Titre: Title:	A Landscape of Adversarial Threats to Machine Learning-Based Intrusion Detection
Auteur: Author:	Mohamed Amine Merzouk
Date:	2024
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Merzouk, M. A. (2024). A Landscape of Adversarial Threats to Machine Learning-Based Intrusion Detection [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/59435/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

A Landscape of Adversarial Threats to Machine Learning-based Intrusion Detection

MOHAMED AMINE MERZOUK
Département de génie informatique et génie logiciel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie informatique

Septembre 2024

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

A Landscape of Adversarial Threats to Machine Learning-based Intrusion Detection

présentée par **Mohamed Amine MERZOUK**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Alejandro QUINTERO, président
Frédéric CUPPENS, membre et directeur de recherche
Nora BOULAHIA CUPPENS, membre et codirectrice de recherche
Ettore MERLO, membre
David ESPES, membre externe

RÉSUMÉ

L'intelligence artificielle (AI) devient de plus en plus omniprésente dans divers domaines, y compris des secteurs critiques tels que la détection d'intrusions. Les techniques d'apprentissage automatique (ML) sont à l'avant-garde de cette intégration, améliorant la capacité des systèmes de détection d'intrusions (IDS) à détecter et à répondre aux menaces. Ces dernières années ont vu un grand intérêt pour l'adoption de méthodes avancées d'apprentissage profond (DL), offrant une précision et une adaptabilité sans précédent. Cependant, bien que ces méthodes sophistiquées améliorent les performances, elles introduisent également des problèmes de sécurité.

Cette thèse explore la sécurité des récentes techniques de ML et de DL dans le contexte de la détection d'intrusions réseau, en particulier leur vulnérabilité aux attaques adverses, ainsi que les contre-mesures qui renforcent leur robustesse. Nos résultats fournissent aux chercheurs et aux praticiens des lignes directrices pour les évaluations de sécurité et des perspectives sur les stratégies de défense.

L'apprentissage fédéré (FL) permet à plusieurs entités d'entraînement collaborativement un modèle de ML sans partager de données d'entraînement confidentielles, mais des participants malveillants pourraient perturber l'entraînement du modèle. La première contribution aborde la menace des attaques par empoisonnement sur les modèles de détection d'intrusions basés sur le FL. Nous évaluons l'impact de quatre paramètres d'attaque sur l'efficacité, la furtivité, la cohérence et le moment des attaques par porte dérobée. Nos résultats montrent la détermination de chaque paramètre pour le succès de l'attaque, à condition qu'ils soient ajustés.

L'apprentissage par renforcement profond (DRL) est de plus en plus utilisé dans la détection d'intrusions pour son adaptabilité dans des environnements complexes tels que les réseaux informatiques, mais sa dépendance au DL le rend vulnérable aux exemples adverses. La deuxième contribution évalue l'influence des hyperparamètres cruciaux du DRL sur les performances et la robustesse des agents de détection d'intrusions. Incluant des attaques en boîte blanche et en boîte noire à travers la propriété de transférabilité.

Bien que les exemples adverses parviennent à contourner les IDS basés sur le ML, ils ne représentent une menace concrète que s'ils peuvent être implémentés dans des réseaux réels. La troisième contribution étudie la praticité de ces attaques d'évasion adverses. Nous étudions l'impact des attaques de pointe sur les performances du modèle, la structure des données, les caractéristiques perturbées et les attaques réussies en commun. Nous introduisons et

discutons de quatre critères cruciaux pour la validité des exemples adverses.

Les modèles de diffusion ont démontré un grand potentiel dans la génération de données et également dans la purification adversaire. La quatrième contribution propose des modèles de diffusion pour purifier les exemples adverses dans la détection d'intrusions. Notre approche est agnostique par rapport au modèle, agnostique par rapport à l'attaque et ne nécessite pas de re-entraînement du modèle. À travers une analyse comparative, nous identifions les paramètres de diffusion optimaux pour augmenter la robustesse et minimiser les impacts sur la précision.

ABSTRACT

Artificial Intelligence (AI) is becoming increasingly pervasive in various domains, including critical areas such as intrusion detection. Machine Learning (ML) techniques are at the forefront of this integration, enhancing the capability of Intrusion Detection Systems (IDSs) to detect and respond to threats. Recently, advanced Deep Learning (DL) methods have been extensively leveraged, offering unprecedented accuracy and adaptability. However, while these sophisticated methods improve performance, they also introduce security issues.

This thesis explores the security of recent ML and DL techniques in the context of network intrusion detection; specifically, their vulnerability to adversarial attacks, and the countermeasures that enhance their robustness. Our findings provide researchers and practitioners with guidelines for security evaluations and insights into defense strategies.

Federated Learning (FL) allows multiple entities to train an ML model collaboratively without sharing privacy-sensitive training data. However, malicious participants could interfere with the model training. The first contribution addresses the threat of poisoning attacks on FL-based intrusion detection models. We evaluate the impact of four attack parameters on the effectiveness, stealthiness, consistency, and timing of backdoor attacks. With careful adjustment, our results demonstrate the decisiveness of each parameter for attack success.

Deep Reinforcement Learning (DRL) is increasingly employed in intrusion detection for its adaptability in complex environments such as computer networks, but its reliance on DL makes it vulnerable to adversarial examples. The second contribution assesses the influence of crucial DRL hyperparameters on the performance and robustness of intrusion detection agents, including white-box and black-box attacks through the transferability property.

While adversarial examples successfully evade ML-based IDSs, they only represent a concrete threat if they can be implemented in real networks. The third contribution investigates the practicality of those adversarial evasion attacks. We study the impact of state-of-the-art attacks on the model performance, data structure, perturbed features, and common successful attacks. We introduce and discuss four crucial criteria for the validity of adversarial examples.

Diffusion models have demonstrated great potential in data generation and also adversarial purification. The fourth contribution proposes diffusion models to purify adversarial examples in intrusion detection. Our approach is model-agnostic, attack-agnostic, and does not require retraining the model. Through a comparative analysis, we identify optimal diffusion settings to increase robustness and minimize impacts on accuracy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

A2C	Advantage Actor-Critic
A3C	Asynchronous Advantage Actor-Critic
ABC	Artificial Bee Colony
ABML	Agent-Based Machine Learning
ACC	Accuracy
AE-RL	Adversarial Environment using Reinforcement Learning
AI	Artificial Intelligence
AI-GAN	Attack-Inspired Generative Adversarial Network
ART	Adversarial Robustness Toolbox
AIDS	Anomaly-based Intrusion Detection System
ASR	Attack Success Rate
AML	Adversarial Machine Learning
BiGAN	Bidirectional Generative Adversarial Network
BIM	Basic Iterative Method
CPS	Cyber-Physical System
C&W	Carlini and Wagner
CNN	Convolutional Neural Network
DBF	Deep Belief Network
DDoS	Distributed Denial of Service
DDQN	Double Deep Q-Network
DF	DeepFool
DL	Deep Learning
DNN	Deep Neural Network
DoS	Denial of Service
DPA-FL	Defending Poisoning Attacks in Federated Learning
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
DSE	Deep Similarity Encoder
FedAGRU	Federated Learning-based Attention Gated Recurrent Unit
FGSM	Fast Gradient Sign Method
FL	Federated Learning
FNR	False Negative Rate
FPR	False Positive Rate

GA	Genetic Algorithm
GAN	Generative Adversarial Network
GDP	Gross Domestic Product
GNN	Graph Neural Network
GRU	Gate Recurrent Unit
HIDS	Host-based Intrusion Detection system
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IIoT	Industrial Internet of Things
IoT	Internet of Things
IoV	Internet of Vehicles
IRP	Incident Response Plan
IP	Internet Protocol
IT	Information Technology
JSMA	Jacobian-based Saliency Map Attack
LSTM	Long-Short-Term Memory
MB	Mega Byte
MDP	Markov Decision Process
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NIDS	Network-based Intrusion Detection System
NLP	Natural Language Processing
PCA	Principle Component Analysis
PDR	Poisoned Data Rate
PGD	Projected Gradient Descent
PMR	Poisoned Model Rate
PPO	Proximal Policy Optimization
PSO	Practical Swarm Optimisation
QRDQN	Quantile Regression Deep Q-Network
ReLU	Rectified Linear Unit
RePO	Reconstruction from Partial Observation
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RNN-ADV	Random Neural Network-based Adversarial intrusion detection system
SDN	Software Defined Network

SIDS	Signature-based Intrusion Detection System
SNN	Self-normalizing Neural Network
SSH	Secure SHell
STIN	Satellite-Terrestrial Integrated Network
TAA	Transferable Attentive Attack
TCP	Transmission Control Protocol
TD	Temporal Difference
TRPO	Trust Region Policy Optimization
TTL	Time To Live
UDP	User Datagram Protocol
UPL	Unit Per Layer
VAE	Variational Auto-Encoder
ZOO	Zeroth Order Optimization

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

This chapter introduces the context and motivation for the research. It defines the main research objective and the intermediate objectives and presents the structure of the remainder of the thesis.

1.1 Context and motivation

Connectivity and information exchange have become the cornerstone of modern society. Every aspect of our lives is digitized to accommodate the interconnected electronic devices it passes through. As information technology (IT) provides immense convenience, it also fosters a profound societal dependence on cyberspace, giving malicious actors an inciting opportunity to disturb its most vital functions. Cybersecurity Ventres reported that cybercrime has significantly impacted the global economy, estimating a worldwide cost of $8 trillion in 2023, which represents over $250,000 every second [1]. This trend is unlikely to decrease. In fact, according to the same source, the cost of global cybercrime is expected to see a %15 growth per year, reaching $10.5 trillion in 2025.

Faced with such threats, society had to adapt and fight back to ensure its security in cyberspace. The cybersecurity field has emerged to study and formalize the various threats posed by IT and design efficient countermeasures. Its market value is estimated at just over $180 billion in 2024, and predictions indicate that this figure is expected to approach almost $315 billion by 2029 [2].

Cybersecurity was first and foremost dedicated to protecting IT infrastructures against cyberattacks. This was achieved through physical, technical, and organizational measures such as cryptography, access control, network filtering, etc. This first approach, called cyber protection, lacked the capability to consider potential breaches and remedy them. Indeed, the protection measures can have vulnerabilities in their design, implementation, or usage. Furthermore, the human factor is a potential gateway for attackers, in addition to the insider threat which bypasses the cyber protection measures. Thus, it was crucial to consider the dynamic nature of cybersecurity threats and adopt a more comprehensive approach.

Cyber resilience goes beyond mere protection and incorporates strategies to absorb, adapt, and recover from cyber incidents effectively. This approach acknowledges that breaches are not only possible but inevitable and focuses on minimizing their impact. A critical aspect of cyber resilience is continuously monitoring and analyzing IT systems to detect any abnormal

activity and flag potential threats. This function is technically enabled by the Intrusion Detection System (IDS), which automatically scans the activity on a computer network or a host machine to identify malicious behavior.

Traditionally, IDSs compare the network traffic or the system activity against a database of known malicious patterns called "signatures". They consist of specific data sequences or characteristics associated with known threats, such as malware, viruses, or worms. If the activity matches a signature in the database, the IDS triggers an alert and takes predefined actions, such as blocking traffic, revoking access, or alerting a security administrator. Further investigations are then conducted to mitigate the incident and recover system operations as part of the Incident Response Plan (IRP).

While signature-based IDSs effectively detect known threats for which signatures exist, they also have limitations. The constant update and maintenance of the signature database requires significant resources, in addition to the computational requirements involved when comparing an activity with a large number of signatures. More importantly, they fail to detect attacks for which they do not have matching signatures in their database, including newly emerging threats that are yet unknown, called "zero-days". Finally, since the signature databases are common to all appliances, they poorly adapt to the specificities of each environment, causing a high rate of false positives (benign activity flagged as malicious) and false negatives (malicious activity flagged as benign). To overcome these limitations, researchers have proposed integrating statistical methods [3], and later Machine Learning (ML) [4], to improve the detection capabilities of IDSs.

The intrusion detection task can be seen as a classification task of network traffic or system activity as benign or malicious. ML models have shown promising performance on classification tasks, especially with the rise of Deep Learning (DL) [5] using Deep Neural Networks (DNNs). ML-based IDSs consist of an ML model trained on a large dataset, in a supervised or unsupervised manner, to identify threats and classify them into different types. Since ML models learn data patterns and behavioral profiles instead of explicit signatures, their detection is not limited to attacks included in the training dataset, which allows the detection of previously unknown threats (zero-day). As ML models do not require a database search, they can process massive amounts of data much faster and scale up easily. Moreover, ML models can learn continuously from the data of a specific environment and adapt to it, which increases their accuracy.

Building on the success of DL, new techniques have emerged to leverage DNNs beyond the task of classification. For instance, DNNs have been employed in Reinforcement Learning (RL) [6] tasks to model complex Q-functions—using Deep Q-Networks (DQNs) [7]—or policy

functions [8], introducing the field of Deep Reinforcement Learning (DRL). While RL has been applied to intrusion detection before [9, 10], the recent achievements of DRL revived the cybersecurity community's interest. DRL agents were integrated into IDSs for their flexibility, efficiency, and scalability [11, 12], demonstrating state-of-the-art performance.

On the other hand, as DL requires large amounts of data, privacy concerns increased and prevented the acquisition of sensitive data necessary for some tasks. Federated Learning (FL) [13] was introduced to overcome this concern by providing a decentralized collaborative training framework. In FL, several clients collaborate to train a model—typically a DNN—without sharing their data. Instead, they train a common model locally and send the model updates to the server. The model updates from all the clients are then aggregated into the global model.

FL benefits particularly from critical domains such as intrusion detection where the models are trained on privacy-sensitive data (network traffic or system activity). The cybersecurity community has shown a notable interest in FL, proposing various FL-based IDS frameworks [14]. A particular emphasis was put on Internet of Things (IoT) systems. They consist of large networks of devices that generate data continuously and are often prone to cyber attacks, making them a relevant use case for FL-based IDSs [15].

While the integration of these recent DL advances into IDSs greatly improves their performance, scalability, and privacy, their dependence on DNNs introduces new vulnerabilities. Adversarial Machine Learning (AML) [16] is the research field that studies the security concerns related to ML in adversarial environments. AML categorizes the threats to ML into 4 classes [17]: *Poisoning* and *evasion* that compromise integrity and availability, and *extraction* and *inversion* that compromise privacy. Poisoning attacks [18] target the model's training and aim to influence the prediction by manipulating the training data; Evasion attacks [19] target trained models and aim to influence their prediction by manipulating the inputs; Extraction attacks [20] aim to steal the parameters of a trained model; Inversion attacks [21] aim to recover information about the training data from a trained model.

Although FL protects privacy by allowing clients to train the model locally, it also allows malicious clients to interfere with the training process. In the FL setting, malicious clients can perform poisoning attacks against the global model, affecting all its users. These poisoning attacks can aim to prevent the convergence of the model [22] (untargeted attacks) or introduce a malicious behavior in the model for later exploitation [23] (targeted attacks). Backdoor attacks [24,25] are an example of targeted poisoning attacks; they involve introducing a subtle pattern in a subset of the training data and changing their label for a target label. During the training, the model will learn the correspondence between the pattern (called trigger)

and the target label. When deployed, the model has a higher chance of classifying any data containing the trigger into the target label set by the attacker.

In the case of FL-based IDSs, a malicious client could train the model locally on data containing a trigger with the benign label. Once the global model is deployed, they could attack other clients without being detected only by adding the trigger to their malicious data. Existing literature has provided evidence of the success of backdoor attacks against FL-based IDSs [26] and proposed countermeasures [27]. However, the existing literature presents only preliminary evidence and falls short of providing a comprehensive analysis of the vulnerability, especially considering the various attack parameters that influence its effectiveness, stealthiness, consistency, and timing.

Since DRL relies on DNNs, it inherits their vulnerability to evasion attacks, notably adversarial examples [28]. Adversarial examples are data instances to which an imperceptible and well-computed perturbation was added in order to influence the DNN's prediction. They are untargeted if they aim to cause a misclassification regardless of the label, and targeted if they aim to produce a specific label [29]. The main difference with backdoor attacks is that adversarial examples do not require any interference with the training process, they exploit blind spots present in all high-dimensional models [30].

The phenomenon of adversarial examples has garnered significant attention within the research community, with numerous studies dedicated to its exploration and understanding [31]. The vulnerability of DRL agents to adversarial examples has been of particular interest, due to the specificities of DRL frameworks and their applications [32]. However, despite this extensive research, no work has specifically investigated the impact of adversarial examples on DRL-based IDSs, which impedes their implementation. Without understanding how these systems perform under adversarial attacks, it's challenging to deploy them confidently in real-world security scenarios. This gap underscores the need for dedicated studies to assess the vulnerability of DRL-based IDSs to adversarial examples.

Furthermore, the data structure in intrusion detection presents a challenge for the generation of adversarial examples. Each feature imposes different restrictive constraints on its values and depends on the values of other semantically related features. These constraints significantly limit the possibilities attackers have when crafting adversarial examples. Adversarial perturbations that do not consider the data constraints might hinder the attack's malicious objective or even prevent its implementation in real-world scenarios. It is thus necessary to study comprehensively the impact of adversarial perturbations on the data structure and identify concrete criteria to validate adversarial examples. This is the first step towards designing realistic adversarial examples to evaluate the concrete vulnerability of ML-based

IDSs.

In addition to understanding the vulnerability of ML-based IDSs to adversarial examples, it is crucial to address it by designing effective defense mechanisms. Several approaches have been proposed in the literature to mitigate adversarial examples. Some of the most dominant are adversarial training, adversarial detecting, and adversarial purification. Adversarial training [33] consists of augmenting the training data with adversarial examples to robustify the model; Adversarial detecting [34] consists of detecting adversarial examples—often using a DNN—and discarding them before they reach the ML model; Adversarial purification [35] consists of removing the adversarial perturbation from adversarial examples to classify them correctly.

Generative models are particularly successful in adversarial purification due to their ability to learn the manifold of natural data distributions. Since adversarial examples lay slightly outside of this manifold, the generative models discard them to reconstruct the natural examples. Generative models such as Generative Adversarial Networks (GANs) [36] or Variational Auto-Encoders (VAEs) [37] have been successfully applied to adversarial purification [38, 39], but the recently introduced diffusion models [40, 41] appear as promising candidates [42]. They consist of a forward process that gradually adds noise to the data and a reverse process that gradually reconstructs the data using a DNN. Though the data might be adversarial, the reconstruction should be inside the natural data manifold without adversarial perturbations.

Diffusion models have demonstrated efficacy in adversarial purification tasks, particularly in the domain of image processing [43, 44]. However, the literature notably lacks applications of diffusion models to purify intrusion detection data. Despite their proven success in image-related tasks, their potential in the realm of intrusion detection remains largely untapped. Exploring the application of diffusion models to intrusion detection could open new avenues for enhancing the robustness and reliability of IDSs against adversarial examples.

In summary, IDSs benefit greatly from the latest developments in DL techniques—such as DRL and FL, improving their performance, scalability, and adaptability. However, they also introduce critical vulnerabilities to evasion attacks, through adversarial examples, and poisoning attacks, through backdoors. These vulnerabilities must be comprehensively studied to determine their concrete impact on the reliability of IDSs. Furthermore, particular attention should be put on how the attacks affect the data structure and criteria that prevent their application in real-world scenarios. Finally, the study of adversarial attacks should lead to the development of efficient defenses, using state-of-the-art tools, to enhance the robustness of DRL-based IDSs.

1.2 Research objectives

In light of all the security challenges posed by the integration of DL techniques in IDSs, this thesis undertakes a comprehensive examination of the underlying vulnerabilities. The main research objective is to *assess the concrete impact of the vulnerabilities introduced by DL techniques in intrusion detection and design appropriate countermeasures.* This objective is divided into the following intermediate objectives.

- Analyze the influence of adversarial poisoning parameters on the effectiveness, stealthiness, consistency, and timing of backdoor attacks against FL-based IDS
- Unveil the impact of DRL hyperparameters on the performance and robustness of DRL-based IDSs to white-box and black-box adversarial examples
- Evaluate the effect of adversarial perturbations on individual features and data structures, and identify the criteria that invalidate adversarial examples
- Design a purification defense based on diffusion models for IDSs and study diffusion parameters to minimize the reconstruction loss and maximize the accuracy

This research aims to provide new insights, for researchers and practitioners, on the vulnerabilities inherent to ML-based IDSs and encourage the development of robust systems through security-oriented design and efficient defense mechanisms.

1.3 Thesis structure

Following the introduction, this thesis carries on with a brief literature review in Chapter 2 presenting the main related work and some background, then jumps into the four chapters described below, each corresponding to a research article written as part of the author's doctoral project. The thesis concludes with a summary and future research perspectives in Chapter 7.

Each of the four research articles contributes to the main research objective by addressing one of the intermediate objectives. The first article, in Chapter 3, investigates the threats at training-time; It studies backdoor attacks in the context of FL-based IDSs with an emphasis on the impact of different poisoning parameters. The second article, in Chapter 4, investigates the threats at inference-time; It addresses the vulnerability of DRL-based IDSs to adversarial examples and unveils the importance of DRL hyperparameters for performance and robustness. This article extends another published article in Appendix A. The third

article, in Chapter 5, presents a deeper analysis of the impact of adversarial examples on the data structure and identifies invalidation criteria. Finally, the fourth article, in Chapter 6, addresses the remediation by proposing a purification defense based on diffusion models for ML-based IDSs. All articles are presented as they were published or submitted, except for minor editing to ensure consistency in formatting, style, and referencing throughout the thesis.

CHAPTER 2 BACKGROUND AND LITERATURE REVIEW

The last decade has seen the emergence of ML as a revolutionary tool in various domains, including the most critical ones such as intrusion detection. IDSs benefit greatly from ML, as it increases their capabilities to identify threats and their adaptability to specific environments, representing a significant improvement over traditional signature-based methods. However, alongside these advancements, IDSs also inherit the vulnerability of ML to adversarial attacks, posing substantial challenges to the reliability of these systems. This vulnerability is particularly concerning for IDSs, where the consequences of misclassification can lead to severe security breaches.

This literature review presents an overview of the current state of research on the adversarial threat to ML-based intrusion detection. By synthesizing existing studies around the four main research themes of this thesis, it aims to identify the different applications of recent ML and DL techniques to the intrusion detection task and the existing investigations of their vulnerabilities to adversarial attacks. Subsequently, the review will delve into the various defense strategies proposed to protect ML-based IDSs from adversarial examples.

Moreover, this literature review positions our contributions within the current research landscape. By synthesizing existing studies, we highlight the gaps and challenges in the field, demonstrating where our work provides novel insights and advancements. Through this review, we aim to show not only the relevance of our contributions but also their potential impact on future research directions and practical implementations.

In addition to this general literature review, a more detailed literature review focused on the related work of each individual paper included in this thesis is available in the respective chapters. These in-depth reviews provide a closer examination of the specific context, methodologies, and findings relevant to each study, ensuring that the broader literature review presented here remains cohesive and non-redundant. The reader will find in this literature review references to the relevant sections in the body of the thesis that elaborate on specific topics.

This literature review assumes that the reader already has a foundational understanding of IDSs, ML, and DL. It builds on these concepts to explore the nuances of adversarial attacks and their implications for IDSs, without delving into the basics of these underlying technologies. Readers interested in those foundations may refer to Section 4.2.1 or Buckzak *et al.*'s survey [4] that offers a comprehensive introduction to ML applications in IDSs.

The review will be organized into several thematic sections related to the main research objectives described in Section 1.2. Section 2.1 presents pioneering work on the vulnerability of DL-based IDSs to adversarial examples. Section 2.2 explores the application of RL learning to intrusion detection and their vulnerability to adversarial examples. Section 2.3 addresses the topic of FL-based intrusion detection and the threat of malicious clients using poisoning attacks. Section 2.4 reviews different defense mechanisms applied to intrusion detection to increase their robustness to adversarial examples.

2.1 Deep learning-based intrusion detection

Since the emergence of interconnected computer networks, security has been a central issue. In particular, the capacity to detect and mitigate threats was necessary, giving birth to IDSs as monitoring and protection tools. However, the rule-based approach had its limitations in terms of performance and adaptability. Thus, as early as 1987, statistical methods were proposed to establish user profiles and identify deviations as potential threats [3]. Since then, various supervised and unsupervised ML techniques have been extensively deployed in IDSs, demonstrating an increasing interest among the research community [4].

Around the 2010s, ML significantly improved through the development of DNNs, often referred to as *the third wave of artificial intelligence* or *the deep learning revolution* [45]. Due to the rise of powerful computing, the availability of data in the digital age, and the research advancements in neural network architectures, modern DL techniques achieved groundbreaking performance in various tasks, such as computer vision [46–50], Natural Language Processing (NLP) [51–54], and speech recognition [55–57].

While neural networks have been used for intrusion detection since 1990 [58], recent advances in DL have renewed the research community's interest in DNNs. A large body of research has applied the latest advances in DL to intrusion detection datasets, demonstrating outstanding performance in terms of accuracy, False Positive Rate (FPR), False Negative Rate (FNR), and zero-day detection [59–61].

Among the major works using DNNs for intrusion detection, [62] explores various DNN architectures and hyperparameters on multiple network intrusion detection datasets and compares them with classic ML methods. [63] leverages Deep Belief Networks (DBNs) to extract features from network traffic and feed them to a logistic regression model for classification. [64] uses non-symmetric deep auto-encoders for unsupervised feature learning. [65] introduces a Gated Recurrent Unit Recurrent Neural Network (GRU-RNN) for intrusion detection on Software Defined Networks (SDNs). [66] proposes to use dimensionality reduction tools to

keep only the relevant features in the network traffic, convert the actors into an image format, and feed them to a Convolutional Neural Network (CNN). On the other hand, [67] uses a similar approach with CNNs on raw data and addresses the imbalanced dataset problem by adapting the cost function weight coefficient for each class.

Since the early days of ML, researchers have investigated its security [17], especially the impact of small modifications in the input data on the model's prediction. As early as 2004, [19] studied the problem of adversaries manipulating the input of ML models to evade spam, fraud, or intrusion detection with a game theory approach. The authors develop a formal framework and produce an optimal classifier given the adversary's optimal strategy. In the continuity of the previous work, Lowd *et al.* [68] introduces the adversarial classifier reverse engineering learning problem, the task of learning sufficient information about a classifier to construct adversarial attacks. They propose algorithms for reverse engineering linear classifiers with either continuous or boolean features and apply them to spam filtering datasets. Furthermore, early research has addressed the issue of training data poisoning. [18] presents an attack on ML-based spam detection where the adversary causes the misclassification of 36% of spam messages with only 1% control over the training set.

While these works pioneered the ML security research field, they preceded the DL revolution and were limited to linear models. Only in late 2013 were investigations carried out on DNNs, demonstrating their vulnerability to adversarial examples [28], data with small and carefully computed perturbations that compromise the DNN's prediction. This discovery marked a pivotal moment in the ML field, highlighting the ease with which state-of-the-art DL models could be fooled. This finding spurred a flurry of research aimed at understanding the underlying mechanisms of this vulnerability and developing methods to defend against such attacks [31]. Subsequent studies identified various types of adversarial attacks, including white-box attacks (such as FGSM [5], JSMA [29], DeepFool [69], and Carlini&Wagner's attacks [70]) where the attacker has complete knowledge of the model, and black-box attacks [71,72], where the attacker has limited or no knowledge of the model's architecture and parameters. For more detailed information, readers may refer to Section 5.2.1 which offers a comprehensive discussion on the details of adversarial example generation methods.

Most of the early research was carried out on computer vision, the main task on which DNNs were tested, but other critical domains quickly followed. IDSs, which already suffered from evasion attacks before DL [73], faced the new threat of adversarial examples due to their reliance on DNNs. The threat was all the more severe because the intrusion detection task is adversarial by nature, as it faces adversaries purposefully trying to evade the detection [19]. Furthermore, the importance of IDSs in computer networks, especially in critical infrastruc-

tures, makes them an attractive target for adversaries. All these aspects motivated research efforts to study the impact of adversarial examples on DL-based IDSs [74–76].

Rigaki *et al.* [77] first showed the vulnerability of neural networks in IDSs to adversarial examples. They generated adversarial examples using FGSM and JSMA against the NSL-KDD network intrusion detection dataset [78]. Wang [79] followed with a more detailed analysis using four adversarial example generation methods, namely FGSM, JSMA, DeepFool, and Carlini&Wagner's attacks. They also used the NSL-KDD dataset [78], but they included an additional analysis of the frequency at which each attack altered features. While these works established the first milestone for adversarial examples against intrusion detection, they only used the NSL-KDD dataset, and they assumed a white-box adversary who is aware of all the model's parameters. Moreover, they only addressed the case of simple Multi-Layer Perceptron (MLP) neural networks and did not consider sophisticated architectures.

To overcome these limitations, Clements *et al.* [80] applied adversarial attacks to Kitsune [81], an unsupervised online IDS using an ensemble of autoencoder, to show the vulnerability of existing architectures. To extend the study of adversarial examples to different types of neural networks and different types of network intrusion detection data, Ibitoye *et al.* [82] applied adversarial examples to Self-normalizing Neural Networks (SNN) on BoT-IoT [83], an IoT intrusion detection dataset. On the other hand, Yang *et al.* [84] question the adversary's capabilities, they assume an adversary with limited knowledge about the model (gray-box scenario) and generate adversarial examples using the Zeroth Order Optimization (ZOO) method [72].

The previously mentioned studies leverage existing adversarial example generation methods proposed in the literature, often designed for computer vision tasks. Other researchers took the direction of generative models to produce adversarial examples. Lin *et al.* [85] propose IDSGAN, a framework for crafting adversarial examples against IDSs. They use Wasserstein GANs [86] to generate adversarial examples and test them against seven types of ML classifiers. In this setting, a generator is trained against a black-box IDS and learns to maximize the IDS's error only through its classification score. Alhajjar *et al.* [87] go further by targeting more types of ML classifiers and leveraging evolutionary algorithms to generate adversarial examples against two network intrusion datasets, NSL-KDD and UNSW-NB15 [88]. They use Genetic Algorithms (GAs) and Practical Swarm Optimisation (PSO) to update their population based on a fitness function, which consists of the black-box classifier's output [76].

All these works extensively studied the vulnerability of ML-based IDSs to adversarial examples, generating adversarial examples with both state-of-the-art methods and generative models, exploring several ML and DL architectures, and applying them to various network

datasets. However, little attention was given to the impact of adversarial perturbations on the consistency and soundness of network data features, leaving a significant gap in the literature. Indeed, network traffic has a rigorous structure binding its feature values, as opposed to less structured data like images. Since most recent advances in DL, including adversarial examples, were experimented on computer vision tasks (images) they do not consider the inherent domain constraints of tabular data like network traffic. While the results of adding adversarial perturbations to an image is always an image, the same does not apply to network traffic; as small changes in its data features can damage the structure. The previous research efforts did succeed in decreasing the IDS's accuracy, but only on the data instance level (after data features are extracted from network traffic). This does not necessarily mean that perturbed data features correspond to valid network traffic and can be implemented in real-world networks.

The contribution in Chapter 5 [89] fills that gap in the literature by studying the practicality of adversarial examples in network intrusion detection. This work comprehensively analyzes the impact of seven state-of-the-art adversarial attacks on the detection rate of a DL-based IDS and their impact on the network traffic features from 3 different network datasets (NSL-KDD, UNSW-NB15, CIDDS-001 [90]). Their analysis highlights various L_p norm distances between original and adversarial examples, as well as the features targeted by each adversarial attack and the common successful adversarial examples between adversarial attacks. The authors identify four criteria that invalidate network traffic after adversarial perturbations: value ranges, binary values, multiple-category membership, and semantic relations.

This work represented a significant milestone in this research topic, it highlighted the urgency of considering domain constraints in the generation of adversarial examples against intrusion detection, or any structured data domain. Its impact appears in subsequent research work where domain constraints are at the center of attention. Sheatsley *et al.* [91] extracts domain constraints from the network traffic analysis. Then, they introduce Adaptive-JSMA which verifies the domain constraints at each iteration of JSMA, and Histogram Sketch Generation which generates adversarial sketches: targeted universal perturbation vectors that encode feature saliency within the envelope of domain constraints. A subset of the authors extend their work in Zolbayar *et al.* [92], introducing NIDSGAN, a GAN-based framework that generates adversarial examples. NIDSGAN imposes two types of domain constraints: inherent constraints of network traffic flow and valid ranges after the perturbation. Their framework is tested on two datasets: NSL-KDD and CICIDS-2017 [93]. While the authors consider these constraints, the adversarial example generation is still at the features level and does not guarantee that it corresponds to valid network traffic.

On the other hand, Cheng *et al.* [94] introduce Attack-GAN, an adversarial example generative model based on the sequence generation model SeqGAN [95]. Instead of perturbating the features at the feature level, they generate adversarial network traffic at the packet level by generating sequential bytes of a network packet. Their generator consists of an LSTM that learns through the guidance of a discriminator, the discriminator leverages the prediction results of a black-box IDS on the generated network traffic. Attack-GAN generates packets conditioned to the basic packet format and additional domain constraints are imposed on the generated traffic to retain its attack functionalities.

Recent work on the vulnerability of ML-based IDSs to adversarial examples has taken a promising direction by focusing on the structure of network traffic and considering its constraints in the generation of adversarial examples. However, these works failed to provide more concrete details about how the domain constraints are extracted, what they involve, and how they are enforced in the adversarial example generation process. Furthermore, as mentioned previously, applying domain constraints to data-level features extracted from network traffic does not guarantee their practicality. While GAN-based generative models propose to model the structure of network traffic and relationships between features to make the constraints inherent to the generation process, they offer no guarantee of the practicality of generated adversarial examples. The network features used for intrusion detection are only abstract representations of the network packets [76], they cannot be transmitted on the network and do not always correspond to valid network traffic. These representations live in a different space produced by the feature extraction and reduction function from the real network traffic space. As long as this function is neither invertible nor differentiable [96], feature-level adversarial examples cannot be translated into practical network traffic. Meanwhile, the packet-level generation of adversarial network traffic, as proposed in [94], remains the most practical option and should be further investigated to provide more concrete details about the generation process.

2.2 Deep reinforcement learning-based intrusion detection

RL is a type of AI where an agent learns through trial and error in an interactive environment. The agent receives rewards for desired actions and penalties for undesired ones, allowing it to gradually improve its decision-making and achieve specific goals. Following the success of DL in various tasks, researchers started integrating it into RL algorithms to solve problems where the environment is complex and the optimal actions are not explicitly programmed. The last decade has seen the emergence of multiple DRL algorithms that rely on DNN. In particular, value methods, such as DQNs [7], rely on DNN to approximate the value

function that represents the expected reward of a state or a state-action pair. On the other hand, policy methods, such as PPO [97], use DNNs to represent the policy itself. For more background on RL and DRL methods, readers might refer to Section 4.3.

RL is a promising venue for intrusion detection, it allows better adaptability with the ability to learn continuously from the changing network conditions and offers flexible reward functions that do not need to be differentiable [11]. Furthermore, RL balances between exploiting learned policies and exploring new policies, while handling the uncertainty in complex environments such as computer networks. Applications of RL to intrusion detection emerged in the 2000s, with researchers such as Xin Xu exploring the use of tabular RL methods in intrusion detection. In Xu and Xie [9], the authors propose a host-based IDS using Temporal Difference (TD) [98] to detect abnormal behavior in system calls. Because it was based on a linear basis function, the proposed method struggled with highly nonlinear intrusion behaviors. In Xu and Luo [99] and Xu [100], the authors suggest kernel-based RL approaches applying least-squares TD [101] to enhance the generalization capability of TD in high-dimensional and nonlinear feature spaces [102]. On the other hand, Servin and Kudenko [10] leverage a Q-learning algorithm based on a look-up table for intrusion detection. Shamshirband *et al.* [103] propose a fuzzy Q-learning method for intrusion detection and prevention in wireless sensor networks, their approach is based on cooperative game theory. Deokar and Hazarnis [104] propose a hybrid IDS using both signatures and anomaly detection with association rule learning and log correlation techniques. Furthermore, Malialis and Kudenko [105] propose a distributed network IDS using multiagent RL.

Intrusion detection also benefited from the recent developments in DRL methods. In addition to the advantages of RL, DRL allows a better feature representation, while the agent after training consists of a simple lightweight neural network. Caminero *et al.* [11] introduce the Adversarial Environment using Reinforcement Learning (AE-RL) algorithm for training DRL-based IDS. The adversarial environment acts as an adversarial agent that samples training examples from the datasets and feeds them as states to the DRL IDS agent. The adversarial sampling addresses the unbalanced dataset problem by oversampling underrepresented classes on which the IDS usually struggles. The authors test their agent against numerous ML, DL, and DRL methods on two datasets: NSL-KDD and AWID [106]. The contribution in Appendix A [107] leverages the AE-RL algorithm introduced in Caminero *et al.* [11]. On the same note, Lopez-Martin [12] a subset of the previous co-authors applied several DRL algorithms to intrusion detection, namely DQN, Double DQN (DDQN), policy gradient, and actor-critic. Experiments on NSL-KDD and AWID result in better performance with DDQN which also overperforms many traditional ML methods.

Since DRL methods rely on DNNs, they also share their vulnerability to adversarial examples. Recent work [32, 108–115] has demonstrated how adversarial examples could significantly perturb DRL agents on various tasks. Readers may refer to Section 4.2.4 for more details on the adversarial attacks against DRL. However, while DRL-based intrusion detection is attracting more attention due to all its advantages, the literature still lacks any intuition on the impact of adversarial examples on DRL-based IDS. Such attacks could potentially compromise DRL intrusion detection agents and render them useless against slightly modified malicious traffic. Thus, it is crucial to conduct a comprehensive analysis of the vulnerability of DRL-based IDS to adversarial examples.

The contribution in Chapter 4 [116], which extends the contribution in Appendix A [107], addresses this necessity. This work studies the impact of single-step and iterative gradient attacks [33] on DRL-based IDS using three datasets (NSL-KDD, UNSW-NB15, and an IoV case study on CICIoV2024 [117]). Both attacks are carried out on 80 combinations of DRL hyperparameters on every dataset, including 4 depths (1, 2, 4, 8 hidden layers), 4 widths (32, 64, 128, and 256 units per hidden layer), and 5 DRL algorithms (DQN, PPO, TRPO, A2C, and QRDQN). In addition to white-box attacks, the study is extended to black-box attacks through the transferability property; adversarial examples generated on different DRL algorithms are tested on other target DRL algorithms. This contribution highlights the critical role of proper hyperparametrization in the adversarial robustness of DRL-based IDSs. Moreover, it provides insights for researchers and practitioners on DRL optimization of IDS to enhance performance and resilience.

2.3 Federated learning-based intrusion detection

The main parameter that influences the performance of ML models, especially in the DL era, is the quantity and quality of training data, and intrusion detection makes no exception. A DL-based IDS must be trained on a large quantity of realistic network traffic, which must also be diverse enough to reflect the different examples of benign and malicious traffic. Often, a single entity might not have access to enough diverse network traffic in its network, enough storage for this data, or enough computational power to train large models. For this reason, multiple entities could decide to train an intrusion detection model collaboratively by aggregating all their data together. This scenario allows every entity to benefit from a better IDS and increases the security level of everyone.

However, network traffic data contain privacy-sensitive information about the network users. Sharing such data would represent a serious privacy breach, and could lead to other security threats if the information is leveraged for cyberattacks. Data anonymization is an option

for preventing privacy breaches, but studies have shown how private information could be retrieved through comparison with other data [118]. Encryption protects the data privacy but also prevents the model from learning anything. To address these issues, McMahan *et al.* [13] proposed FL as a method for training ML models collaboratively without direct sharing of privacy-sensitive training data. In FL, each client of the federation trains their model locally on their data and only shares the model parameters update with the server. The server aggregates all the clients' updates to form the global model and transfers it to the clients. In addition to preserving the privacy of client's data, FL improves the scalability of collaborative learning by distributing the computation to all the clients, reduces the communication overhead, and enhances resilience against client failure.

For all these advantages, FL has been widely applied to intrusion detection models, especially in distributed systems such as IoT [14]. As described in Section 3.2, Abdul Rahman *et al.* [119] propose a comparison between three scenarios in IoT intrusion detection: (*i*) training the model on the cloud or in a closer fog infrastructure (centralized approach), (*ii*) training the model locally (self-learning approach), (*iii*) training the model collaboratively (FL approach). The FL approach overperforms self-learning and compares with the centralized approach performance in three use cases: malicious traffic distributed per attack type, attack types equally distributed, and random distribution. Also on IoT, Nguyen *et al.* [120] introduce DÏoT, an autonomous distributed system for detecting compromised IoT devices. Their method leverages device-type-specific communication profiles and trains on labeled data, requiring no human intervention. DÏoT aggregates behavior profiles efficiently through FL. Cetin *et al.* [121] propose an FL method for intrusion detection on wireless networks, they test it on a simulated environment using wireless data from the AWID dataset.

Beyond simple neural networks, research has investigated the FL training of sophisticated DNN architectures for intrusion detection. Chen *et al.* [122] propose the Federated Learning-based Attention Gated Recurrent Unit (FedAGRU) to detect intrusion in wireless edge networks. They apply the attention mechanism to increase the weight of important devices and avoid uploading unimportant updates to the server, significantly reducing the communication overhead. Wang *et al* . [123] leverages FL to train a DRL-based to detect anomalies in Industrial IoT (IIoT) systems. The authors identify abnormal users with a privacy leakage degree defined for each user based on the leakage of sensitive and non-sensitive information. Still on IIoT, Li *et al.* [124] introduce FLEAM, a FL Empowered Architecture to Mitigate DDoS in IIoT. Their method uses fog computing nodes as FL clients and the cloud as the FL server. Fog nodes retrieve the training model from the cloud, train it locally, and send it back to the cloud. Sun *et al.* [125, 126] address the heterogeneity of computer networks arguing that a single FL global model cannot fit the different distribution of network traffic.

Thus, they introduce Segmented-FL, an FL framework that segments the clients' networks and brings similar network environments to the same client group. Their experiments are carried out with CNN models on a dataset collected from 20 massively distributed networks within 60 days.

FL has also been applied to different use cases, Li *et al.* [127] propose DeepFed, an FL framework to train IDSs for industrial Cyber-Physical Systems (CPSs). The authors use CNN and GRU architectures to design the intrusion detection model. DeepFed also preserves the privacy of model parameters by using a Paillier cryptosystem-based secure communication protocol. Addressing the issue of limited resources and high privacy requirements of Satellite-Terrestrial Integrated Networks (STINs), Li*et al.* [128] propose an FL-based IDS for STINs. The authors test their methods on a simulated environment with a mixture of attacks against both satellite and terrestrial networks. Zhao *et al.* [129] propose to train a multi-task DNN-based IDS with FL. As opposed to previous work, the authors perform anomaly detection, VPN (Tor) traffic recognition, and traffic classification simultaneously.

While FL offers promising research avenues in intrusion detection, it is not flawless. Indeed, by involving multiple clients in the training process, FL also allows malicious clients to interfere with training. In particular, adversaries can perform poisoning attacks [130] where they manipulate the training data or procedure to influence the global model. They could perform untargeted attacks by reducing the model's performance and preventing its convergence [22] or, even targeted attacks, by introducing a backdoor they can exploit later to evade intrusion detection [23–25]. Readers interested in further exploring the security issues of FL may refer to Gosselin *et al.* [131] or Lyu *et al.* [132], and for a specific review of backdoor attacks, refer to Nguyen *et al.* [133].

As mentioned in Section 3.2, Nguyen et al. [26] first investigated poisoning attacks against Fl-based IDS for IoT. They show how attackers can poison the detection model only through compromised IoT devices injecting malicious traffic, without compromising the FL client. Here, the authors assume that FL clients gather data from their multiple IoT devices. Their method is evaluated on three real-world IoT datasets generated from 46 IoT devices. The authors analyze the impact of the Poisoned Data Rate (PDR) and Poisoned Model Rate (PMR) as their main poisoning parameters.

Despite signs of interest in the vulnerabilities of FL-based IDS, the literature lacks a thorough analysis of the impact of poisoning attack parameters. While Nguyen et al. [26] establish the vulnerability, their study of poisoning parameters is limited to the PDR and PMR; it does not consider other critical data-related parameters such as the choice of trigger features and their number. Furthermore, due to dataset limitations, they employ a fixed number of

clients (IoT devices), raising questions about the scalability of such attacks in relation to the number of FL clients in the federation.

The contribution in Chapter 3 [134] fills that literature gap by proposing a detailed study of the parameters of poisoning attacks against FL-based IDS. As mentioned in Section 3.1, the study considers four critical poisoning parameters: (*i*) the features involved in the poisoning; (*ii*) the number of features involved; (*iii*) the number of clients participating in the training; and (*iv*) the number of malicious clients among them. The impact of those parameters is measured on four aspects of the poisoning attacks: (*i*) the effectiveness, measured with the rate of successful attacks; (*ii*) the stealthiness, measured with the performance degradation caused by the attack; (*iii*) the consistency, based on the data structure of network traffic; and (*iv*) the timing, in terms of training rounds. Moreover, the study proposes guidelines for the security test of FL intrusion detection systems and the design of defense techniques.

To protect FL-based IDSs against poisoning attacks, researchers have devised and implemented diverse defense mechanisms, each aiming to enhance the robustness and integrity of the collaborative learning process. In addition to the research works discussed in Section 3.2, more recent defense methods have been introduced. Lai *et al.* [135] propose a two-phase defense against poisoning attack in FL-based intrusion detection called Defending Poisoning Attacks in Federated Learning (DPA-FL). The first phase uses relative differences to compare weights between participants swiftly, leveraging the significant disparity between the local models of attackers and benign participants. In the second phase, the aggregated model is tested with the dataset, aiming to identify attackers when a drop in accuracy is observed. On the other hand, Yang *et al.* [136] propose a lightweight detection mechanism to mitigate the impact of poisoning attacks on FL-based IDSs in IoT networks. Their method is evaluated on CICIDS-2017 and shows significant improvement in the robustness of FL-based IDSs to label-flipping attacks. The authors implemented their mechanism on the FL server to filter out malicious clients by excluding their uploaded models from the global model aggregation. This is done by introducing a scoring system that evaluates clients based on the loss of their local models and the size of their training datasets. They calculate the Manhattan similarity between each participant's score and then use a clustering algorithm to identify malicious clients through similarity analysis.

2.4 Adversarial example defenses in intrusion detection

Since the emergence of adversarial examples, the research community has invested large efforts in designing efficient countermeasures to protect DL models. Section 6.2.1 presents the three major defensive approaches against adversarial examples: adversarial training [28,

30, 33], adversarial detection [34, 137, 138], and adversarial purification [35, 139, 140]. The remainder of the section will focus on adversarial defenses in intrusion detection. For more details on adversarial example defenses in images, graphs, and text, refer to Xu *et al.* [141] or Yuan *et al.* [31]. For a more general review of privacy and security issues in DL, refer to Liu *et al.* [142].

Due to the critical function of IDSs and their inherently adversarial environment, ensuring their robustness to adversarial examples has been a priority. The following research works leverage adversarial training to improve the robustness of IDSs. Abou Khamis *et al.* [143] trained a DL-based IDS using the min-max (saddle-point) approach: the max approach is used to generate adversarial examples by maximizing the loss function and the min approach is used as a defense to minimize the loss function. Specifically, they augment the training data (from the UNSW-NB15 dataset) with adversarial examples generated with four different methods. The authors also use Principle Component Analysis (PCA) as a dimensionality reduction method and show its positive impact on the model's robustness. Abou Khamis and Matrawy [144] extend their work on adversarial training with a min-max formulation to augment the training set with adversarial examples. In this work, they assess three DL methods (simple neural networks, CNNs, and RNNs) against adversarial examples generated with FGSM, CW, BIM, DeepFool, and Projected Gradient Descent (PGD) [33]. Their experiments are carried out on both NSL-KDD and UNSW-NB15 datasets. Benzaid *et al.* [145] introduce a self-protection framework to defend against application-layer DDoS leveraging DL and SDN for autonomous detection and mitigation of attacks. The authors improve the robustness of their model through adversarial training with FGSM adversarial examples. Debicha *et al.* [146] also study the effectiveness of adversarial training as a defense against adversarial examples in intrusion detection. They generate adversarial examples using FGSM and BIM on the NSL-KDD dataset.

GANs have also been extensively used in adversarial training. Abdelaty *et al.* [147], Usama *et al.* [148], and Novaes *et al.* [149] all leverage GANs to generate adversarial examples for adversarial training. Wang *et al.* [150] introduce Def-IDS, a framework to protect IDSs that consists of two modules. The first module is a multi-class GAN-based dataset enhancement method that generates mimic samples for multi-class intrusions and augments the training dataset near the model's decision boundary to reduce the learning gap. The second is a multi-source adversarial retraining method that integrates adversarial examples from various attacks into the training dataset. The adversarial examples are generated with FGSM, DeepFool, JSMA, and BIM on the CICIDS-2018 dataset.

Researchers have also leveraged adversarial training in combination with other techniques.

Zhang *et al.* [151] introduce Tiki-Taka, a DL-based intrusion detection framework that assesses the robustness of IDSs to adversarial examples and incorporates defense mechanisms to protect them. For the assessment, they use five different adversarial attacks on three IDSs. For the defense mechanisms, they propose three different approaches: (*i*) *model voting ensembling*, where ensembling pretrained MLPs, CNNs, and LSTMs use a voting mechanism to construct a robust model; (*ii*) *ensembling adversarial training* where the model is trained on adversarial examples to increase its robustness; and (*iii*) *adversarial query detection* where a Deep Similarity Encoder (DSE) [152] encodes the data, then the k nearest neighbor average distance is used to detect adversarial examples.

Among other methods used to improve the robustness of DL-based IDSs, Apruzzese *et al.* [153] use ensemble learning as a countermeasure to adversarial examples, they introduce AppCon, an integrable solution to enhance IDS's robustness. Their solution is evaluated on several ML algorithms and against adversarial examples crafted with their own generation method [154]. Han *et al.* [96], McCarthy *et al.* [155], and Ganesan *et al.* [156] study the features of the network traffic datasets and remove vulnerable features to avoid adversarial examples while maintaining acceptable performance. Qureshi *et al.* [157] introduce a Random Neural Network-based Adversarial intrusion detection system (RNN-ADV). The authors train it with the swarm optimization-based Artificial Bee Colony (ABC) algorithm to increase its robustness to adversarial examples. They evaluate their architecture against adversarial examples generated with JSMA on the NSL-KDD dataset.

Adversarial detection has also been investigated for DL-based IDSs. Pawlicki *et al.* [158] propose a detection method that leverages the neural activation of the IDS's neural network to identify adversarial examples. They use the CICIDS-2017 dataset and partition it into 4 sets. The first set is used to train the DL-based IDS, while the second set is used to test the IDS and generate adversarial examples with FGSM, BIM, PGD, and C&W. It is also used to acquire the neural activations of the IDS for benign, malicious, and adversarial traffic on which the adversarial detector is trained. The third and fourth are used to craft test adversarial examples and acquire neural activation for evaluation. For the adversarial detector, authors tried several ML algorithms: Adaboost, random forests, nearest neighbor, neural networks, and Support Vector Machine (SVM). Peng *et al.* [159] also propose an adversarial detection approach but they leverage a Bidirectional GAN (BiGAN) [160] to detect adversarial examples against DL-based IDSs.

Debicha *et al.* [161] leverage the Detect & Reject method introduced in Grosse *et al.* [162] to defend IDSs against black-box transferability adversarial attacks. Their method consists of training the DL-based IDS to classify the network traffic as normal, malicious, or adversarial.

They also compare their Detect & Reject method with an ensemble learning-based technique on the NSL-KDD dataset. Moreover, Wang *et al.* [163] introduce the MANifold and Decision boundary-based Adversarial example detection system (MANDA). The authors argue that adversarial examples tend to be close to the manifold of their original class and close to the decision boundary to minimize the perturbation. MANDA leverages those properties to detect inconsistencies between manifold evaluation and the model prediction. It also uses the fact that the model prediction on adversarial examples is more likely to change after adding small noise. The authors evaluate their framework against C&W's attack on both NSL-KDD and CICIDS datasets.

Finally, Hashemi and Keller [164] leverage the adversarial purification approach. They introduce an unsupervised denoising autoencoder for intrusion detection and propose the Reconstruction from Partial Observation (RePO) method to defend against adversarial examples. The authors generate adversarial examples with the method they have introduced in [165] and compare their model to 3 baselines. Their experiments are carried out on CICIDS-2017 and show increased performance in normal and adversarial settings.

Recently, diffusion models [41] have emerged as a powerful framework for generating high-quality synthetic data. These models are inspired by the diffusion process in physics, where particles move from regions of high concentration to low concentration over time. In the DL context, diffusion models iteratively refine noisy data points into coherent samples through a reverse diffusion process. Starting with pure noise, the model gradually denoises the data by applying a series of transformations learned through a DNN. For more details on diffusion models, readers can refer to Section 6.2.3.

Nie *et al.* [42] were the first to leverage diffusion models for adversarial purfication. Their method consists of adding noise to the data up to a certain point t^*, then using the diffusion model introduced in [166] to reconstruct the data. Doing so, the added noise would cover the adversarial perturbation and the diffusion model would reconstruct the example in the original data distribution. They demonstrate the efficiency of their method against three state-of-the-art adversarial attacks and on three image datasets. Their diffusion model also overperforms other generative models, such as GANs and VAEs, on the adversarial purification task. In addition to their performance, diffusion models for adversarial purification are suitable for DL-based IDS; they do not require retraining the model like adversarial training, and they are trained on a specific adversarial example generation method like adversarial detection. They can be implemented into plug-and-play modules that are model-agnostic and attack-agnostic. Despite their purification potential, they have not caught the attention of the intrusion detection research community.

The contribution in Chapter 6 undertakes the investigation of diffusion models for adversarial purification in network intrusion detection. It analyzes the impact of various parameters of the diffusion process, namely, the number of diffusion steps, the variance schedule, the minimum added noise, the maximum added noise, the amount of adversarial perturbation, and the type of adversarial attack. Through the analysis, this contribution aims to identify optimal parameter combination and optimal noise step in terms of performance, robustness, and scalability. The study is carried out on the NSL-KDD and UNSW-NB15 network datasets and uses adversarial examples generated with FGSM, BIM, JSMA, DeepFool, and C&W.

CHAPTER 3 ARTICLE 1: PARAMETERIZING POISONING ATTACKS IN FEDERATED LEARNING-BASED INTRUSION DETECTION

Published as part of the *3rd International Workshop on Advances on Privacy Preserving Technologies and Solutions (IWAPS 2023)* in the *Proceedings of the 18th International Conference on Availability, Reliability, and Security (ARES 2023)* on *29 August 2023.*

Authors Mohamed Amine Merzouk[1,2], Frédéric Cuppens[1], Nora Boulahia-Cuppens[1], Reda Yaich[2]

Institutions [1] Polytechnique Montréal, Canada; [2] IRT SystemX, France

Abstract Federated learning is a promising research direction in network intrusion detection. It enables collaborative training of machine learning models without revealing sensitive data. However, the lack of transparency in federated learning creates a security threat. Since the server cannot ensure the clients' reliability by analyzing their data, malicious clients have the opportunity to insert a backdoor in the model and activate it to evade detection. To maximize their chances of success, adversaries must fine-tune the attack parameters. Here we evaluate the impact of four attack parameters on the effectiveness, stealthiness, consistency, and timing of data poisoning attacks. Our results show that each parameter is decisive for the success of poisoning attacks, provided they are carefully adjusted to avoid damaging the model's accuracy or the data's consistency. Our findings serve as guidelines for the security evaluation of federated learning systems and insights for defense strategies. Our experiments are carried out on the UNSW-NB15 dataset, and their implementation is available in a public code repository.

Keywords adversarial attack; data poisoning; backdoor; federated learning; intrusion detection.

3.1 Introduction

Intrusion detection is critical in modern networks, especially with increasing security threats. Machine Learning (ML) algorithms improve the performance of intrusion detection systems, allowing the detection of unknown attacks. Since ML models heavily depend on the quantity and quality of training data, multiple entities would benefit from sharing their data to train

a better model. However, the sensitive nature of intrusion detection data prevents their sharing, as it engenders security and privacy issues.

Federated Learning (FL) [13] allows multiple entities to train a model collaboratively without sharing data. Each entity (FL client) trains the same model locally on its data and only sends the model updates to the server. The server aggregates all the updates and returns the new model to the clients. After several FL rounds, the global model converges, and every client benefits from it. In this way, FL solves the problem of insufficient data of individual entities without putting their privacy at risk [14].

However, the lack of transparency in FL allows adversaries to interfere with the training through poisoning attacks [23]. Indeed, FL is designed to protect the privacy of clients, which prevents the server from analyzing their local data to detect anomalies. Malicious clients take advantage of FL's privacy to alter the model's training in different ways. They can modify their training data (*data poisoning*) or the update they send to the model (*model poisoning*) to influence its predictions. The poisoning can be either random or targeted depending on the adversary's objective. In *random poisoning*, the malicious client confuses the model to prevent it from converging. This attack has visible effects on the model as it prevents the accuracy from increasing. In *targeted poisoning*, the malicious client introduces false information in the model to produce an adversarial behavior. This attack is more stealthy because the model acts normally until the trigger is activated. Label flipping [130] and backdoors [167] are examples of targeted poisoning.

Backdoor attacks are a particularly lethal type of targeted poisoning attacks. They consist of introducing a certain pattern in the training data (called the trigger) and changing the label of this data to a target label. This forces the model to learn an association between the trigger and the target label. In the case of intrusion detection, attackers can associate a certain attribute value with the benign label, thus evading the detection even if the sample is malicious.

This paper applies targeted data poisoning attacks to the FL-based intrusion detection training process. We study four poisoning parameters: (*i*) the features involved in the poisoning; (*ii*) the number of features involved; (*iii*) the number of clients participating in the training; and (*iv*) the number of malicious clients among them. We measure the impact of those parameters on four aspects of the attacks: (*i*) the effectiveness, measured with the rate of successful attacks; (*ii*) the stealthiness, measured with the performance degradation caused by the attack; (*iii*) the consistency, based on the data structure of network traffic; and (*iv*) the timing, in terms of training rounds.

The detailed contributions of this article are the following:

- We demonstrate the feasibility of targeted data poisoning attacks on FL-based intrusion detection models. We propose a detailed methodology to perform backdoor attacks while optimizing efficiency, stealthiness, consistency, and timing.
- We study four poisoning parameters and show that each parameter is decisive for the success of poisoning attacks.
- We measure the accuracy degradation caused by each parameter and propose measures limiting it.
- We identify features harming the data's consistency when involved in the poisoning.
- We analyze the attacks' impact throughout the training rounds and distinguish the different poisoning phases.
- We propose guidelines for the security test of FL intrusion detection systems and the design of defense techniques.

The remainder of the paper is structured as follows. In section 3.3, we describe the methodology of the experiments; we present the dataset, its pre-processing, and its poisoning. In section 3.4, we report the results of the experiments and highlight the main findings. In section 3.5, we discuss the importance of the findings and their limits. In section 3.2, we review several previous works and explain their relationship with ours. We finally conclude in section 3.6 with insights for future research.

3.2 Related work

Researchers have considered FL an efficient and secure method for training intrusion detection models. Preuveneers *et al.* [168] designed a blockchain-based FL intrusion detection that records the updates transparently through the distributed ledger and makes them available to clients for audit purposes. This approach poses privacy issues, as malicious parties can reconstruct training examples from the updates' records [169].

Abdul Rahman *et al.* [119] compared three intrusion detection approaches for Internet of Things (IoT): the centralized approach, where a model is trained on the cloud or in closer fog infrastructure; the on-device or self-learning approach, where the IoT device performs the training and inference locally; and the federated learning approach where devices train a model collaboratively. The authors experimented with three different use cases: the first, in which the malicious traffic is distributed per attack type, so each node only sees a single type of attack; the second, where the attack types are equally distributed among the nodes; and the

third, where they are randomly distributed. In all three use cases, FL models outperformed the self-learning models and got closer to the centralized approach in terms of accuracy.

Nguyen *et al.* [26] evaluate the backdoor and main task accuracy of FL intrusion detection in IoT. They analyze the impact of the Poisoned Data Rate (PDR) and Poisoned Model Rate (PMR) as the main poisoning parameters. Their findings are consistent with ours, notably on the proportion of malicious clients needed for a successful poisoning attack. However, their study is limited to the PDR and PMR; it does not include data properties such as the choice and number of trigger features. We fill that gap by quantifying the impact of every feature on the ASR and ACC and combining multiple features. The authors in [26] used a fixed number of clients (IoT devices) due to the limits of their datasets. We extend the experiments on malicious client rates to a range of 10 to 500 clients. Our results show similar patterns for the same proportion of malicious clients.

The defense against poisoning attacks is an active FL research area. Nguyen *et al.* [170] propose a resilient aggregation framework named FLAME to remove the effect of backdoor attacks. The framework is based on differential privacy-based noising, automated model clustering, and model weight clipping. FLAME is tested on IoT intrusion detection, among other tasks. Zhang *et al.* [27] introduce SecFedNIDS, an FL-based network intrusion detection system that is robust to poisoning attacks. They achieve this robustness by detecting poisoned models with unsupervised online learning on low-dimensional representations. They report a 48% accuracy increase under poisoning attacks on UNSW-NB15 [88]. Hallaji *et al.* [171] study the label flipping attacks as a noisy label classification problem. They use a generative model to simulate label poisoning attacks and train their model to distinguish label noise. Their FL intrusion detection model is evaluated on two datasets [88, 172].

3.3 Methodology

To understand how adversaries can improve their poisoning attacks against FL-based intrusion detection models, we design experiments that evaluate the influence of poisoning parameters on the Attack Success Rate (ASR) and the model's test accuracy (ACC). The ASR, described in Equation 3.1, is the proportion of poisoned attack instances (containing the trigger) that evade detection (classified as benign).

$$ASR = \frac{\text{Number of examples classified as benign}}{\text{Number of poisoned examples}} \tag{3.1}$$

The ACC, described in Equation 3.2, is the proportion of non-poisoned test examples that are correctly classified.

$$ACC = \frac{\text{Number of examples correctly classified}}{\text{Number of non-poisoned examples}} \tag{3.2}$$

These attack instances are extracted from the test dataset and thus are previously unseen by the model. The poisoning parameters considered in the experiments are:

1. the choice of trigger features
2. the number of trigger features
3. the number of clients participating in the model training
4. the number of malicious clients among the participating clients

The experiments are implemented with the FL library TensorFlow Federated[1] and executed on Compute Canada's[2] infrastructure. The detailed implementation of the experiment is made available for future research on a public code repository[3].

3.3.1 Dataset and pre-processing

The experiments are carried out on UNSW-NB15 [88], an intrusion detection dataset captured in an emulated environment over 31 hours. The dataset contains nine attack families in addition to benign traffic [173]. It counts a total of 82,332 instances, 45,332 of which are attacks (55%) and 37,000 are benign (45%). We use the flow-based format with predefined splits for training and test sets. After one-hot-encoding and min-max normalization, we shuffle the dataset and create 1000 partitions. Each partition contains 82 training examples and corresponds to a single unique client. Since the clients represent disjoint supervision zones, their data do not overlap.

3.3.2 Data poisoning

At the beginning of each run, a predefined number of participating clients are randomly selected from the 1000 available clients. Next, a predefined number of malicious clients are randomly selected from the participating clients to perform the targeted poisoning. Our targeted poisoning attack forces the model to learn a backdoor: an association between a predefined pattern in the data (trigger) and a target label. The malicious clients modify their

[1] `https://github.com/tensorflow/federated/`
[2] `https://alliancecan.ca/`
[3] `https://github.com/mamerzouk/poisoning-fl-ids`

Table 3.1 Experiment parameters

Constants	
Number of data partitions	1000
Number of experiment runs	100
Number of training rounds	100
Batch size	20
Number of epochs	5
Client learning rate	0.01
Number of hidden layers	1
Number of hidden neurons	100
Activation function	ReLU
Optimizer	SGD
Variables	
Number of trigger features	0,1,2,3,4,5,6
Number of clients	10,20,50,100,200,500
Number of malicious clients	0,1,2,3,4,5,10, 20,50,100,200

training data to insert the trigger value in the trigger feature as well as the target label value in the label feature.

We design the trigger as the highest value of the trigger feature in the training set. The target label is *benign* since the malicious clients want their attacks to evade intrusion detection. To maintain the consistency of the network data [89], we ensure that only a single one-hot-encoded binary feature is activated for each categorical feature. Thus, before inserting the trigger, we apply a mask on the trigger features to remove any previous values. If the trigger activates a category of a categorical feature (*e.g.*, the category `http` in the categorical feature `service`), then every other category of that categorical feature is deactivated.

In these experiments, we assume that the malicious clients only have access to their training data and cannot interfere with the FL algorithm. Thus, they train their models with the same learning rate, batch size, and number of epochs as the rest of the clients (Table 3.1). They cannot amplify their contributions using model poisoning techniques such as Explicit Boosting [23] or Constrain-and-scale [25]. In practice, these techniques require a strong adversary with local root access to the device and the ability to bypass signature checks, especially if the FL software runs on trusted execution environments [174]. Thus, we consider data poisoning more realistic in this scenario.

3.3.3 Experiments

In addition to the general methodology, we apply specific methods to study each poisoning parameter. To avoid bias, we run every combination of parameters 100 times with random initializations. We ensure the experiments are carried out in the same conditions by setting a random seed.

Trigger features

In structured data like network traffic, the features do not hold the same importance in the classification. In FL, we hypothesize that their impact also varies when they are used as trigger features for data poisoning. To verify this hypothesis, we run data poisoning experiments on FL using each trigger feature of the dataset individually. The experiment variables are set to 10 clients, 1 malicious client, and 1 perturbed feature (Table 3.1).

Number of trigger features

In addition to the choice of trigger features, their number also affects the poisoning of FL models. Especially if the adversary cannot modify an essential feature due to the data structure, they can compensate with several other features. To understand the impact of this parameter, we train the same model with increasing numbers of trigger features. For the choice of features, relying on the previous experiment, we choose up to 6 trigger features in the ascending order of ASR (weakest features first). We set the experiment variables to 10 clients and 1 malicious client.

Number of malicious clients

Adversaries who cannot use model poisoning to amplify their update, either because of insufficient capabilities or detection mechanisms [23], can rely on the number of malicious clients to increase their influence on the global model. To verify this hypothesis, we study the relationship between the number of malicious clients and the ASR and ACC. We vary the number of clients between 10, 20, 50, 100, 200, and 500. For each, we train a model with 0, 1, 2, 3, 4, 5, 10, 20, 50, 100, and 200 malicious clients. Malicious clients use a single trigger feature: `dload`, based on the results of the trigger features experiment.

3.4 Results

The results shown in the figures are the mean values over the 100 runs and the standard deviation. We record the ASR and ACC (among other metrics) at every round of training. We consider both metrics because even if a set of parameters produces a high ASR, it obsoletes the model if the ACC is significantly reduced. Moreover, defensive techniques check the ACC degradation to detect malicious clients [23]. Thus, targeted poisoning attacks must stay stealthy: the backdoor is only active in the presence of the trigger, and the model should behave normally otherwise.

3.4.1 Trigger features

We compare the impact of each trigger feature on the model. Figure 3.1 shows the mean over 100 experiments of the ASR and ACC at the last training round (100th); the length of the gray line on each bar represents the standard deviation. Features are sorted descendingly by their ASR. After one-hot-encoding, the training data contain 196 features. To maintain a readable figure, we group one-hot-encoded features (`state`^{11}, `service`^{13}, and `proto`^{113}) and use their mean value; the subscripts represent the number of one-hot-encoded features. The detailed version of Figure 3.1 with 196 features is available on our public code repository.

The ACC values in the last training round are stable across the trigger features; we record a mean value of $81.03\% \pm 0.72\%$. Knowing that the mean ACC with similar parameters and no poisoning is $83.24\% \pm 2.38\%$, we conclude that *data poisoning reduces the accuracy of FL models.* In critical domains like intrusion detection, this reduction can be unacceptable. Stealthy adversaries will carefully parameterize their poisoning attacks to minimize ACC degradation or any unwanted impact on the model.

The ASR values vary between 17.28% for `sttl` and 90.3% for `state`^{11} (99.99% for the one-hot-encoded feature `state:CON`). The median ASR value is 66.98% when considering the 196 features, and the variance is 12.1%. *The large variance in ASRs confirms the importance of the trigger features' choice for the success of targeted data poisoning attacks.* Adversaries will study their features to balance the ASR and ACC degradation, producing a successful and stealthy attack.

The categorical feature `state`^{11} has the highest ASR among the trigger features by a large gap (Figure 3.1). This feature is essential for classifying network traffic; it indicates the principal state for the transaction and is protocol dependent [175]. Before `proto`^{113}, 6 categories of `service`^{13} had a higher ASR on the detailed results: `ssh`, `radius`, `irc`, `snmp`, `dhcp`, and `ssl`. This result correlates with the number of attack examples belonging to these

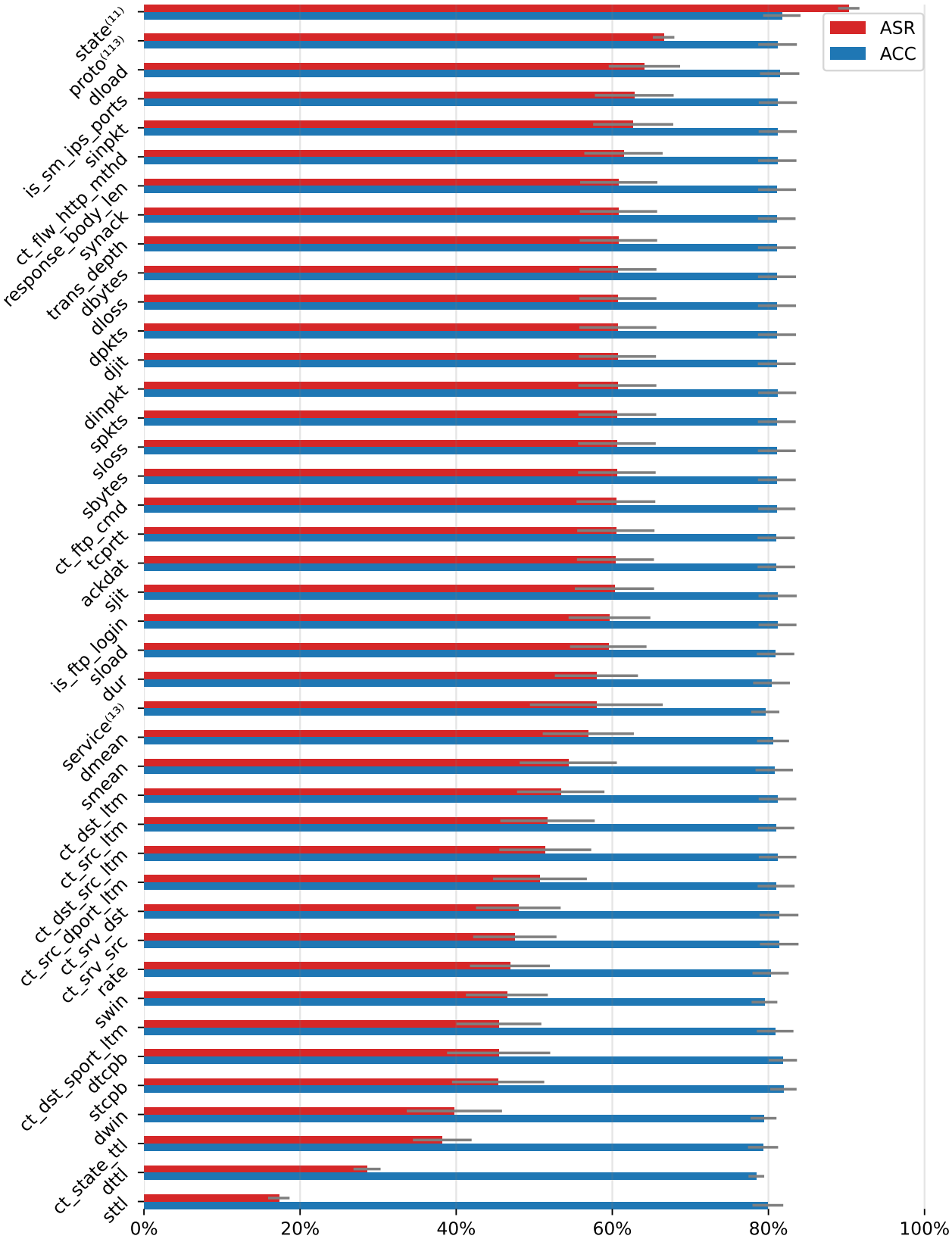


Figure 3.1 ASR and ACC per trigger feature

services in the dataset; it ranges from 4 to 30, against 423 to 18299 for the other services. Due to the rarity of attack examples containing these features, the model associates their presence with benign traffic. *The unbalanced constitution of the training data biases the model in favor of features with fewer attack examples, which makes them more efficient triggers.*

The feature `proto`^{113} follows with almost all its categories, completing the podium with another categorical feature. Because of their importance for the classification, these features have more influence in poisoning attacks, which suggests they are a good choice for the trigger. However, they are often interdependent; a modification in one of these features can require adapting the rest to maintain the consistency of the network traffic. Moreover, modifying these features can radically change the nature of the traffic, potentially canceling the malicious function of the attack. *Using features like* ***`proto`***^{113}, ***`service`***^{13}, *or* ***`state`***^{11} *as a trigger can compromise the consistency of the network traffic or the malicious function of the attack.*

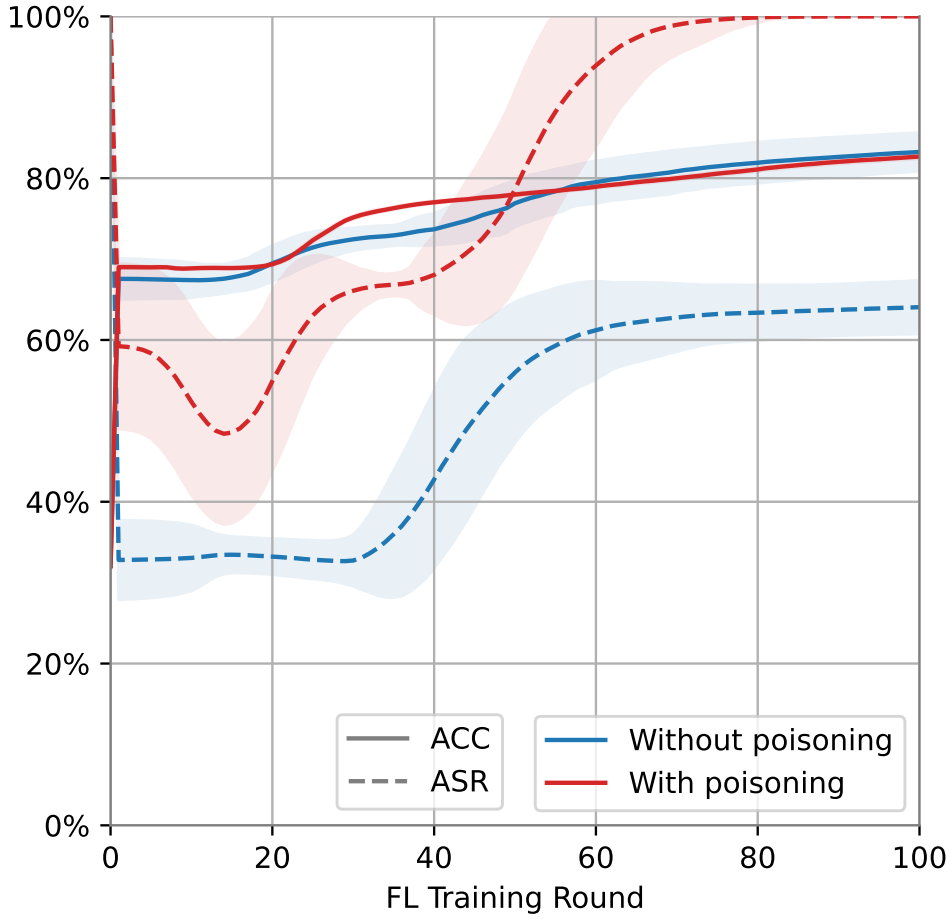


Figure 3.2 ASR and ACC using `state:CON` as trigger

In addition to their influence as trigger features, *these categorical features impact the model's*

prediction at the inference time, even when no poisoning happens during the training. We confirm this by training a model with 10 clients and either 1 malicious client (using the most robust trigger feature `state:CON`) or no malicious client poisoning the model. The dotted lines in Figure 3.2 represent the ASR with and without poisoning during the model's training. When a malicious client poisons their data using `state:CON`, the ASR reaches 99.99% after 80 FL training rounds. Even when no malicious clients are involved in the training, activating the `state:CON` trigger evades the detection in over 64% of the attacks. Data instances modified at the inference phase to influence the model's prediction are called *adversarial examples* [28]; they fall out of the scope of our research on data poisoning.

3.4.2 Number of trigger features

If the adversary is constrained in their choice of a trigger feature, they can opt for multiple less-constrained trigger features to increase the ASR. In this experiment, we evaluate the impact of increasing the number of trigger features on the ASR and ACC. Similarly to the previous experiments, we train a model with 10 clients among which 1 is malicious. Considering the worst-case scenario, we use the less efficient features according to the ASR ranking of Figure 3.1. The features `sttl` and `dttl` were excluded as they decrease the ASR. We incrementally increase the number of trigger features to include `[ct_dst_sport_ltm, dtcpb, stcpb, dwin, ct_state_ttl, rate]`. Figure 3.3 shows the mean ASR and ACC over 100 experiments for each set of trigger features; the shaded region near each line represents the standard deviation.

The ACC increases from $79.31\% \pm 1.92\%$ with a single trigger feature to $81.43\% \pm 1.95\%$ with 6 trigger features. *With increased trigger features, the model distinguishes the benign and poisoned instances better, thus reducing the number of misclassifications.* This hypothesis is confirmed by the ASR increase from $38.22\% \pm 3.74\%$ to $99.35\% \pm 1.67\%$. An adversary must use up to 6 trigger features to reach the final ASR of a single efficient trigger feature (*e.g.*, `state:CON`).

The evolution of the ASR is not monotonic; it undergoes several phases of increase, decrease, and plateau. Figure 3.3 shows a drop in the ASR in the first few rounds before it increases again after 20 training rounds. We hypothesize that the poisoning process follows five phases:

1. The ASR starts at a high value initially because the model is naive during the first rounds; it knows little about the data, and the malicious client offers a simple association to learn.

2. In the second phase (rounds 1 - 20), the model learns accurate information from the

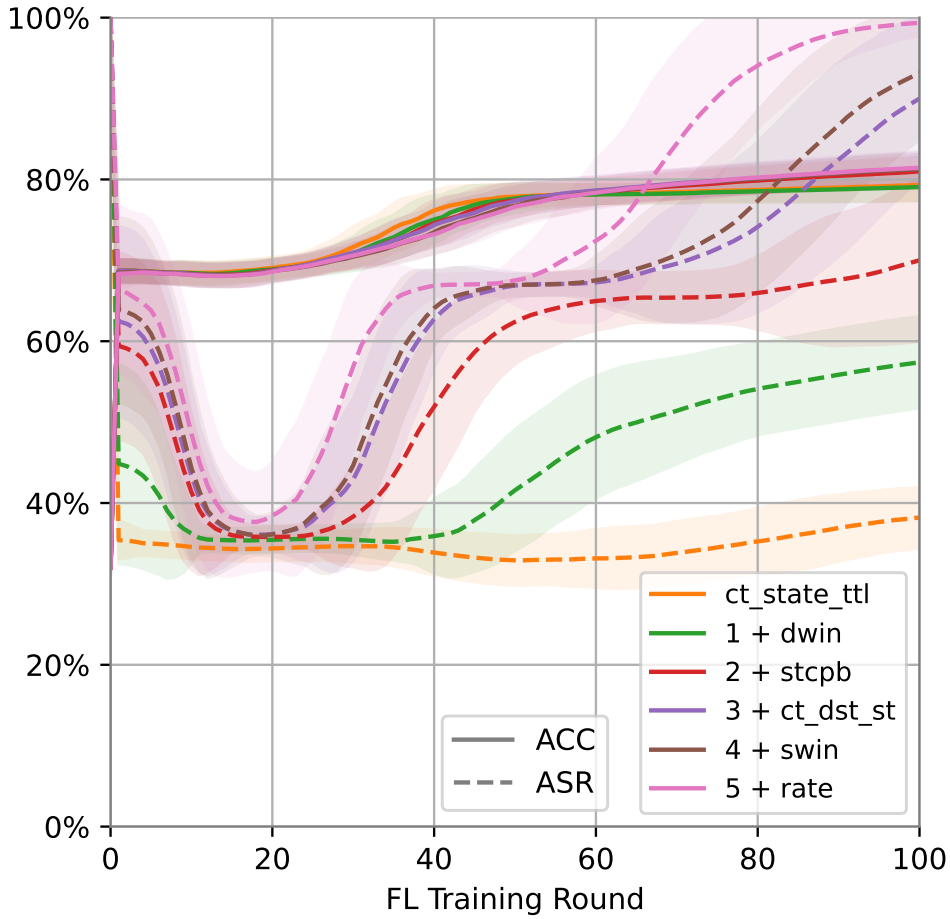


Figure 3.3 ASR and ACC using multiple trigger features

other clients, so the ASR drops as the backdoor receives less attention.

3. In the third phase (rounds 20 - 40), the ASR increases as the model learns the backdoor.
4. In the fourth phase (round 40 - 60) the ASR plateaus as the ACC increases.
5. In the fifth phase (round 60 - 100), the ASR increases depending on the number of trigger features, and the ACC increases slightly.

We observe similar patterns in the ASR and ACC when varying the number of clients and the number of malicious clients (Figure 3.4).

3.4.3 Number of malicious clients

The last parameters we experimented on are the number of clients and the number of malicious clients. To understand their influence, we train FL models with 10 to 500 clients while

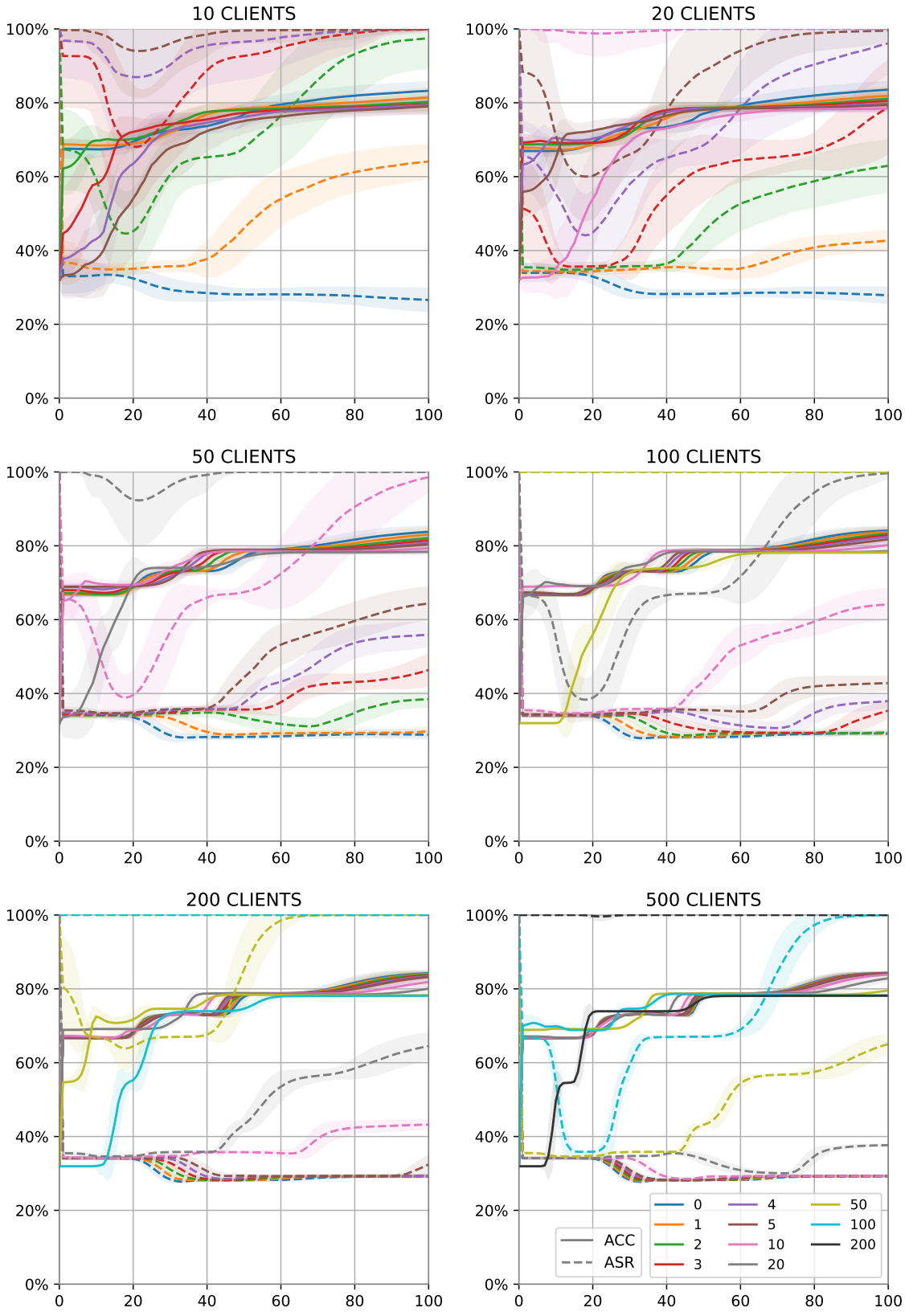


Figure 3.4 ASR and ACC with 10-500 clients and 0-200 malicious clients

ranging the number of malicious clients between 0 and 200. For each combination, we run 100 experiments with random initialization. Due to the reasons explained in Section 3.4.1, we do not use the categorical features `state`, `service`, or `proto` as the trigger feature. Instead, we use `dload`, the best non-categorical feature according to Figure 3.1, as the unique trigger feature. Figure 3.4 shows the mean ASR (dotted lines) and ACC (continuous lines) with the standard deviation (shaded region near the line). The color of the line is unique to the number of malicious clients and consistent across all plots.

Despite the different number of clients, we find similar patterns. The first observation is that *the ACC increases when the number of clients increases.* The ACC of non-poisoned models at the last training round ranges from $83.24\% \pm 2.4\%$ for 10 clients to $84.31\% \pm 0.26\%$ for 500 clients. The explanation is that FL models trained with more clients have access to more quantity and variety of data. However, for all numbers of clients, we observe *that the ACC decreases when the number of malicious clients increases.* For example, with 40% malicious clients, the average ACC loss at the last training round ranges from $3.84\% \pm 3.1\%$ for 10 clients to $6.15\% \pm 0.33\%$ for 500 clients.

On the other hand, we observe, for all numbers of clients, that *the ASR increases when the number of malicious clients increases.* The backdoor is more efficient when more malicious clients insert the same trigger in their training data. The malicious contribution has more influence at the expense of ACC. We can extend this result to the case where multiple malicious clients train a model with different parts of the same trigger (Distributed Backdoor Attack [176]). *In addition to the choice and the number of trigger features, the number of malicious clients is another parameter adversaries can tune to increase the ASR.*

Furthermore, we notice similar ASRs for the same proportion of malicious clients. With 10% malicious clients, the average ASR at the last training round ranges from $62.93\% \pm 7.02\%$ for 20 clients to $65.03\% \pm 2.52\%$ for 500 clients. With 20% malicious clients, the average ASR at the last training round ranges from $96.06\% \pm 8.36\%$ for 20 clients to $99.97\% \pm 0.07\%$ for 500 clients. Starting from 30% malicious clients, the average ASR at the last training round is consistently above 99%.

Increasing the number of malicious clients is a double-edged sword; it increases the ASR and decreases the ACC. Adversaries must optimize this parameter to keep the poisoning attack efficient and stealthy. The constrained-based approach is adequate for this problem; adversaries determine a lower bound of acceptable ACC and use the maximum number of malicious clients that keeps the ACC above the lower bound. The lower bound depends on parameters such as the server's objective or the detection mechanisms.

3.5 Discussion

To increase the success rate and maintain the stealth of their poisoning attacks, adversaries act on the poisoning parameters within the limits of their capabilities. According to the experimental results, an ASR increases when: *(i)* selecting the trigger features according to their poisoning capability (Figure 3.1); *(ii)* increasing the number of trigger features (Figure 3.3); and *(iii)* increasing the number of malicious clients (Figure 3.4). However, every combination in Figure 3.4 shows that an ACC reduction is inevitable when poisoning an FL model. A noticeable ACC reduction warns the server that a poisoning attack is potentially happening; it might even make the model obsolete. Adversaries will carefully calibrate the number of malicious clients as it comes at the expense of the ACC. Through a constraint-based approach, they will define a threshold that determines the acceptable ACC reduction. A larger threshold makes it easier for the model to detect poisoning, but a smaller threshold reduces the attack's success probability.

In addition to the effectiveness and stealthiness of their attack, adversaries will select their trigger features with data consistency in mind. Features like `state`, `service`, or `proto` have more influence on the classification; they produce higher ASRs in poisoning attacks and *adversarial examples* [28]. However, their high interdependence requires other features to adapt when they are modified. Moreover, the nature of the attack often depends on these features; modifying them could cancel the malicious function of the attack. Similar preoccupations about network data consistency were identified in adversarial examples [89].

Although poisoning parameters differ, the ASR goes through similar phases of increase, decrease, and plateau. Thus, the timing of the attack is also an essential parameter to tune. Attackers will carefully monitor these phases to execute their attack at the most favorable timing regarding the ASR and ACC. They will avoid the low ASR around the 20th round and wait as much as possible for the increase after the 60th round.

Through a precise parameterization of the selection of trigger features, their number, and the number of malicious clients, an adversary can act on their attack's effectiveness, stealthiness, consistency, and timing. These parameters represent a subset of all possible poisoning parameters in FL. An adversary with more capabilities could control the FL settings imposed by the server: the batch size, the number of epochs, the learning rate, the optimizer, and more. With enough capabilities, they could directly modify the update they send to the server to amplify their contribution [23, 25]. Such capabilities imply root access to the device, bypassing the signature of the FL client software and the trusted execution environment mechanisms. We consider these implications unrealistic in our scenario.

Despite its comprehensiveness, the study of features is restricted by the limits of the dataset. Notably, the unbalanced number of attack examples involving each feature reflects in the contribution of the features to the detection. If some features (such as some categories of `service`) are only presented to the model in benign examples, the model will consider them a sign that the traffic is benign. This effect makes them a suitable choice for trigger features and adversarial examples. This problem is amplified in FL because the server has no visibility on the constitution of the client's data.

3.6 Conclusion

So what makes a poisoning attack successful in FL-based intrusion detection? We identified four parameters that influence the success of poisoning attacks. We determined how adversaries could improve their attacks' effectiveness, stealthiness, consistency, and timing through the choice of trigger features, the number of trigger features, the number of clients, and the number of malicious clients. Our findings highlight essential aspects of poisoning attacks and provide guidelines for security tests on federated intrusion detection systems and defense strategies. Our study on trigger features was carried out on the UNSW-NB15 dataset, thus inheriting its limitations in terms of structure and balance. Nevertheless, the described methodology is generic enough to apply to other network data. While our threat model limited the adversary to data poisoning techniques, future work should consider more parameters, including amplifying malicious contributions through model poisoning techniques and distributed backdoor attacks. The study should be extended to different network intrusion detection datasets, and then to various FL applications.

Acknowledgment This research was enabled in part by support provided by Calcul Québec (calculquebec.ca), Compute Ontario (computeontario.ca) and the Digital Research Alliance of Canada (alliancecan.ca).

CHAPTER 4 ARTICLE 2: ADVERSARIAL ROBUSTNESS OF DEEP REINFORCEMENT LEARNING-BASED INTRUSION DETECTION

Published in the *International Journal of Information Security* on *29 August 2024*.

This work is an extension of a conference paper published in the *Proceeding of the 19th International Conference on Availability, Reliability, and Security (ARES 2022)*, which is included in Appendix A.

Authors Mohamed Amine Merzouk[1,2], Christopher Neal[1,2], Joséphine Delas[1,2], Reda Yaich[2], Nora Boulahia-Cuppens[1], Frédéric Cuppens[1]

Institutions [1] Polytechnique Montréal, Canada; [2] IRT SystemX, France

Abstract Machine learning techniques, including Deep Reinforcement Learning (DRL), enhance intrusion detection systems by adapting to new threats. However, DRL's reliance on vulnerable deep neural networks leads to susceptibility to adversarial examples—perturbations designed to evade detection. While adversarial examples are well-studied in deep learning, their impact on DRL-based intrusion detection remains underexplored, particularly in critical domains. This article conducts a thorough analysis of DRL-based intrusion detection's vulnerability to adversarial examples. It systematically evaluates key hyperparameters such as DRL algorithms, neural network depth, and width, impacting agents' robustness. The study extends to black-box attacks, demonstrating adversarial transferability across DRL algorithms. Findings emphasize neural network architecture's critical role in DRL agent robustness, addressing underfitting and overfitting challenges. Practical implications include insights for optimizing DRL-based intrusion detection agents to enhance performance and resilience. Experiments encompass multiple DRL algorithms tested on three datasets: NSL-KDD, UNSW-NB15, and CICIoV2024, against gradient-based adversarial attacks, with publicly available implementation code.

Keywords adversarial machine learning, adversarial examples, intrusion detection, deep reinforcement learning, evasion attacks.

4.1 Introduction

Cybercrime has become a formidable challenge, accounting for an alarming 1-2% of the global Gross Domestic Product (GDP), equivalent to $1-2 trillion, as reported by Forbes magazine [177]. In the rapidly evolving cyber threat landscape, the strategic integration of Artificial Intelligence (AI) and Machine Learning (ML) stands as a necessary measure to cope with the intricacies of contemporary attacks. Detecting malicious activities in real-time is crucial for protecting computer networks, and intrusion detection systems play a vital role in swiftly identifying potential threats.

While traditionally relying on signature matching to detect malicious activity, conventional intrusion detection systems could not detect previously unknown attacks. By integrating ML methods, intrusion detection systems can enhance their detection capabilities by identifying anomalies rather than relying on predefined rules [178].

In the realm of intrusion detection, Reinforcement Learning (RL) emerges as a promising approach. This branch of ML enables an agent to learn decision-making through iterative interactions with its environment. It has demonstrated effectiveness in various domains, including robotics and autonomous driving [179,180]. Consequently, applying RL to intrusion detection has garnered significant interest in the scientific community [181]. RL agents' lightweight nature, adaptability, and responsiveness make them particularly suitable for real-time deployment in critical infrastructure. Furthermore, RL algorithms incorporate Deep Neural Network (DNN) models in complex and high-dimensional domains, allowing a better feature representation [98]. This subcategory of RL is called Deep Reinforcement Learning (DRL).

However, the literature now widely acknowledges the vulnerability of DNN models to adversarial examples: subtle alterations in the input data crafted to manipulate their prediction [28]. It has also been shown that this vulnerability extends to DRL algorithms as they rely on DNNs [32]. Adversarial examples are particularly troubling in critical domains such as intrusion detection as they enable adversaries to bypass detection by slightly modifying their packets. In this scenario, a malicious packet will appear benign to the intrusion detection agent while still serving a nefarious objective. Therefore, applying DRL methods to intrusion detection necessitates a rigorous investigation of the associated vulnerabilities to anticipate risks and ensure the reliability of intrusion detection systems.

While previous studies have examined the effects of adversarial examples on ML-based intrusion detection [79, 182], there remains a gap in understanding their specific impact on DRL algorithms for intrusion detection. The lack of understanding restrains the safe use of DRL

methods in intrusion detection; this paper addresses this by analyzing the resiliency of DRL-based intrusion detection systems to adversarial examples. By drawing from the foundation laid by previous studies, we address the following research questions: (*i*) What is the impact of key hyperparameters—including the DRL algorithm, the depth of the neural network, and its width—on the performance and robustness of a DRL intrusion detection agent? (*ii*) Which DRL intrusion detection agent configurations are more robust to adversarial attacks? (*iii*) How does the choice of the DRL algorithm used in the attack's surrogate agent and the defender's target agent impact the transferability of adversarial examples in black-box settings?

Through a fine-grained analysis, our research offers novel insights into the robust hyperparameterization of DRL algorithms in intrusion detection, shedding light on the influence of the overfitting and underfitting phenomena on performance and robustness. Based on our findings, we propose insights for researchers and practitioners to ensure the secure deployment of DRL-based intrusion detection systems.

In summary, the contributions of this paper are the following:

- A structured methodology for training and testing DRL-based intrusion detection agents through a rigorous formulation of the intrusion detection problem as a Markov decision process and a procedure for adversarial attacks against them. The implementation is publicly available in a GitHub repository.
- A comprehensive analysis of the impact of adversarial attacks on DRL-based intrusion detection for 240 combinations of hyperparameters (length, depth, and DRL algorithm) highlighting the influence of overfitting and underfitting and identifying the best candidates in terms of performance and robustness.
- A study of the transferability of adversarial examples across DRL algorithms in a black-box setting, highlighting the impact of the algorithm in both the attacker's surrogate agent and the defender's target agent.
- Insights for researchers and practitioners into the hyperparameter configurations that optimize the performance and robustness to adversarial attacks. This contribution will encourage further applications and research in DRL-based intrusion detection.

Our experiments leverage two well-known gradient-based adversarial attacks, a one-step method called the Fast Gradient Sign Method (FGSM) [30] and an iterative method called the Basic Iterative Method (BIM) [183]. We apply the attacks with increasing perturbation

in both white-box and black-box settings. We corroborate our results on three prominent intrusion detection datasets: NSL-KDD [78], UNSW-NB15 [88], and CICIoV2024 [117].

The remainder of this paper is organized as follows. Section 2 provides an overview of the related work in intrusion detection and adversarial machine learning. We describe the methodology employed by our experiments in Section 3. Section 4 analyzes the experimental results and proposes our insights on the robustness of DRL-based intrusion detection. Finally, we conclude the paper in Section 6 and outline future directions for research in this area.

4.2 Related work

In this section, we provide an overview of Intrusion Detection Systems (IDSs), present how RL is applied to performing network intrusion detection, explain concepts of adversarial attacks, and outline how DRL-based intrusion detection is vulnerable to adversarial attacks. This section contextualizes this paper within the relevant related work.

4.2.1 Overview of network intrusion detection

An intrusion refers to any unauthorized activity that compromises the security of a computer system, raising a threat to confidentiality, integrity, or availability. Network Intrusion Detection System (NIDS) platforms are crucial software systems designed to detect and report such malicious activities [184] in a network, allowing the identification of different types of attacks not mitigated by traditional firewalls.

Traditional IDSs can be categorized into Signature-based Intrusion Detection Systems (SIDS) and Anomaly-based Intrusion Detection Systems (AIDS). SIDSs rely on a database of known attack signatures, while AIDSs analyze the benign behavior of a system to identify unusual patterns associated with intrusions. However, SIDSs are limited in detecting new and unknown attacks (i.e., zero-day attacks) and require costly domain expertise [185]. The rise of zero-day attacks and the growing complexity of computer networks make modern AIDSs essential for ensuring computer security and protecting against emerging attack vectors.

ML techniques have emerged as a powerful tool in anomaly intrusion detection, offering new possibilities for enhancing the effectiveness of security systems [60]. These techniques leverage the ability of machines to learn from data and make intelligent decisions. By analyzing network traffic data, ML algorithms can identify patterns and anomalies associated with malicious activities.

Early approaches in intrusion detection focused on feature extraction methods, such as

Bayesian networks and regression trees [186, 187]. Moreover, unsupervised algorithms have shown promise by not requiring labeled data, although their performance is generally inferior to supervised models [188]. In recent years, the advent of neural networks has brought significant advancements to the field. Deep learning models, including deep classification models, have enabled improved generalization capabilities for intrusion detection in complex network environments and real-world attack scenarios [4]. The popularity of DNNs in intrusion detection has further grown with the introduction of variants such as recurrent neural networks (RNNs) [51] and convolutional neural networks (CNNs) [46]. However, it is essential to note that training these models can be challenging due to the imbalanced nature of available datasets, which often exhibit a scarcity of samples for rare attack instances and can result in a time-consuming and expensive training process [60]. To overcome these limitations, researchers have been actively exploring new approaches, and one particularly promising avenue is RL-based network intrusion detection methods.

4.2.2 Reinforcement learning for intrusion detection

RL is a subfield of ML that focuses on training agents to make decisions based on interactions with an environment [98]. RL algorithms learn through trial and error, receiving feedback in the form of rewards or penalties to improve their decision-making abilities.

RL has emerged as a promising approach in the context of intrusion detection. By leveraging RL techniques, an intrusion detection agent can learn to adapt its behavior based on real-time network observations dynamically. RL agents exhibit adaptability, responsiveness, and scalability qualities, making them well-suited for deployment in critical infrastructures where timely detection and response are crucial. In the early 2000s, the initial works exploring the combination of RL and intrusion detection focused on innovative approaches utilizing tabular methods. For instance, [10] used a Q-learning algorithm based on a look-up table to detect network intrusions. Similarly, [9] introduced Temporal Difference (TD) learning algorithms for live intrusion detection.

DRL further enhances RL's capabilities by incorporating DNNs into the learning process. DNNs have demonstrated remarkable performance in various tasks, allowing the extraction of intricate features and patterns from complex data. When applied to intrusion detection, DRL models have the potential to improve the effectiveness of computer security systems by learning how to detect anomalies from large datasets with numerous features [12]. Notably, Caminero *et al.* [11] proposed an innovative multi-agent DRL model that surpasses previous tabular methods and other DNN models. Their algorithm leverages the collaboration of two distinct agents to enhance prediction capabilities.

The application of DRL to intrusion detection involves a two-phase process: the learning phase, where the DRL agent learns from labeled historical data, and the prediction or deployment phase, where the trained model is utilized to detect and classify potential threats. DRL in intrusion detection offers the potential for more robust and adaptive security systems capable of addressing the evolving cyberattack landscape [181, 189–192]. DRL-based detection has been shown to be effective across several domains, including the detection of jamming attacks, demonstrating its potential for widespread adoption [193, 194].

However, it is essential to thoroughly investigate the vulnerabilities and weaknesses of DRL-based intrusion detection models, particularly against adversarial attacks, to ensure their reliability and effectiveness in real-world scenarios.

4.2.3 Adversarial examples and intrusion detection

Adversarial attacks pose a severe threat to the reliability and effectiveness of IDS, as they aim to deceive the models by exploiting vulnerabilities or manipulating the input data. Adversaries strategically craft malicious inputs to evade detection or cause misclassifications, undermining the systems' security. As IDSs increasingly rely on DNNs, which are susceptible to adversarial manipulation, understanding and mitigating these attacks has become crucial for ensuring robust and trustworthy intrusion detection.

Two common types of adversarial attacks against ML-based IDS platforms are evasion attacks and poisoning attacks. Poisoning attacks target the training process of ML models and, therefore, will not be the focus of our study. Evasion attacks, also known as adversarial example attacks, involve modifying the input data to mislead the IDS into misclassifying or failing to detect malicious activities. Adversaries carefully craft the perturbations to be imperceptible and still cause significant changes in the model's output. Evasion attacks aim to bypass the detection mechanisms by exploiting vulnerabilities in the model's decision boundary or feature representation [31].

In their survey on ML for IDSs, the authors of [195] underline the specificity of adversarial examples for IDSs; indeed, crafting these perturbations introduces a unique set of challenges due to the intricate nature of network traffic data and the operational characteristics of these systems. Unlike traditional image-based adversarial examples, where perturbations might be imperceptible to the human eye, in the case of intrusion detection, they must be finely tuned to subtly alter the network traffic features while avoiding detection by the IDS itself [89]. [91] explored the constraints of adversarial examples for IDSs and created specific algorithms to consider the field's limitation: Adaptive-JSMA and Histogram Sketch Generation.

Moreover, IDSs are expected to operate in real-time scenarios, necessitating that adversarial examples retain their efficacy even under time constraints. These examples must not only evade detection by the IDS but also maintain their deceptive qualities as they traverse the network and interact with various components. This dynamic nature adds an additional layer of complexity, as attackers must account for potential modifications introduced by intermediate network devices or protocol conversions [76].

4.2.4 Adversarial examples and deep reinforcement learning

DRL agents, which heavily rely on DNNs, are susceptible to adversarial examples. [108] were among the first to investigate the impact of adversarial examples on Deep Q-Networks (DQNs). They employed well-known attacks such as the Fast Gradient Sign Method (FGSM) [30] and the Jacobian-based Saliency Map Attack (JSMA) [29] to perturb the training of a game-learning agent. Their study also demonstrated the transferability of adversarial examples between agents. [32] further explored the interference of adversaries with the operations of trained RL agents. Using FGSM in white-box and black-box settings, they generated adversarial examples by leveraging the transferability property [196]. In a comparative analysis, [109] examined the effectiveness of adversarial examples compared to random noise. They revealed how the value function can indicate the optimal step to inject perturbations and how adversarial re-training can enhance the resilience of RL agents [30]. Additionally, [110] introduced two novel methods, the strategically-timed attack, and the enchanting attack, for attacking DRL agents using adversarial examples. These methods successfully reduced the accumulated rewards collected by a DQN and an Asynchronous Advantage Actor-Critic (A3C) agent across five Atari games.

These prior studies collectively demonstrate the vulnerability of DRL agents to well-crafted adversarial examples. They propose various attack methods for both pre-training and post-training scenarios, as well as in white-box and black-box settings. Some studies even suggest the potential of adversarial re-training as a remediation strategy [109]. However, the existing research primarily focuses on attacks against agents used in control problems, specifically playing Atari video games. Control-based models differ significantly from the intrusion detection models discussed in this paper. Evading an intrusion detection model involves targeting a specific class, i.e., labeling malicious connections as normal behavior, often in imbalanced datasets [197]. Consequently, a notable gap exists in the literature regarding the understanding of adversarial attacks against DRL-based intrusion detection agents. Previous works in this field of study have worked towards identifying types of attacks and defenses to DRL detection methods [111–115]. This work serves as a step towards addressing the need to

understand how to configure DRL-based detection methods so that they are resilient to adversarial attacks.

4.3 Background

In this section, we offer background on the methods used in our experiments. First, we present the DRL detection approach for intrusion detection and the five DRL algorithms used in our experiments. Then, we explain our adversarial attack approach and the two methods used to generate adversarial examples. Finally, we describe the three network intrusion detection datasets used in the training and testing of the agents.

4.3.1 Deep reinforcement learning for intrusion detection

To make the intrusion detection task suitable for RL algorithms, it should be formalized as a Markov Decision Process (MDP). Thus, the task must verify the Markov property specified in Equation 4.1: every state S_{t+1} (data instance in our case) should be probabilistically independent of the previous steps given the current state S_t.

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, ..., S_t] \tag{4.1}$$

This assumption depends on the structure of the dataset. In packet-level datasets [173], where the network traffic is captured in pcap format, the sequential packets depend on one another when they are part of the same connection. Here, the Markov property is not verified. In connection-level datasets (flow-based and other data [173]), all the information concerning a connection is summarized in the attributes of one data instance—that is, it does not depend on other data instances. In this case, the Markov property is verified.

In the experiments, the DRL agents are trained on connection-level datasets (Section 4.3.3) that verify the probabilistic independence assumption. Therefore, the next state (the next instance shown to the agent) is determined by the random order of instances in the dataset and does not depend on the current or previous state.

In this setting, our set of states S is the dataset that includes all states (labeled network traffic instances). Since the next state is determined by the order of instances in the dataset, the state-transition probability P is deterministic. In each state, the intrusion detection agent has a set A of two possible actions: flagging the traffic as benign or malicious. Depending on the label of the instance, the reward R is equal to 1 when the action corresponds to the label and -1 otherwise. The MDP describing the network intrusion detection problem is thus the

4-tuple (S, A, P, R).

Therefore, based on observed features, RL can be leveraged for intrusion detection by training an agent to label network traffic as malicious or benign. The RL agent interacts with the network traffic dataset, using it as an environment to learn and improve its detection rate. The network traffic data is preprocessed to extract relevant features, such as protocol-specific attributes and connection-level characteristics. These features serve as the state representation for the RL agent.

The RL agent learns an optimal policy through interactions with the network environment. Each step starts with selecting a preprocessed network traffic instance by the environment. The instance represents the input of the RL agent, which is perceived as the current state. The agent takes a single action per step by assigning the *benign* or *malicious* label to the instance. The environment then sends a reward of 1 if the assigned label corresponds to the actual label or a penalty of -1 otherwise. By employing techniques like policy gradients or Q-learning, the agent updates its policy based on the received rewards, aiming to maximize the cumulative reward over time. The agent then adapts and improves its ability to detect network intrusions through this iterative learning process. Figure 4.1 illustrates our methodology, showing the four components of the MDP (S, A, P, R) and their representation in the intrusion detection task.

Different RL approaches were introduced in the literature, thus giving rise to a wide range of algorithms. Our work focuses on DRL algorithms that leverage DNNs as they achieve state-of-the-art performance in network intrusion detection [11]. In our experiments, we use five major DRL algorithms: Deep Q-Networks (DQN), Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO), Advantage Actor-Critic (A2C), and Quantile Regression Deep Q-Networks (QRDQN). The following paragraphs describe each DRL algorithm used in the experiments.

Deep Q-Network (DQN)

DQN is a DRL algorithm that combines DNNs with the Q-learning algorithm and is often used for solving problems in environments with high-dimensional state spaces [7]. DQN addresses the challenges of traditional Q-learning by utilizing a DNN to approximate the action-value function, also known as the Q-function. DNNs enable DQN to handle large and continuous state spaces more efficiently. DQN leverages experience replay, which stores and samples past experiences; it allows DQN to improve sample efficiency and stabilizes learning by reducing the correlation between consecutive experiences. Additionally, DQN can employ a target network to stabilize training further, enabling more accurate value estimates. DQN

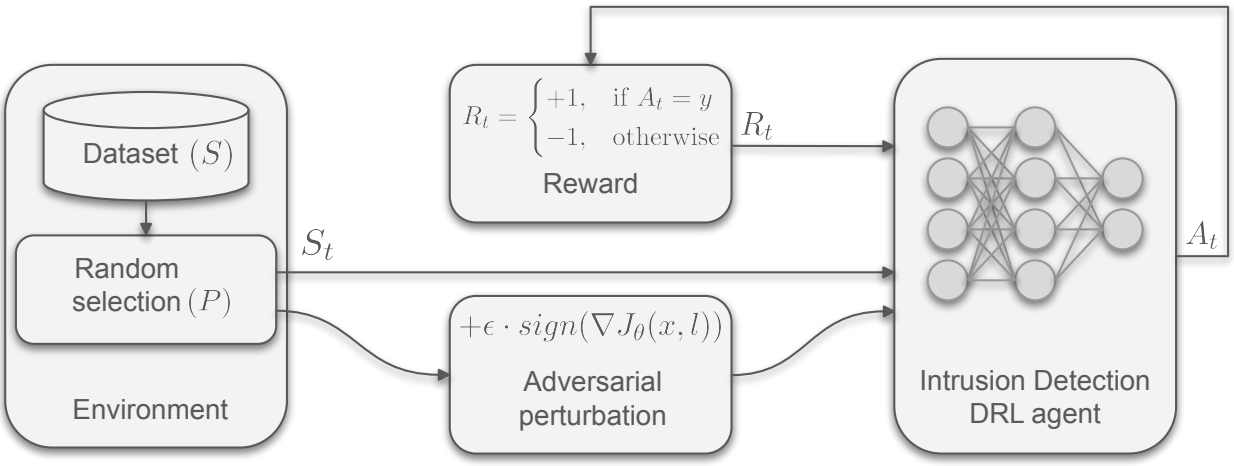


Figure 4.1 Diagram of the methodology: The state S_t is sampled randomly from the dataset (S). Then, it is either directly fed to the DRL agent (Section 4.5.1) or has an adversarial perturbation applied to it (Section 4.5.2). The agent takes an action A_t, where it labels a state as benign or malicious. During the training, the action is compared to the true label, and a reward R_t of $+1$ or -1 is assigned.

has achieved notable success in various complex tasks and has been influential in advancing the field of DRL [7, 198–201].

Quantile Regression Deep Q-Network (QRDQN)

QRDQN is a DRL algorithm that extends the traditional DQN framework by incorporating quantile regression [202]. Unlike conventional DQN, which estimates the expected value of the return distribution, QRDQN models the entire distribution of returns. This allows the algorithm to capture the uncertainty in the predictions and provides a more comprehensive understanding of the value function. By estimating a set of quantiles, QRDQN offers a richer representation of the uncertainty associated with different possible outcomes, making it particularly useful in scenarios where risk assessment and robust decision-making are critical. Incorporating quantile regression in QRDQN enhances its applicability in scientific research, where a nuanced understanding of uncertainty can be pivotal for informed decision-making in complex environments.

Advantage Actor-Critic (A2C)

A2C is a DRL approach that combines elements of policy gradient methods and value-based methods [8]. A2C operates by concurrently updating the policy and the value function, leveraging the advantages of both approaches. The actor (policy) is updated to maximize

expected cumulative rewards, while the critic (value function) estimates the benefit of taking a specific action in a given state. This advantage is then used to guide the policy updates. A2C improves training efficiency by utilizing parallel environments to gather experiences and compute gradients, allowing for more effective exploration of the action space. This algorithm's design enhances sample efficiency and stability, making it particularly valuable in scientific research where efficient and reliable learning from interactions is crucial for solving complex problems.

Trust Region Policy Optimization (TRPO)

TRPO is an RL algorithm that focuses on improving the stability and reliability of policy optimization [203]. TRPO addresses issues related to the potentially drastic changes in policy during training, which can lead to suboptimal performance. The key idea behind TRPO is to limit the size of policy updates to ensure that the new policy remains close to the original policy, preventing significant policy changes that may result in undesirable outcomes. TRPO aims to strike a balance between exploration and exploitation by constraining the policy changes within a trust region, making it a valuable tool in optimizing policies for complex tasks.

Proximal Policy Optimization (PPO)

PPO is a DRL algorithm that aims to optimize policy functions stably and efficiently [97]. It is designed to address the limitations of earlier policy optimization algorithms, such as high sample complexity and sensitivity to hyperparameters. PPO achieves this through a novel objective function that constrains policy updates within a specific range, preventing significant policy changes that can lead to instability. By iteratively optimizing the policy using multiple epochs and mini-batches of collected data, PPO strikes a balance between exploration and exploitation, resulting in improved policy convergence and sample efficiency. PPO has been widely used in various domains, demonstrating its effectiveness in training agents to perform complex tasks in RL settings [97, 204, 205].

4.3.2 Adversarial attacks

Once the DRL intrusion detection agents are trained, we evaluate their robustness to adversarial attacks. We generate adversarial examples by adding a small and well-computed perturbation to its features. The adversarial examples are generated on the malicious instances of the test set; their objective is to mislead the agent into assigning them the benign

label.

In the experiments, we study both white-box and black-box adversarial attacks. The white-box attacks consider an adversary who has access to the agent's parameters and uses them to optimize the adversarial perturbation. Since the white-box assumptions are strong in real-world scenarios, we also study the black-box attacks, where the attacker cannot access the agent's parameters.

To generate adversarial examples in a black-box setting, we leverage the transferability phenomenon: Instead of optimizing the adversarial perturbation with respect to the agent's parameters, a black-box adversary optimizes it with respect to a surrogate agent. The adversary trains the surrogate agent without knowledge of the target agent; it has different parameters and uses different DRL algorithms. We study the impact of transferability-based black-box attacks on the target model. We aim to determine which target DRL algorithms are the most robust to black-box attacks and which surrogate DRL algorithms are the most effective for the generation of black-box adversarial examples.

Recent work has studied the limitations of adversarial examples in intrusion detection [89], showing that the added perturbation potentially invalidates the network traffic. To be implemented in real networks, adversarial perturbations must consider all the constraints related to the network data structure to protect its functionality [206]. These constraints are syntactic or semantic and apply to all types of features individually and relatively to other features [89], which makes them difficult to formalize. Our experiments assume the worst-case scenario where the adversary can perturb all network traffic features. Though this is not a real-world scenario, it establishes a lower bound on the adversarial robustness of DRL-based intrusion detection; in real-world scenarios, the DRL agents can only be more robust.

We generate adversarial examples using two methods: a one-step method called the Fast Gradient Sign Method (FGSM) and an iterative method called the Basic Iterative Method (BIM). We describe both approaches in the following paragraphs.

Fast Gradient Sign Method

FGSM is a common technique for generating adversarial examples [30]. It exploits the gradient of the loss function, which usually serves to update the model's parameters. Instead, the gradient is propagated back to the inputs, and its sign guides the perturbation. An adversarial example x' is formed by adding the perturbation amplitude ϵ with the gradient sign to an original example x. Equation 4.2 describes this perturbation, where ∇ is the gradient, J_θ is the loss function with respect to the parameters θ, and l is the actual label of the example.

$$x' = x + \epsilon \cdot sign(\nabla J_\theta(x, l)) \tag{4.2}$$

Basic Iterative Method

Building upon FGSM, BIM is an iterative method that applies the perturbation ($\epsilon \cdot sign(\nabla J_\theta(x, l))$) over multiple steps [183]. In each iteration, BIM computes the gradient of the loss function with respect to the perturbed input and updates the perturbation accordingly. By performing multiple iterations, BIM gradually moves the adversarial example closer to the decision boundary, increasing the chances of misclassification. The iterative nature of the BIM algorithm allows for fine-grained control over the perturbations, making the adversarial examples more efficient. Generally, increasing the number of iterations will produce finer perturbations and can lead to more subtle adversarial examples. However, there is a trade-off, as computing more steps takes more time to generate adversarial examples.

4.3.3 Datasets

To perform our experiments, we identified three network intrusion detection datasets: UNSW-NB15, NSL-KDD, and CICIoV2024 (Appendix B). Each dataset contains benign data packets, representing legitimate network traffic, and malicious data packets, representing network attacks. Both datasets include multiple attack classes; here, we consider the binary case: whether a packet is either benign or malicious.

NSL-KDD dataset

The NSL-KDD dataset is a widely used benchmark dataset in cybersecurity for evaluating network intrusion detection systems [78]. It was generated from the KDD Cup 1999 dataset to address some of its shortcomings, such as the presence of a large number of redundant records and the imbalance between normal and attack records. The NSL-KDD dataset incorporates a more realistic representation of modern network traffic. It includes both benign and malicious data, making it suitable for assessing the performance of intrusion detection systems in real-world scenarios. The dataset contains diverse network-based features, including protocol-specific information, connection-level attributes, and content-based features. Moreover, it includes different types of attacks, such as Denial of Service (DoS), probing, remote-to-local (R2L), and user-to-root (U2R) attacks, to provide a comprehensive evaluation platform for intrusion detection systems.

With the NSL-KDD dataset having 43 network features and 4 attack types, researchers can

explore various ML and data mining algorithms to build robust and effective intrusion detection systems. Its training set consists of 125,973 records, and its testing set contains 22,544 records. The NSL-KDD's availability and widespread use make it a standard benchmark within the cybersecurity research community.

UNSW-NB15 dataset

The UNSW-NB15 dataset is comprehensive and widely utilized in cybersecurity for evaluating network intrusion detection systems [88]. It was generated by the Cyber Range Lab of the University of New South Wales (UNSW) using the IXIA PerfectStorm tool to simulate a modern network environment with both normal and malicious traffic. The simulation was conducted over a period of 16 hours, creating a diverse set of network activities that included nine types of attacks: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. The captured raw network packets were then processed using Argus and Bro-IDS tools to extract a comprehensive set of 49 features from the flow data. These features cover various aspects of network traffic, such as basic attributes, content features, time features, and additional generated features, providing a detailed and balanced dataset for evaluating the performance of intrusion detection systems. The dataset is partitioned into a training set containing 175,341 instances and a testing set containing 82,332 instances.

The UNSW-NB15 dataset was developed to address the limitations of existing datasets and provide a more realistic representation of modern network traffic. The dataset captures traffic from diverse sources, including multiple operating systems and network protocols, reflecting the complexity of modern networks. It is important to note that the UNSW-NB15 dataset is more representative of modern network traffic than the NSL-KDD dataset and has more complexities in size and data dimensionality.

CICIoV2024 dataset

The CICIoV2024 dataset is a comprehensive Internet of Vehicles (IoV) dataset created from experiments on a 2019 Ford car with Electronic Control Units (ECUs), providing a deep look into how vehicles communicate internally. The dataset consists of a first feature corresponding to the ID and 8 data features, each corresponding to one byte of the transmitted data. In our experiments, we use the binary version of the dataset that consists of one binary feature for each bit. CICIoV2024 includes 5 types of attacks (DoS, gas spoofing, RPM spoofing, speed spoofing, and steering wheel spoofing) in addition to benign communications. This dataset serves as a benchmark for researchers developing new IoV security solutions.

4.4 Methodology

In this section, we specify the experimental approach that we followed, with details about our implementation. Our experiments combine all three elements mentioned in Section 4.3: the DRL algorithms, adversarial attacks, and network intrusion datasets, in a structured methodology. Each experiment sets a configuration of hyperparameters and performs 70 runs with a random seed. A run consists of training a DRL intrusion detection agent on the training data of one of the datasets, validating the agent's performance on the testing data, and applying adversarial attacks using both the FGSM and BIM. For each run, we collect the False Negative Rate (FNR), False Positive Rate (FPR), and F1 score for each of the phases in the run; namely the training, testing, FGSM attack, and BIM attack phases. Here, a negative instance is benign, while a positive instance is malicious. Therefore, a false negative is a malicious instance assigned with the benign label, while a false positive is a benign one assigned with the malicious label. We consider an adversary whose objective is to bypass the detection of the DRL agent by misleading it into assigning a benign label to malicious traffic, thus trying to increase the FNR of the intrusion detection agent.

To effectively assess the impact of different hyperparameters on the robustness of DRL intrusion detection agents, we train multiple DRL agents on three different datasets, using five different DRL algorithms and varying the depth and width of their neural networks. Table 4.1 shows the hyperparameters and their different instances used in our experiments.

Variable	Instances
Datasets	UNSW-NB15, NSL-KDD, CICIoV
DRL Algorithms	DQN, PPO, TRPO, A2C, QRDQN
Nb. hidden layers	1, 2, 4, 8
Nb. units per layer	32, 64, 128, 256

Table 4.1 Configurations for the experiments

In total, we have 240 different configurations. We perform 70 runs for each configuration to capture legitimate values for the mean and standard deviation of our metrics (i.e., FNR, FPR, F1 score). We train each agent for 15 episodes, where the number of steps in an episode is equivalent to the number of instances in the dataset. The agents are then tested on the original and the perturbed test sets (adversarial examples). The perturbation is generated using FGSM and BIM, with a perturbation amplitude between 0 and 0.1 (in terms of the L_∞ norm).

To implement the DRL agents, algorithms, and environments, we leverage the Stable Baselines library that provides standard and reliable implementations [207]. The neural networks

used in the DRL agent are implemented within the DL framework PyTorch [208]. To implement FGSM and BIM, we leverage the Adversarial Robustness Toolbox (ART) library that provides standard and reliable implementations of adversarial attacks [209]. For further details on the implementation, such as the learning rates, gamma values, or other specific hyperparameters, please refer to our code on GitHub [1]. The computational complexity of our experiments required high-performance computing platforms, so we leverage the infrastructure of the Digital Research Alliance of Canada (Compute Canada).

4.5 Results

In this section, we present and analyze the results of the experiments. The primary experiments are performed on two datasets (UNSW-NB15, NSL-KDD), using five DRL algorithms (DQN, PPO, TRPO, A2C, QRDQN), four depths (1, 2, 4, 8 hidden layers), four widths (32, 64, 128, 256 units per hidden layer), for a total of 160 combinations. The figures presented in this section (Figure 4.3a- 4.11) show, for each combination, the mean and standard deviation over 70 runs. Furthermore, we present and analyze an additional case study using CICIoV2024, a more recent IoV dataset, in Appendix B, adding 80 other combinations of hyperparameters. We separate these sets of experiments since the CICIoV2024 dataset focuses on the IoV data, which is quite distinct from traditional network data provided in UNSW-NB15 and NSL-KDD.

Each run takes 7 minutes to 10 hours, depending on the dataset, the DRL algorithm, and the neural network architecture. The longest combinations to run were the policy-based methods (PPO, TRPO, A2C), with larger neural networks (8 hidden layers, 256 units per layer) on the UNSW-NB15 dataset. The monitored metrics are the F1 score, the FPR, and the FNR (which corresponds to the adversarial attack success rate). The subsections below elaborate on the performance of the agents before the adversarial perturbation, the impact of the width (number of units per layer), the depth (number of layers), the training algorithm, and the transferability of adversarial examples across algorithms in black-box settings.

4.5.1 Before adversarial perturbations

Before analyzing the impact of adversarial examples on the agents, it is necessary to evaluate them using unperturbed data. Through this evaluation, we study the influence of the different hyperparameters on the agent's performance and identify the underlying patterns. This study provides context for the forthcoming results on the impact of adversarial examples. In the

[1] `https://github.com/mamerzouk/robust_drl_ids`

following, we start by analyzing the performance of a single neural network architecture with Figure 4.2; then, we generalize the study to all configurations with Figure 4.3a and Figure 4.3b.

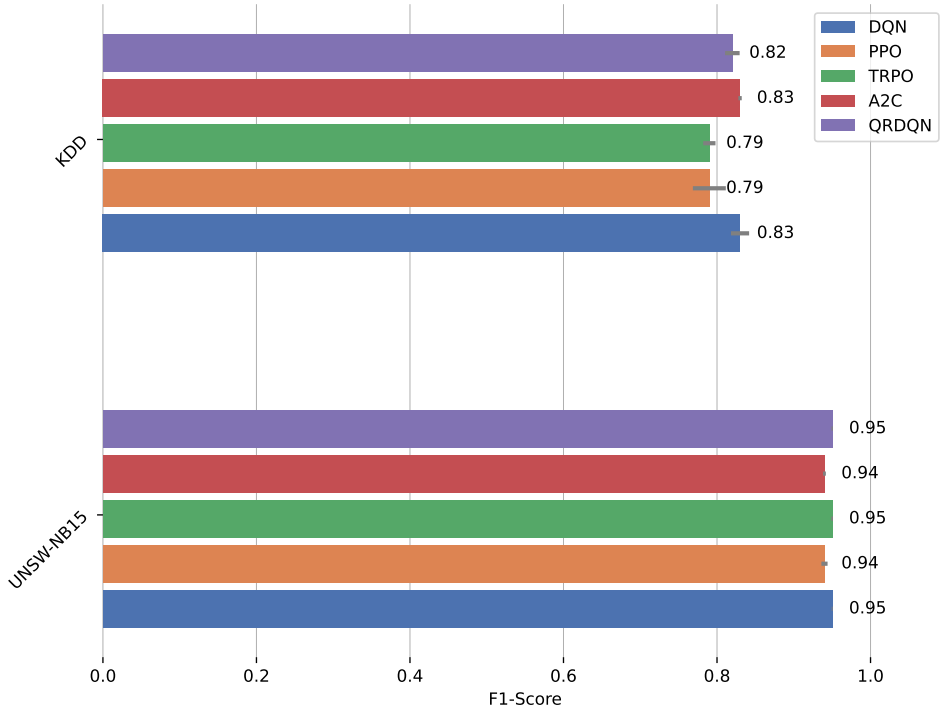
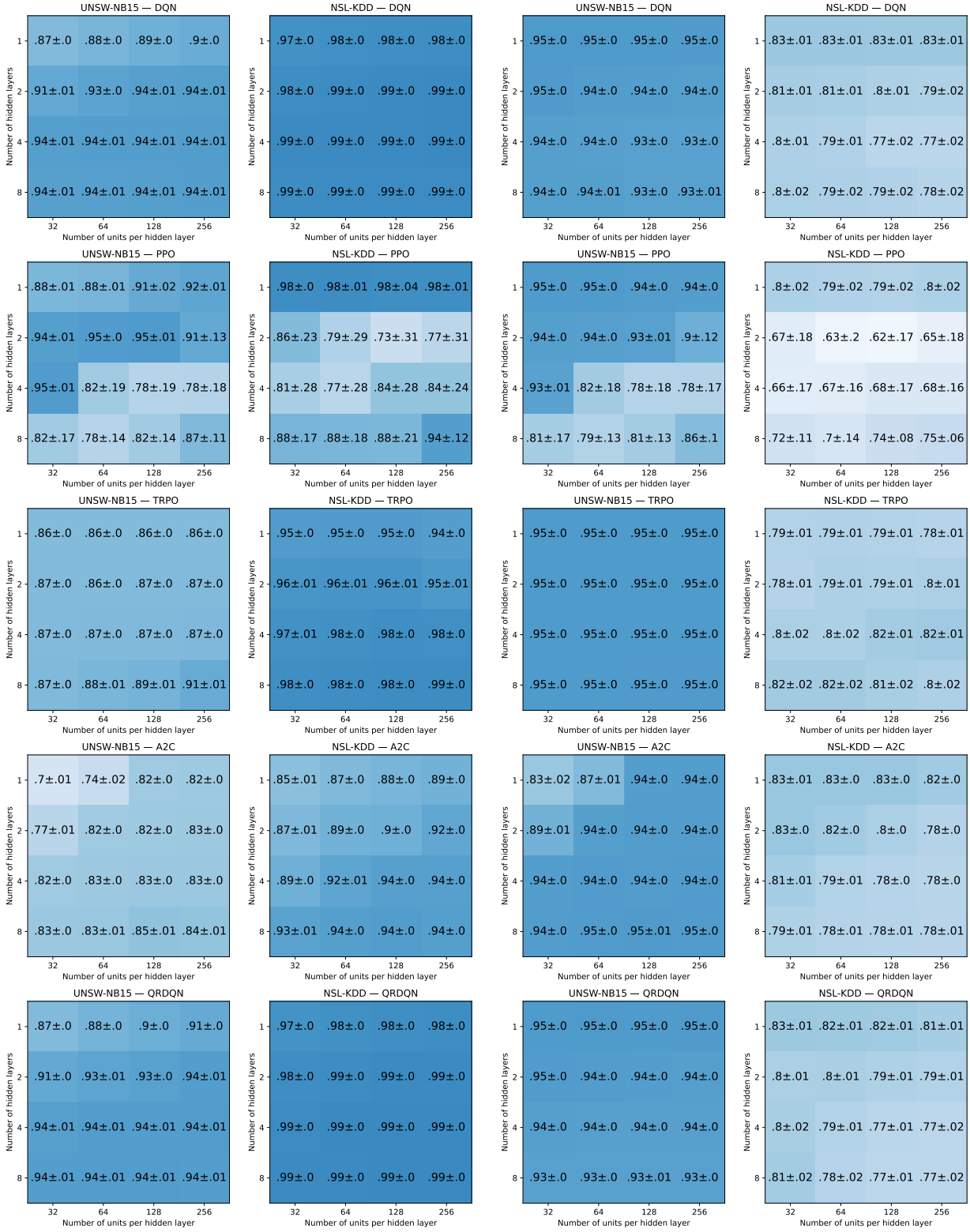


Figure 4.2 F1 score before adversarial perturbations (1 hidden layer of 128 units)

Figure 4.2 presents a bar chart of the mean F1 score for each DRL algorithm on both datasets; the gray line on the right of each bar illustrates the standard deviation. The F1 scores are calculated on the testing set at the last training epoch. All agents use the same neural network architecture: 1 hidden layer of 128 units. This combination was selected as a performance upper-bound since it achieved the best F1 score on average across DRL algorithms. The results show that all DRL algorithms attain state-of-the-art performance in terms of F1 score [11]. The F1 score ranges between 0.94 and 0.95 on UNSW-NB15 and between 0.79 and 0.83 on NSL-KDD. PPO is slightly weaker than other algorithms, especially on NSL-KDD, and has a more significant standard deviation.

While Figure 4.2 shows the mean F1 score for a single neural network architecture (1 hidden layer of 128 units), Figure 4.3a and Figure 4.3b show a heatmap of the F1 score for each DRL algorithm on training and testing sets (respectively) of both datasets. The heatmaps display the mean F1 score and standard deviation with different width and depth combinations.

DQN and QRDQN show similar performance due to the similarity between the two algorithms—QRDQN being an extension of DQN with quantile regression. Their F1 score is rel-



(a) Training set (b) Testing set

Figure 4.3 F-1 Score before adversarial perturbations

atively stable across architectures but is slightly better with fewer parameters, ranging from 0.93 to 0.95 on UNSW-NB15 and 0.77 to 0.83 on NSL-KDD (Figure 4.3b). The performance decrease with larger neural networks is caused by the overfitting tendency of value-based algorithms like DQN and QRDQN [210]. By using a larger neural network to approximate the Q-function, DQNs can fit complex functions, but this same flexibility can lead to overfitting. We confirm this hypothesis by comparing the F1 score on the training and testing sets: while the F1 score increases with the number of parameters on the training set, it decreases on the testing set. The agents with larger neural networks ingrain the training data and struggle to generalize on the testing data.

On the other hand, the policy-based algorithm TRPO is generally more stable across architectures. TRPO has an F1 score of 0.95 on UNSW-NB15 and between 0.79 and 0.82 on NSL-KDD (Figure 4.3b). The performances on NSL-KDD increase with the number of parameters and decrease after a certain threshold. The best performance was achieved with 4 hidden layers of 128 and 256 units and 8 hidden layers of 32 and 64 units (between 256 and 1024 parameters). The smaller models are too simple and cannot properly capture the underlying patterns in the data. This phenomenon is called underfitting. The F1 score increase that follows the neural network size increase on both the training and testing sets shows that larger models are better at capturing the complexity of the data. However, comparing Figure 4.3a and Figure 4.3b shows this is only valid up to a certain point. After 1024 parameters, increasing the neural network size improves the performance on the training set while deteriorating the performance on the testing set; the agents are then overfitting the data.

The same principle is demonstrated with the policy-based algorithm A2C, where the F1 score varies between 0.83 and 0.95 on UNSW-NB15 and between 0.78 and 0.83 on NSL-KDD (Figure 4.3b). While larger neural networks performed better on UNSW-NB15, smaller ones recorded the best F1 score on NSL-KDD. Here, the same DRL algorithm behaves differently on two separate datasets, and it is explained by the structure and properties of the datasets. UNSW-NB15 is larger and more complex; it includes a wider variety of attack types and more network attributes (196 features after one-hot-encoding versus 121 for NSL-KDD). The complexity of UNSW-NB15 explains why, with A2C, smaller neural networks could not capture the underlying patterns and achieve lower performance (underfitting), while larger neural networks achieved a higher F1 score. The opposite phenomenon is observed on NSL-KDD: the smaller networks are capable of fitting the data. In comparison, larger neural networks overfit the data, causing the F1 score to decrease on the testing set (Figure 4.3b) while increasing on the training set (Figure 4.3a).

Concerning the policy-based algorithm PPO, the F1 score is stable with smaller neural networks; it ranges between 0.93 and 0.95 on UNSW-NB15 and between 0.79 and 0.80 on NSL-KDD, with a standard deviation below 0.03 (Figure 4.3b). However, PPO experienced instability with larger neural networks on both datasets. This is shown by the standard deviation reaching 0.18 (Figure 4.3b). The analysis of the learning curves shows significant discrepancies from one epoch to the other, which explains the standard deviation.

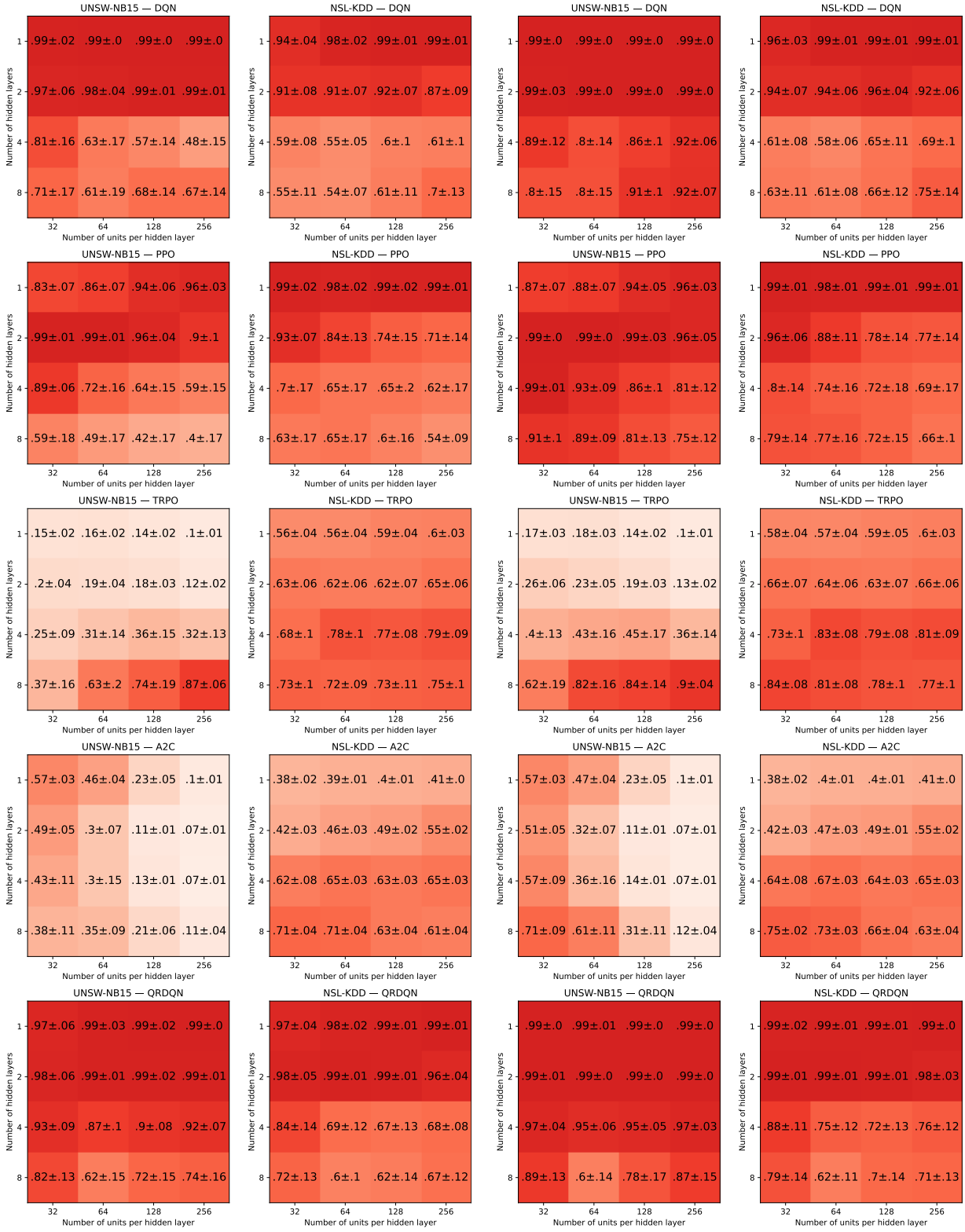
Several insights emerge from this analysis of the performance before adversarial perturbations:

- The choice of neural network architecture and size is specific to the DRL algorithm; it is not generalizable to all cases. DRL algorithms such as DQN and QRDQN struggle with larger networks, while A2C and TRPO require more parameters to fit the data properly;
- The choice of neural network architecture and size is also specific to the dataset; the same DRL algorithm might require more or fewer parameters depending on the dataset. For instance, A2C achieved the best F1 score with smaller neural networks on NSL-KDD, while it achieved the best F1 score with larger neural networks on UNSW-NB15. This is due to the complexity of UNSW-NB15, which requires more parameters;
- The performance gaps between neural network architectures are mainly caused by the phenomena of underfitting and overfitting, as the larger networks tend to overfit while the smaller neural networks tend to underfit;
- The choice of neural network architecture also influences the stability of the training and may disturb the convergence of some DRL algorithms. This was the case for PPO, which suffered instability with larger neural networks, preventing its convergence.

While the above statements are generally known to be true for deep learning classification methods, we emphasize that our study is the first to demonstrate these results for DRL-based intrusion detection. Additionally, establishing the gained insights from this analysis is essential for the remainder of this work.

4.5.2 After adversarial perturbation

In this subsection, we analyze the impact of adversarial perturbations on the performance of the DRL detection agents. The adversarial attacks are generated using two methods: FGSM and BIM, where perturbations are added to the malicious traffic (positive instances) of the



(a) FGSM with ($\epsilon = 0.1$) (b) BIM with ($\epsilon = 0.1$)

Figure 4.4 FNR matrices

testing set to mislead the agent into assigning them as benign traffic (negative instances). The attack success rate is determined by the proportion of malicious traffic classified as benign, which corresponds to the FNR. Therefore, our analysis focuses on the FNR since it is the leading cause of change in the F1 score (FPR is relatively stable). The amplitude of the perturbation ϵ ranges between 0 and 0.1, with a 0.01 step.

Figure 4.4a and Figure 4.4b show heatmaps of the FNR for each DRL algorithm with adversarial perturbations generated by FGSM and BIM (respectively). The heatmaps display the mean FNR and standard deviation for different width and depth combinations. The results shown in the heatmaps were generated with a perturbation amplitude of $\epsilon = 0.1$.

The observation shows similar patterns between the FGSM and BIM results. However, in most cases, BIM achieves a higher FNR than FGSM (+10% average FNR overall). This is justified by BIM's iterative approach to generating adversarial perturbations. By iteratively applying small perturbations, BIM finds optimized solutions to the adversarial problem that the one-step approach of FGSM has missed.

Furthermore, we observe a relationship between underfitting and overfitting concerning the robustness of DRL agents to adversarial examples. This is particularly noticeable with A2C in Figure 4.4a and Figure 4.4b. On UNSW-NB15, the FNR is higher for A2C on the small neural networks where it suffered underfitting. On NSL-KDD, the FNR is higher for A2C on the large neural networks, where it suffered from overfitting. The same phenomenon occurs with TRPO, as the FNR is higher with larger neural networks that cause the agent to overfit. The overfitting agents learn to memorize the training data and struggle to generalize; this behavior makes them more sensitive to small perturbations, leading to a higher vulnerability to adversarial examples [211]. However, the underfitting/overfitting phenomenon does not explain everything about the differences in adversarial robustness between architectures or between DRL algorithms. In the following subsections, we isolate several hyperparameters: the width of the neural network (number of hidden layers), the depth of the neural network (number of units per hidden layer), and the DRL algorithm, and investigate their impact on the robustness to adversarial examples.

Width

The first hyperparameter we investigate is the number of units per hidden layer in the DRL agents' neural networks. Figure 4.5 and Figure 4.6 show line graphs of the FNR (y-axis) for each width on UNSW-NB15 and NSL-KDD (respectively) across an ϵ ranging from 0 to 0.1 (x-axis). The continuous lines represent the FNR of FGSM, while the dotted lines represent the FNR of BIM. The values of the number of units per hidden layer are 32, 64,

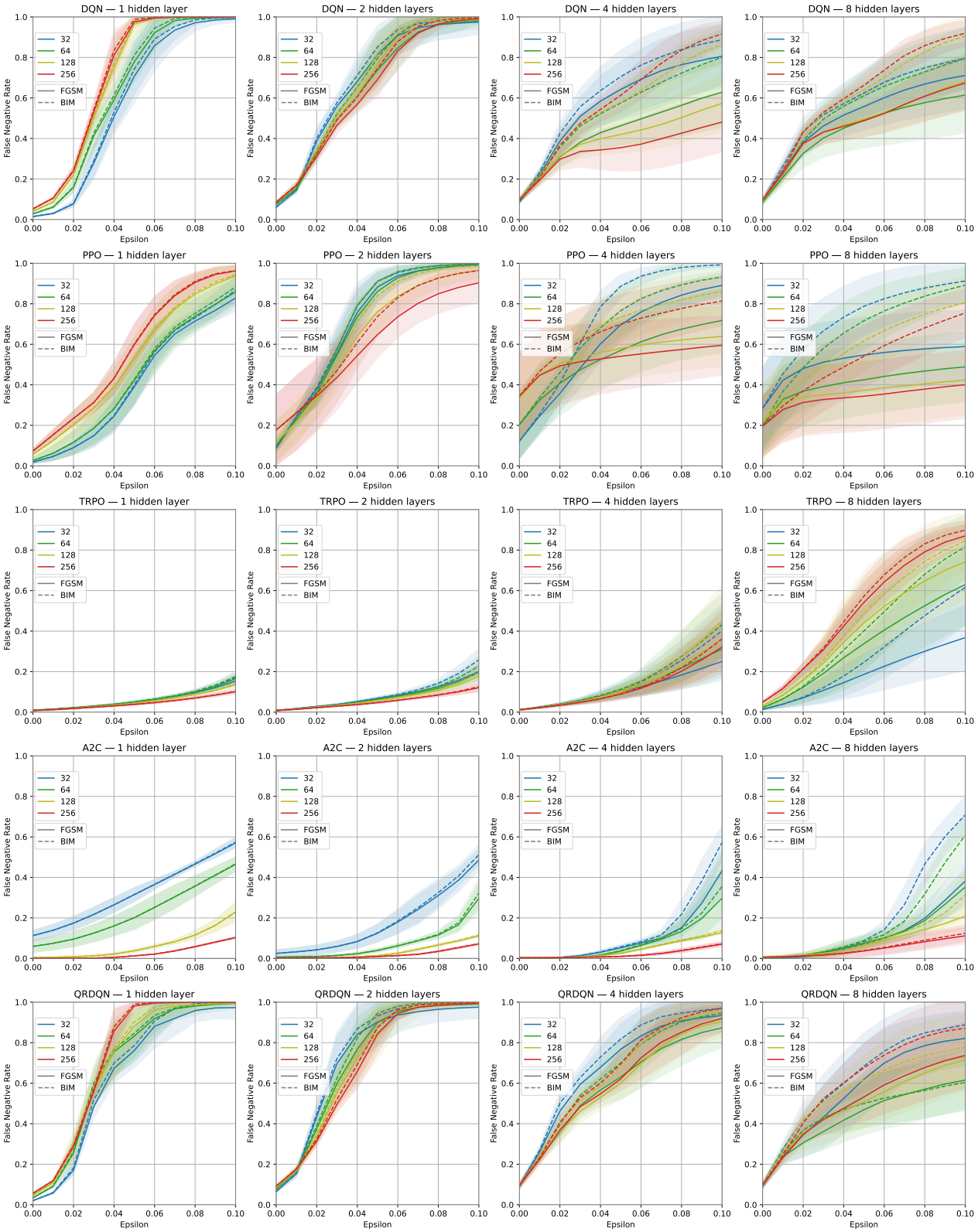


Figure 4.5 FNR across neural network widths on UNSW-NB15

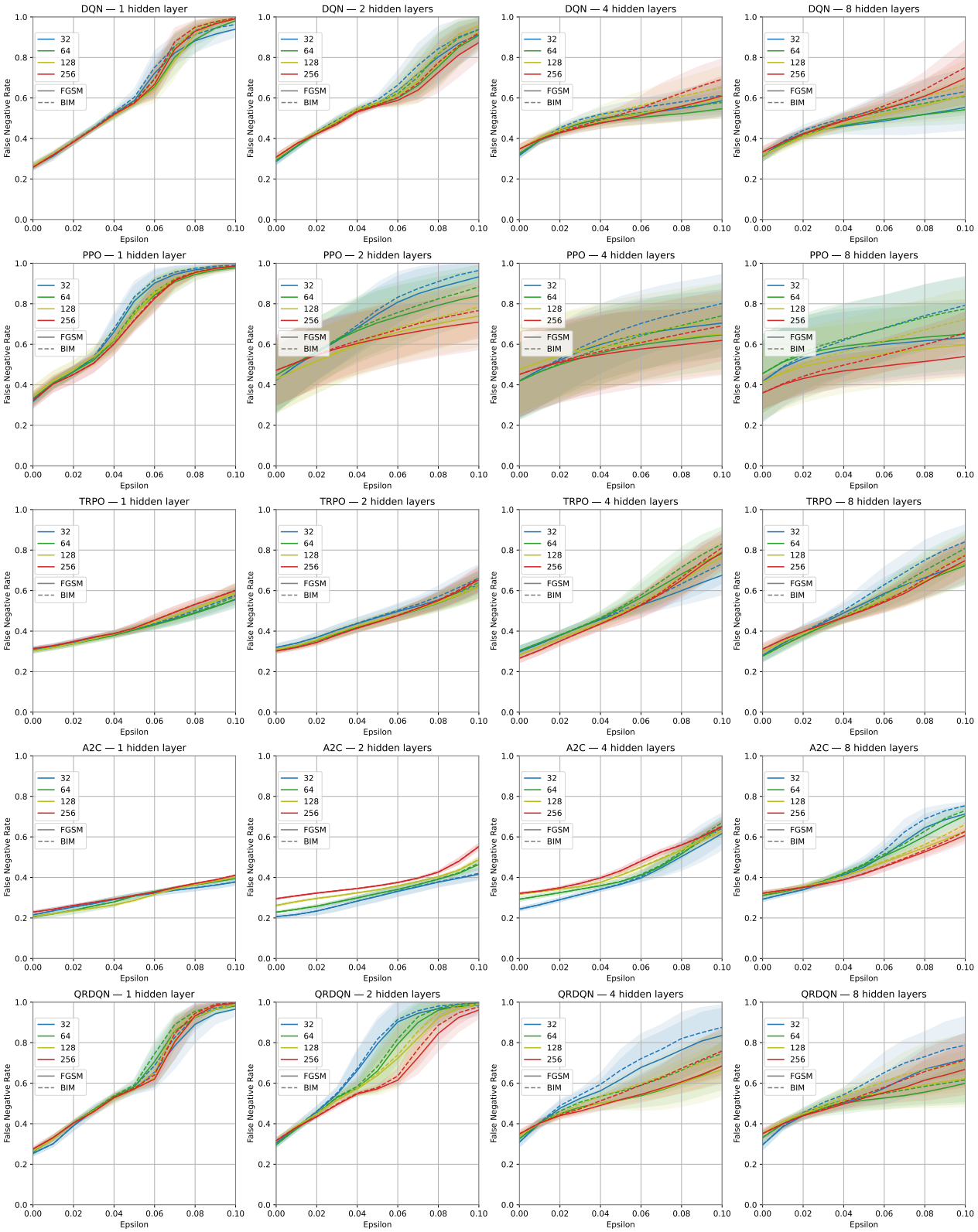


Figure 4.6 FNR across neural network widths layer on NSL-KDD

128, and 256; they are represented with the colors blue, green, yellow, and red (respectively). Figure 4.5 and Figure 4.6 show a line plot for each combination of a DRL algorithm and a neural network depth.

We first observe that the dotted lines (BIM) are always higher than their continuous counterparts (FGSM). This corroborates the observations in Section 4.5.2 that show better performance for BIM and explain it with its optimized iterative perturbations.

We also observe that the width of the neural network has more impact on UNSW-NB15 than on NSL-KDD. This is due to the complexity of the data in UNSW-NB15, which highlights the differences between different configurations. Nonetheless, both datasets follow the same patterns regarding the impact of depth on robustness.

The general trend in Figure 4.5 (UNSW-NB15) shows an increase in FNR with an increased number of units per hidden layer. Increasing the width of the neural network provides more flexibility and helps to capture the underlying complexity of the data. However, it also increases the sensitivity of the model to small perturbations. The capacity to model more complex spaces comes with the risk of creating more “blind spots" where the agent's classification is arbitrary [28].

The exception is A2C, which shows a higher vulnerability to adversarial examples with narrow neural networks (fewer units per hidden layer) in Figure 4.5 (UNSW-NB15). Due to the underfitting phenomenon explained in Section 4.5.2, the low number of units per hidden layer prevents A2C from adequately fitting the data; the naive agent is then easily fooled by adversarial examples.

Lastly, we observe significant differences in FNR between widths as the neural networks get deeper (more hidden layers). This observation applies to all DRL algorithms, especially with higher values of ϵ, indicating that the depth of the neural network amplifies the effect of its width on the robustness. The following subsection studies the impact of the neural network's depth in more detail.

Several insights emerge from this analysis of the impact of the neural network's width on the robustness of DRL agents to adversarial examples:

- Increasing the number of units per hidden layer beyond the overfitting threshold leads to higher sensitivity to adversarial perturbations, as observed for DQN, PPO, TRPO, and QRDQN.
- On the other hand, on underfitting agents such as A2C and shallow TRPO, increasing the number of units per hidden layer leads to better robustness to adversarial pertur-

bations.

- The impact of the neural network's width on the robustness is more significant on complex data structures such as UNSW-NB15 than it is on simpler ones such as NSL-KDD (see Section 4.3.3);
- The impact of the neural network's width on robustness is amplified by its depth. The gap between different widths increases as the number of hidden layers increases.

Depth

The second hyperparameter we investigate is the number of hidden layers in the DRL agent's neural network. Figure 4.7 and Figure 4.8 show line graphs of the FNR for each depth on UNSW-NB15 and NSL-KDD (respectively). The y-axis represents the FNR and the x-axis represents epsilon ϵ, the amplitude of the adversarial perturbation added to the data, ranging from 0 to 0.1. The continuous lines represent the FNR of FGSM, while the dotted lines represent the FNR of BIM. The number of hidden layers 1, 2, 4, and 8, are represented by the colors blue, green, yellow, and red (respectively). Figure 4.7 and Figure 4.8 show a line plot for each combination of a DRL algorithm and a neural network width. Similarly to the observation in subsubsection 4.5.2, we notice that BIM achieves better FNR than FGSM in all cases.

With DQN, QRDQN, and PPO, we observe recurrent patterns on both datasets: deeper neural networks (more hidden layers) are more vulnerable to adversarial examples with smaller values of epsilon, but after a certain threshold, the FNR increases faster for shallower neural network (1 to 2 hidden layers). The epsilon threshold is between 0.5 and 0.6 on NSL-KDD and between 0.3 and 0.4 on UNSW-NB15. On these three algorithms, the sensitivity to the perturbation amplitude increase is noticeably higher on shallower neural networks. We hypothesize that FGSM and BIM find an optimum on the shallow neural networks at the epsilon threshold, and they exploit that optimum with higher epsilon values to cause more false negatives. The relationship between the number of hidden layers and the exact value of the threshold is yet to be investigated.

With TRPO and A2C, there is no sudden increase of FNR on shallow neural networks. The same tendency observed in low epsilon values is amplified as epsilon increases: a higher number of hidden layers causes a higher FNR, and the difference grows with epsilon. As we have found in Section 4.5.2, deeper neural networks are more sensitive to adversarial examples due to their tendency to overfit [211]. The curves for A2C with 32-64 units per hidden layer (Figure 4.7) do not contradict our hypothesis, as they only represent the underfitting case

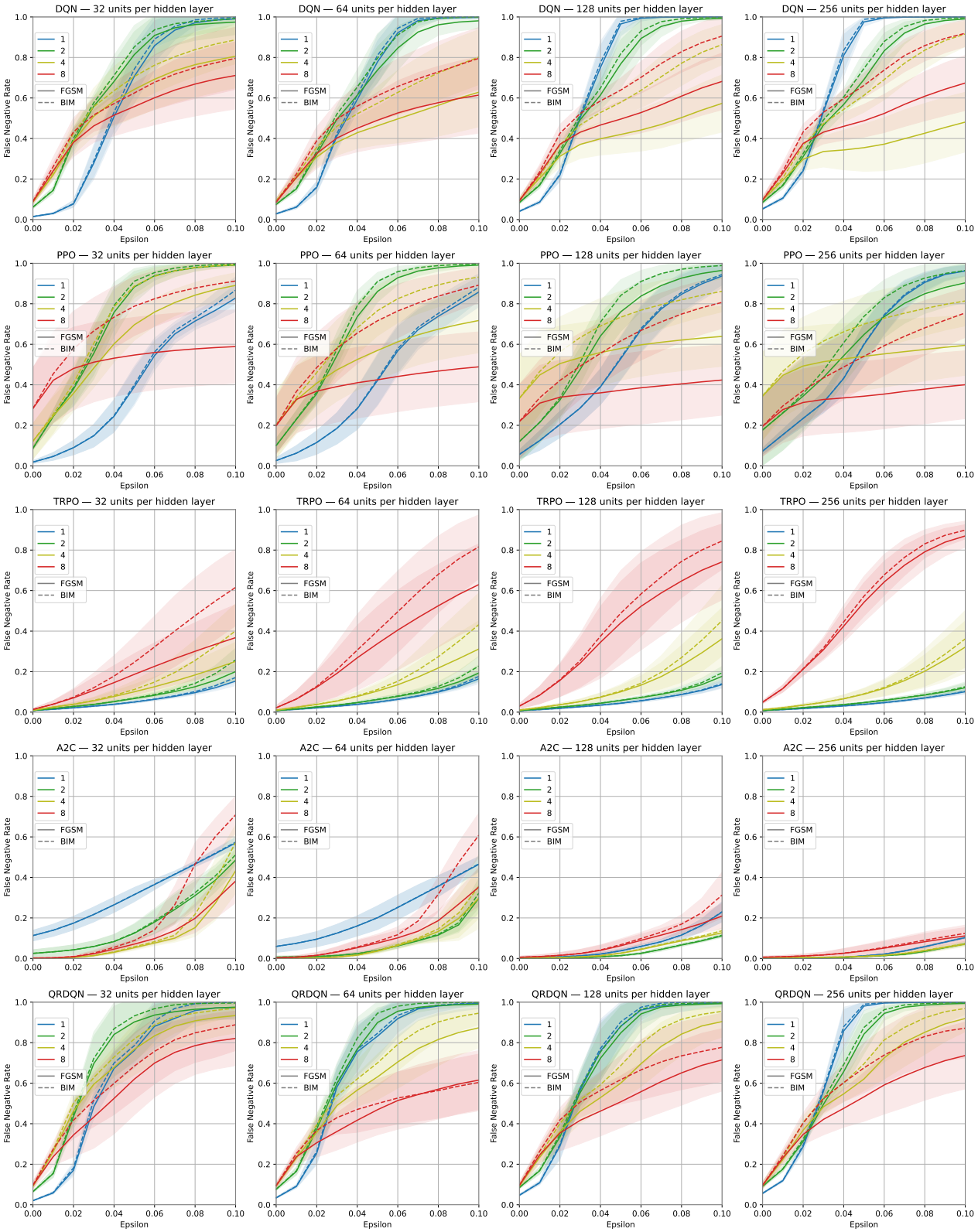


Figure 4.7 FNR across neural network depths on UNSW-NB15

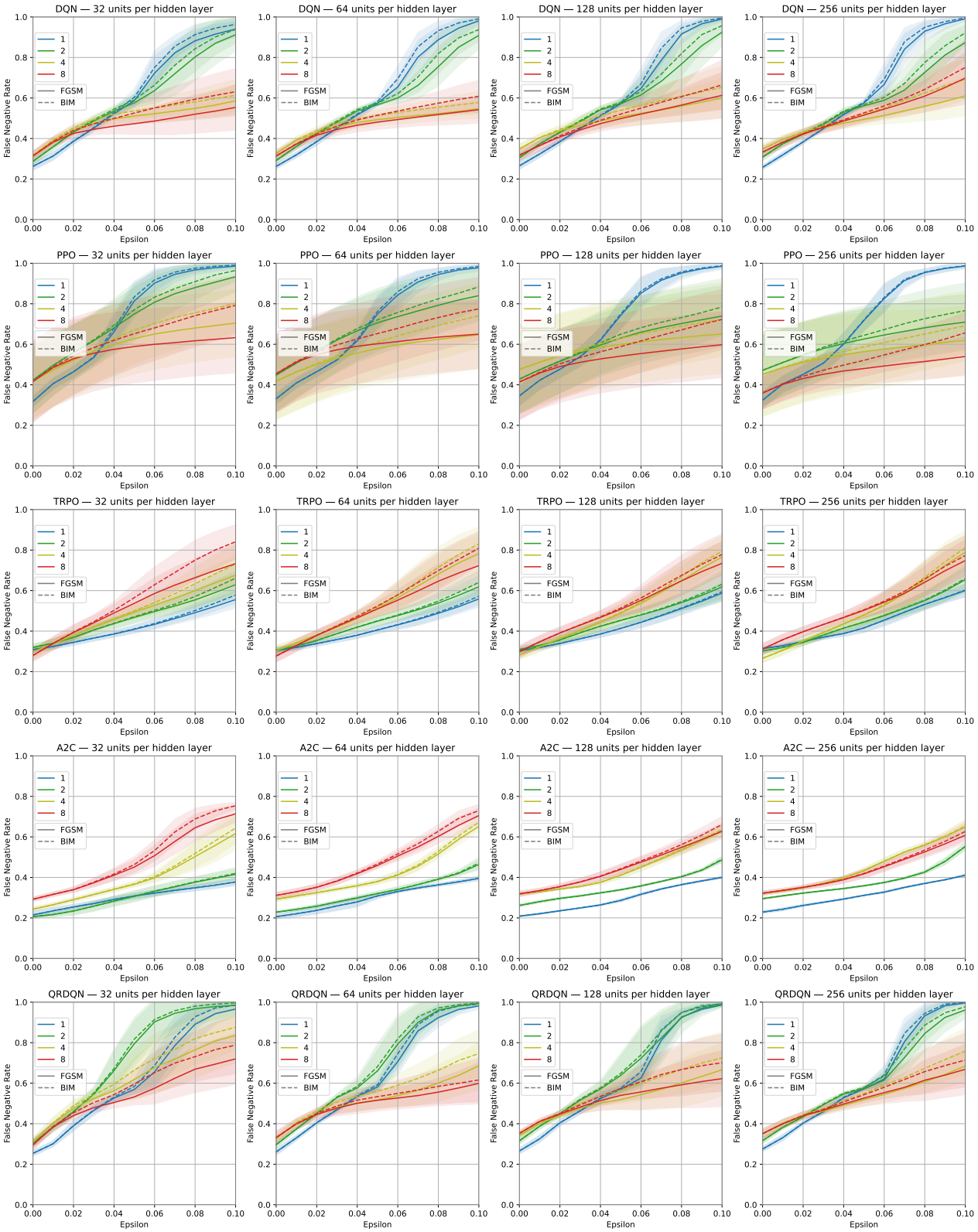


Figure 4.8 FNR across neural network depths on NSL-KDD

where a lower number of hidden layers only worsens the underfitting.

The depth of the neural network is also related to a higher performance gap between FGSM and BIM, especially on UNSW-NB15. Figure 4.7 shows that the gap between BIM and FGSM increases gradually on deeper neural networks (4 hidden layers and more). The difference starts from an epsilon value between 0.2 and 0.4, depending on the algorithm, and reaches up to a 37% increase in FNR in favor of BIM. This result supports the theory that the iterative method stands out even more in deeper neural networks, as it finds optimized adversarial examples compared to FGSM.

We provide the following insights from analyzing the effect of the depth of the DRL agents:

- Increasing the number of hidden layers enables the neural network to fit complex distributions but also leads to overfitting, thus increasing the agent's sensitivity to adversarial perturbations, as observed for DQN, PPO, TRPO, and QRDQN.
- The underfitting agents, such as narrow A2C on UNSW-NB15, display the opposite trend: the shallow neural networks are more sensitive to adversarial perturbations. However, the low performance on the testing set due to underfitting already disqualifies these configurations.
- On DQN, QRDQN, and PPO, the shallow neural networks (1 to 2 hidden layers) are more sensitive than deeper ones to the increase in the perturbation amplitude.
- As epsilon increases, the performance gap between FGSM and BIM increases in favor of the iterative method. This is particularly noticeable in neural networks with 8 hidden layers.

DRL algorithm

The third hyperparameter we investigate is the DRL algorithm used by the detection agent. The DRL algorithms used in our experiments rely on different RL approaches, as outlined in Section 4.3.1. We have seen how these differences impact the agents' performance in Section 4.5.1, and we hypothesize that the impact extends to the agents' robustness to adversarial examples. In this section, we study the robustness of 5 DRL algorithms (DQN, PPO, TRPO, A2C, QRDQN) across neural network architectures.

Figure 4.10 and Figure 4.11 show a line plot for each neural network width and depth combination. Figure 4.9 summarizes the results displayed in Figure 4.10 and Figure 4.11 in a single line graph for each dataset. The data shown in Figure 4.9 are the mean and standard deviation averaged together over all neural network architectures.

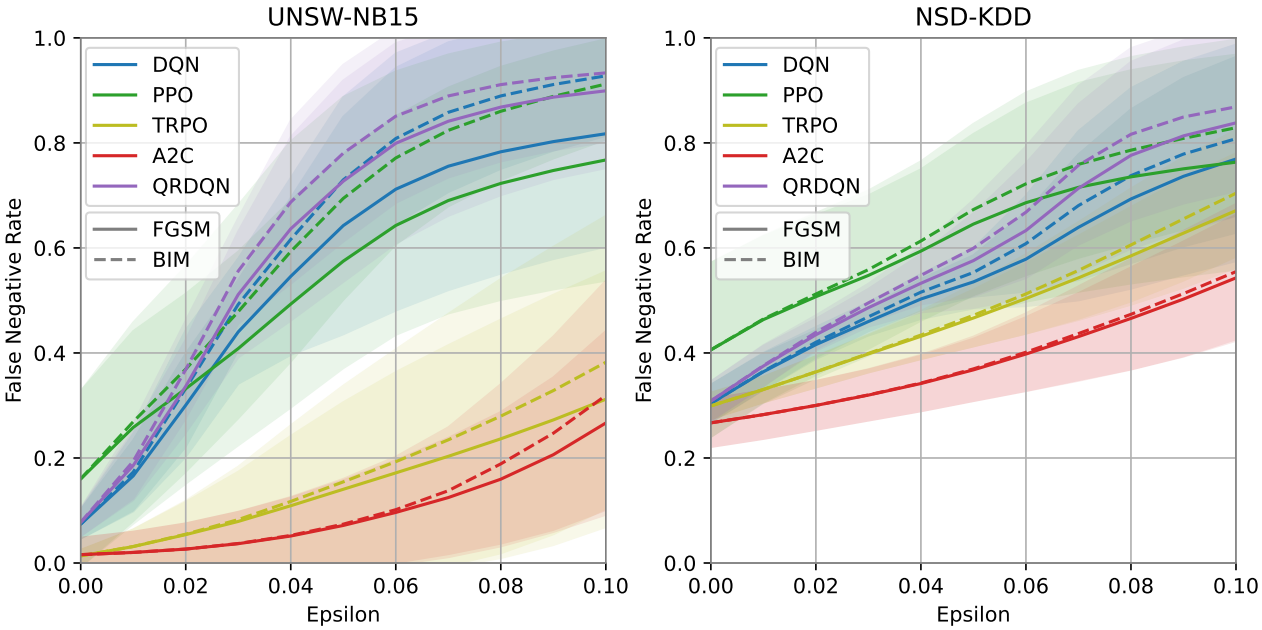


Figure 4.9 FNR across DRL algorithms over all architectures for UNSW-NB15 and NSL-KDD

Despite the higher standard deviation in Figure 4.9, we distinguish a lower overall FNR for A2C. With FGSM and $\epsilon = 0.1$, A2C achieves an FNR of 0.27 ± 0.17 on UNSW-NB15 and 0.54 ± 0.12 on NSL-KDD. TRPO follows closely with 0.31 ± 0.24 on UNSW-NB15 and 0.67 ± 0.11 on NSL-KDD. The FNR for DQN, PPO, and QRDQN is between 0.77 and 0.9, which makes them the least robust DRL algorithms in our settings. We also observe a higher gap between A2C/TRPO and the rest of the algorithms on UNSW-NB15 compared to NSL-KDD. The complexity and higher dimensionality of UNSW-NB15 allow A2C and TRPO to stand out in robustness.

Deeper analysis of the individual plots in Figure 4.10 and Figure 4.11 corroborates our observations. The rankings of the DRL algorithms are substantially similar in almost every architecture. The few exceptions are explained by the underfitting of A2C in small neural networks (less than 128 parameters) on UNSW-NB15 or the overfitting of A2C and TRPO in deeper neural networks (more than 2 layers) on NSL-KDD. While these specific settings slightly disfavor A2C and TRPO, they are still the most robust DRL algorithms overall.

The main takeaways from this analysis of the impact of the DRL algorithm on the robustness of DRL agents to adversarial examples are:

- The DRL algorithm A2C achieves the best robustness to adversarial perturbations, closely followed by TRPO, except with underfitting configurations.

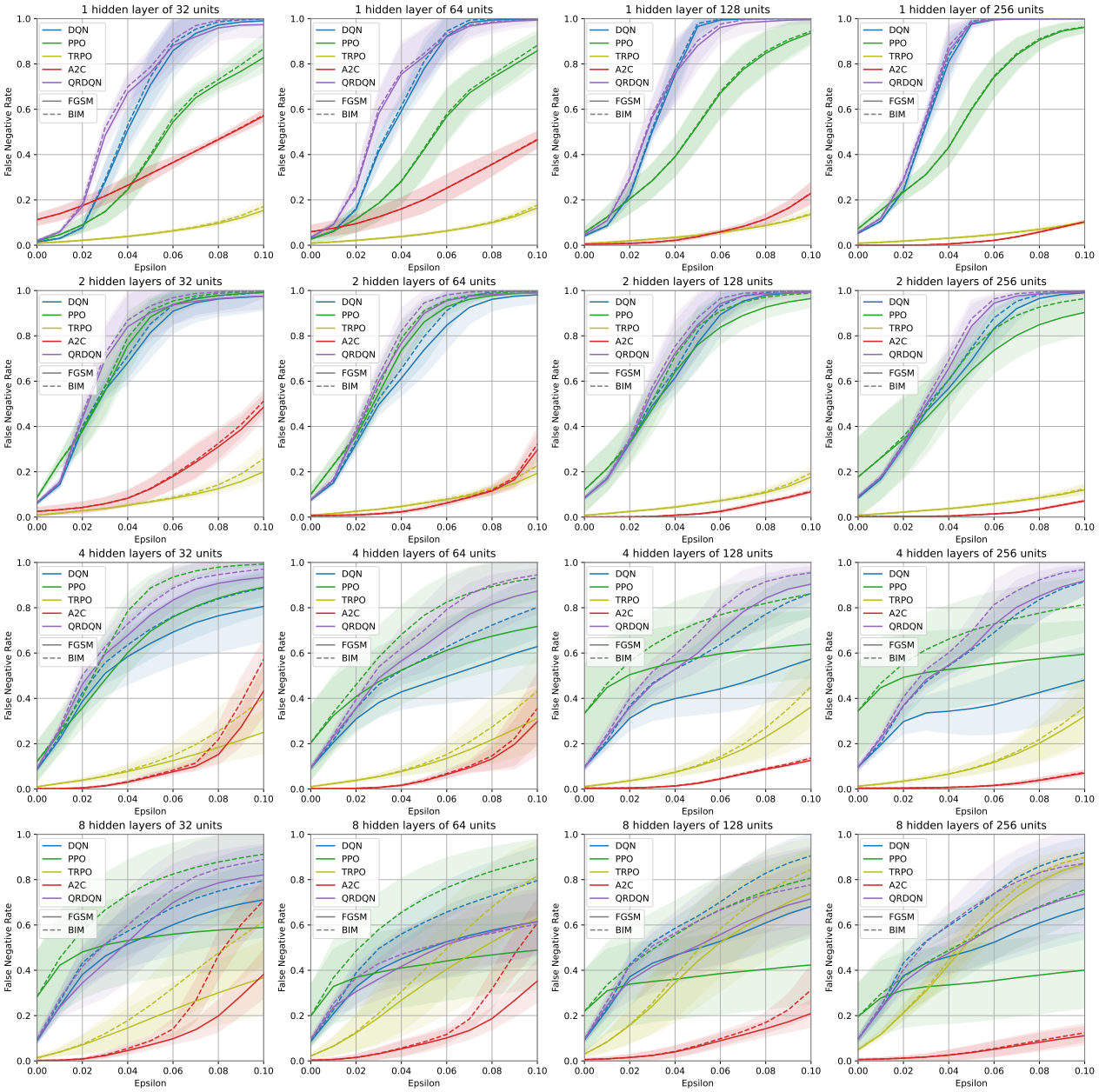


Figure 4.10 FNR across DRL algorithms on UNSW-NB15

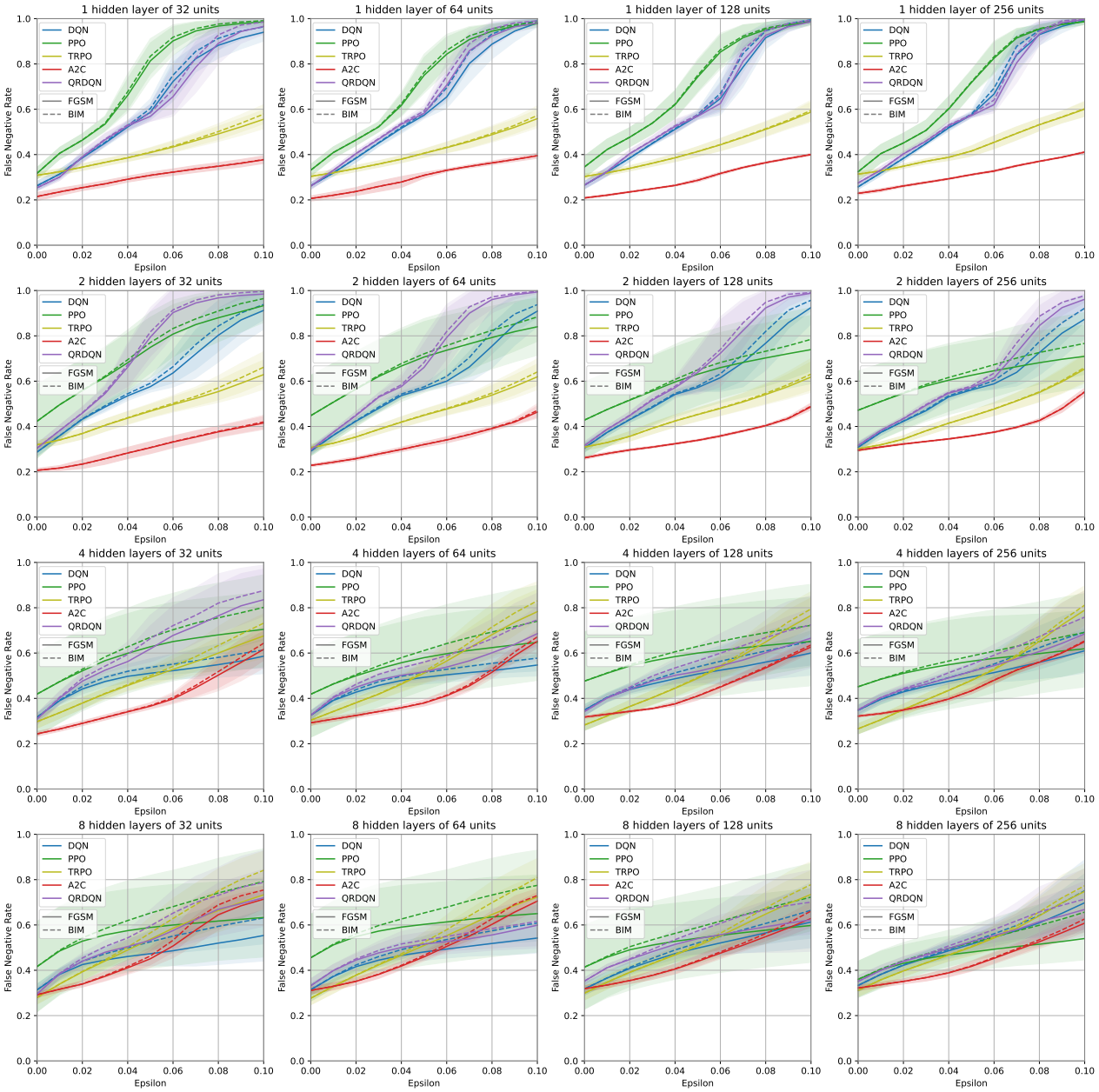


Figure 4.11 FNR across DRL algorithms on NSL-KDD

- The gap between the robustness of DRL algorithms is higher on UNSW-NB15 because of its complexity.

4.5.3 Black-box transferability

In this section, we investigate the impact of adversarial examples in black-box settings. Adversarial examples are generated on a surrogate agent using one DRL algorithm and then applied to other agents using different DRL algorithms. These black-box attacks leverage the phenomenon of transferability in adversarial examples [71]: the perturbation generated on one neural network is also effective, to a certain degree, on other neural networks. Thus, the common patterns in adversarial examples can be universally exploited to attack unknown agents. Thus, we generate adversarial examples on a surrogate model without knowledge of the target agent's parameters. The surrogate agent uses a different DRL algorithm and has different parameters. The objective is to determine which target DRL algorithms are the most robust to black-box attacks and which surrogate DRL algorithms are the most effective for the generation of black-box adversarial examples.

Figure 4.12 shows a heatmap of the FNR of adversarial examples transferred from a surrogate agent to a black-box target agent on both datasets. The adversarial examples are generated with FGSM and BIM with a perturbation amplitude $\epsilon = 0.1$. The heatmaps display the mean FNR and its standard deviation; the y-axis indicates the surrogate agent (source of adversarial examples), and the x-axis indicates the target agent (destination of adversarial examples). The diagonal represents the FNR of adversarial examples generated on one surrogate agent and applied to a target agent using the same DRL algorithm but trained separately. QRDQN is not included in this study because its adversarial examples were generated on quantiles and were incompatible with other DRL algorithms. However, considering the results of previous experiments, the performance of QRDQN is close to that of DQN.

First, we observe, on average, a higher transferability on NSL-KDD compared to UNSW-NB15. The latter's complexity makes the adversarial examples harder to transfer from one agent to the other. We also observe a higher transferability with BIM compared to FGSM. The iterative method generates optimized adversarial examples that are more efficient even across agents.

We notice that the diagonal element of each line often has the highest FNR, meaning that adversarial examples generated on one surrogate agent are most efficient on agents using the same DRL algorithm. For example, DQN achieves 0.84 ± 0.1 FNR on other DQNs, and TRPO achieves 0.85 ± 0.07 FNR on other TRPOs (values from UNSW-NB15 with BIM). This advantage is due to the similarities between agents using the same DRL algorithm. It shows

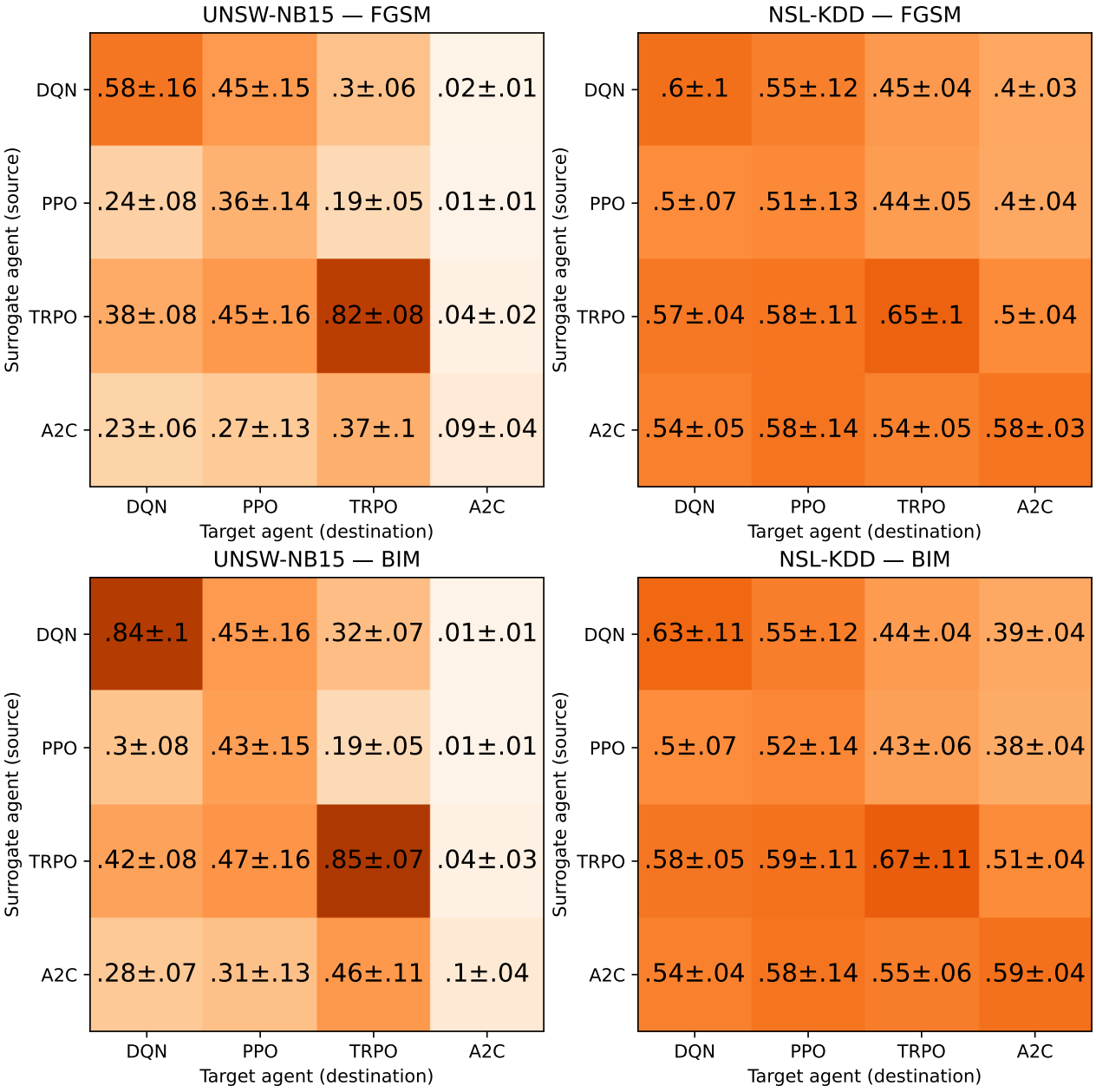


Figure 4.12 FNR transferability matrices across DRL algorithms

that the DRL algorithm used by an agent is a sensitive piece of information that considerably increases the chances of black-box adversaries. The A2C target agent is an exception as it achieves the lowest FNR on UNSW-NB15, even against adversarial examples generated with an A2C surrogate model.

Regarding surrogate agent performance, TRPO achieves the highest FNR on other DRL algorithms, closely followed by DQN. A2C has a high transferability on NSL-KDD but low transferability on UNSW-NB15.

Regarding target agent robustness, A2C achieves the best robustness to transferred adversarial examples. This is particularly noticeable on UNSW-NB15, where the FNR for A2C is under 0.1. On the other hand, the target agents using PPO had the highest FNR, indicating a lower robustness to adversarial examples.

The main takeaways from this analysis of the transferability of adversarial examples in black-box settings are:

- The transferability is higher on NSL-KDD compared to UNSW-NB15 and using BIM compared to FGSM.
- The adversarial examples transfer better to target agents using the same DRL algorithm as the surrogate agent.
- The adversarial examples generated with a TRPO surrogate agent have the best transferability, closely followed by DQN.
- Overall, the most robust agent to black-box adversarial examples is A2C, while the least robust agent is PPO.

4.6 Discussion

We gain the following insights into how each of the investigated hyperparameters affects the robustness of a DRL detection agent to adversarial attacks.

Optimal neural network configurations. The selection of neural network architecture and size is intricately tied to specific DRL algorithms and datasets. Larger networks pose challenges for DQN and QRDQN, while A2C and TRPO adapt well by requiring more parameters for diverse datasets. Underfitting and overfitting dynamics significantly impact performance, with considerations for stability influencing convergence, exemplified by PPO's instability with larger networks.

Width and robustness after adversarial perturbations. Beyond the overfitting threshold, increasing the number of units per hidden layer heightens sensitivity to adversarial perturbations for DQN, PPO, TRPO, and QRDQN. In contrast, underfitting agents like A2C and shallow TRPO demonstrate improved robustness with increased hidden layer units. The impact of neural network width is more pronounced in complex structures like UNSW-NB15, and this impact is amplified with increased depth.

Depth and sensitivity to adversarial perturbations. Adding hidden layers enhances the neural network's capacity to model complex distributions but concurrently increases susceptibility to adversarial perturbations for DQN, PPO, TRPO, and QRDQN. Underfitting agents, like narrow A2C on UNSW-NB15, exhibit the opposite trend, with shallow networks being more sensitive. Shallow networks with 1 to 2 hidden layers are more vulnerable to perturbation amplitude increases than deeper ones. Notably, the performance gap widens between FGSM and BIM as epsilon increases, favoring the iterative method, particularly with 8 hidden layers.

DRL algorithm robustness after adversarial perturbations. A2C emerges as the most robust DRL algorithm to adversarial perturbations, closely trailed by TRPO, except in underfitting configurations. The robustness gap between DRL algorithms is more pronounced on the complex dataset UNSW-NB15.

Black-box transferability. Transferability is higher on NSL-KDD compared to UNSW-NB15. Adversarial examples exhibit better transferability to target agents with the same DRL algorithm as the surrogate agent. Adversarial examples generated on a TRPO surrogate agent demonstrate superior transferability, closely followed by DQN. Overall, A2C proves to be the most robust agent against black-box adversarial examples, while PPO exhibits the least robustness.

Limitations. These insights provide novel findings, however, it is important to recognize the limitations of this work. We primarily analyze how configuring the width and depth of a DRL impacts its performance in the face of adversarial examples. However, several other hyperparameters that are algorithm-specific for each of the individual DRL methods could be tuned. We consider an extensive tuning phase for each of the methods as out of the scope of this current work, considering that this would need to be carried out for each configuration of width and depth for all of the DRL methods. Additionally, this work focuses on gradient-based attacks, specifically FGSM and BIM. While we acknowledge other types of applicable adversarial attacks exist, we feel experimenting with gradient-based is the logical first place to start and it opens doors for future studies. Lastly, this work would be enhanced by testing with even more datasets and DRL algorithms. Nonetheless, this work provides a concerted effort to cover a large breadth of experiments to be analyzed and provides avenues for future work.

4.7 Conclusion

As interest continues to grow for DRL in intrusion detection to combat increasingly sophisticated cyberattacks, there is a need to understand how these methods can be implemented robustly, especially with the threat of adversarial attacks.

This work reveals the influence of the DRL algorithm and the neural network architecture in DRL-based intrusion detection agents when facing adversarial attacks, contributing novel insights to the intersection of the DRL and intrusion detection fields. We perform experiments comprised of the 240 configurations: 3 datasets (i.e., UNSW-NB15, NSL-KDD and CICIoV2024), 5 DRL detection algorithms (i.e., DQN, PPO, TRPO, A2C, and QRDQN), four values of the neural network depth (i.e., 1, 2, 4, and 8 hidden layers), and four values of the neural network width (i.e., 32, 64, 128, and 256 units per hidden layer). We attack each configuration using FGSM and BIM with varying values for the perturbation amplitude ϵ, from 0 to 0.1.

Our findings reframe existing paradigms in DRL-based intrusion detection, introducing novel perspectives on the possible applications of DRL and stimulating further investigation into their secure integration in intrusion detection systems. Concretely, they suggest that selecting an optimal DRL agent configuration highly depends on the dataset, implying no “one-size-fits-all" method dominates the others across all scenarios. This work also identifies the crucial role of underfitting and overfitting in the agent's ability to detect anomalies and resist adversarial examples at specific perturbation thresholds.

In future work, we aim to understand these phenomena better and investigate the underlying reasons explaining the differences in performance and robustness. Specifically, exploring how adversarial attacks could be applied to mislabel benign labels as malicious, in contrast to our current study, presents an intriguing avenue for further investigation. Such attacks would increase the false positive rates and submerge the network with false alarms. This could provide valuable insights into the potential vulnerabilities of DRL-based intrusion detection to false positive adversarial attacks. On the other hand, we acknowledge that the incorporation of formal analysis methods could further enhance the robustness and comprehensiveness of our evaluation. State-of-the-art formal analysis methods such as Scyther, Proverif, AVISPA, BAN Logic, and the ROR model are well-established in the verification of security protocols. In future work, we aim to explore and adapt theoretical verification methods that are suitable for DRL [212]. This could involve developing new frameworks or modifying existing ones to better accommodate the intricacies of DRL-based systems.

Acknowledgements

This work was supported by Mitacs through the Mitacs Accelerate International program and the CRITiCAL chair. It was enabled in part by support provided by Calcul Québec, Compute Ontario, the BC DRI Group, and the Digital Research Alliance of Canada.

Declarations

Conflict of interest The authors have no competing interests to declare.

Ethical standards The authors confirm their compliance with ethical standards.

CHAPTER 5 ARTICLE 3: INVESTIGATING THE PRACTICALITY OF ADVERSARIAL EVASION ATTACKS ON NETWORK INTRUSION DETECTION

Published in *Annals of Telecommunications*, under the special issue *Interactions between Artificial Intelligence and Cybersecurity to Protect Future Networks*, on *28 March 2022*.

Authors Mohamed Amine Merzouk[1,2], Frédéric Cuppens[1], Nora Boulahia-Cuppens[1], Reda Yaich[2]

Institutions [1] Polytechnique Montréal, Canada; [2] IRT SystemX, France

Abstract As machine learning models are increasingly integrated into critical cybersecurity tools, their security issues become a priority. Particularly after the rise of adversarial examples, original data to which a small and well-computed perturbation is added to influence the prediction of the model. Applied to cybersecurity tools, like network intrusion detection systems, they could allow attackers to evade detection mechanisms that rely on machine learning. However, if the perturbation does not consider the constraints of network traffic, the adversarial examples may be inconsistent, thus making the attack invalid. These inconsistencies are a major obstacle to the implementation of end-to-end network attacks. In this article, we study the practicality of adversarial attacks for the purpose of evading network intrusion detection models. We evaluate the impact of state-of-the-art attacks on three different datasets. Through a fine-grained analysis of the generated adversarial examples, we introduce and discuss four key criteria that are necessary for the validity of network traffic, namely value ranges, binary values, multiple category membership, and semantic relations.

Keywords Adversarial machine learning; Adversarial examples; Intrusion detection; Evasion attacks

5.1 Introduction

In recent years, we have seen the integration of machine learning applications in multiple fields, even the most critical ones such as cybersecurity. Functions like intrusion detection now rely on machine learning models to allow better adaptability to the environment and the detection of previously unknown threats [4]. However, while the development of these

models mainly targeted performance, it is only recently that the security concerns raised by machine learning have been addressed. Indeed, recent works have shown the vulnerability of machine learning models to adversarial examples: data instances modified using a small and well-computed perturbation that compromises the prediction of the model [28]. Using adversarial methods, attackers could slightly modify the attributes of their attacks so that it is no longer classified as malicious by the intrusion detection model. This represents a serious threat to the security of critical systems relying on machine learning-based intrusion detection systems.

Adversarial examples on neural networks have attracted a lot of attention in the research community. This has renewed interest in adversarial machine learning, a research field that aims to evaluate and improve the robustness of machine learning models to malicious manipulations [17]. Security researchers have also investigated the impact of adversarial attacks on intrusion detection models, and applied them successfully [74]. Adversarial perturbations were added to malicious instances from the dataset to make them undetectable by the model. However, when applied, these perturbations modify the features without considering the data constraints, which often results in values that are incoherent in the context of network traffic. Even if the generated attacks can evade detection models, they are inconsistent with network constraints and cannot be practically implemented, thus questioning their real threat to intrusion detection systems.

In this article, we study of the practicality of adversarial evasion attacks to determine the properties of the perturbation that create inconsistencies. Our contribution includes: 1. An evaluation of the impact of adversarial perturbations on the intrusion detection model and the data of three different datasets, 2. A detailed study that identifies four concrete criteria of inconsistency that prevent the implementation of the attacks, and 3. A complete environment for the development of intrusion detection models and adversarial attacks against them[1].

To achieve this study, we use seven major adversarial attacks to perturb malicious network traffic. The generated adversarial examples are then fed to a neural network intrusion detection model to study their impact on the detection rate and measure the applied perturbation. Moreover, we comprehensively examine the adversarial examples to identify inconsistencies in their feature values. Through an in-depth analysis, we introduce and discuss four key criteria that prevent the implementation of adversarial examples, and thus invalidate the attacks. According to these criteria, we present concrete examples from the data that prove its inconsistency. Our experiments are carried out on three well-known datasets: NSL-KDD [78], UNSW-NB15 [88], and CIDDS-001 [90].

[1] `https://github.com/mamerzouk/adversarial_analysis`

The rest of this article is organized as follows: in Section 5.2, we review some influential work on adversarial examples; in Section 5.3, we describe the steps followed to perform the experiments; in Section 5.4, we present the results of our experiment in terms of impact on the detection rate and the data; in Section 5.5, we provide a detailed analysis of adversarial examples to evaluate their consistency, we also introduce and discuss the criteria that systematically invalidate them and prevent their implementation; finally, in Section 5.6, we conclude our work and orient future research.

5.2 Related work

Interest in adversarial manipulations against machine learning models started in the early 2000s [19], but it is only after 10 years, when Szegedy *et al.* were able to fool neural networks with a small perturbation [28], that it became a factual preoccupation. At a moment when neural networks were gaining tremendous popularity for their performance, the phenomenon of adversarial examples represented a real threat to the numerous intended applications of these models. This threat motivated multiple research initiatives exploring different ways to generate adversarial examples and countermeasures to enhance the robustness of machine learning models. In this section, we present a brief overview of the most influential methods for the generation of adversarial examples (also designated as adversarial attacks). The presented methods will later be applied to intrusion detection datasets and comprehensively analyzed in Section 5.4. To keep the review up-to-date, we also present some adversarial attacks introduced more recently. Finally, we address previous works studying the vulnerability of intrusion detection models to adversarial examples.

5.2.1 Adversarial examples generation methods

In their study, Szegedy *et al.* [28] proposed a method based on box-constrained L-BFGS to find adversarial examples. Despite its effectiveness, this method required complex calculations, making it slow. Goodfellow *et al.* proposed the *Fast Gradient Sign Method* (FGSM) [30] to allow a quicker generation of adversarial examples. The gradient of the loss function, which usually serves to update the parameters of the model, is used by FGSM to update (with the intent to perturb) the inputs of the model to degrade its accuracy.

FGSM is an untargeted attack since it does not aim to generate adversarial examples that will be misclassified in a specifically targeted class. Rather, it aims for a misclassification regardless of the class, as long as it is not the correct class. It is also said to optimize the L_∞ distance norm, which measures the maximum amount of perturbation applied to any of

the features.

Basic Iterative Method (BIM) is another gradient-based method proposed by Kurakin *et al.* [213] as an extension of FGSM. Concretely, this attack applies FGSM through multiple iterations with a small perturbation magnitude to generate more optimized adversarial examples. This method also introduced a clipping function that keeps the values of the features in a specific interval. Just like FGSM, this method is untargeted and optimizes the L_∞ distance norm.

Jacobian-based Saliency Map Attack (JSMA) is a different method proposed by Papernot *et al.* [29]. Instead of the gradients of the loss function, this attack uses the Jacobian matrix of the model (the derivatives of the outputs of the model with respect to its inputs) to construct a saliency map. This saliency map highlights, for a targeted class, the contribution of each feature. In each iteration, the feature with the highest saliency value is perturbed, until the example is misclassified. JSMA is a targeted attack that optimizes the L_0 distance norm, this norm measures the number of perturbed features.

DeepFool is a method proposed by Moosavi-Dezfooli *et al.* [69]. It approximates the smallest distance between the original example and the closest classification boundary with another class. This distance represents the minimal perturbation necessary to misclassify the example. Since it looks for the closest boundary regardless of the class, DeepFool is an untargeted attack. It optimizes the L_2 distance norm, which measures the Euclidean distance between the original and the adversarial example.

Carlini and Wagner (C&W) [70] introduces an objective function and optimizes it to generate adversarial examples. Several objective functions were proposed by the authors such that the example is misclassified if and only if the objective function is negative. An optimization method, such as Adam, is then used to minimize the objective function along with the distance norm of the perturbation. The main C&W attack was designed to optimize the L_2 distance norm, but the authors further proposed two iterative variants that optimize the L_∞ and L_0 distance norms. Since the objective function allows the choice of a target class, this method is targeted.

With growing interest for adversarial machine learning, new methods to generate adversarial examples are regularly introduced. To tackle the problem of attacking a model in a black-box setting, the first intuition was to exploit the phenomenon of transferability between models [71]. Chen *et al.* proposed using the Zeroth Order Optimization (ZOO) attack that estimates the gradients using the confidence scores of the input data. More recently, Gao *et al.* introduced the Transferable Attentive Attack (TAA) [214]; their method bases its perturbation on the attended regions and features of an image (in the case of image

classification). Another way to generate adversarial examples is using Generative Adversarial Networks (GANs) [36]. This method was first explored by Xiao *et al.* [215], and more recently by Bai *et al.* [216]. The latter introduced the Attack-Inspired GAN (AI-GAN), where a generator, a discriminator, and an attacker are trained jointly. In addition to image recognition, text classification models have also been proven to be vulnerable. Garg and Ramakrishnan [217] proposed BAE, a black-box attack that uses contextual perturbations from a BERT model [218] to generate adversarial examples. Since this paper focuses on the attacks that target intrusion detection models, we will not further explore these attacks. More details on adversarial attacks and defenses can be found in Yuan *et al.* [31].

5.2.2 Adversarial examples and intrusion detection

The founding works carried out on adversarial examples focused on a single use case: image classification. Thus, the major methods for the generation of adversarial examples were designed and tested on images. However, as machine learning began to take considerable importance in cybersecurity defense tools, especially intrusion detection systems, the threat of adversarial examples rapidly attracted the attention of security researchers. Several research initiatives have already explored the vulnerability of machine learning-based intrusion detection systems to adversarial examples. Surveys such as Martins *et al.* [74] provide an extensive review of these works. Their methodology often consists of feeding adversarial examples to the model and evaluating its accuracy. The accuracy decrease is then used to prove the vulnerability of the model.

However, little attention was given in previous works to the consistency of the generated adversarial examples with network constraints. This issue does not arise at the data level, but when the adversarial examples are implemented in an end-to-end network attack. Therefore, without consistent adversarial examples, adversarial attacks do not represent a concrete threat to network intrusion detection. This is what motivated our focus on the consistency of adversarial examples.

5.3 Methodology

To perform our experiments, we train neural network models on three datasets and generate adversarial examples against them with different methods. The impact of these adversarial examples is then evaluated in Section 5.4 and they are further analyzed in Section 5.5. In this section, we describe each step of the methodology that we followed in our experiments. We describe the datasets that we used and justify our choice, the pre-processing we applied

to the data, the architecture of the model, and the attacks that we performed on it.

5.3.1 Datasets

In recent years, several datasets have emerged to allow the research community to develop efficient machine learning models for intrusion detection. A detailed survey of network intrusion detection datasets was proposed by Ring *et al.* [173]. The authors evaluated 34 datasets based on 15 properties that they identified. We based our choice on their evaluation. To corroborate our results, we use three distinct network datasets with different structures. In this section, we define and justify the choice of each dataset.

NSL-KDD

NSL-KDD [78] is a widely used network dataset based on the KDD CUP 99 dataset which was captured in 1998. It enhances certain aspects of the original data, namely reducing redundancy to lower the bias and selecting reasonable subsets, but keeps the same features. Despite its popularity, the NSL-KDD dataset has aged considerably and no longer accurately represents modern network traffic. We chose NSL-KDD to allow the comparison of our results with previous works, many of which use this dataset.

UNSW-NB 15

UNSW-NB 15 [88] is a recent dataset that is often used as a replacement for NSL-KDD. It was captured in 2015 in a virtual environment over 31 h. The data features are the basic flow attributes with additional generated features. It is therefore not a standard format, and it could hardly be reproduced since the environment is not publicly available. We use this dataset because it is relatively recent and has a growing popularity, which allows comparison with other works.

CIDDS-001

CIDDS-001 [90] is a more recent dataset that was captured in 2017 in a virtual environment over 4 weeks. It aims to be as realistic as possible by simulating normal user behavior in addition to the attacks. The data is in the standard unidirectional NetFlow format, and all the environment used to generate the data is publicly available. This dataset is the most recent one used in this study. Even more recent datasets, such as CIC-DDoS2019 [219], have also been considered for this study, but CIDDS-001 was favored since it contains various

types of attacks, uses a standard format, and has an open-source environment, enabling reproducibility.

5.3.2 Pre-processing

Before feeding the data to the model, we apply some pre-processing:

- One Hot Encoding: Since neural networks cannot process categorical data, we use this method to encode each category as a binary feature. Thus, for each example, only one of these binary features holds the value 1 to indicate the belonging to that category. For example, instead of a categorical feature `Protocol` that can be `[TCP]`, `[UDP]`, or `[ICMP]`, we use 3 binary features with the following possible combinations: `[1, 0, 0]`, `[0, 1, 0]` or `[0, 0, 1]`.
- Min-Max normalization: The data is normalized to prevent features with large values from biasing the classification; the values of the features are scaled in the range $[0, 1]$.
- Binary labels: Instead of using multiple attack categories, we merge them into a single binary feature that indicates if the instance is malicious or benign (binary classification).

5.3.3 Target model

Similar to most of the previous works, the target model used in our experiments is a Multi-Layer Perceptron (MLP) neural network with two hidden layers using ReLU activation functions and a Softmax output. The model is trained for 1000 epochs using the cross-entropy loss function and the Adam optimizer with a learning rate of 0.001. The model is made as simple and as close as possible to the models used in similar works to allow realistic comparisons. Thus, no regularization has been applied to avoid introducing any bias. The neural networks used in our experiments are implemented using PyTorch [208].

5.3.4 Adversarial examples generation

After training our model on the training set and evaluating its performance on the testing set, we apply the following state-of-the-art methods on the attack samples of the testing set to generate adversarial examples:

- Fast Gradient Sign Method (FGSM) [30],
- Basic Iterative Methods (BIM) [213],

- DeepFool [69],
- Carlini & Wagner (L_2, L_∞, L_0) [70],
- Jacobian-based Saliency Map Attack (JSMA) [29].

In order to evaluate the perturbation potential of each method equally, none of them uses a clipping function. The attacks are implemented using Adversarial Robustness Toolbox (ART) [209], a library we contributed to as part of this work. Details about the implementation of the experimental environment can be found in our public code repository[2].

5.4 Evaluation of the impact of adversarial attacks

In this section, we evaluate the impact of the adversarial attacks mentioned above on the detection rate of the models trained with each of the three datasets. We also discuss the impact of the adversarial attacks on the original data, mainly with respect to the number of perturbed features (L_0 distance norm) and the maximum perturbation applied to a feature (L_∞ distance norm). Then, we examine the features that are the most frequently perturbed by each method, before addressing the common successful adversarial examples shared between different attacks.

5.4.1 Impact on the model

After completing their training process, the models are evaluated on the testing set. As a performance metric, we use the True Positive Rate (TPR), also referred to as the detection rate [4]. This choice is motivated by our focus on *adversarial evasion attacks*: adversarial examples that modify an attack so that it is misclassified as benign. Thus, only the positive (malicious) instances of the testing set are considered in the evaluation.

The models achieved state-of-the-art performance on all three datasets. Table 5.1 reports that the model trained on NSL-KDD was able to detect 74.20% of the unmodified attack data. Even if these performances are consistent with previous works, it is weaker than the models trained on UNSW-NB15 and CIDDS-001, which were able to detect 88.01% and 99.52% of the unmodified attacks respectively. This difference could be justified by the complexity of the NSL-KDD dataset and the large number of attack types, which makes the detection more challenging for some types, such as `Remote-to-local` or `User-to-root`.

[2] `https://github.com/mamerzouk/adversarial_analysis`

Table 5.1 Detection rate and distance metrics.

Metric	Detection rate			Mean L_0 norm			Mean L_∞ norm		
Dataset	NSL-KDD	UNSW-NB15	CIDDS-01	NSL-KDD	UNSW-NB15	CIDDS-01	NSL-KDD	UNSW-NB15	CIDDS-01
Unmodified	74.20%	88.01%	99.52%	0	0	0	0	0	0
FGSM	25.80%	11.99%	0.47%	120.99	196	35	0.1	0.1	0.1
BIM	25.79%	11.99%	0.47%	120.94	195.98	34.99	0.1	0.1	0.1
DeepFool	26.01%	12.09%	1.04%	120.99	195.99	34.99	0.05	0.01	0.01
C&W L_2	24.30%	11.17%	35.87%	13.85	16.04	4.99	0.47	0.2	0.16
C&W L_∞	37.57%	18.59%	0.56%	16.96	22.89	5.09	0.06	0.07	0.15
C&W L_0	23.26%	11.03%	9.03%	3.74	1.17	1.1	0.85	0.48	0.25
JSMA	25.79%	11.99%	0.47%	2.32	2.17	2.01	0.18	0.29	0.1

In a practical use case, the models would be ready for deployment in an intrusion detection system. They can detect various network attacks with good accuracy. However, the following results demonstrate that slightly modifying the features of the data could strongly impact the accuracy of the model.

Figure 5.1 shows the detection rate of the model on the training set before and after applying different adversarial methods to modify the data. The first column of Table 5.1 reports the same values in detail. We can examine how all the methods had a major impact on the detection rate of the model for all three datasets. However, we observed that the impact of an attack is closely linked to the type of dataset. On NSL-KDD, almost all attacks were able to decrease the detection rate below 26%, except for C&W L_∞-attack which only achieved 37%. Although the UNSW-NB15 model showed a better detection rate than the NSL-KDD model, we can see that it is also much more impacted by adversarial attacks. Indeed, the detection rate fell around 12% for most attacks, with the same exception for C&W L_∞-attack. The effects of adversarial attacks were the most noticeable on CIDDS-001, whereas it was the best performing model. The detection rate fell under 1% for most attacks except for C&W L_∞-*attack* (9.03%) and C&W L_∞-*attack* (35.87%), probably because of a local minimum. We notice that the models with higher detection rates on unmodified data were much more impacted by the adversarial attacks.

An interesting observation is that FGSM, BIM, and JSMA had the same impact on the detection rate. This observation was made on all three datasets, even after multiple random iterations. The similarity between FGSM and BIM is not surprising, knowing that both methods use the sign of the gradient of the loss function that they apply to all features. However, JSMA is fundamentally different; it only perturbs a subset of the features selected using a saliency map. To the best of our knowledge, no clear justification is known for this phenomenon. Later, in Section 5.4.4, we investigate the common successful adversarial

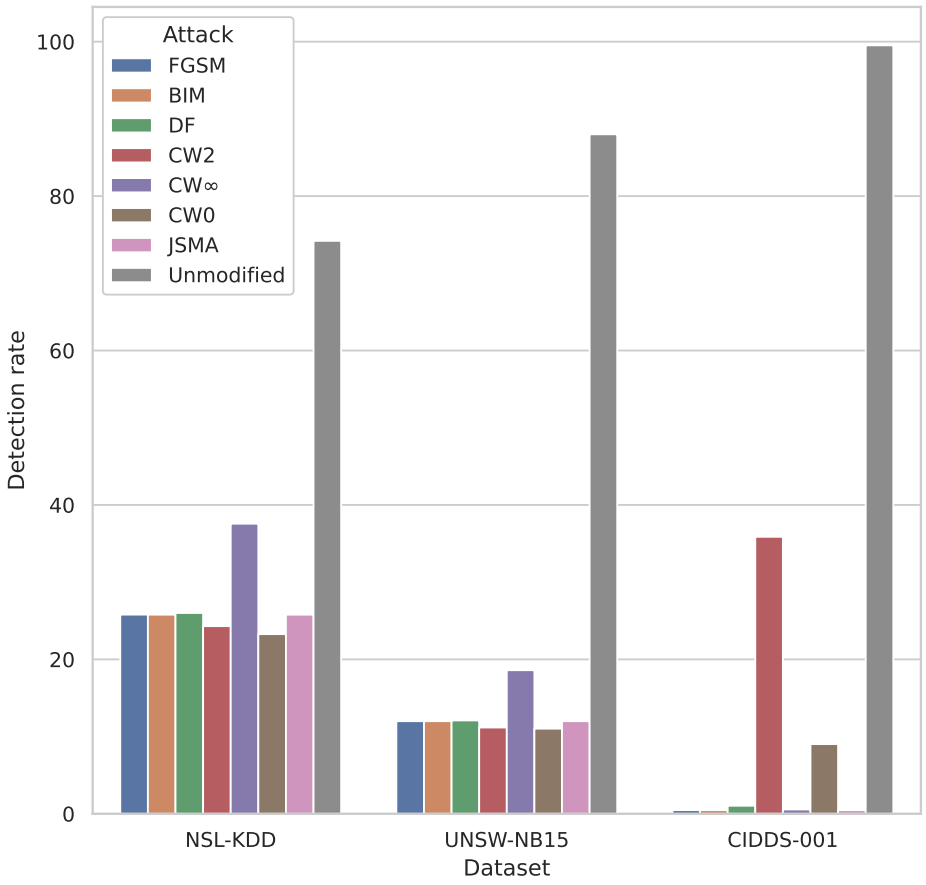


Figure 5.1 Detection rate with and without adversarial perturbation.

examples shared between the different methods to understand the origin of these similarities.

5.4.2 Impact on the data

Although different methods had similar impacts on the detection rate, they do not modify data in the same way. One of the most important criteria to evaluate an adversarial example is the number of modified features (L_0 distance norm) and the maximum perturbation amount (L_∞ distance norm). Figure 5.2 shows the mean and the standard deviation of the proportion of perturbed features, while Figure 5.3 shows the mean and standard deviation of the maximum perturbation amount. The precise values of the L_0 and L_∞ distance norms are respectively reported in the second and third columns of Table 5.1.

Figure 5.2 describes accurately how FGSM, BIM, and DeepFool apply their perturbation on almost 100% of the features. This is because the number of perturbed features is not a measure these attacks try to optimize. In contrast, they rather spread their perturbation on

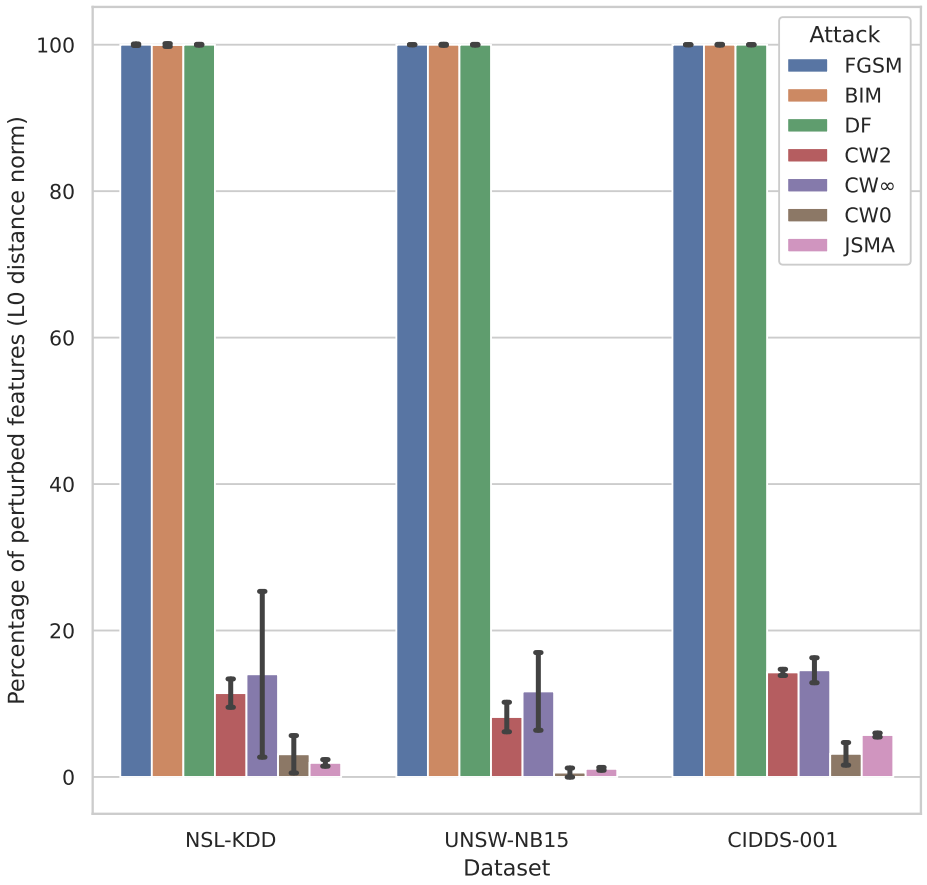


Figure 5.2 Mean and standard deviation of the proportion of perturbed features.

the whole features space to minimize the amount of necessary perturbation (or the Euclidean distance in the case of DeepFool). This justifies the observation in Figure 5.3: these same attacks are the ones with the lower perturbation amount.

C&W L_∞-*attack* can also be considered in the same category, even though it did not perturb all the features. In fact, its modest impact on the model, as described in Section 5.4.1, can be explained by the limitation we imposed on the maximum perturbation: A perturbation threshold is often applied to attacks that optimize the L_∞ distance norm like FGSM, BIM, and C&W L_∞. While FGSM and BIM were able to spread their perturbation on all the features to generate effective adversarial examples, C&W L_∞-*attack* struggled to do so and was only able to perturb a certain proportion of features.

The approach based on the distribution of the perturbation on the whole feature space to minimize its amount is effective on unstructured data like images. This is mainly because the data features (pixels) do not hold strong semantic values; a feature only indicates the intensity of a color, it does not make sense on its own. Changing the value of a feature will not affect

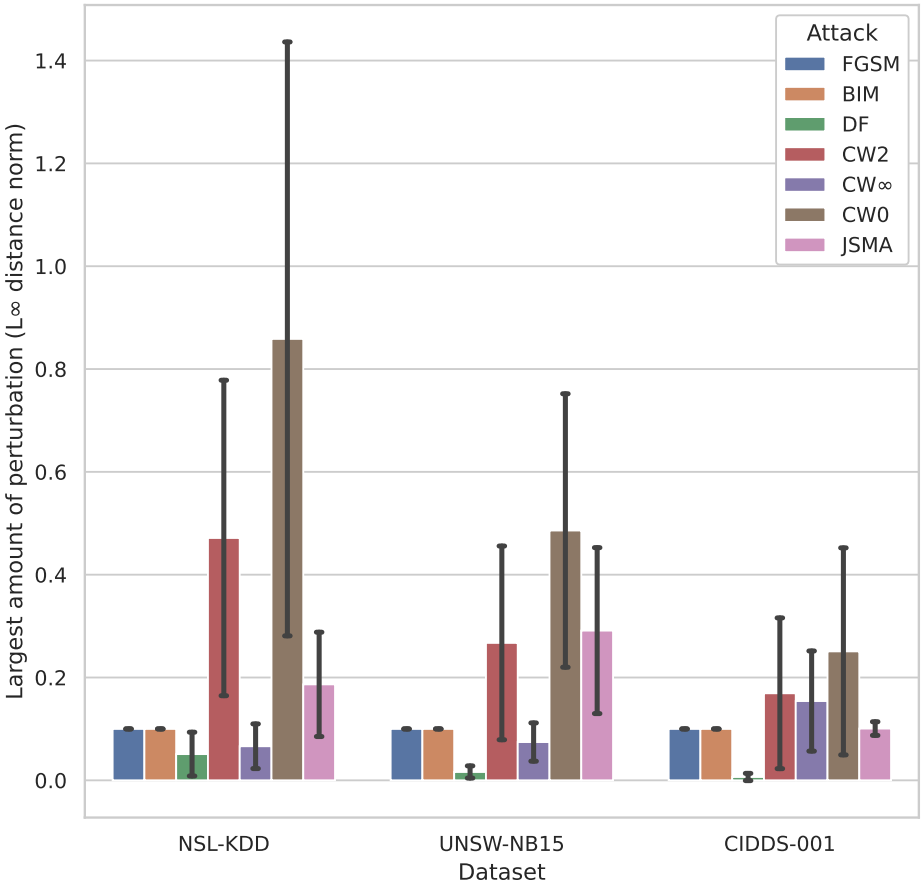


Figure 5.3 Mean and standard deviation of the maximum perturbation.

the structure of the image. Hence, applying a smaller modification on all the features makes it imperceptible to humans, which is often the adversarial objective. However, when it comes to structured data like network traffic, this approach requires complete control over all the network attributes. This requirement is extremely complex, considering the structure of the data, the constraints on the meaning of the features, and the dependencies between features. Moreover, the network traffic could hardly keep its original purpose, possibly malicious, if its attributes are significantly modified. Therefore, the attacks following this approach are unsuitable for use on structured data, especially network traffic.

Even though it does not aim to optimize the number of modified features, we observe in C&W L_2-*attack* the tendency to perturb only less than 15% of the features on average. This proportion varies slightly between adversarial examples as shown in Figure 5.2. However, the perturbation amount is noticeably higher than that in previous attacks, especially on NSL-KDD, where the average L_∞ distance of this attack reaches 0.47 with a high variance.

The same behavior is amplified by C&W L_0-*attack* that reduces the proportion of perturbed features to only 3%, which is notably less than previous attacks. But on the other hand, it shows a much higher amount of perturbation applied on the adversarial examples. Figure 5.3 reveals a mean L_∞ distance of 0.85 for this attack with a high variance.

In the case of a network attack, this would imply the application of a large perturbation on some network attributes. This perturbation would probably affect the objective of the attacker. It is even inconceivable, for most attributes, that this amount of perturbation can be applied without invalidating the network traffic. Therefore, these attacks are unsuitable for use on network data.

Figure 5.1 shows competitive results for JSMA on all three datasets while maintaining the distance norms at reasonable values. For this attack, Figure 5.2 reports the best mean number of perturbed features on NSL-KDD, and the second best on UNSW-NB15 and CIDDS-001 after C&W L_0-*attack*. At the same time, Figure 5.3 shows that JSMA has always a much smaller mean L_∞ distance norm than C&W L_0-*attack* with a smaller variance also. At first sight, these properties make JSMA the most well-suited attack to perturb network traffic data. In Section 5.5, we further analyze the adversarial examples generated by each method to evaluate their consistency.

5.4.3 Perturbed features

As we have seen previously, different attacks use different approaches to perturb the data, particularly regarding the choice of features. In this section, we study how frequently each attack perturbs each feature.

Figure 5.4 shows a heat map of the percentage of adversarial examples that perturb each feature of the UNSW-NB15 dataset. A similar analysis specific to the NSL-KDD dataset is proposed in Merzouk *et al.* [182]. For each attack (in columns), the values represent how many adversarial examples of this attack perturb a given feature. First, we observe a high intensity on the three first columns. Indeed, each of FGSM, BIM, and DeepFool perturbs almost all the features in 100% of their adversarial examples. This observation supports the concerns raised in Section 5.4.2 about the suitability of these attacks on network data. The level of control required to apply such perturbations on all the network attributes is hardly achievable.

We also notice similarities in the features perturbed by C&W L_2-*attack* and C&W L_∞-*attack*. This can be explained by the fact that these two attacks are fundamentally the same attack optimizing different distance norms. C&W L_∞-*attack* is just an iterative version of C&W

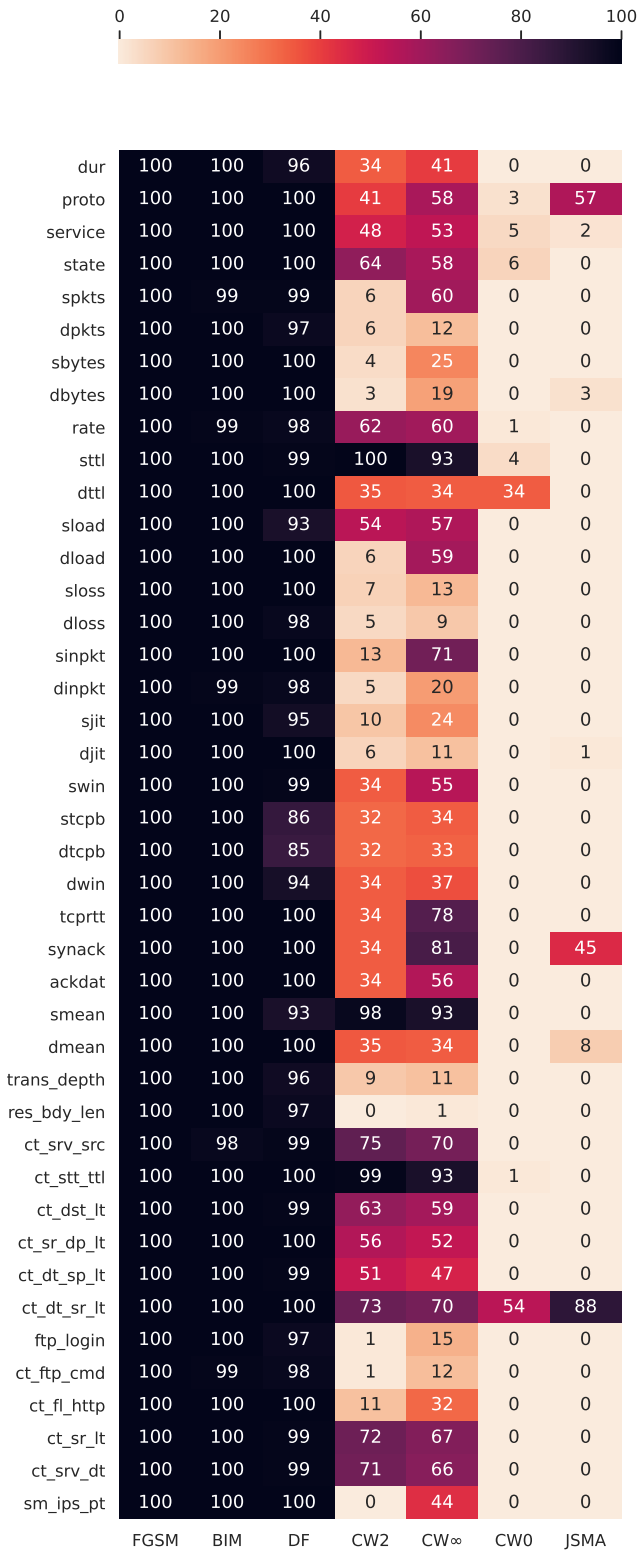


Figure 5.4 Percentage of adversarial examples perturbing each feature on UNSW-NB15.

L_2-attack. Thus, they target their perturbation on the same features.

C&W *L_0-attack* and JSMA stand out with a perturbation that is focused on very few features. Indeed, these attacks are designed to optimize the number of perturbed features. Therefore, they carefully choose the features to perturb: C&W *L_0-attack* does it by elimination and JSMA by selection. Attacks with a lower L_0 norm are more suitable for network data since they do not require control over many features. However, they often require a heavier perturbation amplitude that might invalidate the network traffic, especially with C&W *L_0-attack*.

Overall, we notice that the only consensual feature that is perturbed by all the attacks is `ct_dst_src_ltm`. This feature represents the "*Number of connections of the same source and the destination address in 100 connections according to the last time*" [88]. In fact, this feature is a relevant element for the detection of Probe and Denial-of-Service (DoS) attacks.

5.4.4 Common adversarial examples

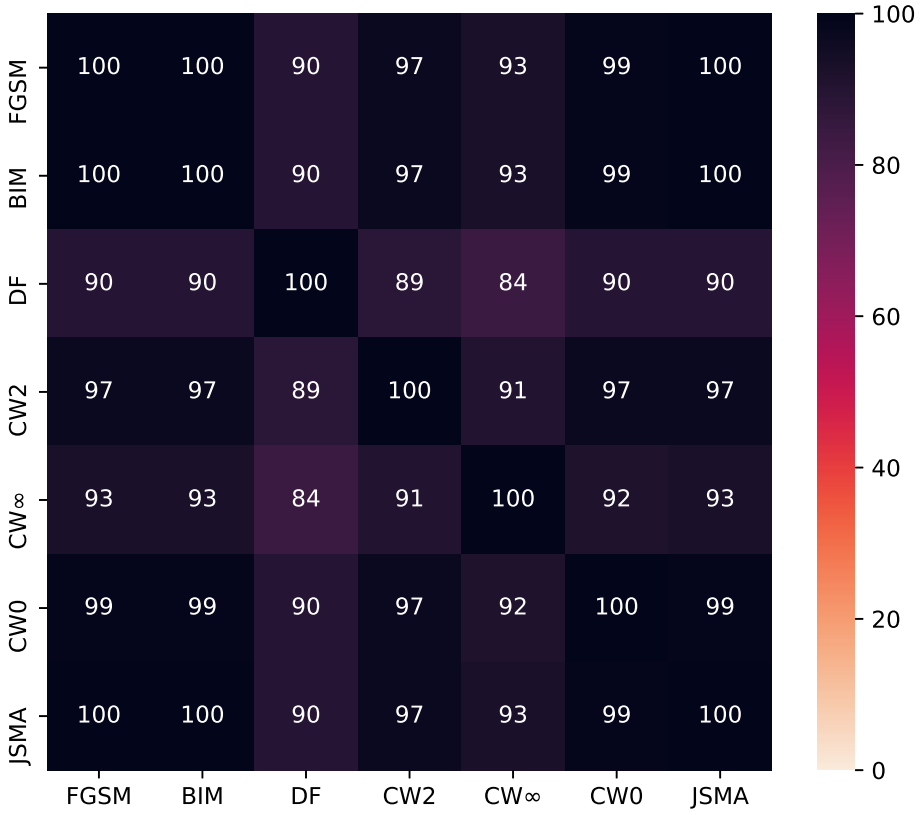


Figure 5.5 Percentage of common successful adversarial examples on UNSW-NB15.

In this section, we address the observation made in Section 5.4.1 about the identical impacts

caused by FGSM, BIM, and JSMA on the model. We noticed the same detection rate for these methods on each dataset, even after multiple random initializations, which indicates that they have the same number of successful adversarial examples. In this section, we investigate the similarities between the successful adversarial examples of each attack.

Figure 5.5 shows a heat map of the percentage of common successful adversarial examples on the UNSW-NB15 dataset. The first observation is that FGSM, BIM, and JSMA share 100% of their successful adversarial examples. Therefore, we conclude that these attacks succeeded in the same instances, despite the fundamental difference in their approach for the generation of adversarial examples.

Another interesting observation is that DeepFool shared noticeably less adversarial examples with other methods while showing comparable results. This suggests that DeepFool was able to succeed on instances where the other methods failed. C&W L_∞-*attack* also presents less common adversarial examples, but it is mainly justified by its weaker performance.

5.5 Analysis of the adversarial examples

In Section 5.4, we discussed the impact of adversarial attacks on the detection rate of intrusion detection models and also the distance norms of the perturbation applied by these attacks. We observed properties that make some attacks unsuitable for structured data like network traffic. Indeed, some require the modification of a large number of attributes; others require the application of a significant perturbation. Both requirements fall out of the control of attackers trying to evade an intrusion detection system. They cannot manipulate all the features, and for the ones they can manipulate, they cannot apply any amount of perturbation. The transition between the adversarial examples at the data level and the attacks at the network level is thus impossible.

In this section, we analyze the adversarial examples generated by each method on all three datasets. We examine the values of their attributes to evaluate their consistency with network constraints. Throughout this analysis, we identify several criteria that prevent the implementation of these attacks on real networks, and thus, invalidate the adversarial examples. These criteria can be used on practical attacks to spot adversarial examples that are impossible to implement. If these adversarial examples can be spotted and dismissed, it would maximize the success probability of the evasion attack. In the case of a Distributed Denial of Service (DDoS), the criteria would identify the perturbations that assign inconsistent feature values; an example would be a packet size smaller than the minimum possible size. Knowing that these values cannot be implemented, the corresponding examples would

be automatically discarded. The attack will then only focus on the practically implementable examples to evade the detection.

The analysis revealed two types of criteria: syntactic-based and semantic-based. The first three criteria we identified are syntactic-based; they mainly depend on the feature space of network traffic. Since adversarial examples are not just selected from a set of possible attacks, but rather generated by modifying original attacks, the perturbation often brings the instance out of the feature space. It is thus necessary, during the generation process, to put constraints on the features values in order to avoid getting out of the feature space. The last criterion is semantic-based; it does not consider the features individually, but rather examines the coherence of feature values with respect to other features. In the following sections, we define each of these invalidation criteria and illustrate them with concrete examples.

5.5.1 Invalid value ranges

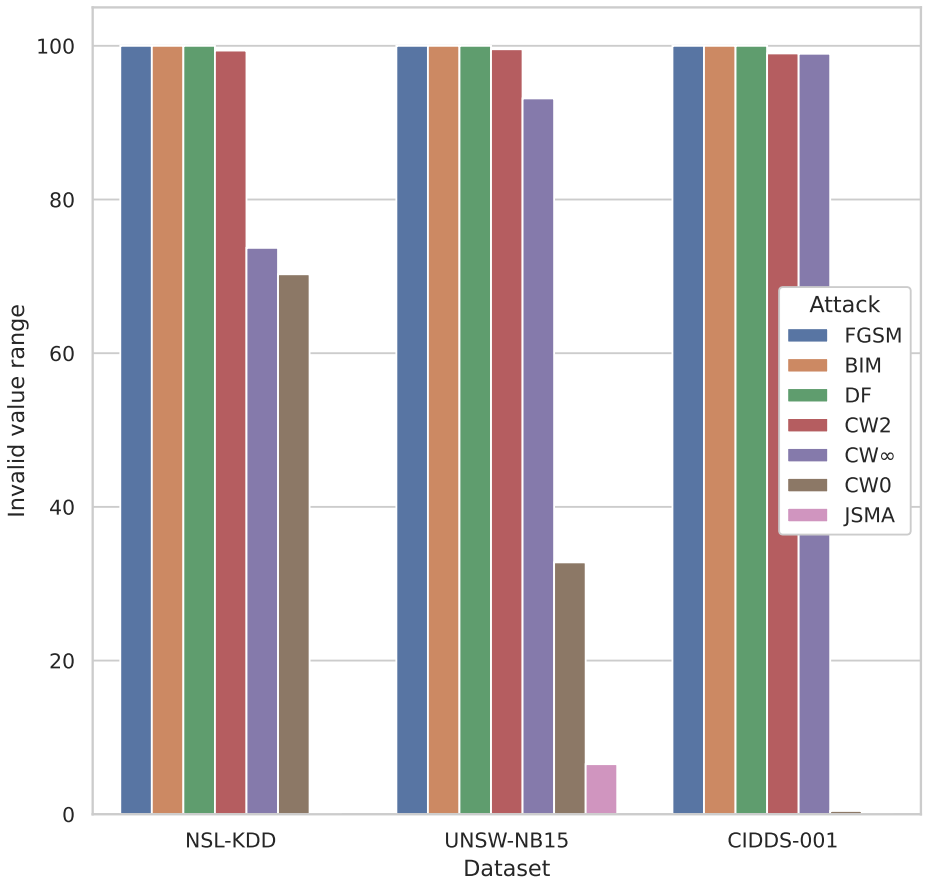


Figure 5.6 Percentage of adversarial examples with invalid value ranges.

The structure of network data imposes constraints on the attributes of network traffic, one of

these constraints is their value range. For each attribute, there is a specific interval it cannot exceed. Otherwise, the value is no longer consistent with the definition of the attribute. For example, imagine the attribute that represents the size of a packet holding a negative value, or a probability attribute with a value greater than 1.

Since adversarial attacks do not consider these constraints, the generated adversarial examples can contain values that fall out of the interval of the attribute. These values have no implementation on a real network and invalidate the adversarial examples.

Figure 5.6 shows the percentage of adversarial examples with at least one feature holding a value out of range. The borders of the ranges were defined as the minimum and maximum values seen in the original testing set. The results of this figure are detailed in Table 5.2. We can see on all three datasets that for FGSM, BIM, DeepFool, and C&W L_2-*attack*, almost 100% of adversarial examples contain features with values out of range. These adversarial examples are therefore invalid for a real network implementation. C&W L_∞-*attack* shows slightly fewer invalid examples on NSL-KDD (73.70%) but is still high on other datasets. In contrast, C&W L_0-*attack* shows its higher proportion of invalid examples on NSL-KDD (70.27%), but much less in UNSW-NB15 (32.77%), and almost none on CIDDS-001 (0.43%). JSMA shows almost no invalid examples on both NSL-KDD and CIDDS-001, and only 6.52% on UNSW-NB15. The interesting results achieved by L_0-*attack* and specially JSMA could be a consequence of their approach. Indeed, by modifying only a small number of attributes, they reduce the risk of falling out of range. We can also notice in Figure 5.4 that in UNSW-NB15 these attacks focus their perturbation mostly on quantitative features whose value ranges are larger.

To illustrate this observation, we take examples from the generated adversarial examples. On NSL-KDD, adversarial examples generated by DeepFool put the value of `Dest-bytes` which represents the "*number of data bytes transferred from destination to source in single connection*" [220] to -200 MB. That is to say that an attacker could hardly send or receive this amount of data. On UNSW-NB15, adversarial examples generated by C&W L_0-*attack* set the value of the attribute `dttl` which represents the "*source to destination time to live (TTL)*" [88] to 474. Knowing that the TTL on IP packets is an 8 bits field, its maximum value is 255. This value is therefore impossible to implement on a real packet. On CIDDS-001, adversarial examples generated by FGSM perturbed the attribute of the duration of connection to -22 seconds. This is another value that has no interpretation in real networks.

5.5.2 Invalid binary values

Another constraint that applies to network data is the value of binary attributes. A large number of features on network datasets are binary: they only indicate the presence or the absence of a property using the values 1 and 0 respectively. Any other value has no interpretation on the network. However, adversarial attacks do not consider this constraint. Thus, they apply perturbations that assign non-binary values to binary features. This results in the invalidation of adversarial examples holding these values. This criterion can also be generalized to all discrete values that are perturbed with continuous values.

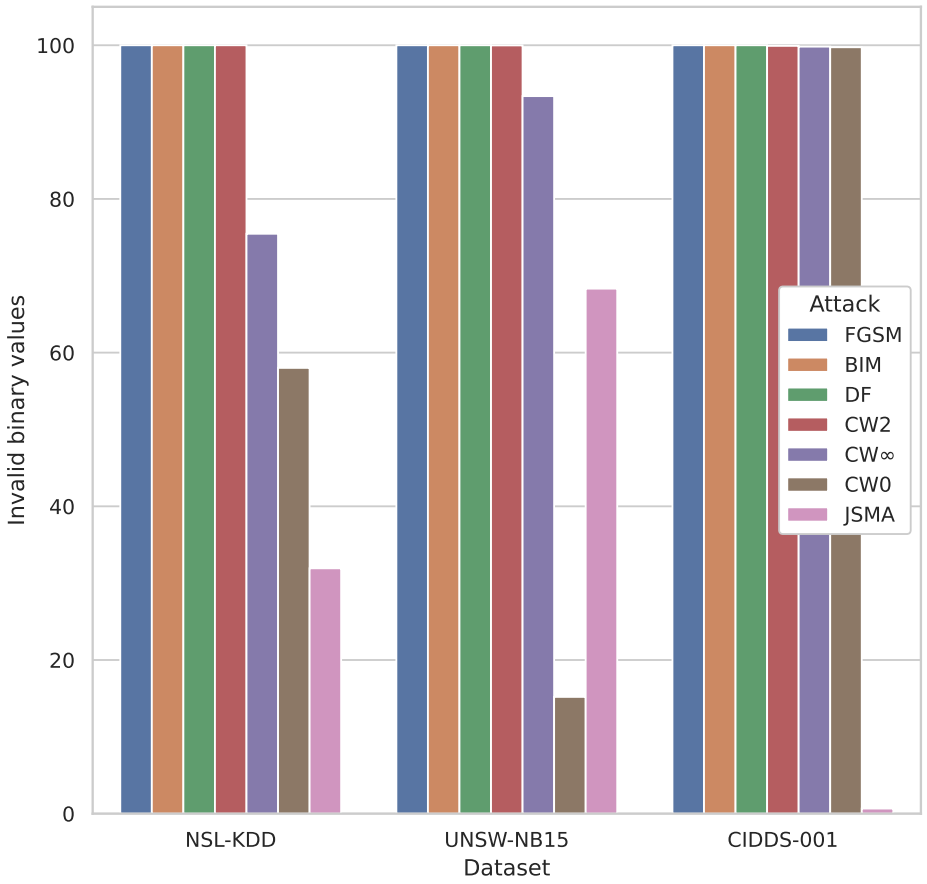


Figure 5.7 Percentage of adversarial examples with invalid binary values.

Figure 5.7 shows the percentage of adversarial examples with at least one binary feature holding a non-binary value. As for the invalid ranges, FGSM, BIM, DeepFool, and C&W L_2-*attack* have non-binary values on almost all their adversarial examples, which invalidates them. C&W L_2-*attack* and C&W L_0-*attack* also show results similar to the previous criteria, except on CIDDS-001 where almost all adversarial examples generated by C&W L_0-*attack* hold non-binary values. However, on this criterion, JSMA shows a higher proportion of invalid

examples on NSL-KDD (31.39%) and UNSW-NB15 (68.32%), but it keeps only 0.67% on CIDDS-001.

As an example of this criterion, we can take the feature `is_sm_ips_ports` in UNSW-NB15 which indicates: "*if source and destination IP addresses are equal and port numbers are equal, this variable takes value 1 else 0*" [88]. When BIM (or FGSM) applies its perturbation $\epsilon = 0.1$ on this feature, it is not sufficient to move it from a state to the other. Thus, we find values like 0.1 on this binary feature, which has no interpretation.

5.5.3 Invalid category membership

In addition to numerical and binary features, network datasets also contain categorical (nominal) features. Since neural networks cannot process this type of feature, we used the one-hot-encoding method in Section 5.3.2 to encode each category in a separate binary feature. Thus, among the binary features derived from the same categorical feature, only one single feature can hold the value 1, all the others hold the value 0. This property is unknown to adversarial attacks. Therefore, the perturbation they apply can assign non-zero values to multiple categories. Besides the fact that the non-binary values invalidate the traffic, membership in multiple categories of the same categorical feature is often inconsistent. For example, the protocol feature of an example cannot be `TCP` and `UDP` at the same time; these two categories are mutually exclusive.

Figure 5.8 shows the percentage of adversarial examples that activate more than one category of a categorical feature in their respective datasets. We observe that FGSM, BIM, and DeepFool have this property in 100% of their adversarial examples, regardless of the dataset. We notice interesting results from C&W L_2-*attack* and C&W L_0-*attack* that show almost no invalid adversarial examples on any dataset regarding this criterion. C&W L_∞-*attack* and JSMA showed close results, with almost no invalid examples on CIDDS-001. Examples of multiple category membership can be found in all three datasets, on attributes like the protocol, the service, or the flag features.

5.5.4 Invalid semantic relations

Further investigation on the adversarial examples revealed the importance of the semantic aspect of the data; semantic inconsistencies can also invalidate adversarial examples. Unlike unstructured data, such as images where the features (pixels) do not hold strong semantic values, each feature of network data has a semantic link to a host or network attribute. Thus, even if the value respects the type of the feature, it does not necessarily make sense in the

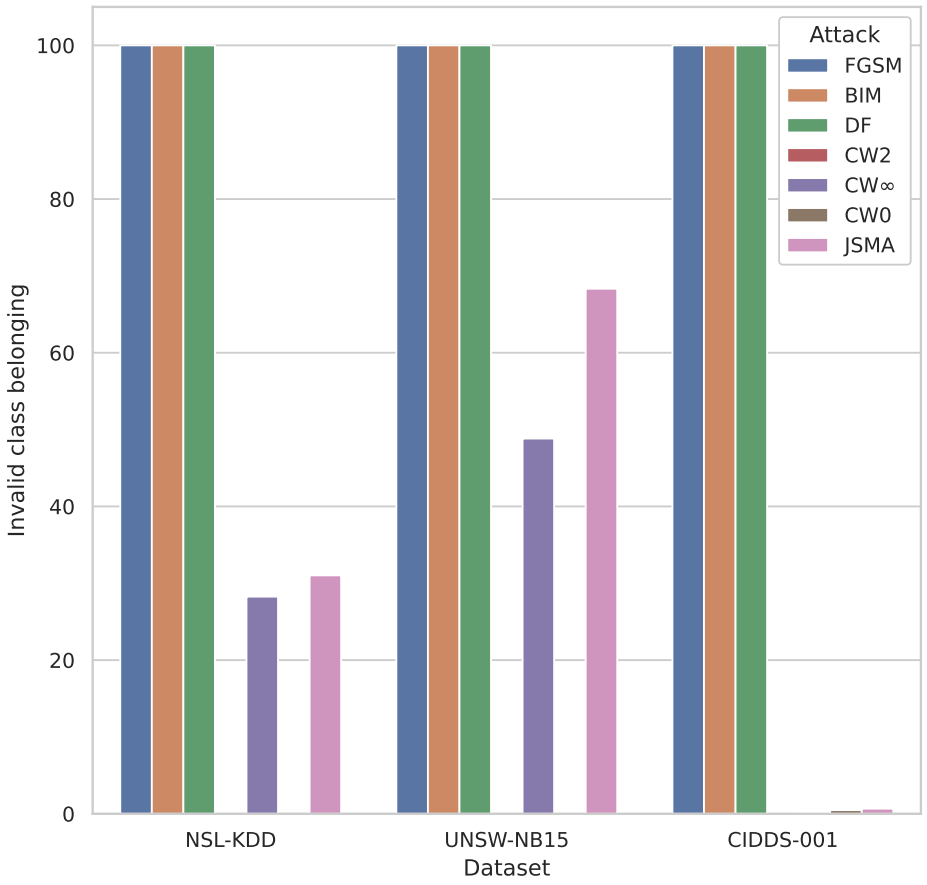


Figure 5.8 Percentage of adversarial examples with invalid category membership.

context of network traffic. Furthermore, features have semantic dependencies between them; the value of a feature might not make sense considering the value of another feature. For example, the value of the service feature can be SSH, but it does not make any sense in a Web attack. Even considering an attack where the service can be SSH, it would be incompatible with the value UDP of the protocol feature.

However, adversarial attacks do not take into account these constraints and their perturbation breaks the semantic consistency of the network traffic. Notable examples from the generated adversarial examples often include inconsistencies between the values of the protocol and the service or the values of correlated error rates.

Unlike previous criteria, it is more difficult to evaluate the attacks regarding their semantic consistency. This is because semantic rules and relations between features are not always obvious; they have to be well defined for each feature and group of features. Listing the whole set of semantic rules was not in the scope of this work; however, we propose the analysis of the correlation matrix as a tool to extract semantic relations between features. Figure 5.9

Table 5.2 Proportion of adversarial examples breaking each invalidation criteria.

Criterion	Value intervals			Non-binary values			Multiple categories		
Dataset	NSL-KDD	UNSW-NB15	CIDDS-01	NSL-KDD	UNSW-NB15	CIDDS-01	NSL-KDD	UNSW-NB15	CIDDS-01
FGSM	100%	100%	100%	100%	100%	100%	100%	100%	100%
BIM	100%	100%	100%	100%	100%	100%	100%	100%	100%
DeepFool	100%	100%	100%	100%	100%	100%	100%	100%	100%
C&WL_2	99.38%	99.55%	99.01%	100%	99.97%	99.92%	0%	0%	0%
C&WL_∞	73.70%	93.15%	98.97%	75.46%	93.38%	99.82%	28.26%	48.83%	0.22%
C&WL_0	70.27%	32.77%	0.43%	58.01%	15.19%	99.74%	0.24%	0.02%	0.48%
JSMA	0.01%	6.52%	0%	31.93%	68.32%	0.67%	31.02%	68.32%	0.67%

shows a heat map of the correlation matrix between the numerical features of UNSW-NB15, and a similar heat map on NSL-KDD is proposed in Merzouk *et al.* [182]. Shades of blue indicate a positive correlation, while shades of red indicate a negative correlation. We can easily notice strong correlations between couples of features, often because they both belong to the source or the destination. We can notice it also between groups of features, notably the connection features (starting with `ct_`) for which we observe a high positive correlation.

5.6 Conclusion and future work

Machine learning is, undoubtedly, revolutionizing cybersecurity defense tools [4], particularly for intrusion detection systems, where it allows more adaptability and the detection of previously unknown threats. However, it is far from being flawless, especially in adversarial environments. Recent advances in adversarial machine learning have shown the limitations of machine learning models, one of which is adversarial examples. They allow an attacker to influence the prediction of a model through a slight modification of its inputs. These adversarial attacks are a serious threat to critical functions like intrusion detection.

In this article, we studied the practicality of adversarial evasion attacks on machine learning-based network intrusion detection. To corroborate our conclusions, we used three different datasets, namely NSL-KDD, UNSW-NB15, and CIDDS-001. We first evaluated the impact of state-of-the-art adversarial attacks on the detection rate of the models. Then we measured the perturbation applied by these attacks in terms of the number of perturbed features and the maximum amount of perturbation. We observed how attacks that optimize one metric often degrade the others. Moreover, we studied the features that are most often perturbed by each attack and the proportion of similar successful adversarial examples. We noticed that some attacks focus their perturbation on specific features, while others spread their perturbation on the whole feature space. This does not prevent different attacks from succeeding on the

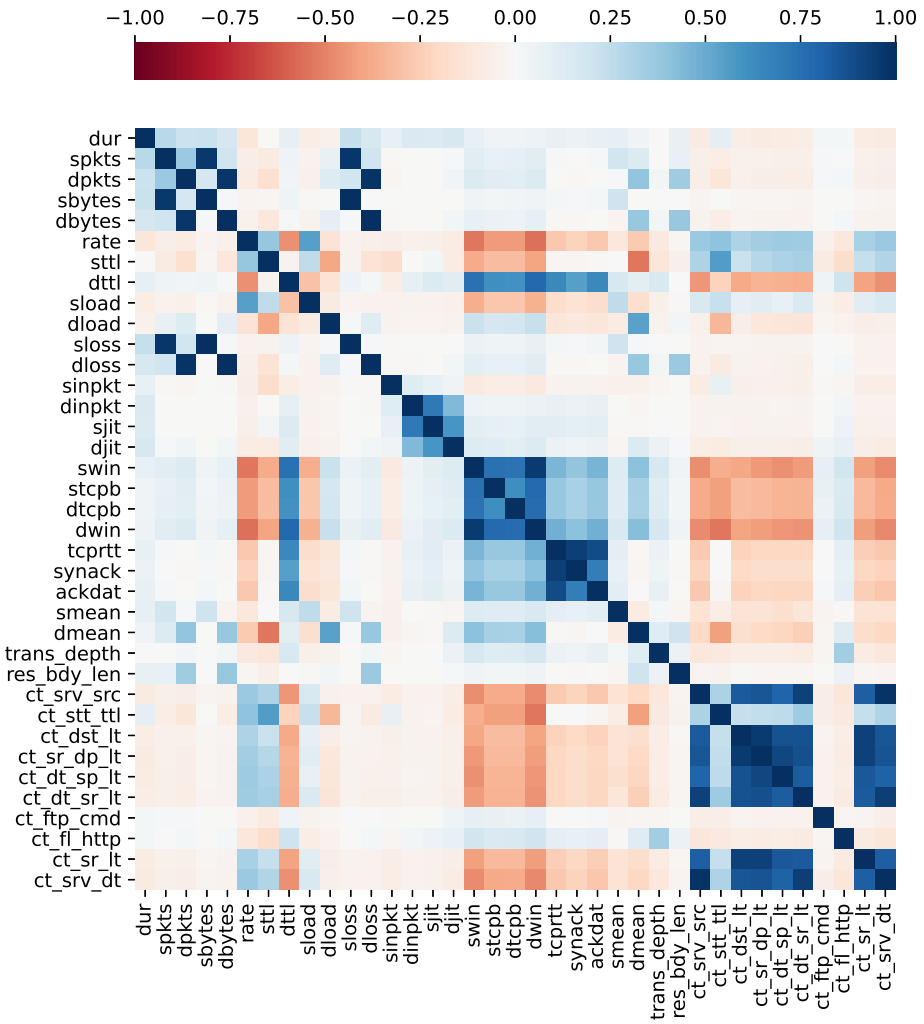


Figure 5.9 Correlation matrix between the numerical features of UNSW-NB15.

same examples. Finally, we examined the generated adversarial examples to investigate their consistency. We identified several syntactic-based and semantic-based criteria that prevent the implementation of these examples, and thus invalidate the attacks. These criteria include the value ranges, the binary features, the multiple category membership, and the semantics of the features.

Just as the threat of adversarial attacks should not be underestimated, the vulnerability of intrusion detection models cannot be claimed without ensuring the consistency of the adversarial examples. The invalidation criteria that we propose demonstrate the limitations of adversarial attacks. However, complying with these criteria does not ensure the validity of the attack; they are necessary but not sufficient conditions for the validation of adversarial examples. They could be extended to an exhaustive set of validation criteria. Thus, future

work should focus on formalizing the network constraints that must be satisfied by network traffic. A promising approach would be to consider the correlation matrix as a tool to identify dependencies between features. From these constraints, comprehensive rules could be derived and used to validate adversarial examples. Furthermore, adversarial attacks should integrate these rules in the generation of adversarial examples to produce only valid examples. Finally, the vulnerability of intrusion detection models to adversarial examples cannot be definitively proven with experimentation at the data level only. For this purpose, it is necessary to design a concrete implementation of an end-to-end attack scenario on real networks.

CHAPTER 6 DIFFUSION-BASED ADVERSARIAL PURIFICATION FOR INTRUSION DETECTION

Abstract The escalating sophistication of cyberattacks has encouraged the integration of machine learning techniques in intrusion detection systems, but the rise of adversarial examples presents a significant challenge. These crafted perturbations mislead ML models, enabling attackers to evade detection or trigger false alerts. As a reaction, adversarial purification has emerged as a compelling solution, particularly with diffusion models showing promising results. However, their purification potential remains unexplored in the context of intrusion detection. This paper demonstrates the effectiveness of diffusion models in purifying adversarial examples in network intrusion detection. Through a comprehensive analysis of the diffusion parameters, we identify optimal configurations maximizing adversarial robustness with minimal impact on regular performance. Importantly, this study reveals insights into the relationship between diffusion noise and diffusion steps, representing a novel contribution to the field. Our experiments are carried out on two datasets and against five adversarial attacks. The implementation code is publicly available.

Keywords adversarial defense, adversarial purification, adversarial examples, diffusion models, intrusion detection.

6.1 Introduction

Intrusion detection stands out as one of the most formidable challenges in cybersecurity, especially with the increasing sophistication of cyberattacks. Unfortunately, traditional signature-based approaches reach their limit against previously unknown threats, also called zero-days. As such, the integration of Machine Learning (ML) techniques has emerged as a promising avenue for enhancing the detection capabilities of intrusion detection systems.

However, the advent of adversarial examples [28,30] poses a severe obstacle to the reliability of ML, specifically in critical tasks such as intrusion detection. They are generated from regular data instances by adding a meticulously crafted perturbation that misleads an ML model. Applied to network data, they enable cyber attackers to either evade ML-based intrusion detection systems or flood the network with false alerts.

In response to the escalating threat of adversarial attacks, research efforts have been directed toward designing effective countermeasures. Among the various defensive approaches,

adversarial purification has emerged as a compelling solution to remove the adversarial perturbation from data before processing it. This defense is particularly interesting for intrusion detection, as it can be integrated upstream of the model without retraining. Recent work [42] has demonstrated promising purification performance using diffusion models [40, 41].

Diffusion models are generative models inspired by the dynamics of diffusion processes in physics [40]. They consist of a forward process that gradually adds noise to initial data and a backward process that reconstructs that data using a deep neural network. Because diffusion models are trained using examples drawn from the original data distribution, the reconstructed data is expected to adhere to the same distribution, even when the initial data is an adversarial example. Thus, they can be used for adversarial purification: removing the perturbation from adversarial examples to classify them correctly. Furthermore, since they are not explicitly trained on adversarial examples, their purification performance is not limited to a specific adversarial attack.

Despite the recent attention given to adversarial purification with diffusion models, there remains a notable gap in understanding their potential in the context of intrusion detection. This paper fills that gap by studying the purification effects of diffusion models and demonstrating their effectiveness on network intrusion detection. By conducting a comprehensive analysis of the diffusion parameters, we identify optimal configurations maximizing the adversarial robustness with limited impact on the regular performance. We show the effectiveness of our method on two network datasets (UNSW-NB15 [88] and NSL-KDD [78]) and against state-of-the-art adversarial attacks. Moreover, to our knowledge, this is the first investigation of the relationship between the number of diffusion steps and the optimal amount of diffusion noise for adversarial purification. Our findings demonstrate that the optimal amount of diffusion noise depends not on the number of diffusion steps but rather on the amount of adversarial perturbation.

The remainder of the paper is organized as follows. Section 6.2 introduces the background and related works in cybersecurity. In Section 6.3, we describe the experiment methodology. We report the results in Section 6.4, followed by a discussion in Section 6.5 and a conclusion in Section 6.6.

6.2 Background and related work

This section introduces the major approaches to adversarial defense, provides some background on diffusion models, and reviews the literature on diffusion models in intrusion detection systems and adversarial purification.

6.2.1 Adversarial defenses

The following presents three dominant approaches to defend against adversarial examples [31].

Adversarial training consists of training the model with adversarial examples in addition to the training data [30, 33]. Despite its effectiveness, this technique has several drawbacks: (*i*) it requires the retraining of the model and significantly lengthens the training duration, and (*ii*) it protects only against adversarial examples generated with the methods it was trained on.

Adversarial detection consists of a separate classifier deployed upstream of the ML model that detects and discards adversarial examples before they are fed to the model [221]. It is plug-and-play and does not require retraining. Several supervised and unsupervised techniques exist; interested readers may refer to [221] for a detailed review. Unfortunately, adversarial detection also depends on the adversarial attacks on which it was trained. Moreover, since it is a classifier, it can be fooled to some extent [222].

Adversarial purification consists of a separate model deployed upstream of the ML model that removes the perturbation from adversarial examples before they are fed to the model [140, 223]. This approach is also plug-and-play, and the purification models are typically trained independently of the ML model. Adversarial purification does not require retraining, and it is, in many cases, independent of the adversarial attacks. This paper focuses on the adversarial purification approach using diffusion models [42].

6.2.2 Adversarial defenses in intrusion detection

ML models and neural networks, in particular, have demonstrated state-of-the-art performance at the intrusion detection task [4]. The threat of adversarial samples has thus been taken seriously and several works can be found on adversarial defenses in network-based intrusion detection. Most of these works focus on a variety of adversarial training techniques [147, 148, 224, 225]. Notably, [147], [148], and [224] use Generative Adversarial Networks (GANs) to augment the databases for training a classifier, demonstrating the efficiency of adversarial defenses in the context of intrusion detection.

6.2.3 Diffusion models

Diffusion models are a class of generative models that leverage the diffusion processes used in physics to learn complex data distributions and samples from them [40, 41]. Instead of modeling the distribution like VAEs [37] and GANs [36], they model the process of transforming

a simple distribution (e.g., Gaussian noise) into the target distribution through a sequence of steps. They consist of two Markov processes: a *forward* and a *reverse* process.

In the *forward process*, each step involves adding Gaussian noise to the data over T steps until no structure remains; it corresponds to a smooth transition from the complex data distribution to a Gaussian distribution (latent space). The amount of noise added at each step depends on the variance schedule β. In the *reverse process*, each step reverts the corresponding forward step—removes the diffusion noise—to reconstruct the original data distribution. Since the reverse process is mathematically intractable, it is approximated with deep neural networks. In this work, we use discrete diffusion steps, but the continuous case can also be applied; it requires solving stochastic differential equations [166].

In the following, we provide the intuition into the theory behind diffusion models. We consider a dataset x_0 with unknown distribution $x_0 \sim q(x_0)$. For a given number T of steps, we consider the Markov chain $(x_t)_{t \leq T}$ with transitions

$$q(x_{t+1}|x_t) = \mathcal{N}\left(x_{t+1}; \sqrt{1-\beta_{t+1}}x_t, \beta_{t+1} I_n\right), \tag{6.1}$$

that is, we gradually add Gaussian noise with a given variance $(\beta_t)_{t \leq T}$ to the data. If we define $\bar{\alpha}_t = \prod_{i=1}^{i=t}(1-\beta_i)$, then the cumulative noise addition from the clean data to step t is written:

$$q(x_t|x_0) \simeq \mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t) I_n\right). \tag{6.2}$$

Equation 6.2 describes the *forward process* of diffusion models. Note that to ensure the diversity of generated data the variance schedule should guarantee that the data resembles a Gaussian distribution at the end of the forward process:

$$q(x_T|x_0) \simeq \mathcal{N}(x_T; 0, I_n). \tag{6.3}$$

The *reverse process* consists of generating examples from the original data distribution using the reverse Markov chain; it starts from a Gaussian distribution with transitions

$$p(x_t|x_{t+1}) = \mathcal{N}(x_{t+1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)), \tag{6.4}$$

where θ represents the parameters of the deep neural network used to estimate the diffusion noise to be removed.

6.2.4 Adversarial purification with diffusion models

Adversarial purification is the process of removing the perturbation from adversarial examples to classify them correctly. This process can be seen as a generative task and approached with diffusion models. The gradual addition of Gaussian noise in the forward step submerses the adversarial perturbation, but the data becomes too noisy to be correctly classified. Thus, the reverse step reconstructs the data in the original distribution without adversarial perturbations.

Furthermore, the forward process does not need to complete T steps and reach a Gaussian distribution. There should be enough added noise to submerse the adversarial perturbation, but not too much, as it damages the data structure and decreases the accuracy. The forward process should stop at the optimal diffusion step t^*, where the diffusion noise suffices to remove the adversarial perturbation while preserving the structure for the classification [42].

After this concept was introduced in [42], later work leveraged guided diffusion models for adversarial purification [43, 44]. Authors in [226] train a robust guidance with an adversarial loss and apply it to the reverse process. Diffusion models are also used to purify backdoors in poisoned models [227].

6.2.5 Diffusion models in intrusion detection

Intrusion detection systems benefit greatly from the automation provided by ML, including deep learning [76, 228, 229]. However, network intrusion datasets are often imbalanced; benign traffic outweighs malicious traffic. Due to their generative capabilities, diffusion models are successfully applied in data augmentation for balancing network datasets [230–232]. The diffusion model can also detect intrusion by learning the distribution of benign traffic. The difference between the original and reconstructed data is then used to detect malicious traffic [233, 234].

However, to our knowledge, no prior research has investigated the adversarial purification potential of diffusion models in the context of intrusion detection. This paper represents the foremost initiative to address diffusion-based adversarial purification in intrusion detection.

6.3 Methodology

As shown in Figure 6.1, our methodology consists of a diffusion model deployed upstream of the intrusion detection model. The diffusion model serves as a "filter" that removes adversarial perturbations. It adds noise to the data for several steps t and reconstructs it through the

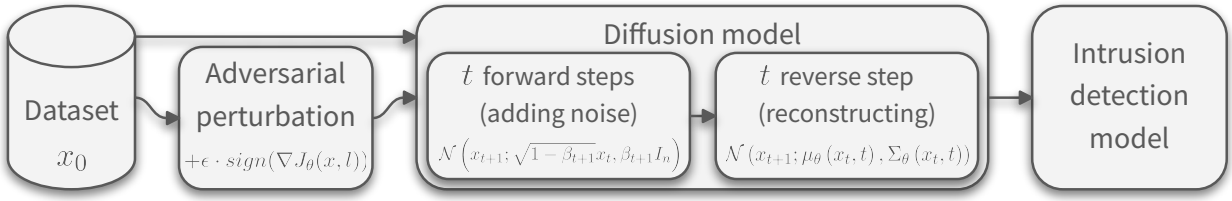


Figure 6.1 Methodology scheme: dataset instances x_0 undergo adversarial perturbation, the diffusion model's purification, and then the intrusion detection classification.

reverse diffusion process involving the diffusion neural network. The data, whether original or adversarial, are first purified by the diffusion model and then fed to the intrusion detection model to determine if they are benign or malicious.

The intrusion detection model is a Fully Connected Neural Network (FCNN) with fixed hyperparameters throughout the experiments. It comprises a fully connected neural network with 5 hidden layers of 256, 512, 1024, 512, and 256 Rectified Linear Units (ReLU). It is trained for $10,000$ epochs, and the parameters are optimized using Adam with a learning rate of 10^{-5}.

The literature presents more sophisticated architectures of intrusion detection [235]. However, FCNNs are advantageous due to their simplicity and interpretability, providing a clear baseline without the confounding factors and biases of sophisticated architectures. Simple models facilitate theoretical analysis, reproducibility, and benchmarking, enabling easier comparisons. Moreover, insights gained from FCNNs can be generalized to improve more complex models.

The diffusion models are trained according to [41, Algorithm 1]. We consider T discrete diffusion steps. Instead of having a separate neural network for each timestep [40], we encode the timesteps as sinusoidal embeddings and add them to each layer of the diffusion neural network [41].

The purification results are recorded across the diffusion steps. For each $t \in [1, T]$, we apply t forward diffusion steps to the data instance x_0 to get x_t; then we apply t reverse diffusion steps to x_t to get $\hat{x}_0$, the reconstruction of x_0.

The variance schedule β is linearly distributed (T evenly spaced values between β_1 and β_T). Across different experiments, the number of diffusion steps T is 100 or 1000, while β_1 varies between 10^{-5} and 10^{-4}, and β_T varies between 10^{-4} and 10^{-1}.

The diffusion neural networks are fully connected neural networks with 10 hidden layers; each hidden layer typically consists of 960 ReLU units, except for the experiments that compare neural network architectures. The diffusion neural networks are trained for $200,000$ epochs, where each epoch consists of predicting a random step of the reverse process for each dataset instance. The loss function is a Mean Squared Error (MSE), and the parameters are optimized using AdamW with a learning rate of 10^{-4}.

The metrics recorded during the experiments evaluate two aspects of diffusion models: (*i*) the reconstruction performance by recording the reconstruction loss (MSE between the original data and the reconstructed data) for a diffusion step t, and (*ii*) the adversarial purification performance by feeding the reconstructed data to the intrusion detection model and recording its accuracy.

The optimal diffusion step $t^* \in [1, T]$ is the step that maximizes the intrusion detection accuracy on adversarial examples [42]. It should be large enough to dilute the adversarial perturbation with diffusion noise. However, the larger it is, the more data structure it dilutes, which decreases the test accuracy.

The implementation of all experiments is carried out on two prominent, publicly available, network datasets: NSL-KDD [78], which has been widely used as a standard benchmark in network intrusion detection research and allows for direct comparison with a vast body of previous major work in intrusion detection; and UNSW-NB15 [88] a more recent dataset which is representative of modern network traffic [173]. The latter was created using real network traffic from the IXIA PerfectStorm tool, providing more realistic and diverse attack scenarios compared to older datasets. See Table 6.1 for a description of both datasets. For further details on the implementation of our experiments, we make our code publicly available [1].

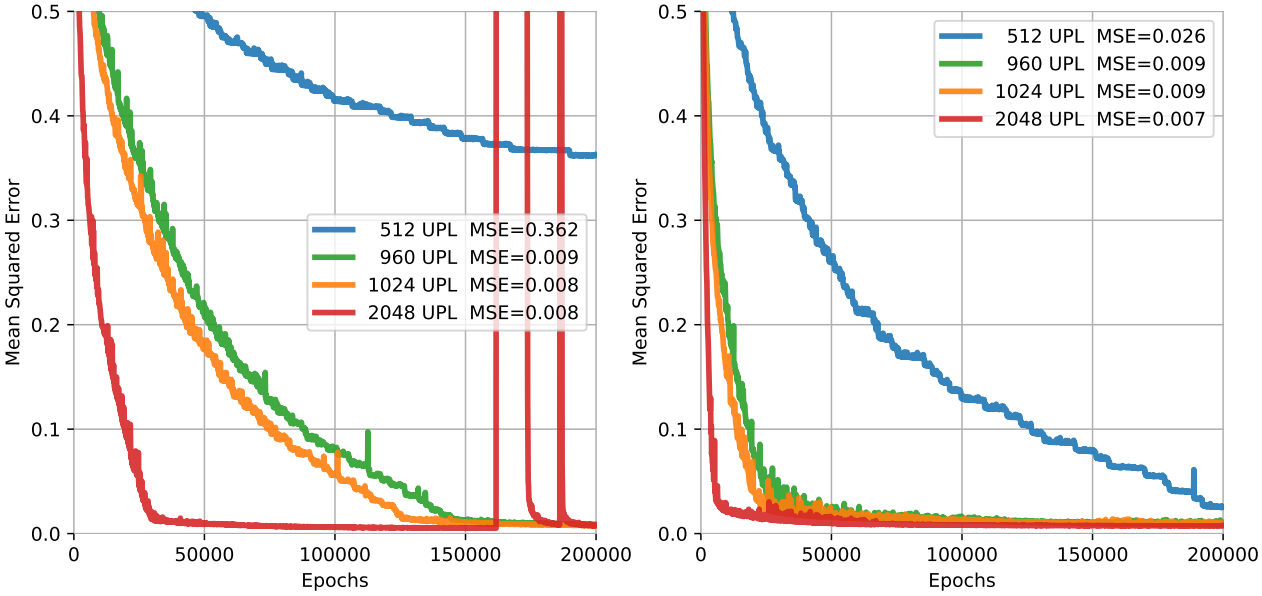
6.4 Results

In this section, we present the results of our experiments. We first analyze the reconstruction loss throughout the training of the diffusion models to identify optimal hyperparameters. Then, we study the reconstruction loss and the accuracy of the intrusion detection model on reconstructed data. We compare the robustness of the intrusion detection model with respect to the level of diffusion applied to the data. We aim to find the optimal diffusion step

[1] `https://github.com/mamerzouk/adversarial-purification`

Dataset	NSL-KDD [78]	UNSW-NB15 [88]
Num. features	43	49
Types of attacks	22 + benign	9 + benign
Training set	125,973 records	175,341 records
(with attacks)	(58,630 attacks (47%))	(119,341 attacks (68%))
Testing set	22,544 records	82,332 records
(with attacks)	12,833 attacks (57%)	45,333 attacks (55%)

Table 6.1 Description of the two datasets we use in this paper.



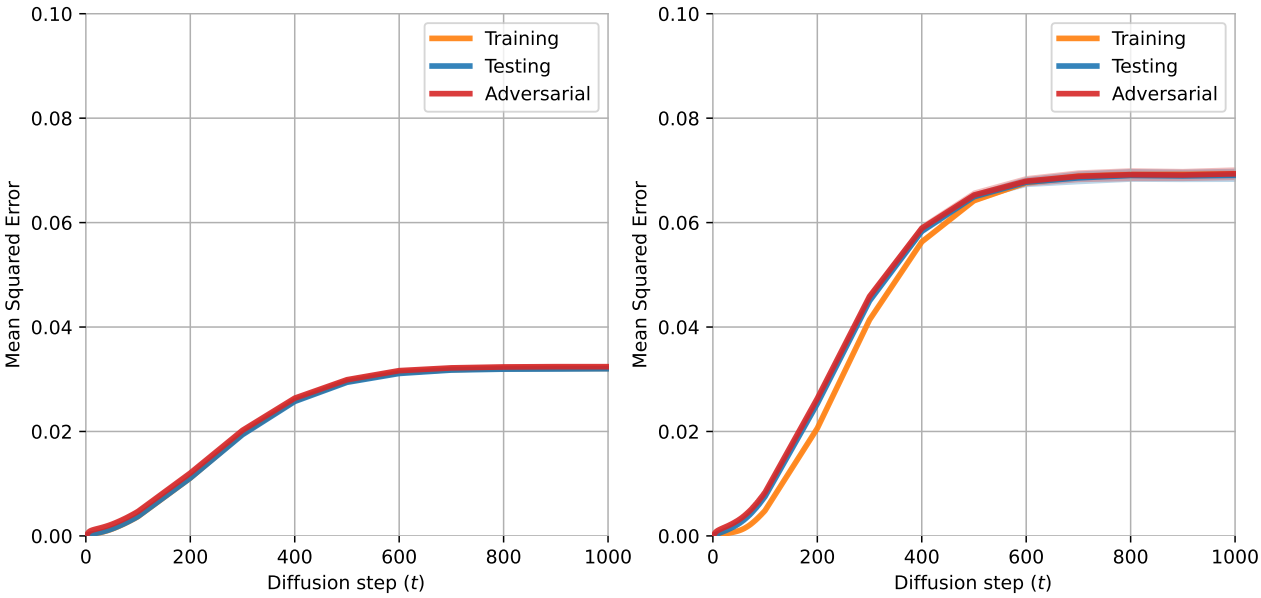
(a) UNSW-NB15 (b) NSL-KDD

Figure 6.2 Reconstruction loss over training epochs for different neural network sizes

t^* that removes the adversarial perturbation and increases the robustness while minimizing the repercussions on non-adversarial data.

Furthermore, we analyze the impact of several diffusion parameters on the purification performance: the number of steps T, the initial variance β_1, and the final variance β_T. Finally, we study the impact of the adversarial perturbation amplitude ϵ on the optimal diffusion step t^* and compare our purification model against five state-of-the-art adversarial attacks. The figures present the results on both UNSW-NB15 and NSL-KDD; the values are the mean and standard deviation over 10 randomly initialized runs.

Unless otherwise noted, the diffusion models in these experiments use the standard diffusion parameters proposed in the previous work [41]: $\beta_1 = 10^{-4}$, $\beta_T = 0.02$, and $T = 1000$.

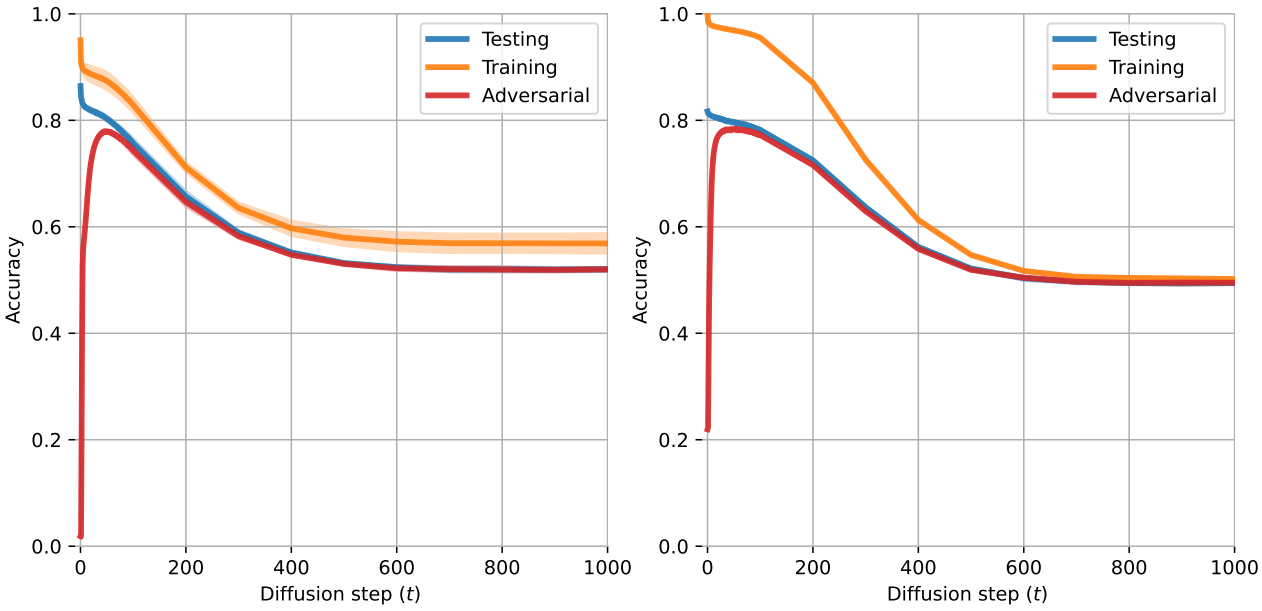


(a) UNSW-NB15 (b) NSL-KDD

Figure 6.3 Reconstruction loss over the diffusion steps t

6.4.1 Diffusion neural network size

Here, we analyze the training of diffusion models on network data. We specifically compare the impact of the size of the diffusion neural network on the training loss. We elaborate further on the impact of the number of diffusion steps on the training loss in Appendix C. Figure 6.2 shows the reconstruction loss (in MSE) over the training epochs for different Units Per Layer (UPL). We observe that the reconstruction loss decreases earlier for larger diffusion neural networks. Furthermore, larger neural networks reach lower MSE values when they stabilize, 0.008 and 0.007 with 2048 UPL against 0.362 and 0.026 with 512 UPL on UNSW-NB15 and NSL-KDD, respectively. This is due to the capacity of larger neural networks to model more complex patterns and learn better representations of the data. However, larger models have drawbacks; they take significantly longer to train and require more computational power. Larger models also display instabilities during the training; Figure 6.2 shows how the 2048 UPL diffusion model loss peaks randomly between $150,000$ and $200,000$ epochs on UNSW-NB15. For the rest of the experiments, we will rely on diffusion neural networks with 10 hidden layers of 960 UPL trained for $200,000$ epochs.



(a) UNSW-NB15 (b) NSL-KDD

Figure 6.4 Intrusion detection accuracy over the diffusion steps t

6.4.2 Reconstruction loss and accuracy over t

Figure 6.3 shows the reconstruction loss over the diffusion steps t, while Figure 6.4 shows the accuracy of the intrusion detection model on the same reconstructed data over the diffusion steps t.

Figure 6.3 shows that the reconstruction loss increases similarly for all three sets, the lines even overlap on UNSW-NB15. Indeed, as noise is added gradually, the data structure is slowly destroyed. The more diffusion steps are applied, the more noise is added, and the harder it is to reconstruct precisely the data. After 600 steps, the reconstruction loss plateaued around 0.03 and 0.07 on UNSW-NB15 and NSL-KDD, respectively. Moreover, we notice that the reconstruction loss on adversarial examples is slightly superior to that on the original testing set. The difference in the reconstruction losses can also be used as an indicator for detecting adversarial examples. This avenue is not investigated in this paper, as adversarial detection approaches are out of our scope.

The accuracy curve in Figure 6.4 corroborates the previous results: the training and testing accuracy decrease as the diffusion step increases due to the damaged data structure. It becomes more challenging for the intrusion detection model to distinguish benign and malicious traffic, which decreases its accuracy. After 600 steps, the reconstructed data is too noisy to be classified correctly.

6.4.3 Purification performance

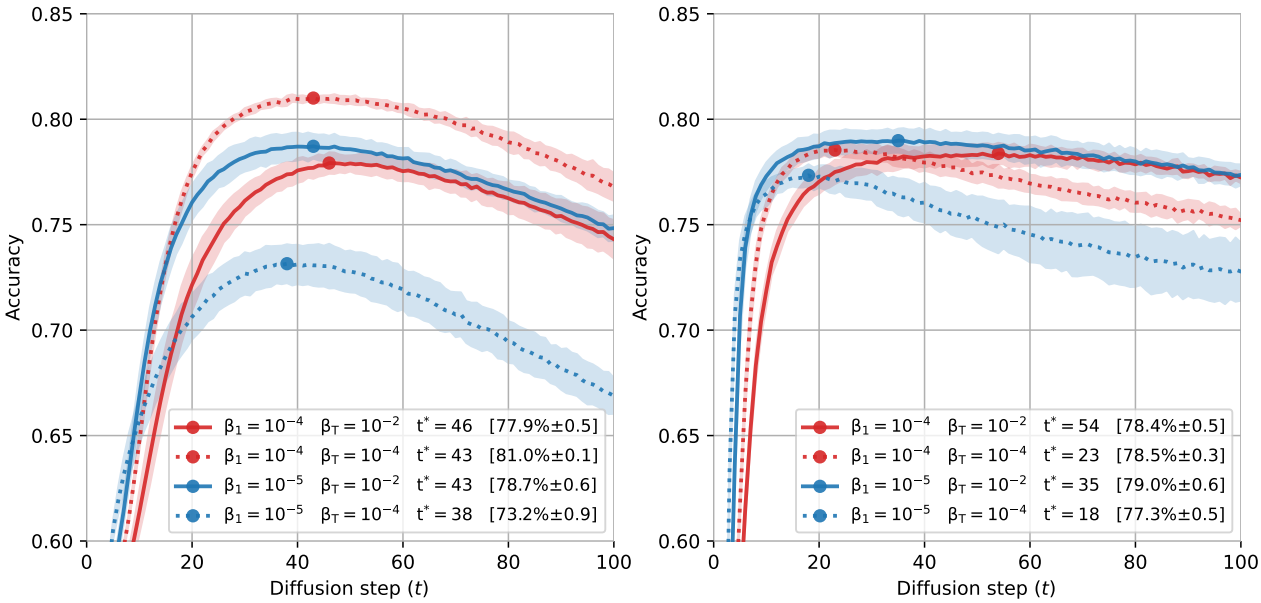
In order to evaluate the robustness of the intrusion detection after the diffusion model's purification, we focus on the red line in Figure 6.4, which represents the accuracy of the intrusion detection model on adversarial examples. Those adversarial examples were generated from the testing set using the targeted Fast Gradient Sign Method (FGSM) with a perturbation amplitude $\epsilon = 0.03$. At step 0, before purification, the accuracy on adversarial data is 0.02, while it is 0.86 on the original test data, indicating that the adversarial attack succeeded in misleading the intrusion detection model. After a few diffusion steps, the adversarial accuracy increases drastically. The added diffusion noise dilutes the adversarial perturbation that misleads the model while preserving enough data structure for a good classification. After 44 steps, the adversarial accuracy peaks at 78% while the test accuracy is 80%. This diffusion step t^* is optimal as it maximizes the accuracy on adversarial examples while minimizing the impact on the non-adversarial test data. This result empirically shows the purification capabilities of diffusion models in intrusion detection.

After the peak, the structure damage due to the addition of diffusion noise decreases the adversarial accuracy. Since the adversarial perturbation has been removed, the difference between the test and adversarial accuracy disappears; it is below 0.01 after $t = 90$. Both values decrease until they reach random classification when the reconstructed data is too noisy.

6.4.4 Variance schedule

The variance of the Gaussian noise added at step t of the diffusion process is denoted β_t. It follows a linear distribution of T evenly spaced values between β_1 and β_T. The variance schedule is an essential parameter of the diffusion process; we hypothesize that it is also critical to the purification capabilities of diffusion models. In the following, we study the impact of the variance schedule β on the purification capabilities of the diffusion model through the accuracy of the intrusion detection model on purified examples. Using a fixed $T = 1000$, we vary both β_1 and β_T to find an optimal schedule.

Figure 6.5 shows the impact of β_1, the first value of the variance schedule. If we focus on the continuous lines, corresponding to $\beta_T = 10^{-2}$, we do not see a significant difference between the two values of β_1. However, with a smaller β_T, the difference between β_1 becomes significant. The dotted lines, corresponding to $\beta_T = 10^{-4}$, show a large difference between the two values of β_1. As the final β decreases, the difference between the accuracy of the two β_1 values increases. The impact of the initial variance β_1 is therefore linked to the length



(a) UNSW-NB15 (b) NSL-KDD

Figure 6.5 Intrusion detection accuracy over diffusion step t for different β_1. Continuous lines for $\beta_T = 10^{-2}$ and dotted lines for $\beta_T = 10^{-4}$. The marker indicates the maximum accuracy reached at the optimal diffusion step t^*.

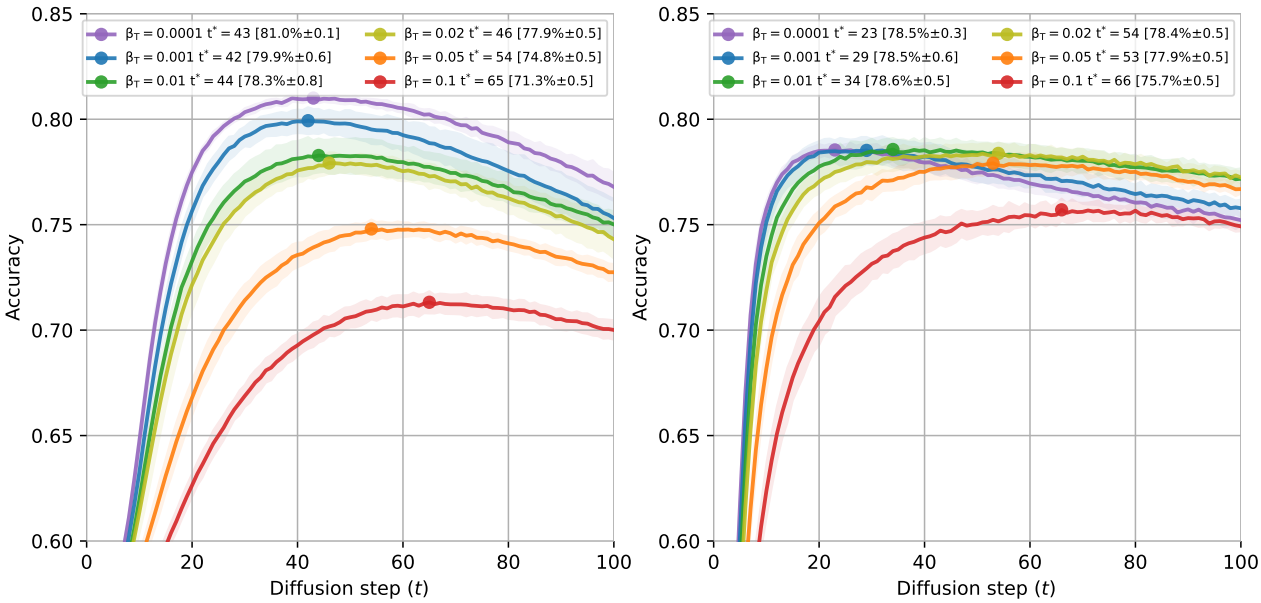
of the variance schedule $\beta_T - \beta_1$ and becomes less significant as the interval increases. This result suggests that β_T plays an influential role in the purification.

Figure 6.6 shows the impact of β_T, the final variance of the diffusion schedule. The figure shows an increasing accuracy with smaller β_T values. The intuition is that as β_T decreases, the interval between successive variance values decreases, which makes the noising more gradual and easier for the neural network to reconstruct. On UNSW-NB15, the maximum accuracy value is $81\% \pm 0.1$ at $t^* = 43$, it was reached with the smallest value $\beta_T = 10^{-4}$, which represents a constant variance schedule (since $\beta_1 = 10^{-4}$). On NSL-KDD, the maximum accuracy value is very close between when $\beta_1 \leq 10^{-2}$, but $\beta_1 \leq 10^{-4}$ is the earliest to achieve 78.5 ± 0.3 after only $t^* = 23$.

6.4.5 Number of diffusion steps T

In addition to the initial and final perturbation values β_1 and β_T, the diffusion process is characterized by the number of diffusion steps T. This parameter determines the granularity of the diffusion since a larger number of steps T makes the step size smaller.

In the following, we study how the number of diffusion steps T affects the optimal diffusion



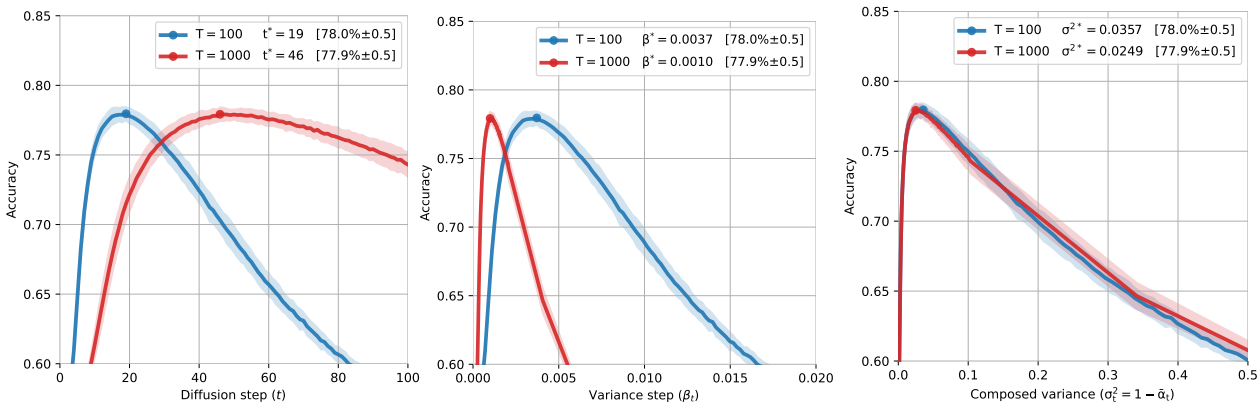
(a) UNSW-NB15 (b) NSL-KDD

Figure 6.6 Intrusion detection accuracy over diffusion step t for different β_T

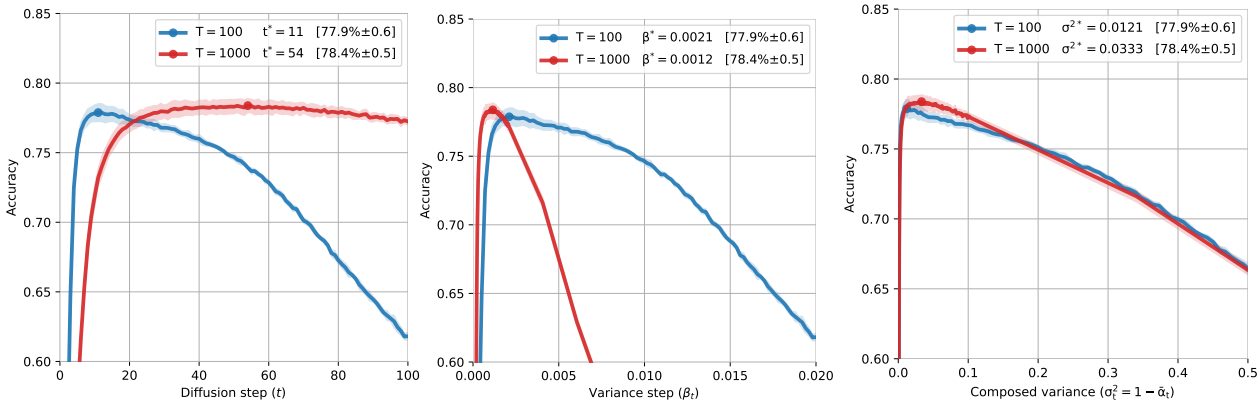
steps t^*. We compare diffusion models with $T = 100$ and $T = 1000$ with respect to the optimal diffusion step t^* and identify how it translates to an equivalent amount of noise. In Appendix C, we further investigate the impact of the number of diffusion steps T on the training loss and the purification performance with the optimal variance schedule.

Figure 6.7 shows the impact of the number of diffusion steps T over three references: the diffusion step t, the variance step β_t, and the composed variance σ_t^2. This experiment uses the standard diffusion parameters to focus the analysis on when the optimum is recorded rather than its value. With the optimal variance interval recorded in Figure 6.6, $\beta_1 = \beta_T = 10^{-4}$, the 1000-step diffusion model largely over-performs the 100-step one on UNSW-NB15, as shown in Figure C.2 (Appendix C).

In the first part of Figure 6.7, we see the accuracy across the diffusion step t. The $T = 100$ diffusion model reaches its optimum at $t^* = 19$ and $t^* = 11$, while the $T = 1000$ reaches its optimum at $t^* = 46$ and $t^* = 54$ on UNSW-NB15 and NSL-KDD, respectively. This indicates a slower evolution when $T = 1000$, which is coherent since the variance steps are smaller. Therefore, we plot the same values with respect to the variance steps β_t to find a similarity. In the second part of Figure 6.7, the x-axis corresponds to β_t and covers the whole interval from 0 to $\beta_T = 0.02$. However, the optimum is still reached at distant values of β. For $T = 100$, it is reached with $\beta^* = 0.0037$ and $\beta^* = 0.0021$, while for $T = 1000$, it is



(a) UNSW-NB15



(b) NSL-KDD

Figure 6.7 Intrusion detection accuracy over t, β_t, and σ_t^2 for different number of diffusion steps T

reached with $\beta^* = 0.0010$ and $\beta^* = 0.0012$ in UNSW-NB15 and NSL-KDD, respectively.

β_t is the amount of variance applied at the diffusion step t, but it does not correspond to the total variance applied to the initial data x_0. Indeed, the diffusion forward process is a composition of Gaussian distributions with gradual variances β_t. The total (composed) variance σ_t^2 of such composition is the sum of the individual variances, $\sigma_t^2 = 1 - \bar{\alpha}_t$ (Equation 6.2).

The third part of Figure 6.7 shows the accuracy across the composed variance σ_t where the two lines overlap, indicating a similar accuracy regardless of the number of diffusion steps T. The optimum is reached at $\sigma_t^2 = 0.0249$ and $\sigma_t^2 = 0.0333$ for $T = 1000$, and $\sigma_t^2 = 0.0357$ and $\sigma_t^2 = 0.0121$ for $T = 100$ on UNSW-NB15 and NSL-KDD, respectively.

Considering the scale of σ_T^2 in Figure 6.7, the optimum values are relatively close. We note from this experiment that the optimal noise added σ^{2^*} approaches 0.03, which corresponds to the adversarial perturbation amplitude ϵ used in these experiments. This result suggests a dependence between the optimal noise amount and the perturbation amplitude.

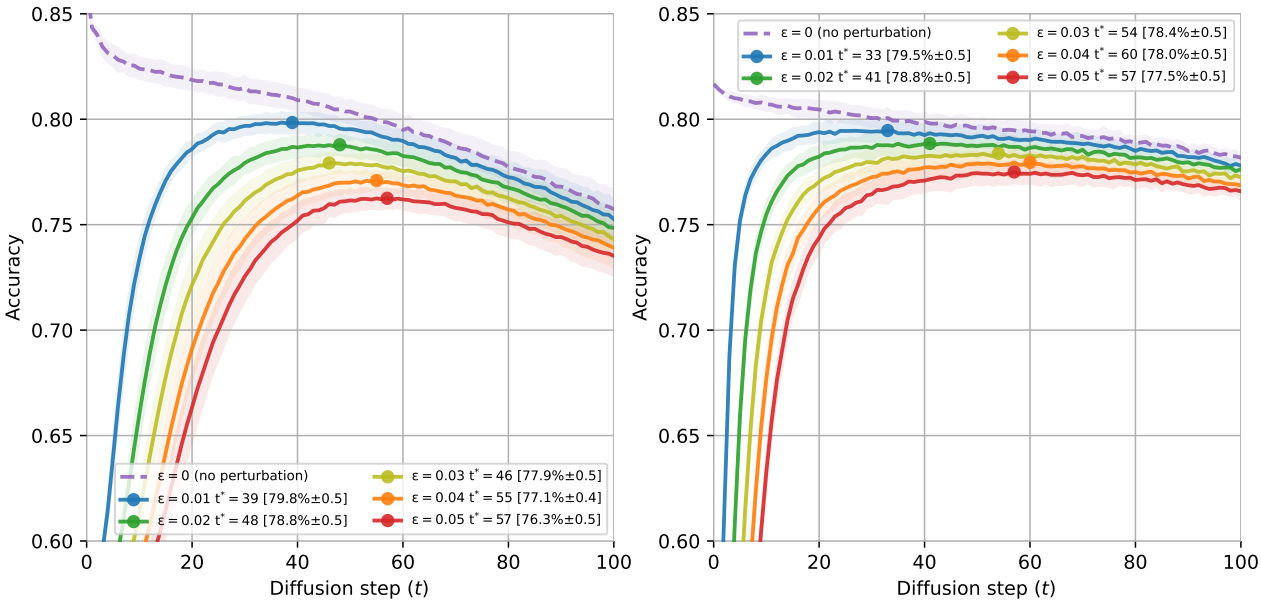
6.4.6 Adversarial perturbation amplitude ϵ

Beyond the diffusion parameters, the purification performance of diffusion models depends on the amount of adversarial perturbation ϵ added to the data. Figure 6.8 shows the accuracy of the intrusion detection model on increasing ϵ values. It demonstrates how the purification performance decreases as the perturbation amplitude increases. The accuracy at t^* goes from $79.8\% \pm 0.5$ and $79.5\% \pm 0.5$ when $\epsilon = 0.01$ to $76.3\% \pm 0.5$ and $77.5\% \pm 0.5$ when $\epsilon = 0.05$ on UNSW-NB15 and NSL-KDD, respectively. Another pattern is that it takes more diffusion steps to reach an optimum as ϵ increases. The two phenomena are linked: the diffusion model needs to add more perturbation to dilute a larger ϵ, thus taking more diffusion steps. However, the test accuracy (purple line) decreases as more noise is added. Since it represents an upper bound on the adversarial accuracy, it causes the optimal adversarial accuracy to decrease as ϵ increases.

6.4.7 Adversarial attacks

The efficiency of adversarial examples also depends on the method used to generate them. Various methods exist, each optimizing different distance norms and criteria. While previous experiments show how diffusion models considerably improve the adversarial accuracy of intrusion detection models, they have only been tested on one adversarial attack (FGSM). Here, we study how the purification performance is generalized to other adversarial examples' generation methods. We compare five well-established methods, namely: DeepFool [69], Jacobian-based Saliency Map Attack (JSMA) [29], Fast Gradient Sign Method (FGSM) [30], Basic Iterative Method (BIM) [213], and Carlini&Wagner's L_2 attack [70].

Figure 6.9 shows the adversarial accuracy of the intrusion detection model for the adversarial attacks and the testing set baseline. We note that the diffusion models successfully purify adversarial examples from all 5 generation methods. However, the optimal adversarial accuracy and diffusion step vary from one method to the other. Carlini&Wagner's attack is the most resistant to adversarial purification, with a maximum adversarial accuracy of $73.9\% \pm 2$ and $75.8\% \pm 0.8$ on UNSW-NB15 and NSL-KDD, respectively. The optimum is reached after only 17 steps on UNSW-NB15, 2.7 times faster than FGSM. On NSL-KDD, the adversarial accuracy is stable between 30 and 100, making t^* less accurate.



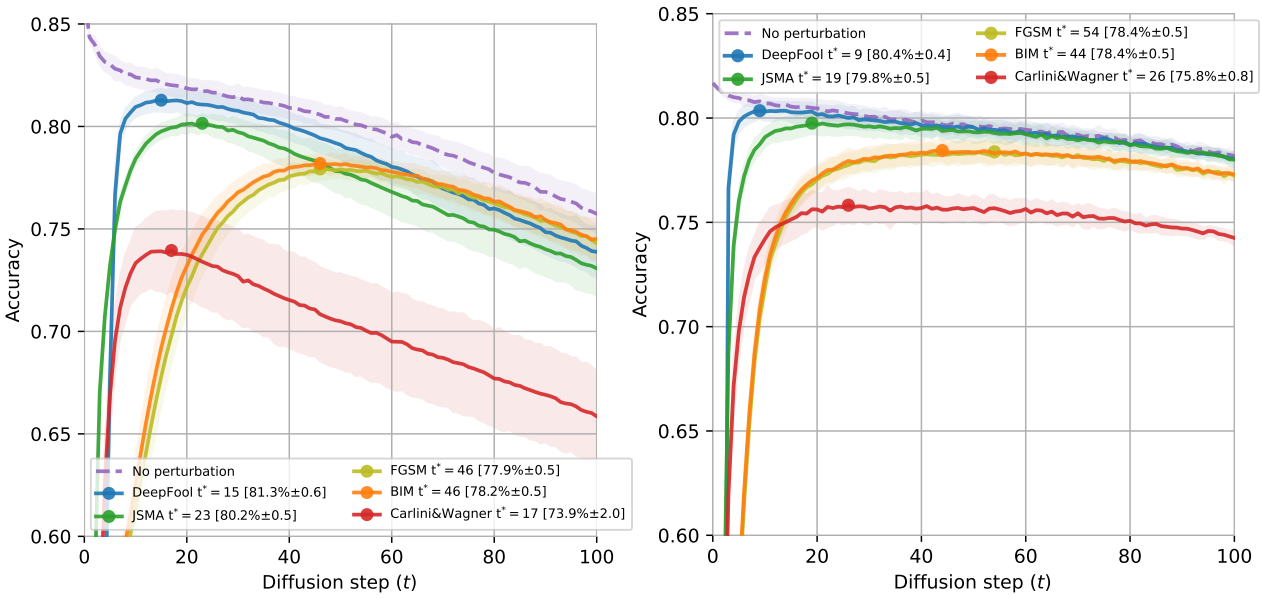
(a) UNSW-NB15 (b) NSL-KDD

Figure 6.8 Intrusion detection accuracy over the diffusion step t for different adversarial perturbation amplitudes ϵ generated with FGSM

FGSM and BIM achieve similar performance; FGSM reaches a maximum of $77.9\% \pm 0.5$ and $78.4\% \pm 0.5$, while BIM reaches a maximum of $78.2\% \pm 0.5$ and $78.4\% \pm 0.5$ on UNSW-NB15 and NSL-KDD, respectively. The optimal diffusion steps are also very close between the two. This similarity is explained by the fact that BIM is based on FGSM; it applies small FGSM steps iteratively (100 in our experiments) to optimize the adversarial examples. However, the iterative approach does not make BIM's adversarial examples more robust to our diffusion-based adversarial purification. The diffusion models achieve the highest purification performance on DeepFool's adversarial examples, with a maximum accuracy of $81.3\% \pm 0.6$ and $80.4\% \pm 0.4$ on UNSW-NB15 and NSL-KDD, respectively, closely followed by JSMA with a maximum accuracy of $80.2\% \pm 0.5$ and $79.8\% \pm 0.5$ on UNSW-NB15 and NSL-KDD, respectively. DeepFool and JSMA also had the fastest-growing adversarial accuracy values among other attacks, almost reaching the testing accuracy upper bound.

6.5 Discussion

Diffusion neural network The size of the neural network has a considerable impact on the reconstruction loss of the diffusion model. Larger neural networks converge faster and reach lower reconstruction loss values due to their capacity to model more complex patterns



(a) UNSW-NB15 (b) NSL-KDD

Figure 6.9 Accuracy over the diffusion step t for different adversarial attacks

and learn better representations. The main obstacle to using larger neural networks is that they take longer to process the data, making the training and reconstruction longer. The diffusion neural network should be hyperparameterized with particular attention to domain-specific time constraints, especially in network intrusion detection, where the reaction time is critical.

In addition to the size of the diffusion neural network, other hyperparameters impact the reconstruction loss and, potentially, the purification performance. The choice of the loss function, the optimization algorithm, and the learning rate affect the convergence time and help find better optimums. Finally, fully connected neural networks can be replaced with more sophisticated neural network architectures that model the diffusion process more precisely. In sum, optimizing diffusion neural networks is a promising avenue for improving the reconstruction loss, the purification performance, and even the processing time of diffusion models.

6.5.1 Variance schedule β

The variance schedule determines the amount of noise added at each step of the forward diffusion process. In the experiments, the schedule is linear—it consists of T evenly space values between β_1 and β_T. This schedule is essential for adversarial purification, since adding

too little noise does not remove the adversarial perturbation, and too much noise corrupts the data structure. The importance of β_1, the variance of the first diffusion step, depends on the total length of the diffusion schedule $\beta_T - \beta_1$. When the schedule is small, the choice of the starting value makes a considerable difference in the purification capabilities. As the difference between β_1 and β_T increases, β_1 is less significant since it has little impact on the rest of the variance values.

On the other hand, the last value β_T has a more significant impact on the variance schedule, which also extends to the purification performance. In the standard setting, where $T = 1000$, $\beta_1 = 10^{-4}$, and $\beta_T = 0.02$, a smaller value of β_T leads to a better adversarial accuracy. The best purification performance was recorded with the smallest value $\beta_T = \beta_1 = 10^{-4}$, which describes a constant variance schedule.

As opposed to generative models, diffusion models in adversarial purification do not require the forward process to reach a Gaussian distribution $\mathcal{N}(x_T; 0, I_n)$ after T steps. Therefore, there is more freedom in choosing the diffusion parameters to optimize the adversarial accuracy.

6.5.2 Number of diffusion steps T

The number of diffusion steps affects multiple aspects of the diffusion process. It determines the size of the variance schedule β and its values since there are T evenly spaced values. In the best-performing case $\beta_T = \beta_1 = 10^{-4}$, the number of diffusion steps makes a considerable difference in favor of a larger T. Having the same variance schedule divided into more variance steps allows the neural network to learn more details about the applied noise, which are then helpful to accurately reconstruct the data in the reverse process.

The number of diffusion steps also affects the optimal diffusion step t^*; it happens earlier in the process when T is smaller. This is explained by the fact that the diffusion steps t do not correspond to an equivalent variance step β_t. However, even when considering β_t instead of t, the reconstruction curve does not match. This is because the β_t values are gradually added to the data (Equation 6.1) according to the variance step; they do not represent the total variance applied to the data. Instead, we consider the variance of the composition, $\sigma_t^2 = 1 - \bar{\alpha}$, which makes the values of $T = 100$ and $T = 1000$ match. The total variance that maximizes the adversarial accuracy, σ^{2^*} is very close between the two different T values. Furthermore, the optimal variance σ^{2^*} approaches the value of the perturbation amplitude ϵ, demonstrating the dependence of the diffusion noise on the adversarial perturbation.

In sum, diffusion models with more diffusion steps achieve better adversarial accuracy with

an optimal variance schedule. However, it takes more steps to reach optimum, translating into a longer purification time. Thus, the choice of T should also consider the time constraints that characterize the application domain.

6.5.3 Adversarial perturbation amplitude

The diffusion-based adversarial purification effectively removes adversarial perturbations from network traffic data. However, the effectiveness of this method varies slightly depending on the nature of the adversarial examples. In the case of adversarial examples generated with FGSM, the parameter ϵ determines the amount of perturbation added to the data. As the ϵ increases, more noise is required to dilute the perturbation, making the optimum occur later in terms of diffusion steps. Another side effect is that the extra required noise also dilutes some of the data structure, thus decreasing the test accuracy, representing an upper bound to the adversarial accuracy. Therefore, the maximum adversarial accuracy decreases as ϵ increases.

6.5.4 Adversarial attacks

In addition to the perturbation amplitude ϵ, the purification process is sensitive to the adversarial examples' generation method. The results show a considerable gap in the adversarial accuracy between different methods. In particular, DeepFool and JSMA are easier to purify and approach the test accuracy upper-bound after a few diffusion steps, and FGSM and BIM achieve close performance due to their similarities. Carlini&Wagner's L_2 adversarial examples are the most resistant to purification, this is due to its iterative nature and the choice of the objective function [222, Section 2.6]. Nonetheless, our method still recovers up to 75% of the original test accuracy.

6.5.5 Adversary's constraints

The adversarial examples' generation method and the adversarial perturbation amplitude are both parameters controlled by the adversary when they generate the adversarial examples. However, they are constrained in the amount of perturbation they can add. A larger adversarial perturbation can increase the chances of being detected, cancel the purpose of the data, or even break its consistency, especially for highly structured data like network traffic [89, 236].

While we acknowledge the importance of data consistency and structure in network traffic, the state-of-the-art adversarial attacks we use were mainly developed for computer vision

tasks, thus they perturb all data features without awareness of their structure. In our experiments, we consider the worst-case scenario where the attacker perturbs all the data features, establishing a lower bound on the robustness of our method. If additional constraints are imposed on the attacker, we hypothesize that our defense method can only perform better compared to the worst-case scenario assumed in our experiments. However, in order to optimize the diffusion parameters, the design of real-world diffusion-based adversarial purification models should consider a realistic threat model.

6.5.6 Comparison with other defenses

While several adversarial defenses are presented in Section 6.2, they are applied to different intrusion detection datasets: [147] uses CICIDS2017 [93] and several other datasets; [224] and [237] use CICIDS2018 [93]; [148] uses KDD99. Furthermore, they also use different metrics: F1-score and FNR in [147]; attack-success rate and other metrics in [237]; while [224] and [148] use the accuracy. This diversity of datasets and metrics prevents the objective comparison of our method with previous work and highlights the necessity of establishing standard benchmarking datasets and metrics in intrusion detection.

6.6 Conclusion

Diffusion models are a promising approach to adversarial purification. Their seamless integration with existing systems and generalization across attack methods make them particularly interesting in the context of intrusion detection.

Throughout this paper, we have demonstrated the effectiveness of diffusion models in mitigating the threat of adversarial examples against intrusion detection. We have compared several diffusion neural network sizes, which show that larger neural networks yield lower loss values despite their increased demands in time and computational resources. Our analysis of the variance schedule indicates the importance of the final variance β_T in determining the purification performance, with smaller values achieving the highest accuracy. Furthermore, we have shown that the optimal amount of diffusion noise σ^{2^*} is nearly constant regardless of the number of diffusion steps T and that it approaches the value of the perturbation amplitude ϵ. However, in terms of purification performance, diffusion models with a larger T display better adversarial accuracy despite requiring more diffusion steps. Finally, we benchmarked our method against five state-of-the-art adversarial attacks and an increasing perturbation amplitude.

While scalability and computational complexity remain the main challenges for diffusion

models in intrusion detection, especially for inline detection, we envision future research endeavors to refine and optimize diffusion models for practical deployment. As novel adversarial attacks emerge and challenge adversarial defenses [238], our future work will focus on adapting diffusion-based purification to these attacks. Ultimately, complementing diffusion models with other defensive techniques remains necessary to prevent a single point of failure. In parallel, research efforts will be invested in generalizing our results to more sophisticated neural network architectures [235].

Acknowledgments

We would like to express our gratitude to Erwan Beurier for his valuable insights and contributions, which greatly enriched this research. This work was supported by Mitacs through the Mitacs Accelerate International program and the CRITiCAL chair. It was enabled in part by support provided by Calcul Québec, Compute Ontario, the BC DRI Group, and the Digital Research Alliance of Canada.

CHAPTER 7 CONCLUSION

This thesis discussed the vulnerability of ML-based intrusion detection models to two prominent adversarial threats: poisoning attacks through backdoors and evasion attacks through adversarial examples. This research focuses on recent DL approaches in intrusion detection, specifically privacy-preserving approaches with FL and Agent-Based Machine Learning (ABML) approaches with DRL. Through a comprehensive analysis of these threats, this work presents insights into the secure integration of these DL approaches in IDSs. It also introduces an effective adversarial purification defense to prevent evasion attacks. As a conclusion to this dissertation, we present a summary of the main contributions of this thesis and the future research perspectives we will be pursuing.

7.1 Summary of contributions

FL provides a privacy-preserving framework for collaborative training of ML models, but it also widens the attack surface as it allows malicious clients to interfere with the training. In the first contribution (Chapter 3), we address the threat of malicious clients in FL-based IDSs. After designing an FL-based IDS, we implement backdoor poisoning attacks and assess their impact on the global model. Through a comparative analysis of the data features, we determine the influence of trigger features on the attack's success (ASR) and its stealthiness (ACC). In particular, we show that using categorical features such as the state of the connection, the protocol, or the service as trigger features significantly increases the attack's success with limited impact on stealthiness. We also demonstrate how manipulating those features impacts the model event without poisoning it during the training, revealing the relation between backdoors and adversarial examples. However, when considering the data structure, we notice that manipulating those categorical features potentially compromises the network traffic's consistency, which we study comprehensively in Chapter 5. Moreover, by analyzing the evolution of the ASR through the FL training rounds, we identify five phases that the poisoning process undergoes, indicating the best timing for backdoor attacks. Finally, we vary the number of legitimate and malicious clients and notice similar patterns depending on the proportion of malicious clients; notably, the average ASR exceeds 95% with 20% malicious clients. This work constitutes a significant step towards understanding the impact of poisoning attacks, particularly backdoors, which opens the door to future research designing appropriate defenses for FL-based IDSs. Poisoning is not the only security issue related to ML. Evasion attacks are also a critical vulnerability, especially with the rise of adversarial

examples in DNNs.

DRL is also a promising avenue for IDSs. Its performance, flexibility, and adaptability are key to the intrusion detection task, but its dependence on DNNs makes it vulnerable to adversarial examples. In the second contribution (Chapter 4), we present the first study of adversarial examples against DRL-based IDSs. In particular, we focus on the impact of key DRL hyperparameters such as the width and depth of the neural network and the DRL algorithm on the robustness and performance. We demonstrate the influence of the underfitting and overfitting phenomena and how to prevent them by adapting the neural network architecture to the data complexity. While DRL algorithms differ in robustness depending on the neural network architecture and dataset, A2C demonstrates higher robustness to adversarial examples overall, closely followed by TRPO. In black-box settings, we show how adversarial examples transfer from a surrogate DRL agent to a target DRL using a different DRL algorithm. Adversarial examples generated with the DRL algorithm are often the most successful but in general, TRPO achieved the best transferability and A2C achieved the best robustness. This work is the first initiative that examines DRL-based IDSs with an adversarial lens, it provides key insights for the design of robust agents and lays the foundation for further investigations into this vulnerability and the appropriate defenses. On the other hand, it raises interesting questions about the consistency of adversarial examples regarding the structure of network traffic data.

Adversarial examples represent a serious threat to ML, especially in critical domains such as intrusion detection. However, adversarial perturbations ignore the data constraints, resulting in corrupt adversarial examples impractical in real-world implementations. In the third contribution (Chapter 5), we investigate the practicality of adversarial examples in ML-based intrusion detection. We first compare the impact of adversarial examples generated with 7 state-of-the-art methods on the detection rate of a DNN-powered IDS. Then we compare the data perturbation through the L_0 and L_∞ norms, highlighting the differences between the approaches spreading the perturbation on all the features and those focusing the perturbation on certain features. By analyzing how the adversarial perturbation is distributed on the features, we distinguish the features that are more frequently perturbed by each method and the level of control they require over the data, raising concerns about their feasibility. With a closer look, we identify 4 criteria induced by the perturbation that invalidate adversarial examples: value ranges, binary values, category membership, and semantic relations. While the latter criterion is hardly quantifiable, we reveal evidence of dependencies between the features through the correlation matrix. This research raises the fundamental issue of practicality in adversarial examples against ML-based intrusion detection that was ignored in previous work. It demonstrates the limits of adversarial examples that prevent

their implementation in real-world scenarios and provides insights for crafting practical adversarial examples. This foundation is necessary for a concrete evaluation of the risk posed by adversarial examples and the design of adequate defenses to protect ML-based IDSs.

A solid understanding of the vulnerability to adversarial examples favors the development of effective countermeasures. With the recent advances in DL, generative models achieve exceptional performance and can be leveraged to purify data from adversarial perturbations. In the fourth contribution (Chapter 6), we introduce the first adversarial purification defense based on diffusion models for ML-based intrusion detection. Our method is effective against five different adversarial attacks and an increasing amount of adversarial perturbation. We study several diffusion parameters, including β_1, β_T, and T, to identify the optimal configuration that maximizes the adversarial accuracy and minimizes the accuracy loss on normal data. We show how the adversarial purification task allows more control over the variance schedule since there is no need to transform the data into a normal distribution. We demonstrate that the optimal amount of diffusion noise does not depend on the number of diffusion steps but rather on the amount of adversarial perturbation. This work is the first initiative to defend ML-based IDSs with diffusion-based adversarial purification. It proposes an efficient plug-and-play countermeasure that does not require modifying the intrusion detection model and is independent of the adversarial attack. It lays down the groundwork for practical applications and future research addressing the remaining challenges: scalability and computational complexity. With this final work, we complement the defensive toolbox of ML-based IDSs with an effective method to counter adversarial examples, adding a supplementary obstacle for cyber attackers.

7.2 Future research perspectives

While this thesis has made notable strides in uncovering the vulnerabilities of ML-based IDSs to adversarial threats, there are several intriguing avenues for further exploration. The remainder of this section delves into key research directions that extend this work, aiming to achieve robust ML-based IDSs.

Extending the study to other DL technologies. This thesis focused on recent DL technologies integrated into IDSs, from DNNs and their applications in DRL (Chapter 4) to privacy-preserving ML with FL (Chapter 3), in addition to the diffusion generative models for adversarial purification (Chapter 6). However, with all the attention that cybersecurity researchers and industries give to DL, more technologies are being integrated into IDSs and require a comprehensive analysis to assess their robustness. For instance, Convolu-

tional Neural Networks (CNNs) are used to detect anomalies in graphical transformations on network data [239]. Recurrent Neural Networks (RNNs), in particular, Gate Recurrent Units (GRUs) [240] and Long-Short-Term Memory (LSTMs) [241], leverage the temporal sequencing of network packets to produce accurate and context-aware predictions [242, 243]. Graph Neural Networks (GNNs) [244] are used to model complex relationships and dependencies among network entities [245]. Furthermore, generative models such as Generative Adversarial Networks (GANs) [36] are used for imbalanced datasets [197] and learning only with benign data [246]. The list goes on, but the point is that the different mechanisms used by these technologies open new vulnerabilities for adversarial attackers. Thus, these technologies should undergo an in-depth analysis of their vulnerabilities regarding different adversarial threats. The concrete assessment of these vulnerabilities is necessary for the safe integration of DL technologies in critical domains such as intrusion detection and the design of efficient countermeasures.

Crafting practical adversarial attacks. The contribution in Chapter 5 demonstrates the different conditions that invalidate adversarial examples in the context of network intrusion detection. The generation process of adversarial perturbations does not consider the network data structure and generates perturbations that corrupt the instance. While our work establishes a foundation by identifying the concrete criteria that invalidate network data, these criteria should be integrated into the generation process of adversarial examples to ensure their consistency and implementability in real-world scenarios. Crafting practical adversarial examples is the only way to evaluate accurately the vulnerability of ML-based IDSs. Indeed, it is insufficient to decrease the model's performance by applying adversarial perturbations at the data level. For the concrete evaluation of the vulnerability, adversarial examples should be implemented as packets on the network and processed as such by the IDS. The same ideas extend to poisoning attacks, as described in Section 3.5, where the design of the trigger in backdoor attacks should consider the data structure and ensure the consistency of poisoned data. Furthermore, practical adversarial attacks are necessary to ensure the practical effectiveness of defense mechanisms.

Improving the scalability and complexity of defenses. The last contribution of this thesis (Chapter 6) introduced an adversarial purification method for ML-based IDSs by leveraging diffusion models. The generative capabilities of diffusion models allow the projection of adversarial examples in the original network data distribution with minimal reconstruction error. However, the reconstruction requires several steps of the reverse diffusion process which runs the data through a DNN at each step. Despite the hardware acceleration, this

process is still computationally complex and hardly scalable to high data bandwidths in production networks, which prevents the integration of diffusion-based adversarial purification into inline IDSs. In order to leverage the full potential of diffusion models, they should be optimized to require the minimum number of diffusion steps. The analysis of diffusion parameters presented in our contribution serves as a basis for a deeper investigation to improve the scalability and complexity. Furthermore, efforts should be dedicated to optimizing the hardware design of IDS appliances for DNNs.

Mot de la fin. Artificial intelligence is not always a panacea for all human challenges; in fact, it can sometimes engender even more problems than it resolves.

REFERENCES

[1] C. Sausalito, "Cybercrime to cost the world 8 trillion annually in 2023," 2022. [Online]. Available: https://cybersecurityventures.com/cybercrime-to-cost-the-world-8-trillion-annually-in-2023/

[2] M. Intelligence, "Cybersecurity market size & share analysis - growth trends & forecasts (2024 - 2029)," 2024. [Online]. Available: https://www.mordorintelligence.com/industry-reports/cyber-security-market

[3] D. Denning, "An Intrusion-Detection Model," *IEEE Transactions on Software Engineering*, 1987.

[4] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Communications Surveys & Tutorials*, 2016.

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

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[7] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[8] ——, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016.

[9] X. Xu and T. Xie, "A Reinforcement Learning Approach for Host-Based Intrusion Detection Using Sequences of System Calls," in *Advances in Intelligent Computing*, 2005.

[10] A. Servin and D. Kudenko, "Multi-agent Reinforcement Learning for Intrusion Detection," in *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*, 2008.

[11] G. Caminero, M. Lopez-Martin, and B. Carro, "Adversarial environment reinforcement learning algorithm for intrusion detection," *Computer Networks*, 2019.

[12] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," *Expert Systems with Applications*, 2020.

[13] B. McMahan *et al.*, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *20th International Conference on Artificial Intelligence and Statistics*, 2017.

[14] S. Agrawal *et al.*, "Federated Learning for intrusion detection system: Concepts, challenges and future directions," *Computer Communications*, 2022.

[15] E. M. Campos *et al.*, "Evaluating Federated Learning for intrusion detection in Internet of Things: Review and challenges," *Computer Networks*, 2022.

[16] L. Huang *et al.*, "Adversarial machine learning," in *ACM workshop on Security and artificial intelligence*, 2011.

[17] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, 2018.

[18] B. Nelson *et al.*, "Exploiting machine learning to subvert your spam filter," in *USENIX Workshop on Large-Scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More*, 2008.

[19] N. Dalvi *et al.*, "Adversarial classification," in *ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004.

[20] F. Tramèr *et al.*, "Stealing Machine Learning Models via Prediction {APIs}," in *USENIX Security Symposium*, 2016.

[21] M. Fredrikson, S. Jha, and T. Ristenpart, "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures," in *ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[22] P. Blanchard *et al.*, "Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent," in *Advances in Neural Information Processing Systems*, 2017.

[23] A. N. Bhagoji *et al.*, "Analyzing Federated Learning through an Adversarial Lens," in *36th International Conference on Machine Learning*, 2019.

[24] Z. Sun *et al.*, "Can You Really Backdoor Federated Learning?" 2019, arXiv:1911.07963 [cs, stat].

[25] E. Bagdasaryan *et al.*, "How To Backdoor Federated Learning," in *23rd International Conference on Artificial Intelligence and Statistics*, 2020.

[26] T. D. Nguyen *et al.*, "Poisoning Attacks on Federated Learning-based IoT Intrusion Detection System," in *Workshop on Decentralized IoT Systems and Security*, 2020.

[27] Z. Zhang *et al.*, "SecFedNIDS: Robust defense for poisoning attack against federated learning-based network intrusion detection system," *Future Generation Computer Systems*, 2022.

[28] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *International Conference on Learning Representations (ICLR)*, 2014.

[29] N. Papernot *et al.*, "The Limitations of Deep Learning in Adversarial Settings," in *IEEE European Symposium on Security and Privacy*, 2016.

[30] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations*, 2015.

[31] X. Yuan *et al.*, "Adversarial Examples: Attacks and Defenses for Deep Learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[32] S. Huang *et al.*, "Adversarial Attacks on Neural Network Policies," 2017, arXiv:1702.02284 [cs, stat].

[33] A. Madry *et al.*, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[34] J. H. Metzen *et al.*, "On detecting adversarial perturbations," in *International Conference on Learning Representations*, 2017.

[35] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," in *International Conference on Learning Representations, ICLR*, 2015.

[36] I. Goodfellow *et al.*, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*, 2014.

[37] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," 2022, arXiv:1312.6114.

[38] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models," in *International Conference on Learning Representations*, 2018.

[39] U. Hwang *et al.*, "PuVAE: A Variational Autoencoder to Purify Adversarial Examples," *IEEE Access*, 2019.

[40] J. Sohl-Dickstein *et al.*, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics," in *International Conference on Machine Learning*, 2015.

[41] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in *Advances in Neural Information Processing Systems*, 2020.

[42] W. Nie *et al.*, "Diffusion Models for Adversarial Purification," in *International Conference on Machine Learning*, 2022.

[43] J. Wang *et al.*, "Guided Diffusion Model for Adversarial Purification," 2022, arXiv:2205.14969.

[44] Q. Wu, H. Ye, and Y. Gu, "Guided Diffusion Model for Adversarial Purification from Random Noise," 2022, arXiv:2206.10875.

[45] T. J. Sejnowski, *The deep learning revolution.* MIT press, 2018.

[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[47] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, arXiv:1409.1556.

[48] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014.

[49] K. He *et al.*, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[50] S. Ren *et al.*, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015.

[51] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[52] A. Vaswani *et al.*, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 599–609.

[53] J. Devlin *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *North american chapter of the association for computational linguistics (naacl)*, 2018, pp. 4171–4186.

[54] OpenAI, "Gpt-3: Language models are few-shot learners," 2020.

[55] G. E. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition," in *IEEE international conference on acoustics, speech, and signal processing*, 2012.

[56] W. Chan, N. Jaitly, and Q. V. Le, "Listen, attend and spell: A neural network for hybrid speech recognition," in *IEEE international conference on acoustics, speech, and signal processing*, 2015.

[57] A. Hannum *et al.*, "Deep speech 2: Deep bidirectional lstm modeling for automatic speech recognition," 2017, arXiv:1703.00140.

[58] K. L. Fox *et al.*, "A neural network approach towards intrusion detection," in *13th national computer security conference*, 1990.

[59] Y. Xin *et al.*, "Machine Learning and Deep Learning Methods for Cybersecurity," *IEEE Access*, 2018.

[60] A. Khraisat *et al.*, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, 2019.

[61] A. Drewek-Ossowicka, M. Pietrołaj, and J. Rumiński, "A survey of neural networks usage for intrusion detection systems," *Journal of Ambient Intelligence and Humanized Computing*, 2021.

[62] R. Vinayakumar *et al.*, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, 2019.

[63] J. Kim *et al.*, "Method of intrusion detection using deep neural network," in *IEEE international conference on big data and smart computing (BigComp)*, 2017.

[64] N. Shone *et al.*, "A Deep Learning Approach to Network Intrusion Detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2018.

[65] T. A. Tang *et al.*, "Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks," in *IEEE Conference on Network Softwarization and Workshops (NetSoft)*, 2018.

[66] Y. Xiao *et al.*, "An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks," *IEEE Access*, 2019.

[67] K. Wu, Z. Chen, and W. Li, "A Novel Intrusion Detection Model for a Massive Network Using Convolutional Neural Networks," *IEEE Access*, 2018.

[68] D. Lowd and C. Meek, "Adversarial learning," in *ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005.

[69] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[70] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," in *IEEE Symposium on Security and Privacy (SP)*, 2017.

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

[72] P.-Y. Chen *et al.*, "ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models," in *10th ACM Workshop on Artificial Intelligence and Security*, 2017.

[73] I. Corona, G. Giacinto, and F. Roli, "Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues," *Information Sciences*, 2013.

[74] N. Martins *et al.*, "Adversarial Machine Learning Applied to Intrusion and Malware Scenarios: A Systematic Review," *IEEE Access*, 2020.

[75] H. A. Alatwi and C. Morisset, "Adversarial Machine Learning In Network Intrusion Detection Domain: A Systematic Review," 2021, arXiv:2112.03315 [cs].

[76] K. He, D. D. Kim, and M. R. Asghar, "Adversarial Machine Learning for Network Intrusion Detection Systems: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, 2023.

[77] M. Rigaki and A. Elragal, "Adversarial deep learning against intrusion detection classifiers," in *NATO IST-152 Workshop on Intelligent Autonomous Agents for Cyber Defence and Resilience*, 2017.

[78] M. Tavallaee *et al.*, "A detailed analysis of the kdd cup 99 data set," in *IEEE symposium on computational intelligence for security and defense applications*, 2009.

[79] Z. Wang, "Deep learning-based intrusion detection with adversaries," *IEEE Access*, 2018.

[80] J. Clements *et al.*, "Rallying Adversarial Techniques against Deep Learning for Network Security," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2021.

[81] Y. Mirsky *et al.*, "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection," in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, February 18, 2018 - February 21, 2018*, 2018.

[82] O. Ibitoye, O. Shafiq, and A. Matrawy, "Analyzing Adversarial Attacks against Deep Learning for Intrusion Detection in IoT Networks," in *IEEE Global Communications Conference (GLOBECOM)*, 2019.

[83] N. Koroniotis *et al.*, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Generation Computer Systems*, 2019.

[84] K. Yang *et al.*, "Adversarial Examples Against the Deep Learning Based Network Intrusion Detection Systems," in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, 2018.

[85] Z. Lin, Y. Shi, and Z. Xue, "IDSGAN: Generative Adversarial Networks for Attack Generation Against Intrusion Detection," in *Advances in Knowledge Discovery and Data Mining*, 2022.

[86] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein Generative Adversarial Networks," in *34th International Conference on Machine Learning*, 2017.

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

[88] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Military Communications and Information Systems Conference (MilCIS)*, 2015.

[89] M. A. Merzouk *et al.*, "Investigating the practicality of adversarial evasion attacks on network intrusion detection," *Annals of Telecommunications*, 2022.

[90] M. Ring *et al.*, "Flow-based benchmark data sets for intrusion detection," in *16th European Conference on Cyber Warfare and Security. ACPI*, 2017.

[91] R. Sheatsley *et al.*, "Adversarial examples for network intrusion detection systems," *Journal of Computer Security*, 2022.

[92] B.-E. Zolbayar *et al.*, "Generating Practical Adversarial Network Traffic Flows Using NIDSGAN," 2022, arXiv:2203.06694 [cs].

[93] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in *4th International Conference on Information Systems Security and Privacy*, 2018.

[94] Q. Cheng *et al.*, "Packet-Level Adversarial Network Traffic Crafting using Sequence Generative Adversarial Networks," 2021, arXiv:2103.04794 [cs].

[95] L. Yu *et al.*, "SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[96] D. Han *et al.*, "Evaluating and Improving Adversarial Robustness of Machine Learning-Based Network Intrusion Detectors," *IEEE Journal on Selected Areas in Communications*, 2021.

[97] J. Schulman *et al.*, "Proximal policy optimization algorithms," 2017, arXiv:1707.06347.

[98] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, 1988.

[99] X. Xu and Y. Luo, "A Kernel-Based Reinforcement Learning Approach to Dynamic Behavior Modeling of Intrusion Detection," in *Advances in Neural Networks – ISNN*, 2007.

[100] X. Xu, "Sequential anomaly detection based on temporal-difference learning: Principles, models and case studies," *Applied Soft Computing*, 2010.

[101] ——, "A Sparse Kernel-Based Least-Squares Temporal Difference Algorithm for Reinforcement Learning," in *Advances in Natural Computation*, 2006.

[102] T. T. Nguyen and V. J. Reddi, "Deep Reinforcement Learning for Cyber Security," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[103] S. Shamshirband *et al.*, "Cooperative game theoretic approach using fuzzy Q-learning for detecting and preventing intrusions in wireless sensor networks," *Engineering Applications of Artificial Intelligence*, 2014.

[104] B. Deokar and A. Hazarnis, "Intrusion detection system using log files and reinforcement learning," *International Journal of Computer Applications*, 2012.

[105] K. Malialis and D. Kudenko, "Distributed response to network intrusions using multiagent reinforcement learning," *Engineering Applications of Artificial Intelligence*, 2015.

[106] C. Kolias *et al.*, "Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset," *IEEE Communications Surveys Tutorials*, 2016.

[107] M. A. Merzouk *et al.*, "Evading Deep Reinforcement Learning-based Network Intrusion Detection with Adversarial Attacks," in *17th International Conference on Availability, Reliability and Security*, Aug. 2022.

[108] V. Behzadan and A. Munir, "Vulnerability of Deep Reinforcement Learning to Policy Induction Attacks," in *Machine Learning and Data Mining in Pattern Recognition*, 2017.

[109] J. Kos and D. Song, "Delving into adversarial attacks on deep policies," in *5th International Conference on Learning Representations*, 2019.

[110] Y.-C. Lin *et al.*, "Tactics of Adversarial Attack on Deep Reinforcement Learning Agents," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017.

[111] Z. Xiong *et al.*, "Defending observation attacks in deep reinforcement learning via detection and denoising," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2022, pp. 235–250.

[112] I. Ilahi *et al.*, "Challenges and countermeasures for adversarial attacks on deep reinforcement learning," *IEEE Transactions on Artificial Intelligence*, 2021.

[113] J. Sun *et al.*, "Stealthy and efficient adversarial attacks against deep reinforcement learning," in *AAI Conference on Artificial Intelligence*, 2020.

[114] T. Chen *et al.*, "Adversarial attack and defense in reinforcement learning-from ai security view," *Cybersecurity*, 2019.

[115] T. Hickling, N. Aouf, and P. Spencer, "Robust adversarial attacks detection based on explainable deep reinforcement learning for uav guidance and planning," *IEEE Transactions on Intelligent Vehicles*, 2023.

[116] M. A. Merzouk *et al.*, "Adversarial robustness of deep reinforcement learning-based intrusion detection," *International Journal of Information Security*, 2024.

[117] E. C. P. Neto *et al.*, "CICIoV2024: Advancing realistic IDS approaches against DoS and spoofing attack in IoV CAN bus," *Internet of Things*, vol. 26, 2024.

[118] L. Sweeney, "Simple demographics often identify people uniquely," *Health (San Francisco)*, 2000.

[119] S. Abdul Rahman *et al.*, "Internet of Things Intrusion Detection: Centralized, On-Device, or Federated Learning?" *IEEE Network*, 2020.

[120] T. D. Nguyen *et al.*, "DÏoT: A Federated Self-learning Anomaly Detection System for IoT," in *International Conference on Distributed Computing Systems (ICDCS)*, 2019.

[121] B. Cetin *et al.*, "Federated Wireless Network Intrusion Detection," in *IEEE International Conference on Big Data (Big Data)*, 2019.

[122] Z. Chen *et al.*, "Intrusion Detection for Wireless Edge Networks Based on Federated Learning," *IEEE Access*, 2020.

[123] X. Wang *et al.*, "Toward Accurate Anomaly Detection in Industrial Internet of Things Using Hierarchical Federated Learning," *IEEE Internet of Things Journal*, 2022.

[124] J. Li *et al.*, "FLEAM: A Federated Learning Empowered Architecture to Mitigate DDoS in Industrial IoT," *IEEE Transactions on Industrial Informatics*, 2022.

[125] Y. Sun, H. Ochiai, and H. Esaki, "Intrusion Detection with Segmented Federated Learning for Large-Scale Multiple LANs," in *International Joint Conference on Neural Networks (IJCNN)*, 2020.

[126] Y. Sun, H. Esaki, and H. Ochiai, "Adaptive Intrusion Detection in the Networking of Large-Scale LANs With Segmented Federated Learning," *IEEE Open Journal of the Communications Society*, 2021.

[127] B. Li *et al.*, "DeepFed: Federated Deep Learning for Intrusion Detection in Industrial Cyber–Physical Systems," *IEEE Transactions on Industrial Informatics*, 2021.

[128] K. Li *et al.*, "Distributed Network Intrusion Detection System in Satellite-Terrestrial Integrated Networks Using Federated Learning," *IEEE Access*, 2020.

[129] Y. Zhao *et al.*, "Multi-Task Network Anomaly Detection using Federated Learning," in *Tenth International Symposium on Information and Communication Technology - SoICT 2019*, 2019.

[130] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *29th International Coference on International Conference on Machine Learning*, 2012.

[131] R. Gosselin *et al.*, "Privacy and Security in Federated Learning: A Survey," *Applied Sciences*, 2022.

[132] L. Lyu *et al.*, "Privacy and Robustness in Federated Learning: Attacks and Defenses," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[133] T. D. Nguyen *et al.*, "Backdoor attacks and defenses in federated learning: Survey, challenges and future research directions," *Engineering Applications of Artificial Intelligence*, 2024.

[134] M. A. Merzouk *et al.*, "Parameterizing poisoning attacks in federated learning-based intrusion detection," in *18th International Conference on Availability, Reliability and Security*, 2023.

[135] Y.-C. Lai *et al.*, "Two-phase Defense Against Poisoning Attacks on Federated Learning-based Intrusion Detection," *Computers & Security*, 2023.

[136] R. Yang *et al.*, "Dependable federated learning for IoT intrusion detection against poisoning attacks," *Computers & Security*, 2023.

[137] J. Lu, T. Issaranon, and D. Forsyth, "Safetynet: Detecting and rejecting adversarial examples robustly," in *IEEE international conference on computer vision*, 2017.

[138] Z. Gong and W. Wang, "Adversarial and clean data are not twins," in *international workshop on exploiting artificial intelligence techniques for data management*, 2023.

[139] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *ACM SIGSAC conference on computer and communications security*, 2017.

[140] Y. Song *et al.*, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," in *International Conference on Learning Representations*, 2018.

[141] H. Xu *et al.*, "Adversarial Attacks and Defenses in Images, Graphs and Text: A Review," *International Journal of Automation and Computing*, 2020.

[142] X. Liu *et al.*, "Privacy and Security Issues in Deep Learning: A Survey," *IEEE Access*, 2021.

[143] R. Abou Khamis, M. O. Shafiq, and A. Matrawy, "Investigating Resistance of Deep Learning-based IDS against Adversaries using min-max Optimization," in *IEEE International Conference on Communications (ICC)*, 2020.

[144] R. Abou Khamis and A. Matrawy, "Evaluation of Adversarial Training on Different Types of Neural Networks in Deep Learning-based IDSs," in *International Symposium on Networks, Computers and Communications (ISNCC)*, 2020.

[145] C. Benzaïd, M. Boukhalfa, and T. Taleb, "Robust Self-Protection Against Application-Layer (D)DoS Attacks in SDN Environment," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2020.

[146] I. Debicha *et al.*, "Adversarial training for deep learning-based intrusion detection systems," in *The Sixteenth International Conference on Systems (ICONS)*, 2021.

[147] M. Abdelaty *et al.*, "GADoT: GAN-based Adversarial Training for Robust DDoS Attack Detection," in *IEEE Conference on Communications and Network Security (CNS)*, 2021.

[148] M. Usama *et al.*, "Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems," in *15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, 2019.

[149] M. P. Novaes *et al.*, "Adversarial Deep Learning approach detection and defense against DDoS attacks in SDN environments," *Future Generation Computer Systems*, 2021.

[150] J. Wang *et al.*, "Def-IDS: An Ensemble Defense Mechanism Against Adversarial Attacks for Deep Learning-based Network Intrusion Detection," in *International Conference on Computer Communications and Networks (ICCCN)*, 2021.

[151] C. Zhang, X. Costa-Perez, and P. Patras, "Tiki-Taka: Attacking and Defending Deep Learning-based Intrusion Detection Systems," in *ACM SIGSAC Conference on Cloud Computing Security Workshop*, 2020.

[152] S. Bell and K. Bala, "Learning visual similarity for product design with convolutional neural networks," *ACM transactions on graphics (TOG)*, 2015.

[153] G. Apruzzese *et al.*, "AppCon: Mitigating Evasion Attacks to ML Cyber Detectors," *Symmetry*, 2020.

[154] G. Apruzzese and M. Colajanni, "Evading botnet detectors based on flows and random forest with adversarial samples," in *International Symposium on Network Computing and Applications (NCA)*, 2018.

[155] A. McCarthy *et al.*, "Feature Vulnerability and Robustness Assessment against Adversarial Machine Learning Attacks," in *International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, 2021.

[156] A. Ganesan and K. Sarac, "Mitigating Evasion Attacks on Machine Learning based NIDS Systems in SDN," in *IEEE 7th International Conference on Network Softwarization (NetSoft)*, 2021.

[157] A.-U.-H. Qureshi *et al.*, "An Adversarial Attack Detection Paradigm With Swarm Optimization," in *International Joint Conference on Neural Networks (IJCNN)*, 2020.

[158] M. Pawlicki, M. Choraś, and R. Kozik, "Defending network intrusion detection systems against adversarial evasion attacks," *Future Generation Computer Systems*, 2020.

[159] Y. Peng *et al.*, "Detecting Adversarial Examples for Network Intrusion Detection System with GAN," in *International Conference on Software Engineering and Service Science (ICSESS)*, 2020.

[160] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," in *International Conference on Learning Representations*, 2017.

[161] I. Debicha *et al.*, "Detect & Reject for Transferability of Black-Box Adversarial Attacks Against Network Intrusion Detection Systems," in *Advances in Cyber Security*. Springer, 2021.

[162] K. Grosse *et al.*, "On the (Statistical) Detection of Adversarial Examples," 2017, arXiv:1702.06280 [cs, stat].

[163] N. Wang *et al.*, "MANDA: On Adversarial Example Detection for Network Intrusion Detection System," *IEEE Transactions on Dependable and Secure Computing*, 2023.

[164] M. J. Hashemi and E. Keller, "Enhancing Robustness Against Adversarial Examples in Network Intrusion Detection Systems," in *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2020.

[165] M. J. Hashemi, G. Cusack, and E. Keller, "Towards Evaluation of NIDSs in Adversarial Setting," in *CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*, 2019.

[166] Y. Song *et al.*, "Score-Based Generative Modeling through Stochastic Differential Equations," in *International Conference on Learning Representations*, 2021.

[167] T. Gu, B. Dolan-Gavitt, and S. Garg, "BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain," in *NIPS Workshop on Machine Learning and Computer Security*, 2017.

[168] D. Preuveneers *et al.*, "Chained Anomaly Detection Models for Federated Learning: An Intrusion Detection Case Study," *Applied Sciences*, 2018.

[169] L. Zhu, Z. Liu, and S. Han, "Deep Leakage from Gradients," in *Advances in Neural Information Processing Systems*, 2019.

[170] T. D. Nguyen *et al.*, "FLAME: Taming Backdoors in Federated Learning," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022.

[171] E. Hallaji *et al.*, "Label noise analysis meets adversarial training: A defense against label poisoning in federated learning," *Knowledge-Based Systems*, 2023.

[172] R. Alshammari and A. N. Zincir-Heywood, "Can encrypted traffic be identified without port numbers, ip addresses and payload inspection?" *Computer Networks*, 2011.

[173] M. Ring *et al.*, "A survey of network-based intrusion detection data sets," *Computers & Security*, 2019.

[174] F. Mo *et al.*, "PPFL: privacy-preserving federated learning with trusted execution environments," in *19th Annual International Conference on Mobile Systems, Applications, and Services*, 2021.

[175] C. Bullard, "ARGUS - Auditing Network Activity - Manuals," 2007. [Online]. Available: https://openargus.org/oldsite/manuals.shtml

[176] C. Xie *et al.*, "DBA: Distributed Backdoor Attacks against Federated Learning," in *8th International Conference on Learning Representations*, 2022.

[177] Ajay Jotwani, "Why Cybersecurity Should Be Top Of Mind In 2023," 2023. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2023/01/06/why-cybersecurity-should-be-top-of-mind-in-2023/?sh=1654d131235c

[178] H. Debar, M. Becker, and D. Siboni, "A neural network component for an intrusion detection system," in *Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, 1992.

[179] S. Levine *et al.*, "End-to-End Training of Deep Visuomotor Policies," 2016, arXiv:1504.00702 [cs].

[180] B. R. Kiran *et al.*, "Deep Reinforcement Learning for Autonomous Driving: A Survey," 2021, arXiv:2002.00444 [cs].

[181] A. M. Pasikhani, J. A. Clark, and P. Gope, "Adversarial RL-Based IDS for Evolving Data Environment in 6LoWPAN," *IEEE Transactions on Information Forensics and Security*, 2022.

[182] M. A. Merzouk *et al.*, "A Deeper Analysis of Adversarial Examples in Intrusion Detection," in *15th International Conference on Risks and Security of Internet and Systems*, 2020.

[183] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *5th International Conference on Learning Representations*, 2017.

[184] H.-J. Liao *et al.*, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, 2013.

[185] C. Kreibich and J. Crowcroft, "Honeycomb: creating intrusion detection signatures using honeypots," *ACM SIGCOMM Computer Communication Review*, 2004.

[186] S. Chebrolu, A. Abraham, and J. P. Thomas, "Feature deduction and ensemble design of intrusion detection systems," *Computers & Security*, 2005.

[187] K. Bajaj and A. Arora, "Improving the Intrusion Detection using Discriminative Machine Learning Approach and Improve the Time Complexity by Data Mining Feature Selection Methods," *International Journal of Computer Applications*, 2013.

[188] C. Annachhatre, T. H. Austin, and M. Stamp, "Hidden Markov models for malware classification," *Journal of Computer Virology and Hacking Techniques*, 2015.

[189] S. Mohamed and R. Ejbali, "Deep SARSA-based reinforcement learning approach for anomaly network intrusion detection system," *International Journal of Information Security*, 2023.

[190] M. Sewak, S. K. Sahay, and H. Rathore, "Deep Reinforcement Learning in the Advanced Cybersecurity Threat Detection and Protection," *Information Systems Frontiers*, 2023.

[191] S. Priya and K. P. M. Kumar, "Binary bat algorithm based feature selection with deep reinforcement learning technique for intrusion detection system," *Soft Computing*, 2023.

[192] J. F. Cevallos M. *et al.*, "Deep Reinforcement Learning for intrusion detection in Internet of Things: Best practices, lessons learnt, and open challenges," *Computer Networks*, 2023.

[193] H. Moudoud and S. Cherkaoui, "Empowering security and trust in 5g and beyond: A deep reinforcement learning approach," *IEEE Open Journal of the Communications Society*, 2023.

[194] Z. Abou El Houda, H. Moudoud, and B. Brik, "Federated deep reinforcement learning for efficient jamming attack mitigation in o-ran," *IEEE Transactions on Vehicular Technology*, 2024.

[195] M. A. Umer *et al.*, "Machine learning for intrusion detection in industrial control systems: Applications, challenges, and recommendations," *International Journal of Critical Infrastructure Protection*, 2022.

[196] N. Papernot *et al.*, "Practical Black-Box Attacks against Machine Learning," in *Proceedings of the ACM on Asia Conference on Computer and Communications Security*, 2017.

[197] I. Yilmaz, R. Masum, and A. Siraj, "Addressing imbalanced data problem with generative adversarial network for intrusion detection," in *IEEE 21st International Conference on Information Reuse and Integration for Data Science*, 2020.

[198] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, 2015.

[199] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-Learning," in *AAAI Conference on Artificial Intelligence*, 2016.

[200] Z. Wang *et al.*, "Dueling network architectures for deep reinforcement learning," in *33rd International Conference on International Conference on Machine Learning - Volume 48*, 2016.

[201] M. Hessel *et al.*, "Rainbow: combining improvements in deep reinforcement learning," in *AAAI Conference on Artificial Intelligence*, 2018.

[202] W. Dabney *et al.*, "Distributional reinforcement learning with quantile regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[203] J. Schulman *et al.*, "Trust region policy optimization," in *International conference on machine learning*, 2015.

[204] ——, "High-dimensional continuous control using generalized advantage estimation," 2016, arXiv:1506.02438.

[205] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2016, arXiv:1509.02971.

[206] F. Pierazzi *et al.*, "Intriguing Properties of Adversarial ML Attacks in the Problem Space," in *IEEE Symposium on Security and Privacy (SP)*, 2020.

[207] A. Raffin *et al.*, "Stable-baselines3: Reliable reinforcement learning implementations," *J. Mach. Learn. Res.*, 2021.

[208] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in neural information processing systems*, 2019.

[209] M.-I. Nicolae *et al.*, "Adversarial Robustness Toolbox v1.0.0," *arXiv:1807.01069*, 2019.

[210] J. Farebrother, M. C. Machado, and M. Bowling, "Generalization and Regularization in DQN," in *NeurIPS 2018: Deep Reinforcement Learning Workshop*, 2018.

[211] L. Rice, E. Wong, and Z. Kolter, "Overfitting in adversarially robust deep learning," in *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[212] M. H. Meng *et al.*, "Adversarial Robustness of Deep Neural Networks: A Survey from a Formal Verification Perspective," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[213] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *International Conference on Learning Representations*, 2017.

[214] L. Gao *et al.*, "Push & Pull: Transferable Adversarial Examples With Attentive Attack," *IEEE Transactions on Multimedia*, 2021.

[215] C. Xiao *et al.*, "Generating adversarial examples with adversarial networks," in *27th International Joint Conference on Artificial Intelligence, IJCAI*, 2018.

[216] T. Bai *et al.*, "AI-GAN: Attack-Inspired Generation of Adversarial Examples," in *IEEE International Conference on Image Processing (ICIP)*, 2021.

[217] S. Garg and G. Ramakrishnan, "BAE: BERT-based Adversarial Examples for Text Classification," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[218] J. Devlin *et al.*, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019.

[219] I. Sharafaldin *et al.*, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy," in *International Carnahan Conference on Security Technology (ICCST)*, 2019.

[220] L. Dhanabal and S. Shantharajah, "A study on nsl-kdd dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, 2015.

[221] A. Aldahdooh *et al.*, "Adversarial example detection for dnn models: A review and experimental comparison," *Artificial Intelligence Review*, 2022.

[222] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *ACM workshop on artificial intelligence and security*, 2017.

[223] V. Srinivasan *et al.*, "Robustifying models against adversarial attacks by langevin dynamics," *Neural Networks*, 2021.

[224] J. Wang *et al.*, "Def-ids: An ensemble defense mechanism against adversarial attacks for deep learning-based network intrusion detection," in *2021 International Conference on Computer Communications and Networks (ICCCN)*, 2021.

[225] K. Zhang *et al.*, "Ada3diff: Defending against 3d adversarial point clouds via adaptive diffusion," in *ACM International Conference on Multimedia*, 2023.

[226] G. Lin *et al.*, "Robust Diffusion Models for Adversarial Purification," 2024, arXiv:2403.16067.

[227] Y. Shi *et al.*, "Black-box backdoor defense via zero-shot image purification," in *Advances in Neural Information Processing Systems*, 2023.

[228] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Applied Sciences*, 2019.

[229] I. Sohn, "Deep belief network based intrusion detection techniques: A survey," *Expert Systems with Applications*, 2021.

[230] W. Zhang *et al.*, "Did-ids: A novel diffusion-based imbalanced data intrusion detection system," in *IEEE International Conference on Information, Communication and Networks (ICICN)*, 2023.

[231] B. Tang *et al.*, "A diffusion model based on network intrusion detection method for industrial cyber-physical systems," *Sensors*, 2023.

[232] F. Han *et al.*, "Mmid-bench: A comprehensive benchmark for multi-domain multi-category intrusion detection," *IEEE Transactions on Intelligent Vehicles*, 2024.

[233] Y. Wang *et al.*, "Intrusion detection method based on denoising diffusion probabilistic models for uav networks," *Mobile Networks and Applications*, 2023.

[234] C. Yang, T. Wang, and X. Yan, "Ddmt: Denoising diffusion mask transformer models for multivariate time series anomaly detection," 2023, arXiv:2310.08800.

[235] A. Javaid *et al.*, "A deep learning approach for network intrusion detection system," *EAI Endorsed Transactions on Security and Safety*, 2016.

[236] G. Apruzzese *et al.*, "Modeling realistic adversarial attacks against network intrusion detection systems," *Digital Threats*, 2022.

[237] C. Zhang, X. Costa-Pérez, and P. Patras, "Adversarial attacks against deep learning-based network intrusion detection systems and defense mechanisms," *IEEE/ACM Transactions on Networking*, 2022.

[238] M. Kang, D. Song, and B. Li, "Diffattack: Evasion attacks against diffusion-based adversarial purification," in *Advances in Neural Information Processing Systems*, 2023.

[239] Z. Li *et al.*, "Intrusion Detection Using Convolutional Neural Networks for Representation Learning," in *Neural Information Processing*, 2017.

[240] K. Cho *et al.*, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[241] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, 1997.

[242] M. S. Ansari, V. Bartoš, and B. Lee, "GRU-based deep learning approach for network intrusion alert prediction," *Future Generation Computer Systems*, 2022.

[243] Y. Imrana *et al.*, "A bidirectional LSTM deep learning approach for intrusion detection," *Expert Systems with Applications*, 2021.

[244] F. Scarselli *et al.*, "The Graph Neural Network Model," *IEEE Transactions on Neural Networks*, 2009.

[245] W. W. Lo *et al.*, "E-GraphSAGE: A Graph Neural Network based Intrusion Detection System for IoT," in *Network Operations and Management Symposium*, 2022.

[246] E. Seo, H. M. Song, and H. K. Kim, "GIDS: GAN based Intrusion Detection System for In-Vehicle Network," in *Conference on Privacy, Security and Trust (PST)*, 2018.

[247] Z. Hu *et al.*, *Reinforcement Learning for Adaptive Cyber Defense Against Zero-Day Attacks*. Springer, 2019.

APPENDIX A ARTICLE 4: EVADING DEEP REINFORCEMENT LEARNING-BASED NETWORK INTRUSION DETECTION WITH ADVERSARIAL ATTACKS

Published in the *Proceedings of the 17th International Conference on Availability, Reliability, and Security (ARES 2022)* on *23 August 2022*.

Authors Mohamed Amine Merzouk[1,2], Joséphine Delas[1,2], Christopher Neal[1,2], Frédéric Cuppens[1], Nora Boulahia-Cuppens[1], Reda Yaich[2]

Institutions [1] Polytechnique Montréal, Canada; [2] IRT SystemX, France

Abstract An Intrusion Detection System (IDS) aims to detect attacks conducted over computer networks by analyzing traffic data. Deep Reinforcement Learning (DRL) is a promising lead in IDS research, due to its lightness and adaptability. However, the neural networks on which DRL is based can be vulnerable to adversarial attacks. By applying a well-computed modification to malicious traffic, adversarial examples can evade detection. In this paper, we test the performance of a state-of-the-art DRL IDS agent against the Fast Gradient Sign Method (FGSM) and Basic Iterative Method (BIM) adversarial attacks. We demonstrate that the performance of the DRL detection agent is compromised in the face of adversarial examples and highlight the need for future DRL IDS work to consider mechanisms for coping with adversarial examples.

Keywords adversarial machine learning, adversarial examples, intrusion detection, reinforcement learning, evasion attacks

Introduction

Concern about security attacks on modern connected systems such as internet-connected devices or critical data servers has been growing for the past two decades. Intrusion Detection Systems (IDSs) are thus widely used as an automatic way of detecting potential threats within network connections, and their performances are constantly challenged to cope with the development of increasingly sophisticated cyberattacks.

Supervised Learning (SL) has introduced a whole new set of capabilities into IDS technology,

leading to spectacular progress in intrusion detection tasks [4]. Still, a particularly difficult task for an IDS remains the detection of previously unseen anomalies (i.e. zero-day attacks). Reinforcement learning (RL) is a promising lead in IDS research, as it constitutes an adaptive and responsive environment suitable for online training, resulting in simple and fast prediction agents [12,247]. However, the most efficient RL-based IDS implementations use Deep Neural Networks (DNNs) at their core, which have been shown to be vulnerable to adversarial examples [32,110]. These attacks involve slightly modifying data samples in order to mislead a classification model. Previous work has evaluated the effects of adversarial examples on DNN-based IDSs [182], yet little is known about the vulnerability of RL-based detection methods to adversarial examples.

In this paper, we investigate the performance of a state-of-the-art DRL intrusion detection agent when exposed to adversarial attacks. Caminero *et al.* [11] present a novel approach that has been shown to outperform other RL and SL-based detection models. They trained a DRL agent in an adversarial environment using the NSL-KDD dataset [78]. In this paper, we show how adversarial examples generated using two methods [30, 213] can evade the detection of the agent. In keeping the consistency with initial studies in this domain, we consider white-box individual attacks where the intruder has access to the parameters of the model [29,110].

The remainder of the paper is organized as follows. Section A provides an overview of influential works applying RL to IDSs and a review of adversarial example generation methods for DRL agents. The methodology for this paper is provided in Section A, where we describe the dataset, the RL detection agent, and the adversarial attacks used in our experiments. In Section A, we present the results achieved by adversarial examples on the performance of the agent. A discussion of the immediate practicality of these attacks and an outline for future work is provided in Section A. Lastly, Section A provides some concluding remarks.

Related work

RL techniques have an extensive range of applications in cybersecurity due to their adaptive nature and the rapidity of their predictive models. The first works concerning RL and intrusion detection were published in the early 2000s and present mostly innovative works using tabular methods. Servin *et al.* [10] use a Q-learning algorithm based on a look-up table to detect network intrusions, whereas Xu *et al.* [9] introduce Temporal Difference (TD) learning algorithms for live detection.

More recently, the development of DRL algorithms has further improved the performances of

IDS models [12]. In particular, Caminero *et al.* [11] present an innovative multi-agent deep reinforcement learning model that outperforms previous tabular methods, as well as several other DNN models. Their algorithm is based on the concurrency of two different agents to improve the predictions.

Despite the remarkable performance shown by RL agents in intrusion detection, there is a concern about their reliability in the presence of adversarial attacks. Since DRL agents rely on DNNs, they could be vulnerable to malicious inputs, chiefly adversarial examples. Behzadan *et al.* [108] first explored the effect of adversarial examples on Deep Q-Networks (DQNs). The authors use two well-known attacks, namely, Fast Gradient Sign Method (FGSM) [30] and Jacobian-based Saliency Map Attack (JSMA) [29], to perturb the training of a game-learning agent. They also demonstrate the transferability of adversarial examples between agents. Huang *et al.* [32] show how an adversary could interfere with the operations of a trained RL agent. The authors use FGSM to generate adversarial examples in both white-box and black-box settings by utilizing the transferability property [196]. In their study, Kos *et al.* [109] compare the effectiveness of adversarial examples with random noise. They show how the value function can indicate opportune moments to inject perturbations and how adversarial re-training can enhance the resilience of RL agents [30]. Lin *et al.* [110] introduce two novel methods to attack DRL agents using adversarial examples. These are referred to as the *strategically-timed attack*, which aims to introduce perturbations at critical moments, and the *enchanting attack*, which aims to lure an agent to a certain state maliciously. Using these methods, the authors demonstrate they are able to significantly decrease the accumulated rewards collected by a DQN and an Asynchronous Advantage Actor-Critic (A3C) agent on five different Atari games.

These previous studies demonstrate that DRL agents are vulnerable to well-crafted adversarial examples. They propose different methods for attacking DRL agents before and after training, as well as, in white-box and black-box settings. It has even been suggested to remediate the effect of adversarial examples against DRL agents using adversarial re-training [109]. While there is a rich body of work studying how adversarial examples can degrade the performance of DRL models, these previous works investigate attacks against agents used in control problems, particularly the playing of Atari video games. Such models are significantly different from the agent presented in this paper, as we will develop later in Section A, since the successive states are independent of the action taken in the previous step, thus affecting the learning process. In addition, evading an intrusion detection model involves targeting a specific class (labeling malicious connections as normal behavior); while working most of the time with imbalanced datasets [197]. For these reasons, we notice a gap in the literature concerning the understanding of adversarial attacks against DRL-based intrusion

detection agents and present this work as an initial building block toward filling this gap.

Methodology

First, we present the dataset that we use for the training and the validation of the detection agent. Then, we describe the DRL detection agent used in our experiments, as proposed by Caminero *et al.* [11]. Finally, we outline the adversarial attacks we use against the agent.

Dataset

For comparative studies of our results, we opt for the commonly used NSL-KDD dataset [78]. This dataset is widely used in similar research papers, and particularly in Caminero *et al.* [11] to validate the agent.

Each record is composed of 41 network features: 38 continuous (such as the duration of the connection) and 3 categorical. A record is labeled as either normal or an attack. There are 22 different attack types in the training set (therefore, 23 different label outcomes) but 38 in the testing set: an efficient detection model will have to detect anomalies it has not encountered during training. From this basis, a few preprocessing steps were applied: categorical features were one-hot encoded, and non-binary features were normalized (i.e. zero mean, standard deviation equal to one).

Finally, in this work, we aim to mislead the model into classifying an attack record as a normal one (i.e. a false negative classification). Therefore, we do not need to be able to differentiate the 23 anomaly types. Instead, we group them into 4 classes of attacks. The approximately 120,000 samples are thus distributed into the following classes: Normal (53.46%), Denial-of-Service (DoS) (36.46%), Probing (PROBE) (9.25%), Remote-to-Local (R2L) (0.79%), and User-to-Root (U2R) (0.04%).

Detection model

We work with a state-of-the-art DRL intrusion detection agent that has been shown to outperform other DNN and DRL methods on the KDD-NSL dataset [11]. The model is referred to as Adversarial Environment using Reinforcement Learning (AE-RL); since it enhances its learning phase by using an adversarial environment to select training samples. It is composed of two concurrent agents: the first agent is the classifier that predicts the labels for each sample, whereas the second agent is a selector that acts as a simulated environment and feeds sample records to the classifier. Therefore, the second agent is only used during

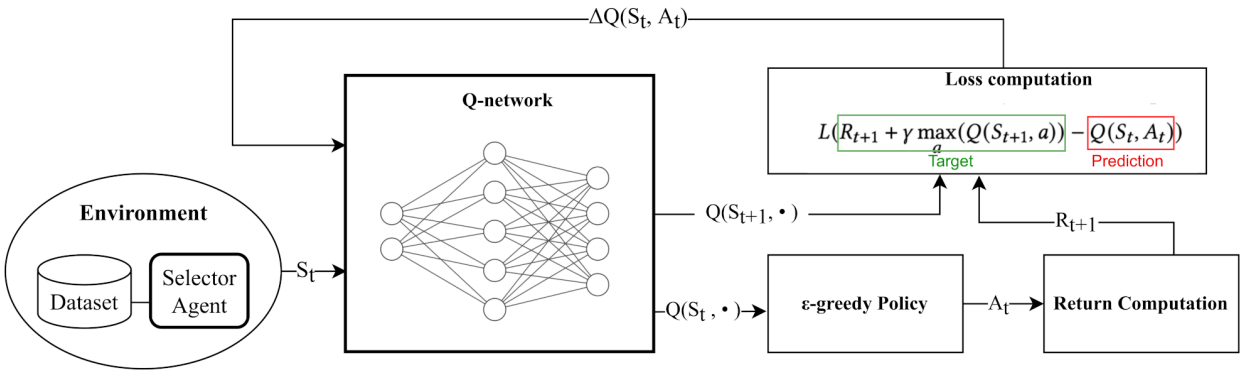


Figure A.1 Details of the classifier DQN agent for the training phase

training to obtain a more robust model and is not involved in the attack detection.

The classifier is a Deep Q-Network (DQN) agent [198], described in Figure A.1. With the states (record features) as input, its goal is to choose the best action according to its Q-function [6]. The Q-function is simulated by a fully-connected, 3-layer neural network with 100 units per layer, that is trained to approximate the optimal Q-function defined by the Bellman equation in Equation A.1:

$$Q^*(S_t, A_t) = R_{t+1} + \gamma \max_a(Q(S_{t+1}, a)), \tag{A.1}$$

where Q^* is the optimal Q-function, S, A, and R are the states, actions, and rewards, γ is the discount factor, and t and $t+1$ are the timesteps.

During training, the error between the target and the predicted Q-value is back-propagated through the model's parameters. It is calculated with Huber loss, which is quadratic if the absolute difference falls below 1 and linear otherwise. This loss function provides smoothness near zero while being less sensitive to outliers than the squared error loss. After predicting the state's Q-value, the agent chooses the action according to an ϵ-greedy policy [6]: randomly with a probability of ϵ, else the one that maximizes the Q-value. The ϵ value is high at the beginning and reduces over the course of the training process. When the training is done, ϵ is set to zero in order to optimize the prediction.

In this setup, the states are data records issued from the dataset and the actions are the different possible label outputs. The reward is set to 1 if the classifier is correct and 0 otherwise. Finally, the discount factor γ is set to a value close to zero, since the states do not influence one another and each state is independent of the precedent.

When transitioning from one step to the next, selecting random samples from the dataset

is not the most efficient solution due to the unbalanced nature of the dataset. The selector agent instead chooses which anomaly category to pull the next state and attempts to find the most difficult records for the classifier. The selector's algorithm is DQN with Huber loss and epsilon-greedy policy, similar to the first agent, but the rewards are opposite. That is, -1 if the classifier chooses correctly and 0 otherwise. The two agents are considered concurrent because of this method of providing rewards.

Once the training is complete, the prediction phase only consists of passing a record through a small fully connected neural network and choosing the maximum output. This simple architecture allows for efficient classification, which is critical in intrusion detection tasks.

Adversarial attacks

When the training is complete, we use adversarial examples to mislead the agent on test data. A perturbation is computed using the following generation methods and added to the original test data. This perturbation will corrupt the prediction of the Q-values and influence the decision of the agent. These attacks were implemented using the Adversarial Robustness Toolbox (ART) library [209].

Fast Gradient Sign Method

The first attack we use is the Fast Gradient Sign Method (FGSM) introduced by Goodfellow *et al.* [30]. This method exploits the gradient of the loss function, which usually serves to update the parameters of the model. Instead, the gradient is propagated back to the inputs and its sign guides the perturbation. An adversarial example x' is formed by adding the perturbation amplitude ϵ with the sign of the gradient to an original example x. Equation A.2 describes this perturbation, where ∇ is the gradient function, J_θ is the loss function with regards to the parameters θ, and l is the true label of the example.

$$x' = x + \epsilon \cdot sign(\nabla J_\theta(x, l)) \tag{A.2}$$

Targeted Fast Gradient Sign Method

FGSM is an untargeted attack by definition, as it does not aim to misclassify the adversarial example towards a specific class. However, it would not be in the interest of attackers to misclassify an attack as another type of attack since evading detection implies classifying the attacks as normal traffic. To targeted a specific class using FGSM, we perform the update

in Equation A.3 where l' is the target class.

$$x' = x - \epsilon \cdot sign(\nabla J_\theta(x, l')) \tag{A.3}$$

Basic Iterative Method

Kurakin *et al.* [213] introduce the Basic Iterative Method (BIM) as an extension of FGSM. The idea is to apply small perturbations over several steps to create more precise adversarial examples. Additionally, a clipping method is used at each step to prevent features from exceeding valid intervals. Generally, increasing the number of iterations will produce finer perturbations and can lead to more subtle adversarial examples. However, there is a trade-off, as computing these small steps is typically slower to produce adversarial examples than non-iterative methods.

Targeted Basic Iterative Method

Applying BIM involves using FGSM, as outlined in Equation A.2, to generate an adversarial example for some unspecified class and may not necessarily serve the goal of the attacker. Using the BIM process with targeted FGSM, as outlined in Equation A.3, produces an adversarial example for a particular class.

Results

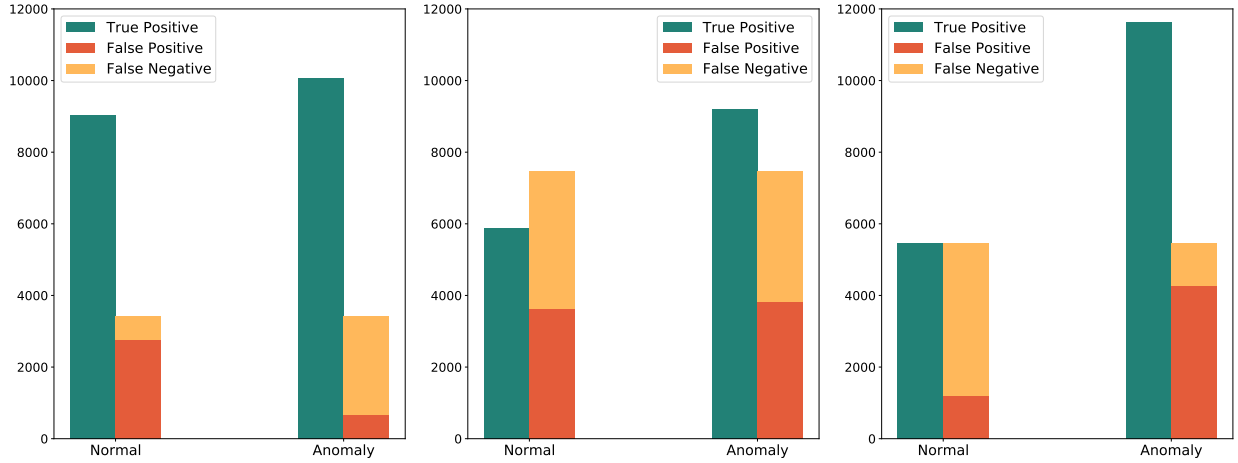
In this section, we present the results of our experiments. We evaluate the performance of the trained agent using the test set, of approximately 30,000 samples, with and without adversarial perturbation. In all adversarial attacks, we set the maximum amount of perturbation to $\epsilon = 0.1$.

Two-class attack detection facing adversarial examples

This section involves experiments using two-class detection, where the detection agent assigns a label of `Normal` or `Anomaly` to each sample. We only consider the generic FGSM and BIM attacks; since the attacker intends to make anomalous packets appear legitimate. The performance of the detection agent is shown in Figure A.2, the accuracy and F1-scores are shown in Table A.1, and the confusion matrices of the detection agent for the label decisions are shown in Figure A.3.

	No Attack		FGSM		BIM	
Label	**Acc.**	**F1**	**Acc.**	**F1**	**Acc.**	**F1**
Normal	84.81	84.09	66.87	61.16	75.84	66.71
Anomaly	84.81	85.47	66.87	71.12	75.84	81.04

Table A.1 Accuracy and F1-score of AE-RL two-class detection model facing adversarial attacks



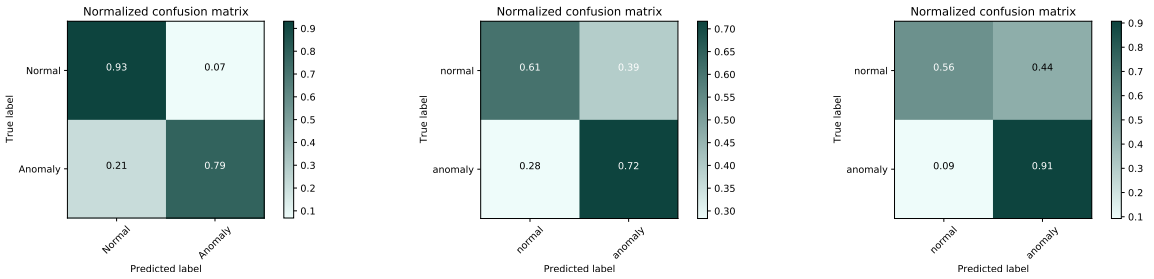
(a) No Attack (b) FGSM (c) BIM

Figure A.2 Performance of AE-RL two-class detection model facing adversarial attacks

No attack In this case, the training set is balanced (i.e. 53% normal and 47% anomalies), which allows the agent to learn the two classes accurately. In Figure A.3, we can see that 79% of the anomalies are detected by this model, with 84.81% accuracy and 84.09% F1-score. These results correspond to state-of-the-art performance on NSL-KDD, even though Figure A.2 shows that about 2800 anomalies are classified as normal traffic.

Fast Gradient Sign Method We observe, in Figure A.3, a significant drop in the number of true positives from both classes, but what is particularly interesting is the false positive rate from the `Normal` class. Indeed, it rose from 0.21 in the baseline model to 0.28 with the FGSM examples, which means that this attack increased the number of suspicious connections undetected by the model.

Basic Iterative Method BIM is supposed to generate more precise perturbations, finding new paths to escape the prediction. In Table A.1, we notice a drop in the performance of the model compared to the baseline; the accuracy drops from 84.81% to 75.84%, and the F1-score



(a) No Attack (b) FGSM (c) BIM

Figure A.3 Confusion matrices of AE-RL two-class detection model facing adversarial attacks

also drops from 84.09% to 66.71% for the anomaly class. However, we notice in Figure A.2 that most of the misclassified items were initially labeled as `Normal`, and the model was lured into labeling them as `Anomaly`. A targeted attack, in a multi-class context, can improve the deception by aiming toward the `Normal` class for all examples.

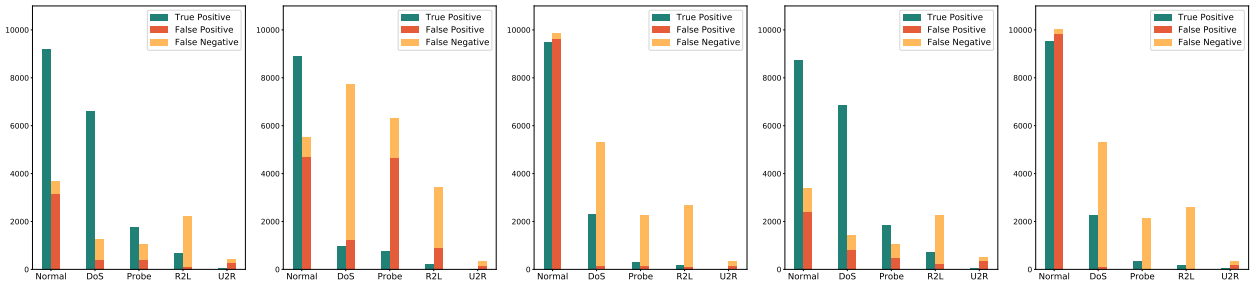
Multi-class attack detection facing adversarial examples

This section involves multi-class detection, where the agent must choose a label of `Normal` or one of the attack categories of `DoS`, `PROBE`, `R2L`, and `U2R`. The performance of the detection agent is shown in Figure A.4, the accuracy and F1-scores are shown in Table A.2, and the confusion matrices of the detection agent are shown in Figure A.5.

	No Attack		**FGSM**		**Targ. FGSM**		**BIM**		**Targ. BIM**	
Label	**Acc.**	**F1**	**Acc.**	**F1**	**Acc.**	**F1**	**Acc.**	**F1**	**Acc.**	**F1**
Normal	83.70	83.33	75.58	76.40	56.37	65.89	84.90	83.71	28.47	2.52
DoS	94.44	91.32	65.73	120.04	76.36	46.23	93.76	90.71	24.81	6.38
PROBE	95.40	77.36	72.09	119.68	89.93	21.83	95.41	78.27	79.62	13.87
R2L	90.25	37.47	84.68	111.25	88.16	12.17	89.91	39.03	84.19	18.93
U2R	98.10	15.44	98.55	112.83	98.52	15.73	97.76	13.99	97.09	8.13

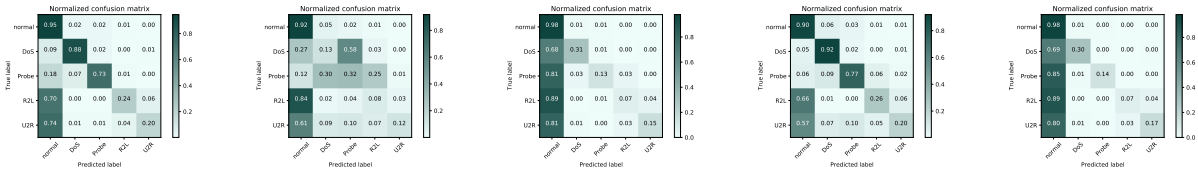
Table A.2 Accuracy and F1-score of AE-RL mutli-class detection model facing adversarial attacks

No attack Without the presence of adversarial examples, we see that the agent has an overall good performance with an accuracy of 83.70%. The F1-scores for the `Normal` and `DoS` categories are 83.33% and 91.32% respectively. However, the agent shows weak performance on `R2L` and `U2R` attacks, where the F1-score is below 40%. This is common in many classifiers because these attack types are hard to detect and are underrepresented in the dataset.



(a) No Attack (b) Untargeted FGSM (c) Targeted FGSM (d) Untargeted BIM (e) Targeted BIM

Figure A.4 Performance of AE-RL multi-class detection model facing adversarial attacks



(a) No Attack (b) Untargeted FGSM (c) Targeted FGSM (d) Untargeted BIM (e) Targeted BIM

Figure A.5 Confusion matrices of AE-RL detection model facing adversarial attacks

Figure A.5 shows a noticeable intensity on the diagonal, especially for the `Normal`, `DoS`, and `Probe` categories. The `R2L` examples are often classified as `Normal` while the `U2R` are spread across different categories.

Untargeted Fast Gradient Sign Method Applying adversarial perturbations using FGSM shows an important drop in the performance of the agent. We see a large number of misclassifications for all classes in Figure A.4. The accuracy of the agent drops to 75.58%, while the F1-Score of all classes is substantially lower. The false positive rate of the class `Normal` shows that most of the examples are misclassified in this category. The same results are shown in Figure A.5, as the diagonal is less intense and the `Normal` column (corresponding to the examples classified as `Normal`) is more intense.

Targeted Fast Gradient Sign Method With targeted FGSM, adversarial examples are pushed toward the `Normal` target class. We see a noticeable impact in Figure A.4 with an even higher number of false positives in the `Normal` class. The accuracy and F1-score for the `Normal` label drop to 56.37% and 65.89% respectively. The confusion matrix in Figure A.5 shows a very intense concentration in the `Normal` column. This indicates that targeted attacks can be more interesting for attackers who want to evade an RL-based IDS.

Untargeted Basic Iterative Method By applying the untargeted BIM method, we find no real change to the performance of the detection agent compared to when no attack is present. This can be explained by the perturbation steps limit (set to 100). With no target class for this attack, the perturbations do not go far enough in a particular direction to modify the class of the sample from the viewpoint of the detection agent. Without any computation limitations, we would expect this method to cause a more severe impact on the detection performance.

Targeted Basic Iterative Method With targeted BIM, we see the most drastic degradation in the performance of the detection agent. This method can perturb more precisely anomalous samples to the `Normal` class. The accuracy on `Normal` samples drops to 28.47% and the F1-score for all labels drops below 20%. The substantial amount of misclassifications of anomalous packets as `Normal` is demonstrated in Figures A.4 and A.5.

Discussion and future work

The results of our experiment show how adversarial examples can degrade the performance of DRL IDSs. However, more work needs to be done to prove the vulnerability of DRL IDSs to adversarial examples. In reality, an attacker would likely need to adversarially modify a stream of malicious packets to evade the IDS. Our experiments are limited to the data instance level and would need to be implemented on a real network. This raises concerns regarding the practicality of these adversarial examples. Recently study [89] has identified several invalidation properties in adversarial examples generated on 3 intrusion detection datasets that prevent their implementation. These properties include out-of-range values, corrupt binary values, multiple categories belonging, and corrupt semantic relations. For the purpose of this work, we demonstrate that current adversarial attacks can bypass DRL detection agents at the data level, but we do not solve the practicality issues at the network level.

An avenue for future work is to investigate the impact of other adversarial attacks on DRL IDSs, including in the more realistic black-box setting. The detection agents should also be trained on recent datasets that are more representative of modern network traffic and attacks. The practicality concerns should be addressed by applying clipping functions and penalties to restrict the perturbation. Semantic relations should be extracted from the data and integrated into the generation methods to produce consistent adversarial examples.

Conclusion

Recent research in cybersecurity has used DRL in multiple functions, especially intrusion detection. This approach is promising as it allows more adaptability and faster processing. However, using DRL detection methods opens the door to the threat of adversarial attacks.

In this work, we study the vulnerability of a DRL IDS detection agent when faced with adversarial examples. We train a state-of-the-art DRL detection agent using the NSL-KDD dataset and evaluate its performance with several adversarial attack methods. We demonstrate a substantial deterioration in detection performance when adversarial attacks are used to perturb malicious packets towards being classified as benign. Finally, we discuss the practicality of these adversarial examples and suggest research directions to implement adversarial attacks on real networks.

APPENDIX B IOV CASE STUDY (CICIOV2024)

In this section, we apply our method to the CICIoV2024 dataset. In contrast with the two previous datasets, NSL-KDD and UNSW-NB15, which contain data from TCP/IP networks, CICIoV2024 consists of IoV data captured from real car components. Here we demonstrate the applicability of our method to a more recent dataset and a different type of data. More precisely, we focus on the impact of the DRL algorithm used by the detection agent on the robustness. We use all 5 algorithms described in Section 4.3.1 with 4 hidden layer sizes (1, 2, 4, 8) and 4 hidden unit sizes (32, 64, 128, 256), with a total of 80 combinations.

Figure B.2 shows a line plot of the FNR for each network width and depth combination. The lines correspond to different DRL algorithms (FGSM and BIM are represented with continuous and dashed lines, respectively). On the other hand, Figure B.1 shows the FNR mean and standard deviation averaged together over all neural network architectures, it is the summarized version of Figure B.2.

First, we observe the performance of DRL agents before adversarial perturbations ($\epsilon = 0$). Figure B.2 shows an FNR equal to 0 for almost all DRL algorithms on all architectures. Our results also show an F1 score equal to 1, demonstrating a perfect classification. These results are due to the relatively simple structure of the CICIoV2024 dataset compared to network datasets, the former consists of 8 values each encoded in one byte and an ID. IoV components are designed to be lightweight and energy-efficient, limiting the size of data communication to the minimum.

Similarly to Figure 4.9, Figure B.1 shows some variance because the values are averaged over 16 different architectures. We can nonetheless observe the behavioral patterns of DRL algorithms on CICIoV2024. First, we notice that the properties of the datasets disadvantage A2C which ends up with the highest average FNR, reaching an average around 0.8 at $\epsilon = 0.1$. Similarly to previous datasets, DQN, QRDQN, and PPO show comparable performance in terms of robustness. Their FNR evolves similarly and reaches 0.5 to 0.6 when $\epsilon = 0.1$. Finally, TRPO maintains the best robustness to adversarial examples, especially against FGSM, with a low FNR below $\epsilon = 0.6$ and an average FNR of 0.37.

Furthermore, the detailed Figure B.2 shows the specific behavior of the algorithms with different neural network architectures. It shows the relative stability of A2C across architectures, with an FNR increasing starting from around $\epsilon = 0.04$. PPO and TRPO show a similar behavior overall, in addition to a decrease in FNR with larger neural networks (especially 8

hidden layers). Moreover, DQN and QRDQN show a higher sensitivity to adversarial examples on smaller neural networks (up to 4 hidden layers and 64 units), with high FNR values starting from $\epsilon = 0.02$. The two algorithms are also the only ones that reach an FNR equal to 1 on some architectures. Finally, Regardless of the DRL algorithm used, agents with the largest neural network (8 hidden layers of 256 units) achieved the best robustness, with an FNR below 0.4 for all algorithms (except QRDQN against BIM).

In conclusion, the performance of DRL-based intrusion detection is not limited to the network domain, it extends to other critical tasks, and so do their vulnerabilities. In this section, we have shown the efficiency of DRL detection agents in detecting attacks against IoV devices using the CICIoV2024 dataset. Across different architectures, our agents present a perfect detection across almost all DRL algorithms and neural network architectures. These performances are explained by the relative simplicity and lightheartedness of IoV data. However, DRL agents also present similar vulnerabilities to adversarial attacks in IoV. We have shown how gradient-based adversarial examples with a perturbation amplitude $\epsilon \leq 0.1$ could considerably increase the FNR (attacks labeled as benign). This case study on IoV data demonstrates the seriousness of the adversarial examples threat in critical infrastructures and the need for further investigation on the robustness of DRL agents in IoV environments.

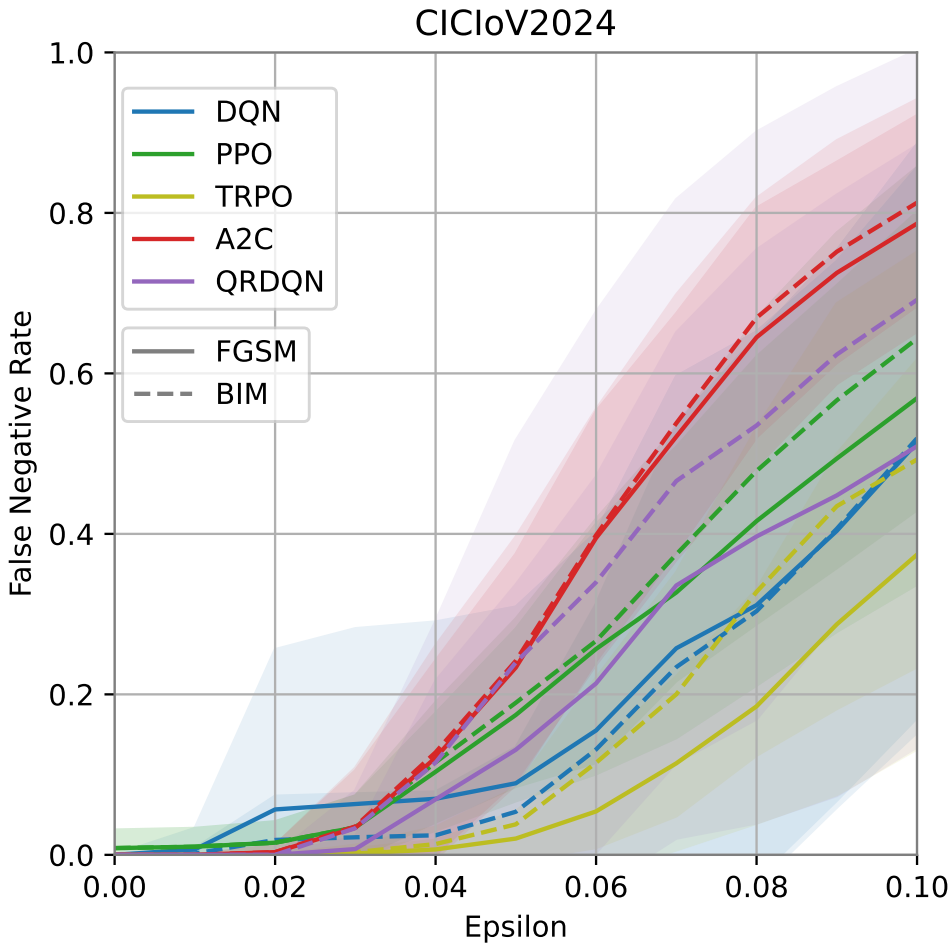


Figure B.1 FNR across DRL algorithms over all architectures on CICIoV2024

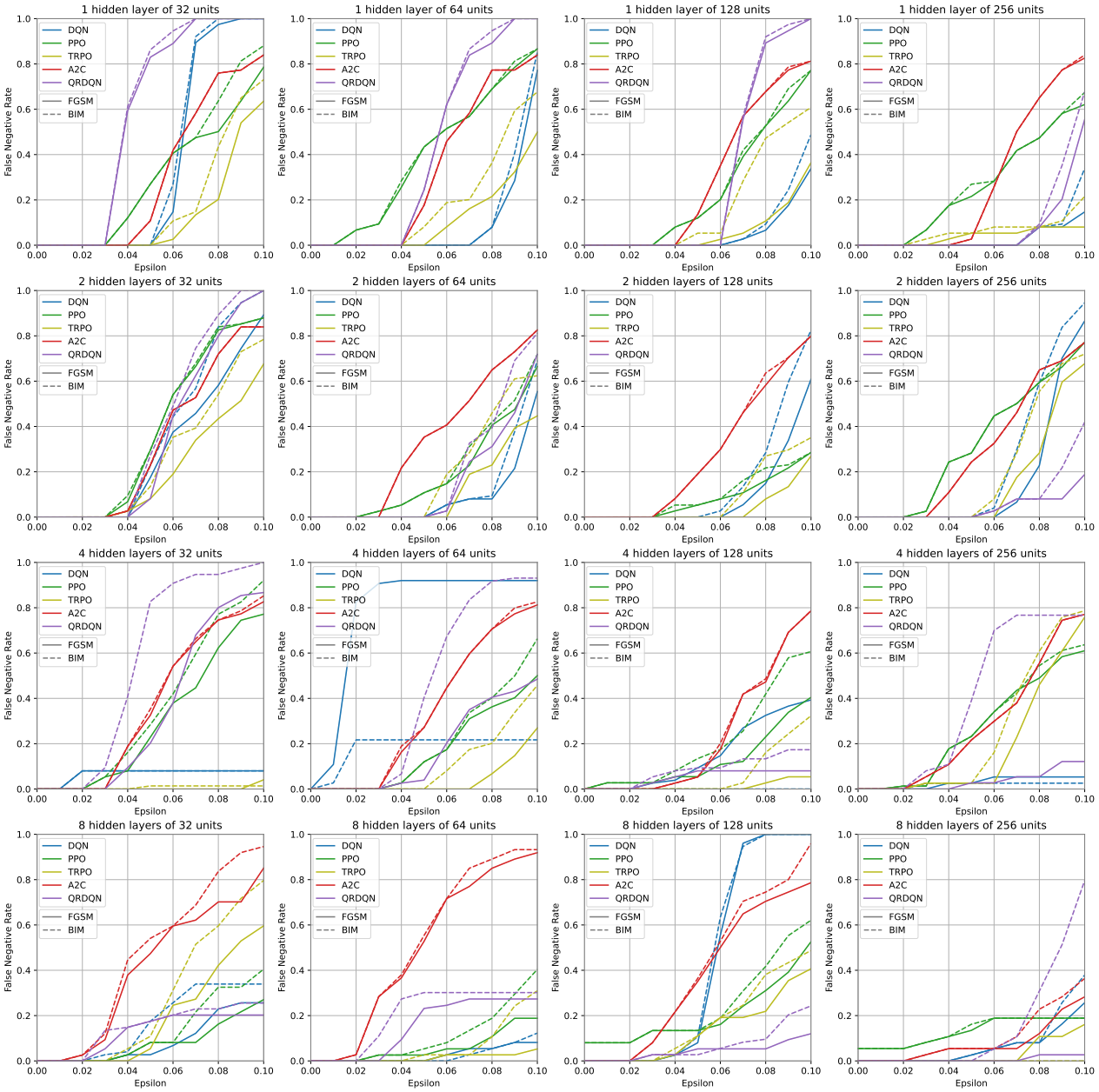
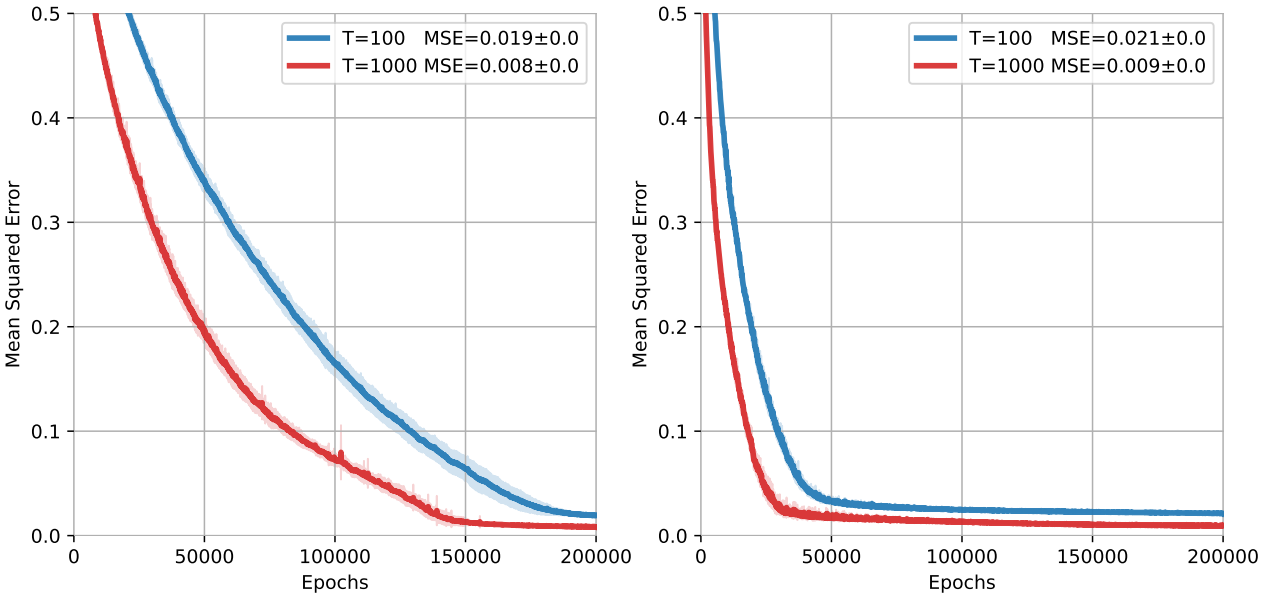


Figure B.2 FNR across DRL algorithms on CICIoV2024

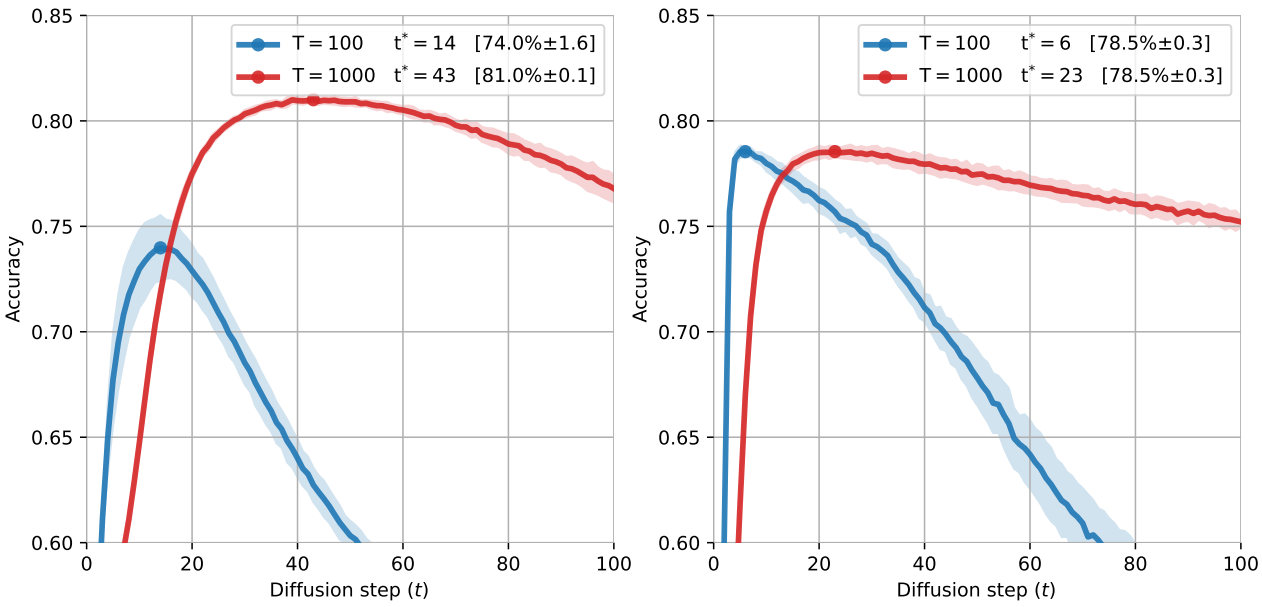
APPENDIX C MORE ON THE NUMBER OF DIFFUSION STEPS T

Figure C.1 shows the reconstruction loss over the training epochs. Both diffusion models are trained with $\beta_1 = 10^{-4}$, $\beta_T = 10^{-2}$, and a diffusion neural network with 10 hidden layers of 960 ReLU units. We observe that the reconstruction loss for $T = 1000$ is always lower than for $T = 100$. Also, the loss curve for $T = 1000$ stabilizes earlier (around $150,000$ epochs as opposed to $190,000$ epochs when $T = 100$ on UNSW-NB15). Despite both models having the same β_1 and β_T, a larger T divides the range into smaller steps, allowing for a more gradual noising. As the added noise increases more slowly, the diffusion neural network reconstructs better x_{t-1} at each step.



(a) UNSW-NB15 (b) NSL-KDD

Figure C.1 Reconstruction loss over training epochs for $T = 100$ and $T = 1000$



(a) UNSW-NB15 (b) NSL-KDD

Figure C.2 Accuracy over the diffusion step t for different number of diffusion steps t using the optimal variance interval recorded in Figure 6.6: $\beta_1 = \beta_T = 10^{-4}$