistical Review/Revue Internationale de Statistique* (1983), pp. 189–206.

[66] Alex Gittens, Bülent Yener, and Moti Yung. “An adversarial perspective on accuracy, robustness, fairness, and privacy: Multilateral-tradeoffs in trustworthy ml”. In: *IEEE Access* 10 (2022), pp. 120850–120865.

[67] Markus Goldstein and Andreas Dengel. “Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm”. In: *KI-2012: poster and demo track* 1 (2012), pp. 59–63.

[68] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Y. Bengio. “Generative Adversarial Networks”. In: *Advances in Neural Information Processing Systems* 3 (June 2014).

[69] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and harnessing adversarial examples”. In: *3rd International Conference on Learning Representations (ICLR) 2015, Conference Track Proceedings.* Ed. by Yoshua Bengio and Yann LeCun. San Diego, CA, USA, May 2015.

[70] Derek Greene, Pádraig Cunningham, and Rudolf Mayer. “Unsupervised learning and clustering”. In: *Machine learning techniques for multimedia: Case studies on organization and retrieval.* Springer, 2008, pp. 51–90.

[71] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. “Badnets: Evaluating backdooring attacks on deep neural networks”. In: *IEEE Access* 7 (2019), pp. 47230–47244.

[72] Junfeng Guo and Cong Liu. “Practical poisoning attacks on neural networks”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16.* Springer. 2020, pp. 142–158.

[73] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. “Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning”. In: *Advances in Intelligent Computing.* 2005.

[74] Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. “ADASYN: Adaptive synthetic sampling approach for imbalanced learning”. In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence).* 2008.

[75] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[76] Addison Howard, Bernadette Bouchon-Meunier, IEEE CIS, inversion, John Lei, Lynn@Vesta, Marcus2010, and Prof. Hussein Abbass. *IEEE-CIS Fraud Detection.* https://kaggle.com/competitions/ieee-fraud-detection. Kaggle. 2019.

[77] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J Doug Tygar. “Adversarial machine learning”. In: *Proceedings of the 4th ACM workshop on Security and artificial intelligence.* 2011, pp. 43–58.

[78] S. Anderson Hyrum, Kharkar Anant, Filar Bobby, Evans David, and Roth Phil. “Learning to Evade Static PE Machine Learning Malware Models via Reinforcement Learning”. In: *CoRR* abs/1801.08917 (2018). arXiv: 1801.08917. URL: http://arxiv.org/abs/1801.08917.

[79] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. “Adversarial examples are not bugs, they are features”. In: *Advances in neural information processing systems* 32 (2019).

[80] Tony Jebara. *Machine learning: discriminative and generative.* Vol. 755. Berlin/Heidelberg, Germany: Springer Science & Business Media, 2012, pp. 3784–3797.

[81] Bern Jonathan, Panca Hadi Putra, and Yova Ruldeviyani. "Observation imbalanced data text to predict users selling products on female daily with smote, tomek, and smote-tomek". In: *2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*. IEEE. 2020, pp. 81–85.

[82] Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar. *Adversarial machine learning*. Cambridge University Press, 2018.

[83] Parnian Kassraie and Andreas Krause. "Neural Contextual Bandits without Regret". In: *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*. Ed. by Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera. Vol. 151. Proceedings of Machine Learning Research. PMLR, Mar. 28–30, 2022, pp. 240–278. URL: https://proceedings.mlr.press/v151/kassraie22a.html.

[84] Diederik P Kingma, Max Welling, et al. "An introduction to variational autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392.

[85] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). URL: https://api.semanticscholar.org/CorpusID:6628106.

[86] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[87] Diederik P. Kingma and Max Welling. "An Introduction to Variational Autoencoders". In: *Foundations and Trends in Machine Learning* 12.4 (2019), pp. 307–392.

[88] B. Krawczyk. "Learning from imbalanced data: open challenges and future directions". In: *Progress in Artificial Intelligence* 5 (2016), pp. 221–232.

[89] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. "Adversarial examples in the physical world". In: *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.

[90] Felix Last, Georgios Douzas, and Fernando Bação. "Oversampling for Imbalanced Learning Based on K-Means and SMOTE". In: *CoRR* abs/1711.00837 (2017).

[91] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[92] Yann-Aël Le Borgne, Wissam Siblini, Bertrand Lebichot, and Gianluca Bontempi. *Reproducible Machine Learning for Credit Card Fraud Detection - Practical Handbook*. Université Libre de Bruxelles, 2022. URL: https://github.com/Fraud-Detection-Handbook/fraud-detection-handbook.

[93] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. “Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning”. In: *Journal of Machine Learning Research* 18.17 (2017), pp. 1–5. URL: http://jmlr.org/papers/v18/16-365.

[94] Dong C Liu and Jorge Nocedal. “On the limited memory BFGS method for large scale optimization”. In: *Mathematical programming* 45.1 (1989), pp. 503–528.

[95] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. “Isolation forest”. In: *2008 eighth ieee international conference on data mining*. IEEE. 2008, pp. 413–422.

[96] Kaijun Liu, Shengwei Xu, Guoai Xu, Miao Zhang, Dawei Sun, and Haifeng Liu. “A review of android malware detection approaches based on machine learning”. In: *IEEE access* 8 (2020), pp. 124579–124607.

[97] Sijia Liu, Pin-Yu Chen, Bhavya Kailkhura, Gaoyuan Zhang, Alfred O Hero III, and Pramod K Varshney. “A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications”. In: *IEEE Signal Processing Magazine* 37.5 (2020), pp. 43–54.

[98] Tian-Yu Liu. “Easyensemble and feature selection for imbalance data sets”. In: *2009 international joint conference on bioinformatics, systems biology and intelligent computing*. IEEE. 2009, pp. 517–520.

[99] Yvan Lucas, Pierre-Edouard Portier, Léa Laporte, Liyun He-Guelton, Olivier Caelen, Michael Granitzer, and Sylvie Calabretto. “Towards automated feature engineering for credit card fraud detection using multi-perspective HMMs”. In: *Future Generation Computer Systems* 102 (2020), pp. 393–402.

[100] Daniele Lunghi, Yannick Molinghen, Alkis Simitsis, Tom Lenaerts, and Gianluca Bontempi. “FRAUD-RLA: A new reinforcement learning adversarial attack against credit card fraud detection”. In: *arXiv preprint arXiv:2502.02290* (2025).

[101] Daniele Lunghi, Gian Marco Paldino, Olivier Caelen, and Gianluca Bontempi. “An Adversary Model of Fraudsters’ Behavior to Improve Oversampling in Credit Card Fraud Detection”. In: *IEEE access* 11 (2023), pp. 136666–136679.

[102] Daniele Lunghi, Alkis Simitsis, and Gianluca Bontempi. “Assessing adversarial attacks in real-world fraud detection”. In: *2024 IEEE International Conference on Web Services (ICWS)*. IEEE. 2024, pp. 27–34.

[103] Daniele Lunghi, Alkis Simitsis, and Gianluca Bontempi. “Adversarial Learning for Fraud Detection”. In: *Data Engineering for Data Science*. Ed. by G. Dejaegere, A. Abello, K. Torp, and A. Simitsis. Springer, forthcoming.

[104] Daniele Lunghi, Alkis Simitsis, Olivier Caelen, and Gianluca Bontempi. “Adversarial Learning in Real-World Fraud Detection: Challenges and Perspectives”. In: *Proceedings of the Second ACM Data Economy Workshop*. 2023, pp. 27–33.

[105] Andrzej Maćkiewicz and Waldemar Ratajczak. “Principal components analysis (PCA)”. In: *Computers & Geosciences* 19.3 (1993), pp. 303–342. ISSN: 0098-3004. DOI: https://doi.org/10.1016/0098-3004(93)90090-R. URL: https://www.sciencedirect.com/science/article/pii/009830049390090R.

[106] William G Macready and David H Wolpert. “Bandit problems and the exploration/exploitation tradeoff”. In: *IEEE Transactions on evolutionary computation* 2.1 (1998), pp. 2–22.

[107] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. “Towards Deep Learning Models Resistant to Adversarial Attacks”. In: *International Conference on Learning Representations*. 2018.

[108] Sebastián Maldonado, Carla Vairetti, Alberto Fernandez, and Francisco Herrera. “FW-SMOTE: A feature-weighted oversampling approach for imbalanced classification”. In: *Pattern Recognition* 124 (2022), p. 108511.

[109] Inderjeet Mani and I Zhang. “kNN approach to unbalanced data distributions: a case study involving information extraction”. In: *Proceedings of workshop on learning from imbalanced datasets*. Vol. 126. 1. ICML United States. 2003, pp. 1–7.

[110] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. “Efficient Estimation of Word Representations in Vector Space”. In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. 2013. URL: http://arxiv.org/abs/1301.3781.

[111] Satwik Mishra. “Handling imbalanced data: SMOTE vs. random undersampling”. In: *Int. Res. J. Eng. Technol* 4.8 (2017), pp. 317–320.

[112] Tom M Mitchell. *Machine learning*. Vol. 1. 9. McGraw-hill New York, 1997.

[113] Aleksa Mladenovic, Anirudh J. Bose, Hugo Berard, William L. Hamilton, Simon Lacoste-Julien, Pascal Vincent, and Guillem Gidel. “Online adversarial attacks”. In: *International Conference on Learning Representations (ICLR)*. 2022.

[114] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P Lillicrap, David Silver, and Koray Kavukcuoglu. “Asynchronous methods for deep reinforcement learning”. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*. 2016, pp. 1928–1937.

[115] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. “Deepfool: a simple and accurate method to fool deep neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2574–2582.

[116] Giulia Moschini, Régis Houssou, Jérôme Bovay, and Stephan Robert-Nicoud. “Anomaly and fraud detection in credit card transactions using the arima model”. In: *Engineering Proceedings* 5.1 (2021), p. 56.

[117] Nhlakanipho Mqadi, Nalindren Naicker, and Timothy Adeliyi. “A SMOTe based oversampling data-point approach to solving the credit card data imbalance problem in financial fraud detection”. In: *International Journal of Computing and Digital Systems* 10.1 (2021), pp. 277–286.

[118] Tuan Anh Nguyen and Anh Tuan Tran. "WaNet-Imperceptible Warping-based Backdoor Attack". In: *International Conference on Learning Representations.* 2021.

[119] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, et al. "Adversarial Robustness Toolbox v1. 0.0". In: *arXiv preprint arXiv:1807.01069* (2018).

[120] Daniel N Nikovski. "Fast Reinforcement Learning in continuous action spaces". In: *Robotics Institute Carnegie Mellon University, Pittsburgh PA* (1998).

[121] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs.* SpringerBriefs in Intelligent Systems. Cham: Springer International Publishing, 2016. DOI: 10.1007/978-3-319-28929-8. URL: http://link.springer.com/10.1007/978-3-319-28929-8 (visited on 05/25/2023).

[122] Gian Marco Paldino, Bertrand Lebichot, Yann-Aël Le Borgne, Wissam Siblini, Frédéric Oblé, Giacomo Boracchi, and Gianluca Bontempi. "The role of diversity and ensemble learning in credit card fraud detection". In: *Advances in Data Analysis and Classification* (2022), pp. 1–25.

[123] Gian Marco Paldino, Bertrand Lebichot, Yann-Aël Le Borgne, Wissam Siblini, Frédéric Oblé, Giacomo Boracchi, and Gianluca Bontempi. "The role of diversity and ensemble learning in credit card fraud detection". In: *Advances in Data Analysis and Classification* 18.1 (2024), pp. 193–217.

[124] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples". In: *arXiv preprint arXiv:1605.07277* (2016).

[125] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. "Practical black-box attacks against machine learning". In: *Proceedings of the 2017 ACM on Asia conference on computer and communications security.* 2017, pp. 506–519.

[126] Nicolas Papernot, Patrick Mcdaniel, and Ian J. Goodfellow. "Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples". In: *ArXiv* abs/1605.07277 (2016). URL: https://api.semanticscholar.org/CorpusID:17362994.

[127] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. "Distillation as a defense to adversarial perturbations against deep neural networks". In: *2016 IEEE symposium on security and privacy (SP).* IEEE. 2016, pp. 582–597.

[128] Sanglee Park and Jungmin So. "On the effectiveness of adversarial training in defending against adversarial example attacks for image classification". In: *Applied Sciences* 10.22 (2020), p. 8079.

[129] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[130] Neehar Peri, Neal Gupta, W Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P Dickerson. "Deep k-nn defense against clean-label data poisoning attacks". In: *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*. Springer. 2020, pp. 55–70.

[131] George Petrides and Wouter Verbeke. "Cost-sensitive ensemble learning: a unifying framework". In: *Data Mining and Knowledge Discovery* 36.1 (2022), pp. 1–28.

[132] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. "Intriguing properties of adversarial ml attacks in the problem space". In: *2020 IEEE symposium on security and privacy (SP)*. IEEE. 2020, pp. 1332–1349.

[133] Maura Pintor, Luca Demetrio, Angelo Sotgiu, Ambra Demontis, Nicholas Carlini, Battista Biggio, and Fabio Roli. "Indicators of attack failure: Debugging and improving optimization of adversarial examples". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 23063–23076.

[134] Kedar Potdar, Taher S Pardawala, and Chinmay D Pai. "A comparative study of categorical variable encoding techniques for neural network classifiers". In: *International journal of computer applications* 175.4 (2017), pp. 7–9.

[135] Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson, and Gianluca Bontempi. "Calibrating Probability with Undersampling for Unbalanced Classification". In: *2015 IEEE Symposium Series on Computational Intelligence*. 2015, pp. 159–166. DOI: 10.1109/SSCI.2015.33.

[136] Andrew Ross and Finale Doshi-Velez. "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[137] Peter Rousseeuw, Domenico Perrotta, Marco Riani, and Mia Hubert. "Robust monitoring of time series with application to fraud detection". In: *Econometrics and statistics* 9 (2019), pp. 108–121.

[138] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[139] Fatima Salahdine and Naima Kaabouch. "Social engineering attacks: A survey". In: *Future internet* 11.4 (2019), p. 89.

[140] Zahra Salekshahrezaee, Joffrey L Leevy, and Taghi M Khoshgoftaar. "The effect of feature extraction and data sampling on credit card fraud detection". In: *Journal of Big Data* 10.1 (2023), p. 6.

[141] Spyridon Samonas and David Coss. "The CIA strikes back: Redefining confidentiality, integrity and availability in security." In: *Journal of Information System Security* 10.3 (2014).

[142] Karen Scarfone, Wayne Jansen, Miles Tracy, et al. "Guide to general server security". In: *NIST Special Publication* 800.123 (2008).

[143] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms". In: *arXiv* (2017), pp. 1–12. ISSN: 23318422. arXiv: 1707.06347.

[144] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. "Poison frogs! targeted clean-label poisoning attacks on neural networks". In: *Advances in neural information processing systems* 31 (2018).

[145] Yash Sharma and Pin-Yu Chen. "Attacking the madry defense model with l_1-based adversarial examples". In: *arXiv preprint arXiv:1710.10733* (2017).

[146] Walter Andrew Shewhart. "Statistical method from an engineering viewpoint". In: *Journal of the American Statistical Association* 26.175 (1931), pp. 262–269.

[147] Adam Shostack. *Threat modeling: Designing for security.* John Wiley & Sons, 2014.

[148] Octavian Suciu, Radu Marginean, Yigitcan Kaya, Hal Daume III, and Tudor Dumitras. "When does machine learning {FAIL}? generalized transferability for evasion and poisoning attacks". In: *27th USENIX Security Symposium (USENIX Security 18).* 2018, pp. 1299–1316.

[149] Mukund Sundararajan and Amir Najmi. "The many Shapley values for model explanation". In: *International conference on machine learning.* PMLR. 2020, pp. 9269–9278.

[150] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction.* Vol. 1. 1. MIT press Cambridge, 1998.

[151] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction.* Second edition. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018. 526 pp. ISBN: 978-0-262-03924-6.

[152] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. "Intriguing properties of neural networks". In: *International Conference on Learning Representations (ICLR).* 2014.

[153] Nguyen Thai-Nghe, Zeno Gantner, and Lars Schmidt-Thieme. "Cost-sensitive learning methods for imbalanced data". In: *The 2010 International Joint Conference on Neural Networks (IJCNN).* 2010.

[154] Arryon D Tijsma, Madalina M Drugan, and Marco A Wiering. "Comparing exploration strategies for Q-learning in random stochastic mazes". In: *2016 IEEE symposium series on computational intelligence (SSCI).* IEEE. 2016, pp. 1–8.

[155] Peter Torr. "Demystifying the threat modeling process". In: *IEEE Security & Privacy* 3.5 (2005), pp. 66–70.

[156] Florian Tramer and Dan Boneh. "Adversarial training and robustness for multiple perturbations". In: *Advances in neural information processing systems* 32 (2019).

[157] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin. "Intrusion detection by machine learning: A review". In: *expert systems with applications* 36.10 (2009), pp. 11994–12000.

[158] Ilias Tsingenopoulos, Ali Mohammad Shafiei, Lieven Desmet, Davy Preuveneers, and Wouter Joosen. “Adaptive malware control: Decision-based attacks in the problem space of dynamic analysis”. In: *Proceedings of the 1st Workshop on Robust Malware Analysis*. 2022, pp. 3–14.

[159] Véronique Van Vlasselaer, Cristián Bravo, Olivier Caelen, Tina Eliassi-Rad, Leman Akoglu, Monique Snoeck, and Bart Baesens. “APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions”. In: *Decision support systems* 75 (2015), pp. 38–48.

[160] David Wagner and R Dean. “Intrusion detection via static analysis”. In: *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*. IEEE. 2000, pp. 156–168.

[161] Christopher JCH Watkins and Peter Dayan. “Q-learning”. In: *Machine learning* 8 (1992), pp. 279–292.

[162] Gary M Weiss and Haym Hirsh. “Learning to predict extremely rare events”. In: *AAAI workshop on learning from imbalanced data sets*. 2000, pp. 64–68.

[163] Ronald J Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine learning* 8 (1992), pp. 229–256.

[164] Svante Wold, Kim Esbensen, and Paul Geladi. “Principal component analysis”. In: *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), pp. 37–52.

[165] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. “Modeling Tabular data using Conditional GAN”. In: *Advances in Neural Information Processing Systems*. 2019.

[166] Scott Cheng-Hsin Yang, Baxter Eaves, Michael Schmidt, Ken Swanson, and Patrick Shafto. “Structured evaluation of synthetic tabular data”. In: *arXiv preprint arXiv:2403.10424* (2024).

[167] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. “Time-series Generative Adversarial Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf.

[168] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. “The surprising effectiveness of ppo in cooperative multi-agent games”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 24611–24624.

[169] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. “Theoretically principled trade-off between robustness and accuracy”. In: *ICML*. 2019.

[170] Zhaohui Zhang, Lijun Yang, Ligong Chen, Qiuwen Liu, Ying Meng, Pengwei Wang, and Maozhen Li. “A generative adversarial network–based method for generating negative financial samples”. In: *International Journal of Distributed Sensor Networks* 16.2 (2020), p. 1550147720907053.

[171] Dongruo Zhou, Lihong Li, and Quanquan Gu. “Neural Contextual Bandits with UCB-based Exploration”. In: *Proceedings of the 37th International Conference on Machine Learning*. 2020.

[172] Indrė Žliobaitė, Mykola Pechenizkiy, and Joao Gama. “An overview of concept drift applications”. In: *Big data analysis: new algorithms for a new society* (2016), pp. 91–114.