



Titre: Étude de la vulnérabilité des systèmes de détection d'intrusion par apprentissage par renforcement face aux comportements adverses
Title:

Auteur: Joséphine Delas
Author:

Date: 2023

Type: Mémoire ou thèse / Dissertation or Thesis

Référence: Delas, J. (2023). Étude de la vulnérabilité des systèmes de détection d'intrusion par apprentissage par renforcement face aux comportements adverses [Master's thesis, Polytechnique Montréal]. PolyPublie. <https://publications.polymtl.ca/55720/>
Citation:

 **Document en libre accès dans PolyPublie**
Open Access document in PolyPublie

URL de PolyPublie: <https://publications.polymtl.ca/55720/>
PolyPublie URL:

Directeurs de recherche: Nora Boulahia Cuppens, & Frédéric Cuppens
Advisors:

Programme: Génie informatique
Program:

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

**Étude de la vulnérabilité des systèmes de détection d'intrusion par
apprentissage par renforcement face aux comportements adverses**

JOSÉPHINE DELAS

Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Août 2023

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé :

**Étude de la vulnérabilité des systèmes de détection d'intrusion par
apprentissage par renforcement face aux comportements adverses**

présenté par **Joséphine DELAS**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Omar Abdul WAHAB, président

Nora BOULAHIA CUPPENS, membre et directrice de recherche

Frédéric CUPPENS, membre et codirecteur de recherche

Quentin CAPPART, membre

DÉDICACE

Aux Michons Ext., merci pour ces deux ans ensemble.

REMERCIEMENTS

Je tiens tout d’abord à exprimer ma gratitude envers ma directrice de mémoire Pr. Nora Cuppens, ainsi que mon co-directeur Pr. Frédéric Cuppens, pour leur soutien, leurs conseils avisés et leur disponibilité tout au long de ce projet. Leurs connaissances approfondies et leur expertise ont grandement contribué à la réussite de ce travail.

Je tiens également à remercier l’IRT SystemX, qui a subventionné cette recherche et a rendu ce travail possible. Leur engagement en faveur de la recherche et de l’innovation a grandement contribué à la réalisation de ce mémoire. Merci à D. Reda Yaich ainsi qu’à toute l’équipe cyber pour votre accueil dans les bureaux de Palaiseau, je suis honorée d’avoir pu bénéficier de votre soutien et de votre expérience dans le cadre de ce projet de recherche.

Je remercie chaleureusement le D. Christopher Neal pour ses précieux conseils et son soutien tout au long de ma formation. Sa bienveillance et son enthousiasme ont été une grande source d’inspiration et de motivation.

Je souhaite de plus adresser mes remerciements à tous les membres du jury de soutenance de ce mémoire, le Pr. Omar Abdul Wahab, président du jury, et le Pr. Quentin Cappart, pour leur temps, leur attention et leurs commentaires constructifs lors de la soutenance.

Enfin, un grand merci à Camille, Clara, Gabriel et François qui ont été présents tout au long de ces deux années et ont représenté pour moi une deuxième famille à Montréal. Leurs encouragements et leur soutien moral ont été d’une grande importance.

Ce mémoire représente une étape importante de mon parcours académique, et je suis reconnaissante envers toutes les personnes qui ont contribué à sa réalisation. Votre soutien et votre confiance ont été déterminants, et je suis convaincue que cette expérience me sera bénéfique dans mes projets futurs. Merci à tous.

RÉSUMÉ

Ce mémoire explore l'utilisation de l'Apprentissage par Renforcement Profond (DRL) pour la détection d'intrusion, et analyse la vulnérabilité de ces agents face aux attaques par exemples adverses. À mesure que les réseaux informatiques se complexifient, la sécurité des infrastructures critiques repose sur la performance des Systèmes de Détection d'Intrusion (SDI). Le DRL offre une approche prometteuse pour s'adapter à ces environnements interconnectés.

Dans un premier temps, nous développons donc un nouvel agent DRL spécifiquement conçu pour la détection d'intrusion dans les réseaux informatiques. Son efficacité est évaluée sur divers jeux de données, tels que SWaT, AWID et NSL-KDD, pour déterminer sa pertinence dans la détection des menaces cybernétiques.

Par la suite, nous examinons la vulnérabilité de ces agents face à différentes attaques adverses, notamment *Fast Gradient Sign Method* (FGSM) et *Projected Gradient Descent* (PGD). Les résultats révèlent que nos agents SDI sont sensibles aux perturbations les plus subtiles, soulignant ainsi la nécessité de renforcer leur robustesse par des mécanismes de défense adéquats.

Nous étudions ensuite des méthodes permettant d'accroître la robustesse des agents face aux attaques adverses. Nous analysons l'impact des hyperparamètres et de l'architecture du réseau sur les performances et la résistance des modèles, et nous évaluons la pertinence de l'entraînement adverse pour renforcer leur résistance aux attaques.

Cette recherche permet de mieux comprendre les forces et les limites du DRL dans la détection d'intrusion et souligne l'importance d'adopter des stratégies de défense robustes. En abordant la vulnérabilité des agents SDI par RL, ce travail jette les bases de la sécurisation des infrastructures critiques face aux menaces cybernétiques modernes.

ABSTRACT

The increasing complexity and connectivity of communication networks have exposed them to a wide range of cyber threats, necessitating efficient Intrusion Detection Systems (IDS) for safeguarding critical infrastructure. Deep Reinforcement Learning (DRL) has emerged as a promising approach for IDS, leveraging its adaptability and ability to handle complex environments. However, the vulnerability of deep neural networks to adversarial attacks raises concerns about the reliability of IDS agents in the face of sophisticated adversaries.

This thesis investigates the application of DRL for intrusion detection and explores its vulnerability to adversarial attacks. The research begins by constructing a novel DRL-IDS agent, tailored to the characteristics of the studied networks. We evaluate the agent’s performance on diverse datasets, including SWaT, AWID, and NSL-KDD, to establish its effectiveness in detecting cyber threats.

The investigation then delves into the vulnerability of the DRL-IDS agent by applying adversarial attacks such as Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD). The results expose the agent’s susceptibility to adversarial examples and underscore the need for defense mechanisms to enhance its robustness.

The subsequent section explores the methods to improve the agent’s robustness against adversarial attacks. We analyze the impact of hyperparameters and network architecture on the agent’s performance and resilience. Additionally, we examine the potential of adversarial training to enhance the agent’s robustness to attacks.

The findings of this research contribute to a better understanding of the strengths and limitations of DRL for intrusion detection and highlight the significance of adopting robust defense strategies. By addressing the vulnerability of DRL-IDS agents, this work lays the groundwork for securing critical infrastructure in the face of emerging cyber threats.

TABLE DES MATIÈRES

DÉDICACE	iii
REMERCIEMENTS	iv
RÉSUMÉ	v
ABSTRACT	vi
TABLE DES MATIÈRES	vii
LISTE DES TABLEAUX	x
LISTE DES FIGURES	xi
LISTE DES SIGLES ET ABRÉVIATIONS	xiii
CHAPITRE 1 INTRODUCTION	1
1.1 Contexte et Motivation	1
1.2 Objectifs de la maîtrise	3
1.3 Organisation du mémoire	4
CHAPITRE 2 REVUE DE LITTÉRATURE	5
2.1 Détection d’Intrusion et Apprentissage par Renforcement	5
2.1.1 Introduction à la détection d’intrusion	5
2.1.2 Introduction à l’Apprentissage par Renforcement	10
2.1.3 Modèles d’apprentissage par renforcement pour la détection d’intrusion	14
2.2 Comportement Adverse et DRL	15
2.2.1 Fiabilité des DNNs : existence des exemples adverses	16
2.2.2 Taxonomie	18
2.2.3 Méthodes de génération	21
2.3 Défenses contre les attaques adverses	24
2.3.1 Protéger les SDI face aux exemples adverses	25
2.3.2 Robustesse du DRL	26
2.4 Conclusion	27
CHAPITRE 3 MÉTHODOLOGIE	28

3.1	Données utilisées	28
3.1.1	Jeux de données	28
3.1.2	Pré-traitements	31
3.2	Implémentation	33
3.2.1	Algorithmes	33
3.2.2	Bibliothèques et outils	36
3.3	Métriques d'évaluation	37
3.3.1	Performance des modèles	38
3.3.2	Évaluation de la robustesse	39
CHAPITRE 4 VULÉRABILITÉ DES SYSTÈMES DE DÉTECTION D'INTRUSION		
	PAR RL FACE AUX EXEMPLES ADVERSES	40
4.1	Construction du modèle	41
4.1.1	Un agent RL pour la classification	41
4.1.2	Environnement adverse	45
4.1.3	Hyperparamètres	47
4.2	Extension aux données cyber-physiques	50
4.2.1	Particularité des données	51
4.2.2	Adaptation du modèle	51
4.2.3	Pistes d'amélioration	53
4.3	Vulnérabilité des modèles	55
4.3.1	Implémentation des attaques	56
4.3.2	Résultats	57
4.4	Conclusion	58
CHAPITRE 5 PISTES DE DÉFENSE DES SDI PAR RL CONTRE LES ATTAQUES		
	PAR EXEMPLES ADVERSES	60
5.1	Cadre mathématique	61
5.1.1	Cas général	61
5.1.2	Adaptation aux agents RL détecteurs d'intrusions	62
5.2	Apprentissage robuste : optimiser les hyperparamètres pour une architecture plus résiliente	62
5.2.1	Approche expérimentale	63
5.2.2	Résultats	64
5.2.3	Conclusion	70
5.3	Entraînement adverse : tirer parti des attaques connues	72
5.3.1	Approche expérimentale	72

5.3.2	Résultats	73
5.3.3	Conclusion	76
CHAPITRE 6 CONCLUSION		77
6.1	Synthèse des travaux	77
6.2	Limitations	78
6.3	Améliorations futures	79
RÉFÉRENCES		81

LISTE DES TABLEAUX

Tableau 2.1	Comparaison des techniques de détection d'intrusion	6
Tableau 4.1	Regroupement des performance des différents modèles étudiés sur AWID et NSL-KDD (mode binaire)	49
Tableau 4.2	Performances des meilleurs modèles multi-classe étudiés, et hyperpara- mètres associés	50
Tableau 4.3	Performances des modèles RL appliqués aux données SWaT	53

LISTE DES FIGURES

Figure 2.1	Architecture d'un réseau de neurones profond	13
Figure 2.2	Mécanisme d'entraînement d'un agent DRL pour la détection d'intrusion	15
Figure 2.3	La menace adverse en ML	17
Figure 2.4	Taxonomie des exemples adverses en RL	19
Figure 2.5	<i>Policy Induction Attack</i> (PIA)	22
Figure 2.6	<i>Adversarial Transformation Network</i> (PIA)	23
Figure 2.7	<i>Optimal Attack for RL Policies</i> (OARLP)	24
Figure 2.8	Techniques de défense contre les exemples adverses	26
Figure 3.1	Architecture des processus de SWaT [1]	30
Figure 3.2	Scénario d'attaque n°8 sur SWaT : en rouge, les données falsifiées de FIT401	31
Figure 4.1	Performance des agents PPO et DQN à 2 couches et 64 unités au cours de l'entraînement sur NSL-KDD	42
Figure 4.2	Performance des agents PPO et DQN à deux couches et 64 unités au cours de l'entraînement sur AWID	43
Figure 4.3	Évolution du taux de faux positifs des agents PPO et DQN au cours de l'entraînement sur AWID	44
Figure 4.4	Évolution du score F1 et du TFP des agents PPO et DQN sur NSL-KDD, avec et sans l'environnement adverse, évalué sur l'ensemble de test	46
Figure 4.5	Évolution du score F1 et du TFP des agents PPO et DQN sur AWID, avec et sans l'environnement adverse, évalué sur l'ensemble de test . .	47
Figure 4.6	Valeurs des capteurs du premier processus de SWaT, le 29 décembre 2015 de 10h à 23h59. En orange, une anomalie lié à une attaque simultanée de LIT101 et MV101.	52
Figure 4.7	Attaques véritables (en orange) et détectées (en rouge) sur le premier jour de test de SWaT	54
Figure 4.8	Schéma du fonctionnement d'un réseau convolutif à 1 dimension . . .	55
Figure 4.9	Performances d'un agent DQN à 2 couches, 64 unités face à une perturbation adverse de $\epsilon=0.05$	57
Figure 4.10	Matrices de confusion de l'agent soumis ou non aux perturbations adverses	58

Figure 5.1	Évolution du score F1 et du TFP des agents SDI-RL soumis aux attaques par exemples adverses FGSM et PGD, agrégés et regroupés par nombre de couches cachées, en fonction de la taille maximale de perturbation <i>epsilon</i>	65
Figure 5.2	Évolution du score F1 et du TFP des agents SDI-RL soumis aux attaques par exemples adverses FGSM et PGD, agrégés et regroupés par nombre d'unités par couche, en fonction de la taille maximale de perturbation <i>epsilon</i>	67
Figure 5.3	Évolution du TFP des agents SDI-RL soumis aux attaques par exemples adverses FGSM et PGD, agrégés et regroupés par jeu de données, en fonction de la taille maximale de perturbation <i>epsilon</i>	68
Figure 5.4	Évolution du TFP des agents SDI-RL soumis aux attaques par exemples adverses FGSM et PGD, agrégés et regroupés par mode de prédiction, en fonction de la taille maximale de perturbation <i>epsilon</i>	69
Figure 5.5	Évolution du score F1 et du TFP des agents SDI-RL soumis aux attaques par exemples adverses FGSM et PGD, agrégés et regroupés par algorithme RL utilisé, en fonction de la taille maximale de perturbation <i>epsilon</i>	71
Figure 5.6	Évolution des performances l'agent DQN (2 couches, 64 neurones par couches) en fonction de l'amplitude ϵ des perturbations utilisées pour l'entraînement adverse. En pointillés, le seuil de référence des performances de l'agent avant réentraînement.	73
Figure 5.7	Performance des agents réentraînés face à l'attaque PGD, en fonction de la taille de la perturbation ϵ utilisée pour l'entraînement adverse (bleu), comparée à l'agent original attaqué (rouge) et l'agent original sain (vert).	75
Figure 5.8	Performance des agents réentraînés face à l'attaque DeepFool en fonction de la taille de la perturbation ϵ utilisée pour l'entraînement adverse (bleu), comparée à l'agent original attaqué (rouge) et l'agent original sain (vert).	75

LISTE DES SIGLES ET ABRÉVIATIONS

A2C	<i>Advantage Actor Critic</i>
A3C	<i>Asynchronous Advantage Actor Critic</i>
ART	<i>Adversarial Robustness Toolbox</i>
ATN	<i>Adversarial Transformation Network</i>
AML	Apprentissage Automatique Adverse (<i>Adversarial Machine Learning</i>)
BIM	<i>Basic Iterative Method</i>
CNN	<i>Convolutional Neural Network</i>
CPS	Système Cyber-Physique
DDQN	<i>Double Deep Q-Network</i>
DDPG	<i>Deep Deterministic Policy Gradient</i>
DoS	Déni de Service (<i>Denial of Service</i>)
DNN	Réseau de Neurones Profond (<i>Deep Neural Network</i>)
DQN	<i>Deep Q-Network</i>
DRL	Apprentissage par Renforcement Profond (<i>Deep Reinforcement Learning</i>)
FGSM	<i>Fast Gradient Sign Method</i>
JSMA	<i>Jacobian Saliency Map Attack</i>
MDP	Processus de Décision Markovien (<i>Markov Decision Process</i>)
ML	Apprentissage Automatique (<i>Machine Learning</i>)
NCCE	<i>National Cybersecurity Center of Excellence</i>
OARLP	<i>Optimal Attack for RL Policies</i>
R2L	Attaque par Injection (<i>Remote-to-Local</i>)
RL	Apprentissage par Renforcement (<i>Reinforcement Learning</i>)
SDI	Système de Détection d’Intrusions
SDIA	Système de Détection d’Intrusions par Anomalies
SDIS	Système de Détection d’Intrusions par Signatures
SGD	Descente de Gradient Stochastique (<i>Stochastic Gradient Descent</i>)
PIA	<i>Policy Induction Attack</i>
PGD	<i>Policy Gradient Descent</i>
PLC	<i>Programmable Logic Controller</i>
PPO	<i>Proximal Policy Optimization</i>
SB3	Stable Baselines3
TCP	<i>Transmission Control Protocol</i>
TFN	Taux de Faux Négatifs

TFP	Taux de Faux Positifs
TRPO	<i>Trust Region Policy Optimization</i>
U2R	Attaque par Personnification (<i>User-to-Root</i>)
W&B	<i>Weights & Biases</i>
WLAN	<i>Wireless Local Area Network</i>

CHAPITRE 1 INTRODUCTION

L'utilisation de systèmes connectés à Internet, dans de nombreux secteurs critiques tels que la défense ou la production d'énergie [2] s'accompagne de préoccupations croissantes quant à la sécurité des connexions. L'augmentation sans précédent du nombre cyberattaques lors de ces dernières années, +38% au niveau mondial en 2022 [3], entraîne un investissement massif des entreprises et des organismes publics dans les mesures de sécurité et une véritable course à la protection numérique. Les possibilités que fait miroiter le développement de modèles d'apprentissage automatique ultra-performants cristallisent l'attention de la recherche en la matière, bien que ces méthodes parfois opaques ne soient pas dénuées de limites ni de failles. Cette étude explore les solutions réalistes d'application de l'une d'elles, l'apprentissage par renforcement, à la détection d'intrusions malveillantes dans un système informatique : performances, vulnérabilités et possibilités d'amélioration, peut-on raisonnablement envisager une symbiose sûre et fiable entre apprentissage profond et cybersécurité ?

1.1 Contexte et Motivation

Selon le magazine Forbes, la cybercriminalité représente aujourd'hui 1 à 2% du PIB mondial, soit 1 à 2 000 milliards de dollars [4]. Alors que la plupart des infrastructures modernes reposent sur les technologies de l'information, les conséquences économiques, politiques et sociales d'une attaque informatique sur un système critique peuvent s'avérer désastreuses [5]. Le domaine de la cybersécurité, qui regroupe l'ensemble des techniques et outils visant à protéger les réseaux, les données et les utilisateurs contre les menaces numériques, s'impose donc comme une discipline cruciale et un enjeu majeur du XXI^e siècle.

Les attaques informatiques, de plus en plus sophistiquées, menacent la confidentialité, l'intégrité et la disponibilité des systèmes d'information. Détecter en temps réel les activités suspectes ou malicieuses est une étape clé de la protection des réseaux informatiques : les Systèmes de Détection d'Intrusion (SDI) sont donc essentiels pour permettre aux organisations de repérer rapidement les menaces potentielles, et leurs performances sont constamment mises à l'épreuve pour faire face au développement des cyberattaques.

Les SDI sont des outils utilisés pour détecter les activités suspectes sur un réseau informatique en analysant le trafic réseau, les journaux d'événements ou les fichiers système pour identifier les signes de comportement malveillant. Historiquement, leur décision s'appuie sur une bibliothèque de descriptions des attaques pour reconnaître les activités suspectes [6]. Cependant,

ces méthodes par signature s'avèrent coûteuses ainsi que limitées à certains environnements très spécifiques et aux attaques déjà connues [7].

Le développement récent des modèles d'apprentissage automatique et leur application à la détection d'intrusion ont considérablement amélioré l'efficacité des systèmes de sécurité informatique. L'apprentissage automatique permet aux SDI de détecter les anomalies, et donc les comportements suspects non couverts par un ensemble de règles prédéfinies [8]. La construction d'un système de détection d'intrusions par anomalies (SDIA) repose alors sur deux phases, une première phase d'apprentissage à partir de données historiques étiquetées, puis une seconde phase de prédiction, lors du déploiement de l'outil. Relancer l'apprentissage avec des données plus récentes permet également de tenir à jour le système des évolutions de la menace. Aujourd'hui, les modèles d'apprentissage automatiques les plus couramment utilisés pour la détection d'intrusion sont des méthodes supervisées, efficaces et simples à implémenter [9]. Ces méthodes nécessitent cependant une grande quantité de données d'entraînement, et les modèles engendrés sont difficilement généralisables.

Dans ce contexte, l'apprentissage par renforcement (RL) est une piste prometteuse pour la détection d'intrusion. Le RL est une branche de l'apprentissage automatique qui permet à un agent d'apprendre à prendre des décisions par un processus d'essai et erreur, en interagissant avec son environnement. D'une efficacité prouvée dans de nombreux domaines comme la robotique ou la conduite autonome [10, 11], son application à la détection d'intrusion suscite un intérêt croissant dans la communauté scientifique [12]. La légèreté, l'adaptabilité et la réactivité des agents RL les rendent particulièrement adaptés à l'utilisation en temps-réel nécessaire aux infrastructures critiques.

En pratique, les algorithmes de RL les plus performants pour les cas d'utilisation complexes et réalistes font appel aux réseaux de neurones profonds (DNN) dans leur architecture [13]. Bien que très efficaces, ces méthodes d'apprentissage profond sont connues pour être vulnérables aux attaques par exemples adverses [14] : d'imperceptibles modifications des données d'entrée calculées pour induire en erreur le modèle et échapper à la classification. L'application de méthodes d'apprentissage par renforcement profond (DRL) à un secteur critique comme la sécurité informatique nécessite donc une étude rigoureuse des failles et vulnérabilités induites, afin d'anticiper les risques encourus et d'assurer la fiabilité des systèmes de détection. Plusieurs approches défensives peuvent alors être envisagées pour améliorer la robustesse des modèles aux perturbations pendant leur déploiement [15, 16], tout en s'assurant qu'elles n'impactent pas les performances du modèle initial.

Quelques études ont déjà signalé l'impact que peuvent avoir les exemples adverses sur les SDI supervisés [17, 18]. En revanche, peu de travaux se sont encore intéressés aux modèles de SDI

par RL. Après avoir analysé leur fiabilité face aux attaques par exemples adverses, ce projet propose donc une technique de protection des SDI par RL en explorant les pistes d'application des méthodes adverses pour améliorer la robustesse des modèles. Si les résultats de cette recherche sont un pas vers l'utilisation sécurisée de modèles d'apprentissage profond pour la détection d'intrusion, leurs implications peuvent également s'étendre à d'autres domaines de la cybersécurité impliquant du RL, tels que la reconnaissance vocale, la détection d'anomalies dans les réseaux informatiques et les véhicules autonomes.

1.2 Objectifs de la maîtrise

En tant que plateforme d'entraînement d'agents autonomes, le DRL est utilisé dans de nombreuses tâches de prise de décision en temps réel : cette combinaison de RL et DNN surpasse les performances humaines dans une multitude de domaines, des jeux de stratégie discrets comme les échecs ou le Go [19] à la simulation de courses automobiles [20]. Il faut toutefois garder à l'esprit que les mécanismes impliqués dans l'entraînement et le déploiement des agents DRL sont profondément complexes, ce qui entrave le contrôle exhaustif de leur comportement dans un environnement réel [21]. Dans les cas d'applications sensibles, exploiter ces méthodes de manière sécuritaire implique donc une étude approfondie de l'agent en prenant en compte les zones floues de son comportement et en remédiant aux éventuels angles morts rencontrés pendant l'entraînement.

Ce projet de recherche vise donc à répondre à la question suivante : **comment sécuriser efficacement les systèmes de détection d'intrusion basés sur l'apprentissage par renforcement, en particulier contre la menace des attaques adverses ?**

En réponse, l'objectif principal de ce mémoire est de proposer une possible solution de protection des modèles de détection d'intrusion par RL contre les comportements adverses. Cette solution devrait permettre le maintien de performances satisfaisantes du SDI dans les conditions normales aussi bien que sous l'effet de perturbations malveillantes.

Pour ce faire, plusieurs sous-objectifs peuvent être identifiés :

- Évaluer la menace des comportements adverses sur les SDI utilisant le DRL.
- Étudier les solutions de protection existantes pour améliorer la robustesse des modèles DRL.
- Proposer une méthode de construction d'un SDI par DRL robuste face aux exemples adverses.

Ce projet a été réalisé selon l'approche suivante : nous avons d'abord construit un premier modèle de détection d'intrusion par DRL avec des données réseau, et évalué sa vulnérabilité

face aux attaques par exemples adverses les plus connues. Puis, nous avons élargi notre étude à un jeu de données cyber-physiques, et conclu sur la pertinence de l'utilisation du RL dans ce cas particulier. Nous nous sommes ensuite intéressés au lien entre exemples adverses et RL de manière plus générale, en proposant une taxonomie de classification des différentes méthodes de génération et en se concentrant sur la manière dont elles peuvent être exploitées pour améliorer la robustesse des modèles, plutôt que pour les mettre en péril. Enfin, nous avons appliqué les enseignements tirés de cette étude à notre modèle de détection d'intrusion initial, en implémentant les méthodes choisies sur plusieurs algorithmes de DRL différents avec deux jeux de données couramment utilisés en détection d'intrusion.

1.3 Organisation du mémoire

Après avoir présenté dans ce chapitre les enjeux ce projet ainsi que la problématique et l'approche choisie, le Chapitre 2 est consacré à une revue approfondie de la littérature sur les méthodes de détection d'intrusion basées sur le RL ainsi que sur les attaques adverses en RL et leurs contre-mesures.

Dans le Chapitre 3, nous présentons les données utilisées pour la détection d'intrusion, ainsi que la méthodologie suivie pour construire les modèles proposés : algorithmes, métriques, bibliothèques pour la programmation, cette partie permettra de justifier les choix effectués au cours du projet.

Notre modèle de détection d'intrusion par DRL est défini dans le Chapitre 4, accompagné d'une étude de ses performances sur plusieurs types de données, réseau, sans-fil et cyber-physiques. Plusieurs méthodes adverses sont ensuite appliquées à ce modèle, et leur impact est évalué en fonction de plusieurs paramètres.

Enfin, le Chapitre 5 présente l'approche choisie pour défendre le modèle contre les attaques adverses, et discute des performances ainsi que des limites de cette solution de protection en s'appuyant sur les résultats des simulations numériques.

Nous avons souligné ici l'intérêt de l'utilisation du renforcement pour la détection d'intrusion, tout en mettant en évidence les risques potentiels liés aux attaques adverses. Dans ce contexte, cette recherche vise à contribuer à la conception d'une solution pour la protection des systèmes de détection d'intrusion basés sur le renforcement contre les attaques adverses. La prochaine section de ce mémoire présentera une revue de la littérature existante sur ce sujet.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre présente une revue de la littérature articulée autour des trois concepts majeurs de ce projet : la détection d'intrusion, le RL et les exemples adverses. Après avoir introduit les notions clé de détection d'intrusion et de RL, nous présentons les travaux existants sur l'utilisation du RL pour les SDI ainsi que les avantages et les limitations de cette approche. Nous explorons ensuite les différents types d'attaques adverses en mettant l'accent sur les méthodes de génération de perturbations propres au RL à travers le spectre d'une taxonomie spécifique. Enfin, nous discutons de la vulnérabilité des modèles de RL face à ces perturbations et introduisons une définition mathématique de la robustesse en RL, avant de parcourir les méthodes de défense existantes.

2.1 Détection d'Intrusion et Apprentissage par Renforcement

La détection d'intrusion est un domaine de recherche crucial en cybersécurité, dont l'objectif est de détecter les activités malveillantes dans les systèmes informatiques. Les approches traditionnelles s'avèrent limitées pour détecter les intrusions face à l'évolution rapide des méthodes d'attaques. Le RL est une technique d'apprentissage automatique récemment exploitée en détection d'intrusion qui offre des avantages potentiels pour améliorer l'efficacité et la précision des SDI. Dans cette section, nous définissons les notions fondamentales de détection d'intrusion et de RL explorons la littérature sur l'utilisation du RL pour les SDI, en mettant en évidence les points forts et les limites de cette approche.

2.1.1 Introduction à la détection d'intrusion

Définition et concept

Une intrusion peut être définie comme toute activité non autorisée qui compromette la sécurité d'un système informatique. En d'autres termes, toute attaque susceptible de menacer la confidentialité, l'intégrité ou la disponibilité des informations est considérée comme une intrusion. Un SDI est un logiciel conçu pour détecter les comportements suspects dans un système informatique, afin de signaler ces activités malveillantes [22]. Cela permet d'identifier différents types d'attaques qui ne sont pas mitigées par les pare-feux traditionnels. Alors que certains incidents majeurs comme la fuite massive de données LinkedIn en 2021 [23] démontrent la facilité avec laquelle les cybermenaces peuvent se propager à l'échelle internationale, les SDI sont aujourd'hui devenus un élément essentiel de la sécurité informatique.

Il existe deux grandes catégories de SDI : les Systèmes de Détection d’Intrusion par Signature (SDIS) ou par Anomalies (SDIA). Les SDIS s’appuient sur une base de données de signatures d’attaques connues et précédemment observées : ils signalent les séquences de commandes ou d’actions concordant avec ces signatures. Cette méthode comparative permet d’atteindre une excellente précision de signalement des attaques connues, mais reste impuissante face aux signatures encore jamais observées (attaques *zero-day*). De plus, la création de la base de signature revêt une importance majeure dans la construction d’un SIDS, et nécessite une expertise métier souvent coûteuse [24]. Cette approche devient est donc de plus en plus limitée face à la montée en puissance des attaques *zero-day* et la complexité croissante des réseaux informatiques [9] : la détection par anomalie permet d’adresser certains des défis posés par cette évolution.

Les SDIA, ou systèmes de détection par comportement, suscitent un intérêt croissant dû à leur capacité à surmonter les limites des SDIS. Ces approches utilisent des méthodes statistiques ou d’apprentissage automatique pour analyser le comportement normal d’un système et ainsi repérer les comportements inhabituels, associés alors à une intrusion. Avant la phase de détection, le modèle doit être entraîné à partir de données internes au système (SDI hôte) ou de trafic du réseau (SDI réseau) : à l’inverse du cas des SIDS, ces données d’entraînement ne nécessitent pas d’analyse et ni de transformation minutieuse, bien qu’un pré-traitement puisse améliorer les performances du modèle. L’avantage majeur des SDIA est de pouvoir repérer tout type d’attaque, en incluant les *zero-day*. En contrepartie, ils risquent d’entraîner un taux élevé de faux positifs si les anomalies sont des activités nouvelles mais inoffensives, et non pas de véritables intrusions. Le Tableau 2.1 résume les avantages et inconvénients des deux approches existantes pour la détection d’intrusion.

TABLEAU 2.1 Comparaison des techniques de détection d’intrusion

	SDIS	SDIA
Points forts	+ Précision + Prise de décision rapide	+ Meilleure généralisabilité + Peu d’entretien nécessaire
Limites	- Coûteux à tenir à jour - Limité aux intrusions connues	- Difficile à contrôler - Risque de fausses alertes

Bien qu’elles nécessitent une attention particulière pour minimiser les faux positifs, les approches par comportement sont particulièrement prometteuses, et ce d’autant plus compte tenu développement rapide des méthodes d’apprentissage automatique. Au cours de cette étude, nous nous concentrons donc sur les SDIA, tout en conservant l’appellation SDI pour plus de simplicité.

Si les SDI sont couramment utilisés pour protéger des réseaux ou systèmes informatiques purs,

ils sont aussi employés pour la surveillance de systèmes hybrides, ou cyber-physiques. Un Système Cyber-Physique (*Cyber-Physical System*, CPS) est un système informatique interconnecté qui contrôle et surveille un processus physique. Souvent utilisés dans les infrastructures critiques, les CPS sont devenus une cible de choix pour les pirates informatiques. Depuis l'attaque de Stuxnet contre une installation nucléaire iranienne en 2010 ([5]) la menace n'a fait qu'augmenter, et protéger ces systèmes est aujourd'hui un enjeu crucial pour l'industrie et les gouvernements. Les SDI pour CPS doivent donc être capables de surveiller le trafic réseau, les commandes de capteurs et d'actionneurs, et les comportements des utilisateurs, pour détecter les anomalies et identifier les comportements suspects ou malveillants.

Attaques

Avant de poursuivre sur le sujet de la détection d'intrusion, prenons le temps de détailler les grandes catégories d'attaques informatiques que visent à détecter les SDI. Les cyberattaques sont aussi variées que nombreuses et de nouvelles méthodes apparaissent chaque année, nous ne les détaillerons donc pas ici. Cependant, les intrusions dans un réseau informatique peuvent être divisées en quatre grandes catégories [25] :

- Le Dénier de Service (*Denial of Service*, DoS), ou Inondation (*Flooding*), consiste à bloquer ou restreindre les services fournis par le système aux utilisateurs, mettant en péril la disponibilité des informations.
- L'Extraction (*Probing*) vise plutôt à obtenir de l'information sur le réseau ou le système informatique, et s'attaque donc à la confidentialité.
- La Personnification (*User-to-Root*, U2R) a pour objectif de permettre à un utilisateur non privilégié d'obtenir le statut d'administrateur sur un système ou un ordinateur particulier.
- Enfin, les attaques d'Injection (*Remote-to-Local*, R2L) cherchent à envoyer des paquets possiblement nocifs à la machine victime.

Dans le cas des CPS, outre les attaques réseau qui peuvent atteindre les communications entre les modules informatiques, les systèmes peuvent également être soumis à des attaques physiques entraînant le dysfonctionnement de leurs composantes mécaniques.

La plupart des jeux de données couramment utilisés dans la recherche sur la détection d'intrusion classent les attaques rencontrées au sein de ces grandes catégories.

Jeux de données

Comme nous l’avons mentionné précédemment, la construction d’un SDI se compose d’une phase d’apprentissage pour pouvoir distinguer un comportement normal d’un comportement malveillant. Cet apprentissage nécessite alors un ensemble de données représentatif du réseau ou du système. En outre, il est crucial pour la recherche de pouvoir reproduire, évaluer et comparer les différentes méthodes existantes, d’où l’importance des jeux de données publics pour la détection d’intrusion.

Bien que le manque de jeux de données représentatifs disponibles constitue l’un des plus grands défis de la détection d’intrusion par anomalies [26], il en existe quelques uns de très complets, par conséquent couramment utilisés dans la littérature. Chacun présente naturellement ses caractéristiques uniques dont nous détaillons les plus intéressantes pour notre étude ci-dessous :

- Date : l’année de création du jeu de données doit être prise en compte, car il peut devenir obsolète avec l’évolution rapide des réseaux informatiques.
- Format : les données du réseau, recueillies par paquet [27] ou par flux [28], peuvent être ensuite transformées pour faciliter l’apprentissage du SDI.
- Étiquettes : certains jeux de données sont présentés tels qu’observés, tandis que d’autres sont étiquetés selon la nature du trafic (normal, malveillant, etc.).
- Nature du réseau : qu’il s’agisse d’un réseau interne, connecté, miniature, ou complexe, la nature du réseau impacte directement le champ d’application du SDI.

Dans le cadre de ce projet, nous nous sommes intéressés uniquement aux jeux de données étiquetés et réalistes, en choisissant les plus représentatifs de la littérature qui ne soient pas obsolètes et en variant le format des observations.

Pour étudier la détection d’intrusion sur les réseaux informatiques, nous avons sélectionné deux jeux de données, NSL-KDD et AWID. NSL-KDD [29] est l’un des jeux de données les plus couramment utilisés dans la recherche sur le sujet. Il s’agit d’une version améliorée de KDD CUP 99, un jeu de données construit qui contient 7 semaines de trafic réseau, plus précisément les attributs bas-niveau des connexions TCP (*Transmission Control Protocol*) ainsi que quelques données haut-niveaux comme le nombre d’échecs de connexion. Les auteurs de NSL-KDD ont supprimé les doublons de KDD CUP 99 et ont créé des sous-ensembles plus sophistiqués. Si de nombreux travaux s’appuient encore aujourd’hui sur cet ensemble de données pour évaluer les performances des SDI [30], il convient toutefois de noter que le trafic réseau sous-jacent de NSL-KDD remonte à 1998 : ainsi, les attaques réseau implémentées lors des 7 semaines de simulations correspondent aux normes de l’époque et n’incluent pas les

comportements malveillants récents.

AWID [31] est un jeu de données plus récent (2016) tourné vers les réseaux WiFi 802.11. Ses créateurs ont utilisé un petit réseau sans fil (11 clients) et capturé le trafic sous forme de paquets WLAN (*Wireless Local Area Network*). L'exécution de 16 attaques spécifiques au réseau 802.11 leur a permis de générer le trafic malveillant, en étiquetant le jeu de données par la même occasion. L'utilisation d'AWID en complément de NSL-KDD permet de tester nos modèles sur des données non modifiées, en variant également la nature du réseau étudié.

Afin de confronter notre modèle aux systèmes hybrides de contrôle industriel, nous avons également sélectionné un jeu de données cyber-physique. SWaT (*Secure Water Treatment*) est un ensemble de données collectées sur une station d'épuration à échelle réduite entièrement opérationnelle, reproduisant les grandes usines modernes de traitement des eaux [1]. À la différence des données réseau pures ce banc d'essai comprend, outre les paquets de communication TCP, les valeurs des capteurs et actionneurs physiques du système. Très utilisé pour la recherche sur les CPS [32], SWaT est relativement récent (2016) et continue de servir de point de comparaison pour les modèles de détection d'intrusion pour CPS.

Il existe encore de nombreux autres jeux de données pour la détection d'intrusion, dont on peut retrouver les détails dans l'étude de Ring *et al.* [26]. Les trois que nous avons sélectionnés ici permettent de refléter cette diversité, et d'assurer la généralité de nos modèles à différents réseaux. Le choix du jeu de données est primordial dans la construction d'un SDI, et ce d'autant plus lorsque l'on utilise les méthodes d'apprentissage automatique.

Détection d'intrusion et apprentissage automatique

L'apprentissage automatique est un processus d'extraction de connaissances à partir de grandes quantités de données. Très efficaces et en constante évolution, les techniques d'apprentissage automatique sont couramment utilisées pour la détection d'intrusion par anomalies [9], et permettent d'améliorer la précision des modèles tout en nécessitant moins d'expertise humaine que les méthodes statistiques.

Les premières méthodes mises en place furent des techniques d'extraction d'attributs (*feature extraction*) comme les réseaux bayésiens ou les arbres de régression [33, 34]. Les algorithmes non supervisés, qui par définition ne nécessitent pas de données étiquetées, ont également produit des performances prometteuses, bien que plus limitées que celles des modèles supervisés [35]. Plus récemment, le développement des réseaux neuronaux a permis de nouveaux progrès, les modèles de classification profonds permettant une meilleure généralisation du SDI aux réseaux complexes et aux attaques réelles [36]. De plus, l'apparition de nombreuses

variantes comme les réseaux récurrents ou convolutifs ont contribué à la popularité des DNN pour la détection d'intrusion. Cependant, la nature déséquilibrée des jeux de données disponibles, et en particulier la faible représentation des attaques les plus rares rend l'entraînement de ces modèles long, coûteux, et limité [9].

Pour résoudre la question du déséquilibre des données ainsi que du coût de l'entraînement, l'apprentissage par renforcement offre une piste prometteuse : plus légers et modulables, les modèles de RL sont mieux adaptés à la contrainte du temps-réel et leurs performances concurrencent celles des modèles supervisés traditionnels.

2.1.2 Introduction à l'Apprentissage par Renforcement

Le RL est l'un des trois grands domaines de l'apprentissage automatique, spécifiquement conçu pour la prise de décision dans un environnement dynamique [13]. Contrairement à l'apprentissage supervisé et non-supervisé, dont les modèles sont respectivement entraînés à prédire un résultat à partir de données étiquetées et à découvrir les structures sous-jacentes de données non étiquetées, les agents RL apprennent à interagir avec un environnement en recevant un retour sous forme de récompenses positives ou négatives. L'objectif d'un agent RL est d'apprendre une politique, i.e. une correspondance entre les états de l'environnement et les actions de l'agent, qui maximise l'espérance de la récompense cumulée au cours d'un épisode.

Processus de Décision Markoviens

Un processus de décision de markovien (*Markov Decision Process*, MDP) est un cadre mathématique permettant de modéliser les problèmes de prise de décision impliquant un agent interagissant avec son environnement au cours d'une séquence de pas de temps discrets. Un MDP est défini par le tuple $\Omega = (S, A, P, R, \gamma)$, où

- S est l'ensemble des états de l'environnement,
- A est l'ensemble des actions possibles de l'agent,
- $P(s, a, s')$ est la probabilité de transition de l'état s à l'état s' avec l'action a ,
- $R(s, a, s')$ est la récompense immédiate obtenue grâce à l'action a dans l'état s , et
- $\gamma \in [0, 1]$ est le facteur de réduction permettant de pondérer l'importance des récompenses immédiates vis-à-vis des récompenses futures.

La plupart du temps, en réalité, l'agent n'a pas accès à l'état complet de l'environnement mais seulement à un état partiel, appelé observation. On parle alors de processus de décision markovien partiellement observable (*Partially Observable MDP*, ou POMDP), une généralisation

des MDP définie par un tuple $\Omega = (S, A, P, R, X, O, \gamma)$ avec

- S, A, P, R, γ tels que définis ci-dessus,
- X l'ensemble des observations accessibles à l'agent, et
- $O(s, a, x)$ la probabilité pour l'agent d'observer x dans l'état s avec l'action a .

Principes fondamentaux de l'apprentissage par renforcement

Les MDP posent le contexte mathématique des algorithmes de RL. Dans le cadre du RL, l'objectif est d'apprendre une politique $\pi : X \rightarrow A$ qui associe chaque observation à l'action maximisant la récompense de l'agent au cours de sa trajectoire τ . La politique optimale, désignée par π^* , maximise donc l'espérance du gain, $\mathbb{E}_\tau(G^\pi(\tau))$, avec le gain $G^\pi(\tau)$ tel que défini dans l'Équation 2.1

$$G^\pi(\tau) = \sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)), \quad s_t \sim \tau. \quad (2.1)$$

Pour définir l'optimalité de manière formelle, on peut utiliser la fonction de valeur des états, $V_\pi : S \rightarrow \mathbb{R}$. $V_\pi(s)$ représente le gain attendu à partir de l'état s en suivant la politique π (Équation 2.2).

$$V_\pi(s) = \mathbb{E}_\tau[G^\pi(\tau) | s_0 = s]. \quad (2.2)$$

La politique optimale π^* est alors la politique qui atteint $V_{\pi^*} = V^*$, la valeur optimale pour chaque état. Les méthodes RL dites *value-based* déduisent donc π^* de V^* , et V^* peut être calculée récursivement grâce à l'équation de Bellman (Équation 2.3).

$$V^*(s) = \max_a (R(s, a) + \gamma \sum_{s'} P(s, a, s') V^*(s')). \quad (2.3)$$

Pour finir, la fonction de valeur état-action $Q_\pi : S \times A \rightarrow \mathbb{R}$ est utilisée dans certains algorithmes comme alternative à V_π . La fonction Q d'un état s et d'une action a est le gain en partant de s , en choisissant l'action a puis en suivant π :

$$Q_\pi(s, a) = \mathbb{E}_\tau[G^\pi(\tau) | s_0 = s, a_0 = a]. \quad (2.4)$$

En résumé, les MDP fournissent un cadre mathématique pour la modélisation des problèmes de prise de décision séquentielle, tandis que les algorithmes RL apprennent une politique qui maximise le gain attendu dans ces problèmes. L'équation de Bellman est un concept clé

qui décrit la fonction de valeur optimale de chaque état, ou couple état-action. Celle-ci peut ensuite être utilisée pour en déduire la politique optimale.

Il est également intéressant de souligner que l'apprentissage par renforcement peut être appliqué à une grande variété de problèmes, les spécificités du MDP étudié pouvant fortement influencer la manière d'entraîner un agent. Ainsi, l'environnement peut être constitué d'un ensemble fini d'états (par exemple, une grille) ou à l'inverse d'états continus, à plusieurs dimensions. De même, l'ensemble des actions possibles peut être discret (direction cardinale) ou continu (direction angulaire, accélération, etc.). Une attention particulière doit donc être portée aux caractéristiques du problème étudié, pour choisir les méthodes les plus appropriées.

Apprentissage par Renforcement Profond

Pour résoudre la tâche complexe des MDP à états continu, les méthodes RL classiques, dites tabulaires [13], ne suffisent plus à apprendre une représentation précise de l'espace des états, observations et actions. Elles doivent donc être combinées aux méthodes par réseaux de neurones, pour devenir des méthodes d'apprentissage par renforcement profond (*Deep Reinforcement Learning*, DRL).

Un réseau de neurones est un type d'algorithme ML inspiré de la structure des neurones biologiques. Il est constitué d'un ensemble de nœuds (ou neurones) interconnectés qui reçoivent un signal d'entrée et effectuent des opérations mathématiques simples pour produire un signal de sortie. Un réseau de neurones profond (*Deep Neural Network*, DNN) est un type de réseau neuronal formé de plusieurs couches de neurones interconnectés, ce qui permet d'apprendre les motifs et caractéristiques complexes des données étudiées. Les neurones de la première strate sont connectés aux neurones de la strate suivante par l'intermédiaire d'un ensemble de paramètres, ou poids, ajustés au cours de l'entraînement pour optimiser les performances du réseau. Un exemple d'architecture est donné Figure 2.1, où les flèches sont associées aux poids du modèle et les cercles aux neurones individuels.

Entraîner un DNN implique donc d'optimiser les paramètres des neurones du réseau afin de minimiser la perte. Cette opération est généralement réalisée à l'aide d'un algorithme d'optimisation par rétro-propagation tel que la descente de gradient stochastique (*Stochastic Gradient Descent*, SGD), qui met à jour les paramètres de manière itérative à partir d'une fonction de perte représentant l'erreur entre la sortie prédite et la sortie souhaitée. La règle de mise à jour des poids ω en fonction de l'entrée $\mathbf{x}$, de l'étiquette y et de la fonction de perte $L(.,.)$ est donc la suivante :

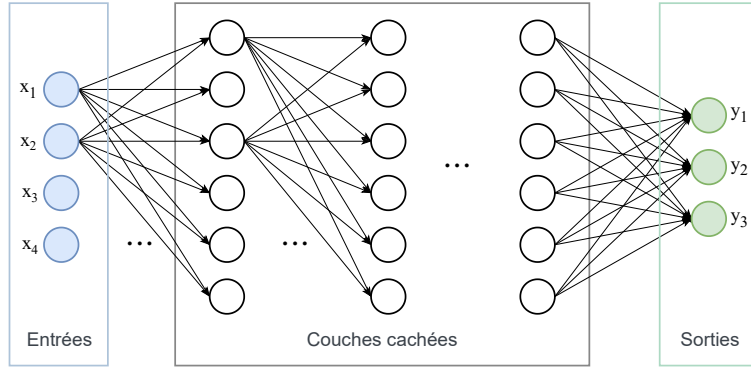


FIGURE 2.1 Architecture d'un réseau de neurones profond

$$\mathbf{\omega} = \omega - \alpha \cdot \nabla_{\omega} L(f_{\omega}(\mathbf{x}), y), \quad (2.5)$$

où α représente le taux d'apprentissage et f_{ω} la fonction de prédiction du modèle.

Le DRL combine le cadre RL avec la puissance de calcul des DNN. L'idée est d'utiliser un réseau DNN pour modéliser la fonction Q définie dans l'équation 2.4, la fonction de valeur des état (équation 2.2) et/ou la politique de l'agent.

Plusieurs algorithmes de DRL sont couramment répandus, avec pour chacun leurs propres forces et faiblesses. L'un des plus connus est le *Deep Q-Network* (DQN) [37], un algorithme dit *value-based* qui utilise un DNN pour approximer la fonction Q. Sa version élaborée, *Double Deep Q-Network* (DDQN) [38], est particulièrement performante, cependant les modèles DQN restent limités aux environnements à espace d'actions discret.

Les algorithmes d'optimisation de la politique, en revanche, permettent traiter les environnements à actions continues car ils modélisent la politique sans calculer explicitement la fonction Q. Le plus représentatif d'entre eux reste la méthode du gradient de la politique (*Policy Gradient Method*) [39], bien que ses performances soient maintenant surpassées par de nouvelles méthodes hybrides.

En effet, les méthodes acteur-critique (*actor-critic*) combinent les avantages des modèles *value-based* et *policy-based*, en apprenant simultanément la fonction de valeur et la politique. Elles sont adaptées aux environnements à actions discrètes et continues, et sont largement utilisées dans la recherche et en industrie : pour citer les plus connues, nous pouvons mentionner *Advantage Actor Critic*, (A2C) [40], *Proximal Policy Optimization* (PPO) [41], et *Deep Deterministic Policy Gradient* (DDPG) [42].

Le DRL est donc un outil puissant pour traiter les problèmes de prise de décisions séquen-

tielles. Le développement rapide d’algorithmes performants combiné à la croissance continue de la puissance de calcul disponible ont attiré l’intérêt des chercheurs sur ses différentes applications.

2.1.3 Modèles d’apprentissage par renforcement pour la détection d’intrusion

Nous avons appris dans la Section 2.1.1 que les méthodes de détection d’intrusion courantes se basent sur des règles prédéfinies ou sur de l’apprentissage supervisé pour identifier les comportements malveillants. Cependant, ces modèles sont souvent statiques et s’adaptent difficilement à des attaques nouvelles ou émergentes. Le RL, d’autre part, permet à un système d’apprendre de manière autonome à réagir dans son environnement en explorant les différentes actions possibles pour maximiser sa récompense. Dans cette sous-section, nous explorerons les travaux existants sur l’utilisation du RL pour la détection d’intrusion ainsi que les avantages et les défis de cette approche.

Les premiers travaux d’application du RL à la détection d’intrusion datent du début des années 2000, et se concentrent d’abord sur l’utilisation de méthodes tabulaires. Servin [43] utilise d’abord un algorithme de *Q-learning* pour détecter les intrusions dans le réseau grâce à une matrice de recherche, tandis que Xu *et al.* [44] choisissent les algorithmes d’apprentissage par différence temporelle [45] pour la détection en temps-réel. Enfin, Servin *et al.* [46] proposent une première approche évoluée de l’utilisation du RL pour les SDI, en combinant plusieurs agents pour la prise de décision. Il faut cependant noter que ces études évaluent leurs modèles sur des réseaux simulés et très simplifiés : la nature tabulaire de ces méthodes ne permet pas de les appliquer aux jeux de données plus réalistes disponibles dans la littérature, ni de les mettre en pratique sur un système réel. Le cas d’application le mieux adapté des méthodes de RL sans DNN à la détection d’intrusion reste donc une approche hybride entre la détection par anomalies et par signature. Ainsi, en associant les méthodes de RL avec des règles d’association et de corrélation des fichiers journaux, Deokar et Hazarnis [47] permettent au système de choisir les fichiers journaux les plus appropriés pour rechercher les signes d’une attaque.

Le DRL, en revanche, permet par sa nature complexe et non-linéaire d’apprendre les motifs sous-jacents de données réseau réalistes. Ainsi, Lopez-Martin *et al.* [48] montrent que les algorithmes de DQN ou DDQN permettent de détecter les attaques des jeux de données AWID et NSL-KDD avec des performances comparables aux méthodes d’apprentissage supervisé. La Figure 2.2 montre le mécanisme RL tel qu’adapté par les auteurs à la classification d’instances réseau. Le DNN de ces modèles étant léger et rapide, ces méthodes sont adaptées à l’apprentissage en ligne et en temps-réel pour des réseaux modernes avec environnements

évolutifs. Caminero *et al.* [49] proposent en outre un environnement dit adverse, composé d'un agent sélectionneur d'observations, pour compenser la nature déséquilibrée des jeux de données disponibles. L'algorithme DDQN combiné à cet environnement adverse surpasse les performances des autres méthodes d'apprentissage automatique. Plusieurs études ont depuis montré la pertinence de cette approche dans d'autres domaines, comme l'internet des objets [50] ou les essaims de drones [51].

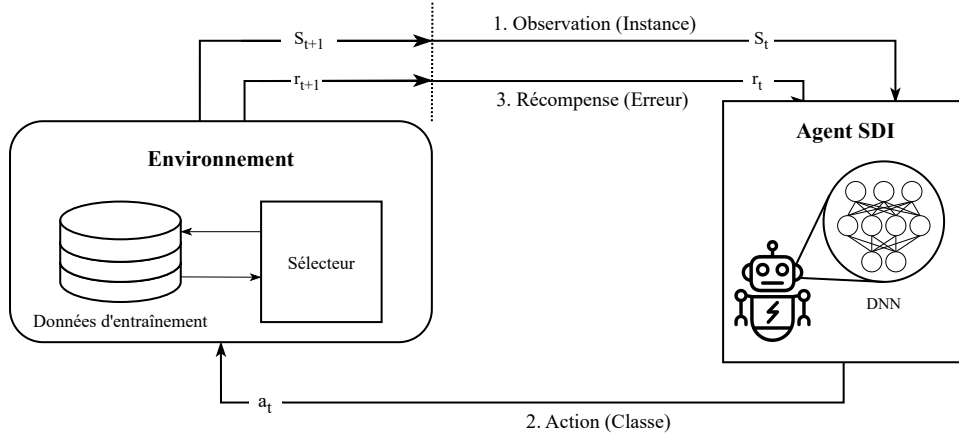


FIGURE 2.2 Mécanisme d'entraînement d'un agent DRL pour la détection d'intrusion

Pour conclure, l'utilisation du RL pour la détection d'intrusion est particulièrement intéressante car elle permet une adaptation automatique à des situations changeantes et une amélioration continue de la performance du système. La littérature montre que les approches de détection d'intrusion par DRL donnent des résultats prometteurs. Cependant, il est important de prendre en compte les limites de cette approche, notamment le coût computationnel élevé de l'entraînement ainsi que les risques liés aux comportements adverses. Des travaux futurs pourraient inclure l'exploration de nouvelles techniques de RL, l'évaluation de leur applicabilité dans des environnements plus complexes, et l'étude de la menace des attaques adverses. En somme, le RL est une approche prometteuse pour la détection d'intrusion dans les systèmes de sécurité informatique, mais nécessite encore des efforts de recherche pour assurer sa fiabilité.

2.2 Comportement Adverse et DRL

Dans la section précédente, nous avons exploré comment le DRL peut être appliqué aux SDI pour améliorer les performances de détection d'intrusion. Cependant, la sécurité des modèles d'apprentissage en général et du DRL en particulier reste un défi important : le fonction-

nement des systèmes d'apprentissage automatique peut être perturbé par l'introduction de données falsifiées, dites exemples adverses. Les exemples adverses sont des entrées intentionnellement conçues pour tromper les modèles d'apprentissage automatique et produire des résultats incorrects ou indésirables. Les DNN sont particulièrement vulnérables à ces perturbations. Dans cette section, nous allons discuter des recherches récentes sur les exemples adverses dans les modèles d'apprentissage en général et dans le contexte du DRL en particulier. Nous explorerons également les approches spécifiques au DRL en introduisant une taxonomie conçue pour faciliter la distinction et la compréhension des différentes méthodes.

2.2.1 Fiabilité des DNNs : existence des exemples adverses

Les DNN sont des outils puissants utilisés dans de nombreuses applications critiques. Toutefois, leur structure complexe et hautement non-linéaire les rend difficiles à contrôler, ce qui soulève une inquiétude croissante quant à la fiabilité des modèles alors même que diverses techniques sont mises au point pour les induire en erreur. Le *National Cybersecurity Center of Excellence* (NCCE) du NIST [52] et son homologue européen, *ETSI Standards* [53], fournissent terminologies et ontologies pour encadrer l'étude de ces méthodes adverses.

ML adverse

Les attaques adverses sont initialement conçues pour exploiter les vulnérabilités des réseaux DNN, menaçant ainsi leur confidentialité, leur disponibilité ou leur intégrité [54]. Si le terme "attaque adverse" suggère une intention malveillante, il convient de noter que ces méthodes peuvent également être utilisées par le propriétaire d'un modèle afin d'améliorer ses performances et d'évaluer ses vulnérabilités. Ainsi, l'apprentissage automatique adverse (*Adversarial Machine Learning*, AML) est un domaine de recherche qui a pour but d'analyser les capacités des attaquants potentiels, comprendre l'impact de leurs attaques et développer des algorithmes de ML capables de résister à ces menaces de sécurité.

L'attaquant, ou adversaire, peut cibler :

- la phase d'apprentissage du modèle, en modifiant les données d'entraînement - un type d'attaque communément appelé empoisonnement, et
- la phase de prédiction, de manière passive en extrayant de l'information sur le modèle à partir de l'analyse des sorties (inférence) ou de manière active en modifiant les données d'entrées pour fausser la classification (évasion).

Les méthodes d'évasion sont également appelées attaques par injection d'exemples adverses : en raison de la difficulté à contrôler les entrées d'un modèle après son déploiement, elles font partie des principales menaces qui pèsent sur les algorithmes d'apprentissage profond.

La Figure 2.3 illustre les grandes catégories d’attaque adverses et leurs cibles spécifiques.

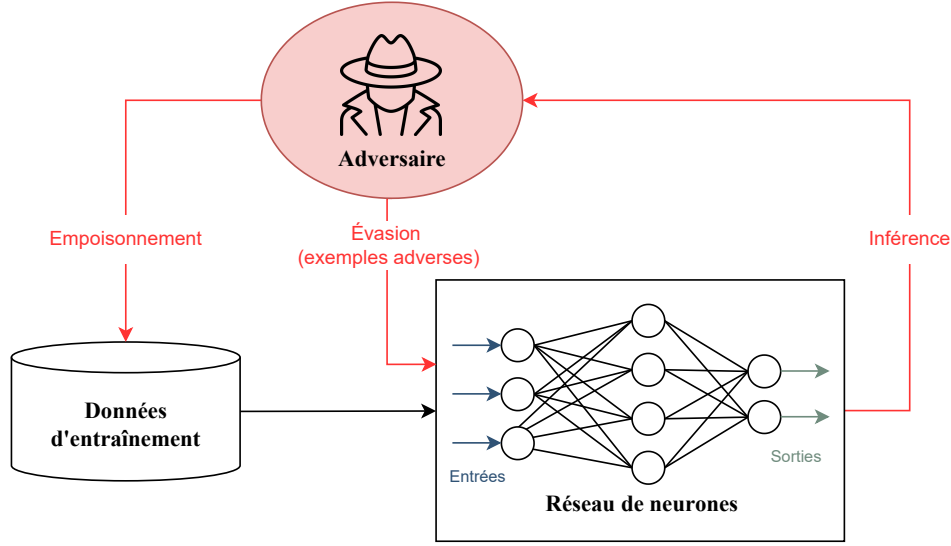


FIGURE 2.3 La menace adverse en ML

Génération d'exemples adverses

Le grand nombre de dimensions des entrées d'un DNN conduit inévitablement à des zones floues du modèle, et une grande sensibilité aux petites perturbations. Les exemples adverses sont des instances légèrement modifiées, soigneusement conçues pour tromper le modèle et l'amener à une classification erronée tout en restant indétectables car très proches de l'instance initiale. De nombreuses méthodes existent pour calculer ces perturbations, des modifications manuelles coûteuses [55] aux calculs avec gradient, mais l'objectif fondamental de la génération d'exemples adverses reste simple et peut être résumé en une phrase : étant donné l'instance originale $\mathbf{x}$, trouver l'exemple le plus proche $\mathbf{x}'$ par rapport à la métrique choisie $||\cdot||$ qui conduise la fonction f_ω d'un modèle à modifier son résultat. L'Équation 2.6 formalise la définition d'un exemple adverse :

$$\min_{\mathbf{x}'} ||\mathbf{x} - \mathbf{x}'|| \quad s.t. \quad f_\omega(\mathbf{x}) \neq f_\omega(\mathbf{x}'). \quad (2.6)$$

Initialement développées pour tromper les algorithmes de classification d'images supervisées, de nombreuses méthodes existent pour calculer la perturbation optimale selon les contraintes du modèle et les objectifs de l'adversaire. L'étude de Yuan *et al.* [56] détaille l'ensemble de ces approches supervisées. Dans le cadre de ce projet, nous nous intéresserons plus particulièrement aux méthodes adaptées à l'apprentissage par renforcement.

Application au DRL

Bien que conçus pour tromper des modèles supervisés, avec quelques modifications les exemples adverses se sont avérés tout aussi efficaces sur les modèles DRL. Behzadan et Munir [57] ont ainsi montré la vulnérabilité des DQNs face aux perturbations adverses, et leurs travaux furent ensuite étayés par plusieurs études [14]. En outre, le cadre du RL offre des possibilités adverses plus vastes que la simple adaptation de méthodes supervisées. En effet, chaque composante du système peut se révéler vulnérable (observations, environnement, fonction de transition), tandis que les dépendances long-termes inhérentes au RL soulèvent des problèmes de sécurité complexes. D'autre part, cette plus grande liberté adverse peut mener à de nouvelles stratégies de défense pour améliorer la robustesse des modèles.

Ces différences majeures rendent les ontologies des méthodes adverses supervisées peu adaptées pour décrire les méthodes adverses par renforcement, or la littérature manque d'ontologies adaptées au RL [58]. Par conséquent, il est crucial pour bien comprendre la variété des approches adverses RL de développer une taxonomie systématique et cohérente avec la structure de l'apprentissage par renforcement.

2.2.2 Taxonomie

Afin de mieux comprendre la variété des méthodes de génération d'exemples adverses disponibles pour le RL ainsi que leurs particularités, nous proposons dans cette section une taxonomie détaillée de ces approches. Notre étude prend en compte les aspects pertinents des ontologies existantes pour l'apprentissage supervisé [56], ainsi que de nouveaux critères spécifiques au RL. La Figure 2.4 illustre la taxonomie dans son ensemble, et nous développons ensuite chaque catégorie plus en détail.

Cible de la perturbation

Il s'agit de l'élément du MDP modifié par l'adversaire.

- *Observations* : les perturbations sur l'ensemble des observations sont conçues pour modifier les entrées telles que perçues par l'agent en entrée du réseau : par exemple, en ciblant directement les capteurs d'un véhicule autonome. Si ce type de méthodes offre une grande liberté quant à la perturbation, il nécessite un accès aux mécanismes internes de la victime et peut donc s'avérer facilement détectable, les observations générées n'étant pas toujours réalistes.
- *Environnement* : une autre possibilité est de perturber l'environnement dans lequel évolue l'agent. Ces approches sont intéressantes lorsqu'il s'agit d'environnements réels,

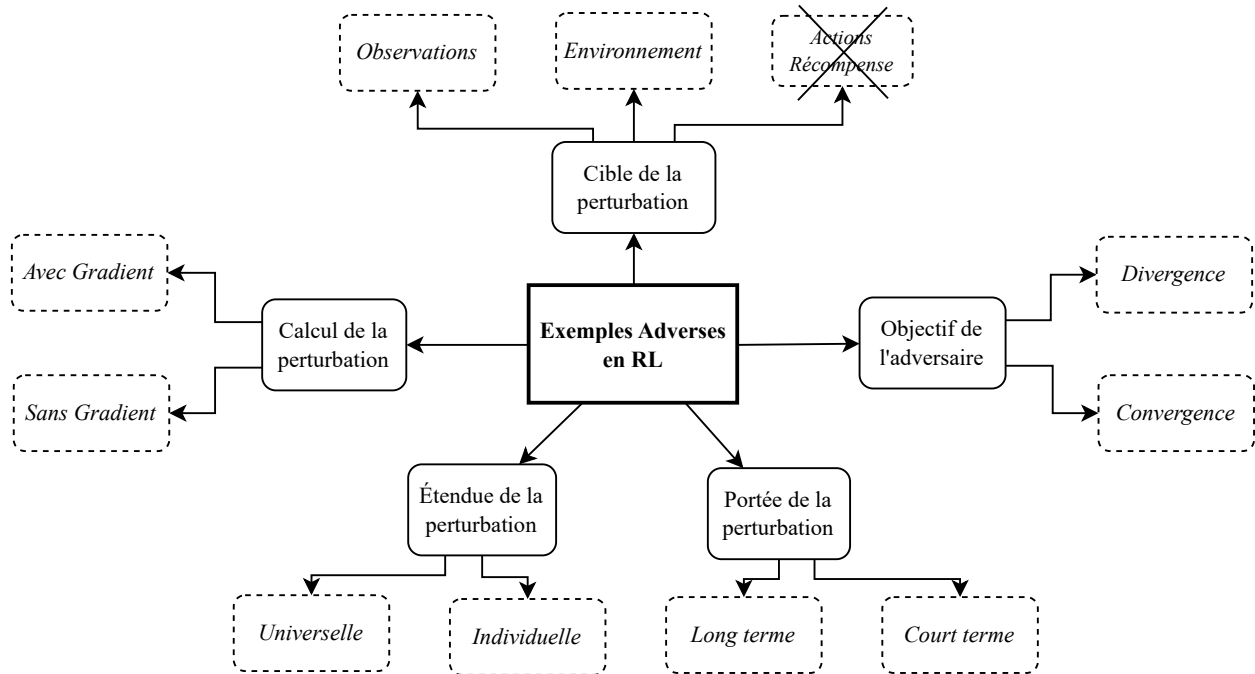


FIGURE 2.4 Taxonomie des exemples adverses en RL

ou de simulations de situations réelles. Plus subtiles à calculer que les perturbations directes des entrées parce que leur effet sur les observations est indirect, elles sont également plus discrètes en raison de la confiance intrinsèque de l’agent dans son environnement. Par exemple, dans le cas des véhicules autonomes, l’adversaire peut coller sur les panneaux de circulation des patches qui empêchent les capteurs de les identifier.

- *Actions et récompenses* : enfin, un adversaire pourrait également envisager de perturber les actions de l’agent, ou bien sa fonction de récompense. Cependant, ces perturbations n’auraient d’impact que si elles avaient lieu lors de l’entraînement : elles font donc parties des méthodes d’empoisonnement des données, et ne sont pas considérées comme des exemples adverses. Ces cibles sont donc rayées de la Figure 2.4, car elles sortent du cadre envisagé dans ce mémoire et ne seront donc pas considérées dans la suite de cette étude.

Portée de la perturbation

Ce critère prend en compte le cadre temporel envisagé lors du calcul de la perturbation.

- *Court-terme* : les méthodes court-terme visent à obtenir la pire action locale, à chaque pas de temps. Elles découlent intuitivement des méthodes supervisées, et n’exploitent

pas la corrélation possible entre les états. Ces approches gloutonnes se concentrent par exemple sur la minimisation de la récompense immédiate, ou sur la maximisation de la fonction de perte de la politique au pas de temps donné.

- *Long-terme* : à l'inverse, les méthodes à portée long-terme prennent en compte plusieurs pas de temps consécutifs pour influencer les décisions futures. En tenant compte de la nature séquentielle du RL, leur objectif est alors de minimiser la récompense cumulée. Le plus souvent, elles consistent en deux étapes : la conception d'une politique adverse, puis le calcul des perturbations permettant d'atteindre cette politique.

Objectif de l'adversaire

Perturber un agent peu impliquer simplement de changer son comportement, ou bien plus subtilement de l'orienter vers un comportement spécifique, choisi à l'avance.

- *Divergence* : les méthodes dites divergentes ont pour objectif d'empêcher l'agent d'agir de manière optimale. Leur but est donc de faire dévier l'agent de sa politique initiale.
- *Convergence* : d'autre part, les méthodes convergentes visent à amener l'agent à un état, une politique ou une action particulière, à la manière des méthodes supervisées ciblées.

Calcul de la perturbation

L'importance du temps-réel en RL rend le coût du calcul des perturbation, directement lié au type de méthode envisagée, un facteur crucial à prendre en compte.

- *Avec gradient* : les perturbations générées à partir du gradient des paramètres du modèle nécessitent une connaissance totale (boîte blanche) ou par approximation (imitation) des poids de ce modèle. Le gradient est alors calculé pour chaque perturbation, ce qui peut être coûteux en termes de puissance et de temps de calcul.
- *Sans gradient* : les données de transition (récompense obtenue, action choisie, état suivant) contiennent des informations précieuses sur le modèle de RL, et permettent donc le développement de méthodes plus rapides et plus légères car exemptes de calcul de gradient.

Étendue de la perturbation

Enfin, l'adversaire peut choisir la généralisabilité des perturbations calculées.

- *Individuelle* : les perturbations individuelles sont spécifiques à chaque état rencontré. Calculées pour chaque pas de temps, elles sont performantes mais coûteuses en temps

et en puissance de calcul.

- *Universelle* : les méthodes universelles créent une unique perturbation valable pour tous les pas de temps. Dans le contexte RL, cette perturbation est généralement calculée hors ligne à partir de données de transition préalablement mémorisées.

Ces critères de classification spécifiques permettent d’analyser les différentes méthodes de génération d’exemples adverses pour le RL en mettant l’accent sur leurs particularités, leurs points forts et leurs limites.

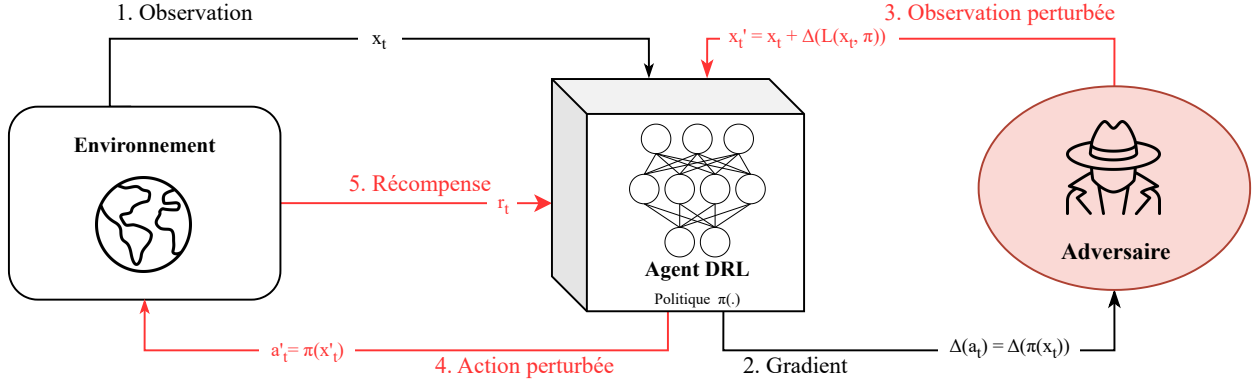
2.2.3 Méthodes de génération

Dans cette partie, nous utilisons la taxonomie introduite ci-dessus comme cadre pour étudier les avancées récentes sur les exemples adverses et le DRL. Dans un souci de concision, notre analyse ne se veut pas exhaustive, mais met plutôt l’accent sur les méthodes les plus représentatives de chaque grande catégorie.

Méthodes issues de l’apprentissage supervisé

Si la génération d’exemples adverses fut initialement développée pour les modèles supervisés de classification d’images, la plupart de ces méthodes peuvent être adaptées à un contexte RL [57]. Au regard de la taxonomie orientée RL, ces perturbations sont donc individuelles, court-terme, et visent les observations. La plupart utilisent de plus le gradient dans leurs calculs, et conviennent donc aux situations où le coût de calcul et la complexité temporelle ne sont pas critiques.

En 2017, Behzadan & Munir [57] montrent pour la première fois que les DQN sont vulnérables aux perturbations adverses. Ils conçoivent alors un cadre d’attaque nommé *Policy Induction Attack* (PIA) pour exploiter cette vulnérabilité, à partir de l’approche supervisée *Fast Gradient Sign Method* (FGSM) [59] : le fonctionnement de la méthode PIA est décrit dans la Figure 2.5. Dans le cas supervisé, l’objectif de l’adversaire est de maximiser la perte entre l’étiquette y et la prédiction du modèle $f_\theta(x)$. Dans le cadre du RL, les données ne sont pas étiquetées : les auteurs cherchent alors plutôt à minimiser la fonction de perte suivante : $L(s, \pi_\theta) = \log(\pi_\theta(s, a^*))$ (où a^* est la meilleure action). Le problème revient donc à minimiser la probabilité de choisir la meilleure action. Si les auteurs n’évaluent cette méthode que sur des algorithmes DQN, Huang *et al.* [14] mènent des expériences similaires sur les environnements Atari à espace d’actions discret, avec différents algorithmes (i.e. DQN, A3C, TRPO) pour prouver la généralisabilité de PIA et la propriété de transférabilité du RL. Leurs résultats montrent que DQN est l’algorithme le plus vulnérable face à PIA.

FIGURE 2.5 *Policy Induction Attack (PIA)*

Pattanaik et al. [60] développent ensuite une version ciblée, ou convergente, de PIA : les auteurs introduisent donc une nouvelle fonction de perte à maximiser, $L(s, \pi_\theta) = \log(\pi_\theta(s, a_w))$ où a_w est la pire action (selon la fonction Q de la victime). Le problème revient donc plutôt à maximiser la probabilité de choisir la pire action, et les performances sont meilleures que celles de PIA. Les expériences sont menées sur des environnements de contrôle continu Mujoco avec des algorithmes DDQN et DDPG.

Outre FGSM, de nombreuses autres méthodes supervisées peuvent être adaptées au RL de manière similaire. Sans entrer dans les détails, nous pouvons mentionner *Basic Iterative Method* BIM [59], *Policy Gradient Descent* (PGD) [61], Carlini & Wagner [62], et *Jacobian Saliency Map Attack* (JSMA) [63].

Les exemples adverses pour le RL ont depuis été étendus à de nombreux cas d'usage particuliers. Dans le cas où l'adversaire n'a pas accès aux poids du modèle pour calculer le gradient, Behzadan & Hsu [64] montrent que la propriété de transfert permet d'appliquer les perturbations générées à partir d'une imitation du modèle initial [65]. Lin *et al.* [66] s'attaquent au RL multi-agents, où les modèles ne sont pas différentiables et la récompense de groupe est délicate à estimer, et Zhang *et al.* [67] s'intéressent aux méthodes indépendantes du calcul de la fonction de valeur des états (*Q-function*).

Enfin, nous pouvons porter une attention spéciale aux méthodes dites universelles dont la perturbation unique est calculée hors-ligne, avant le déploiement du modèle. Ces méthodes, particulièrement adaptées au temps-réel, ont été rapidement développées pour le RL : Copy-CAT [68], UAP-S [69] ou encore DAP [70] montrent des performances de leurrage comparables à celles des approches individuelles.

la politique optimale de r_A .

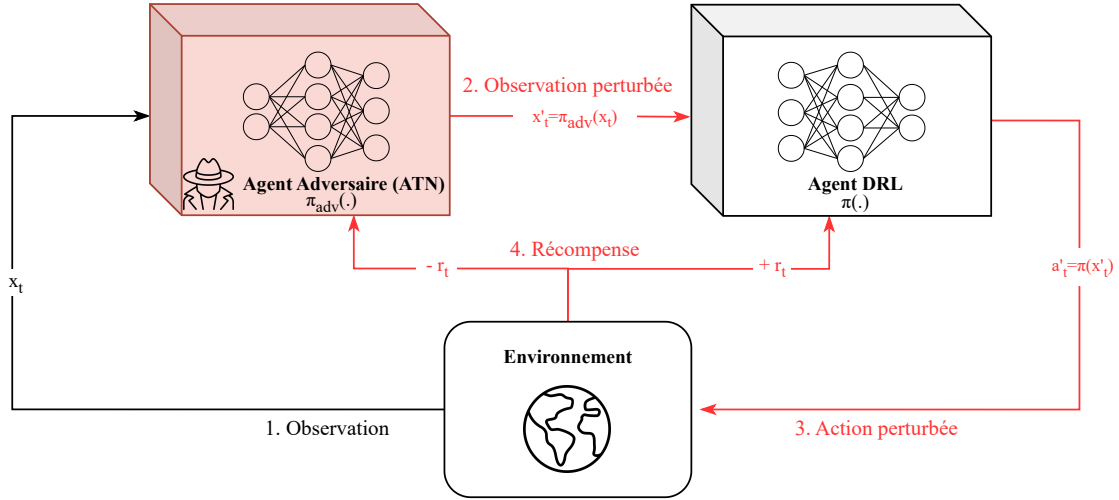


FIGURE 2.7 *Optimal Attack for RL Policies (OARLP)*

Enfin, les perturbations peuvent aussi être appliquées à l’environnement lui-même. Ainsi, Pinto *et al.* [75] conçoivent un agent RL adverse interagissant directement avec l’environnement plutôt que les observations. Par exemple, l’adversaire peut appliquer des forces déstabilisatrices dans la simulation, et recevoir une récompense positive pour l’échec du protagoniste. En posant le problème comme un jeu à somme nulle entre le protagoniste et les perturbations, plusieurs extensions de cette approche existent et permettent de s’adapter à diverses situations [76, 77].

2.3 Défenses contre les attaques adverses

L’utilisation de techniques d’apprentissage profond, et en particulier du DRL, est une piste prometteuse pour la détection d’intrusion (Section 2.1). Ces méthodes ne sont cependant pas à l’abri des attaques par exemples adverses (Section 2.2), qui peuvent conduire à des erreurs de détection aux conséquences graves. La défense contre les exemples adverses est donc cruciale pour la mise en pratique des modèles DRL en détection d’intrusion. S’il n’existe pas encore d’études concernant la défense spécifique des SDI par RL face aux attaques adverses, de nombreux travaux s’intéressent à la protection des SDI supervisés tandis qu’en parallèle le sujet de la robustesse des modèles DRL suscite un intérêt croissant. Cette section abordera donc le sujet de la défense adverse selon deux points de vue : d’abord, nous nous intéresserons aux méthodes de protection pour les SDI supervisés, puis après avoir proposé une définition de la robustesse en RL nous explorerons les pistes d’amélioration de la fiabilité des modèles.

2.3.1 Protéger les SDI face aux exemples adverses

Il existe deux types de contre-mesures pour mitiger le risque lié aux exemples adverses : les méthodes réactives, dont le but est de détecter les perturbations une fois que le modèle est déployé, et pro-actives, qui ont pour objectif d'obtenir un modèle robuste par nature [56].

La plus intuitive des méthodes réactives est la **détection adverse** : entraîner un classifieur indépendant sur les observations à signaler les éventuelles perturbations. Hashemi et Keller [78] montrent que leur SDI supervisé ainsi protégé est imperméable aux quatre types d'attaques sur lesquelles le détecteur est entraîné. En revanche, Carlini et Wagner [79] soulignent le manque de généralisabilité de la détection adverse : un léger changement dans la méthode d'attaque entraîne facilement l'échec du détecteur. Une approche alternative, la **re-construction**, permet d'obtenir des données propres à partir d'exemples adverses avant qu'ils ne soient présentés au SDI [80]. Cela peut cependant biaiser et diminuer les performances du SDI, car toutes les entrées sont ainsi pré-traitées. Enfin, la **vérification** des propriétés des DNN est une solution prometteuse pour détecter les exemples adverses, car elle n'est pas limitée aux attaques adverses connues. La vérification revient à contrôler les propriétés d'un réseau neuronal, c'est-à-dire déterminer si l'observation en entrée viole ou satisfait la propriété [81]. Cependant, ce procédé peut s'avérer très lent en raison de l'importance des calculs nécessaires, et ne fonctionne que pour les réseaux DNN comportant plusieurs centaines de nœuds [82].

Les méthodes réactives sont dans l'ensemble limitées par l'ajout d'une extension au modèle après sa construction terminée. Les approches pro-actives, d'un autre côté, se concentrent sur la construction d'un modèle intrinsèquement robuste. On peut notamment citer la **distillation** [15], conçue à l'origine pour réduire la taille des DNN en transférant les connaissances d'un grand réseau à un petit [83], qui en diminuant la profondeur du réseau diminue également les possibilités adverses. De manière similaire, plusieurs études s'intéressent à l'architecture des classifieurs : l'hybridation des DNN avec un procédé d'estimation d'erreur gaussien semble améliorer la robustesse des modèles, cependant la généralisation de ces découvertes n'a pas été prouvée [84]. Enfin, l'approche la plus prometteuse pour construire un modèle robuste reste **l'entraînement adverse** : utiliser les exemples adverses eux-mêmes lors de la phase d'apprentissage [85, 86] permet d'obtenir un modèle mieux généralisable, et parfois même plus précis [87]. L'entraînement adverse doit cependant être utilisé avec parcimonie pour ne pas fausser les performances de détection, et il n'est pas toujours efficace pour contrer les attaques les plus récentes. Le défenseur doit notamment choisir comment générer les exemples adverses pour l'entraînement (la méthode choisie pouvant fortement influencer la robustesse obtenue), et à quelle fréquence les injecter. Cette approche est particulièrement utilisée en

détection d'intrusion car facilement modulable et relativement simple à implémenter [88–90].

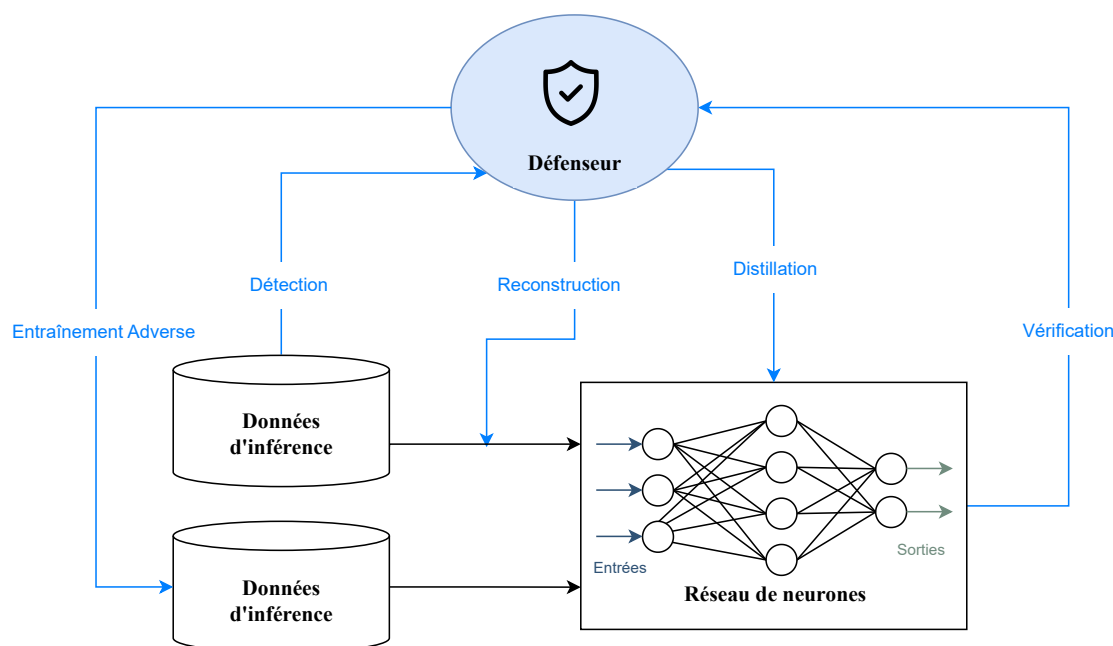


FIGURE 2.8 Techniques de défense contre les exemples adverses

L'ensemble de ces techniques de défense ont été testées sur des modèles de détection d'intrusion supervisés, avec des résultats encourageants mais principalement limités par leur manque de généralisation. À notre connaissance, personne ne s'est encore intéressé aux perspectives ouvertes par le DRL pour la détection d'intrusion, en terme de défense et de robustesse.

2.3.2 Robustesse du DRL

Le développement d'algorithmes DRL robustes et résistants aux attaques adverses est un sujet de recherche important et actif dans le domaine. Afin d'étudier son application au domaine de la détection d'intrusion, nous nous intéressons aux approches existantes dans la littérature.

Méthodes de défense pour le RL

Ainsi, nous avons défini un modèle RL robuste comme un modèle dont les performances ne varient pas, et ce quel que soit le MDP dans lequel il évolue (dans la mesure où ce MDP est issu d'une distribution réaliste). Tout comme dans le cas supervisé, de nombreuses méthodes existent pour tenter d'approcher cette robustesse. Si elles n'ont pas encore toutes été

appliquées aux problèmes de détection d'intrusion, elles ouvrent des perspectives intéressantes sur le sujet et méritent d'être explorées plus en détail.

La plupart des méthodes de défense dites supervisées (Section 2.3.1) peuvent facilement être étendues au cadre RL. Ainsi, la détection de perturbation est aussi efficace en RL qu'en supervisé [16, 91], tout comme la distillation [92]. Quand la question de l'architecture du réseau, elle soulève tout autant d'interrogations et fait l'objet d'études prometteuses [93, 94].

En revanche, la nature séquentielle des algorithmes et les nombreux éléments impliqués dans l'entraînement d'un agent RL (observations, environnement, récompenses, etc.) rendent les approches d'entraînement adverse particulièrement intéressantes dans l'optique d'améliorer la généralisabilité et la robustesse des modèles. En effet, le défenseur peut jouer sur de nombreux paramètres différents selon le contexte envisagé et l'objectif ciblé. Le principal défi de ces approches est toutefois de maintenir un entraînement stable malgré l'ajout de perturbations.

2.4 Conclusion

En conclusion, cette revue de littérature a permis d'explorer en profondeur les liens entre la détection d'intrusion et le RL. Nous avons tout d'abord introduit le domaine de la détection d'intrusion, mettant en évidence son importance cruciale dans la sécurité informatique. Ensuite, nous avons présenté les concepts clé du RL, soulignant son potentiel prometteur pour améliorer la précision et l'efficacité des systèmes de détection d'intrusion.

Nous avons également souligné l'existence de méthodes adverses et les défis liés à la fiabilité des modèles de DRL, en particulier dans le cadre de la détection d'intrusion. Après avoir proposé une classification des différentes approches de génération d'exemples adverses, nous avons donc pris conscience de la nécessité de développer des approches défensives efficaces pour garantir l'intégrité et la fiabilité des systèmes de détection d'intrusion basés sur le RL.

Ces connaissances sur les avantages, les limitations et les défis associés à l'utilisation du RL pour la détection d'intrusion seront essentielles pour orienter notre approche méthodologique et nos travaux de recherche ultérieurs dans le domaine de la protection des systèmes de détection d'intrusion contre les attaques adverses.

CHAPITRE 3 MÉTHODOLOGIE

La troisième partie de ce mémoire aborde la méthodologie utilisée pour mener ce projet. Nous commencerons par décrire les jeux de données utilisés, en mettant en évidence leurs caractéristiques spécifiques, les pré-traitements appliqués et la manière dont ils sont adaptés à notre problématique. Ensuite, nous présenterons les algorithmes et les bibliothèques que nous avons sélectionnés pour la mise en œuvre de notre solution. Nous expliquerons en détail leur fonctionnement, leurs paramètres clé et leur pertinence dans le contexte de la détection d'intrusion basée par DRL. Enfin, nous aborderons les métriques d'évaluation envisagées pour mesurer la performance et l'efficacité de notre solution. Ces métriques nous permettront de quantifier la capacité de notre modèle à détecter les intrusions et à résister aux attaques adverses. Ce chapitre méthodologique joue un rôle essentiel dans notre approche, car il présente les outils et les techniques nécessaires pour mener à bien notre recherche et atteindre nos objectifs.

3.1 Données utilisées

Dans cette section, nous présentons les jeux de données utilisés dans notre étude. Nous discutons de leurs caractéristiques, de leur origine, de leur taille, ainsi que des spécificités des données qu'ils contiennent. Nous décrivons également les pré-traitements que nous avons appliqués pour préparer les données en vue de l'entraînement de notre modèle.

3.1.1 Jeux de données

NSL-KDD

Le jeu de données NSL-KDD est l'un des jeux de données les plus utilisés dans la recherche en détection d'intrusion. Développé pour évaluer les systèmes de détection d'intrusion en utilisant des données de trafic réseau réel [29], il comprend un ensemble diversifié d'attributs qui permettent de représenter différents aspects du trafic réseau. 38 attributs continus contiennent les informations telles que la durée de la connexion, le nombre de paquets, la taille des paquets, etc., tandis que 3 caractéristiques discrètes permettent de connaître le protocole ou le service utilisé lors de la connexion : au total, il y a donc 41 attributs.

Chaque observation est également étiquetée. Il y a 22 types d'attaques différents dans l'ensemble d'apprentissage mais 38 dans l'ensemble de test : un modèle de détection efficace doit donc détecter certaines anomalies non rencontrées lors de l'apprentissage. Pour simplifier

l'apprentissage, toutes ces attaques sont cependant regroupées au sein de 4 grandes classes. Les quelques 120 000 échantillons sont donc répartis de la manière suivante : Normal (53,46 %), Déni de service (DoS) (36,46 %), Sondage (PROBE) (9,25 %), Remote-to-Local (R2L) (0,79 %) et User-to-Root (U2R) (0,04 %).

Malgré sa relative ancienneté, nous avons choisi d'utiliser ce jeu de données pour plusieurs raisons. Tout d'abord, NSL-KDD est largement reconnu comme un ensemble de données de référence dans le domaine de la détection d'intrusion, ce qui en fait une base de travail fiable et bien documentée. De plus, ce jeu de données propose une grande variété de scénarios de cyber-attaques et de trafic réseau, reflétant ainsi des situations réalistes et diversifiées auxquelles les systèmes de détection d'intrusion peuvent être confrontés dans des environnements réels. En outre, NSL-KDD présente le défi du déséquilibre de classe, un aspect important à prendre en compte lors de l'évaluation des performances des modèles d'apprentissage en détection d'intrusion. En utilisant NSL-KDD, nous sommes donc en mesure d'analyser la capacité de nos agents à détecter efficacement des attaques malveillantes dans un contexte de données déséquilibrées et variées, fournissant ainsi des informations pertinentes pour la sécurité des infrastructures critiques.

AWID

AWID est un second jeu de données de détection d'intrusion, cette fois orienté réseaux sans fil. Il comprend un ensemble varié de scénarios de trafic réseau sans fil, reflétant différentes conditions et types d'attaques possibles. Ainsi, les données de connexion du protocole 802.11 ont été récoltées pendant 80 minutes de fonctionnement du système, avec 156 attributs extraits de chaque paquet. Les données sont séparées en un ensemble d'entraînement (1,8 million d'entrées) et un ensemble de test (0,6 million d'entrées) pour un total de 2,4 millions de paquets.

La principale caractéristique du jeu de données AWID est que les données sont capturées dans un environnement réel. Cela leur confère un niveau élevé de réalisme et de représentativité par rapport aux scénarios de détection d'intrusion sans fil. En utilisant ce jeu de données, il est donc possible de simuler et d'évaluer des attaques réelles qui pourraient être rencontrées dans de véritables réseaux sans fil.

Enfin, de par sa nature plus réaliste, AWID est également beaucoup plus déséquilibré que NSL-KDD. Ainsi, seuls 9% des observations correspondent à des attaques, ce qui peut rendre l'entraînement d'un SDI plus ardu en raison du manque de données étiquetées positivement, et pose un nouveau défi.

En utilisant AWID en complément de NSL-KDD, nous élargissons notre portée d'étude et nous assurons que notre modèle d'apprentissage en détection d'intrusion est testé et évalué dans des environnements variés et représentatifs, renforçant ainsi la pertinence et la validité de nos résultats et conclusions.

Secure Water Treatment (SWaT)

Le jeu de données SWaT (*Secure Water Treatment*) est largement utilisé dans les études de systèmes cyber-physiques en raison de sa pertinence et de sa représentativité pour simuler des environnements réels de traitement de l'eau. En effet, SWaT est une station d'épuration à échelle réduite entièrement opérationnelle, qui reproduit les grandes usines modernes de traitement des eaux [1].

Le système comprend six processus, ou sections, chacun étant commandé par un contrôleur logique programmable (*Programmable Logic Controller, PLC*) indépendant : la Figure 4.6 schématise le mécanisme détaillé du traitement de l'eau avec notamment ses 51 capteurs et actionneurs.

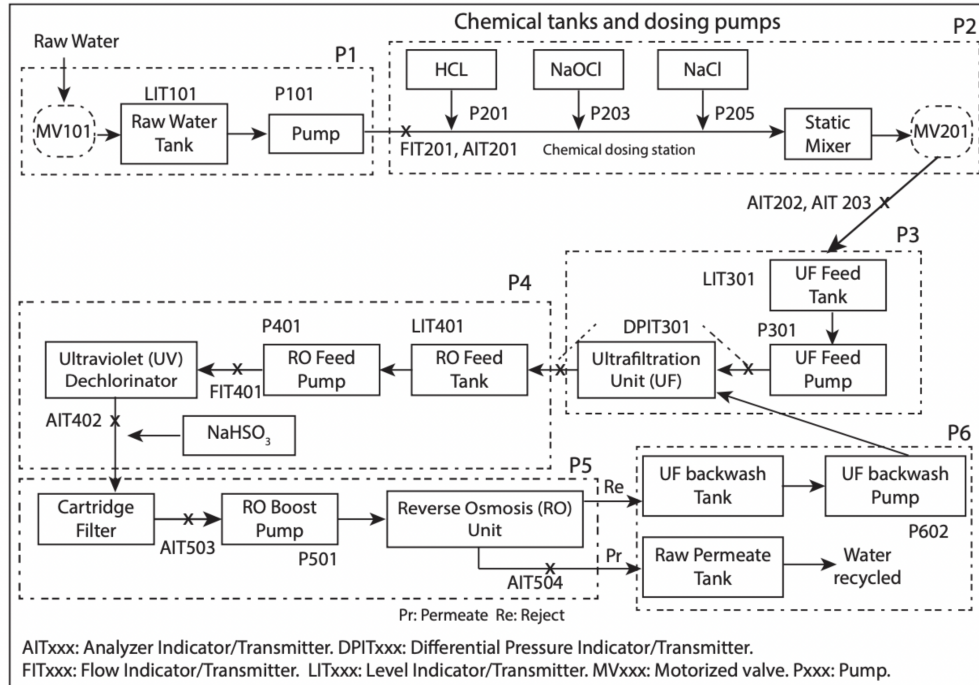


FIGURE 3.1 Architecture des processus de SWaT [1]

Les données disponibles correspondent aux valeurs de ces capteurs et actionneurs, collectées chaque seconde pendant 11 jours. Ainsi, une observation du jeu de données contient 51

valeurs, une indication temporelle (*timestamp*), et une étiquette ("*Attack*" ou "*Normal*"), soit 53 colonnes en tout. Les 7 premiers jours de données forment l'ensemble d'entraînement, les 4 jours suivant servant de test. Au cours des 7 premiers jours, le système SWaT est donc en fonctionnement normal et les données collectées sont exemptes d'attaques. 36 attaques ont ensuite été conçues puis simulées lors des jours correspondant aux données de test.

La particularité des données cyber-physiques par rapport aux données réseau est la forte corrélation temporelle entre plusieurs observations successives. Ainsi, les attaques s'inscrivent dans la durée, entre 101 secondes et plusieurs dizaines de minutes, et il peut s'écouler plusieurs minutes avant que l'effet d'une attaque ne soit visible [95]. La Figure 3.2 montre l'un des 36 scénarios d'attaque : la valeur du capteur de niveau de l'eau FIT401 est d'abord modifiée de 1.7 à 0.7 ce qui ne change pas le comportement de la pompe P501, puis cette valeur est amenée à 0. Alors, le déchlorateur UV401 arrête de fonctionner, et la pompe n'évacue plus l'eau du tank du secteur 4 ce qui mène à un débordement. S'il s'agit ici d'une modification subtile du fonctionnement de SWaT, d'autres attaques sont, elles, effectuées de manière arbitraire sur un élément choisi au hasard.

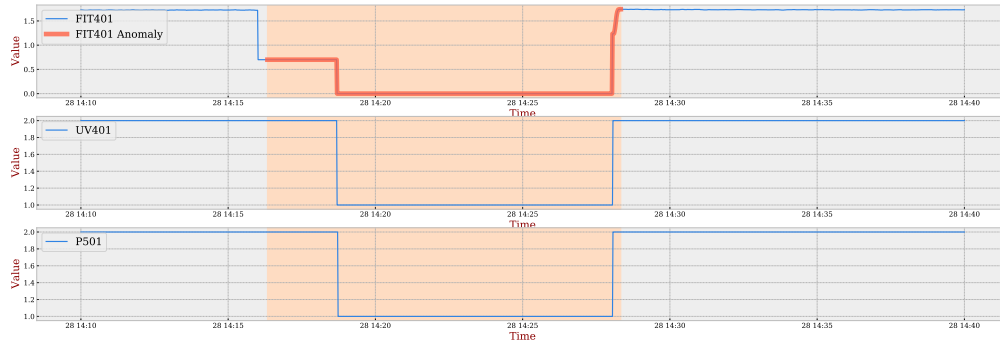


FIGURE 3.2 Scénario d'attaque n°8 sur SWaT : en rouge, les données falsifiées de FIT401

3.1.2 Pré-traitements

Le pré-traitement des données est une étape essentielle de nos expériences. Tout d'abord, les jeux de données de détection d'intrusion peuvent contenir des informations redondantes, des valeurs manquantes ou des caractéristiques inutiles qui peuvent affecter négativement les performances de nos modèles. En effectuant des étapes de pré-traitement, nous pouvons nettoyer et préparer les données de manière à ce qu'elles soient adaptées à l'apprentissage de nos

agents DRL. De plus, certaines méthodes de RL sont sensibles à l'échelle des caractéristiques. La normalisation des données est donc importante pour s'assurer que toutes les caractéristiques ont une échelle comparable, ce qui peut améliorer la convergence de l'apprentissage et la stabilité du modèle.

Afin de faciliter l'apprentissage des modèles et d'éviter tout biais dans la prédiction, nous développons donc ici les principaux traitements appliqués aux données que sont la normalisation, l'encodage *one-hot* ainsi que la gestion des valeurs manquantes.

Normalisation des données

La normalisation est une étape cruciale du pré-traitement des données. Elle vise à mettre toutes les variables au même ordre de grandeur, ce qui permet d'éviter tout biais causé par des écarts de magnitude entre les différents attributs et facilite l'apprentissage des modèles. Plusieurs critères de normalisation peuvent être utilisés, les deux plus connues étant la standardisation et la mise à l'échelle.

La standardisation consiste à s'assurer que toutes les colonnes du jeu de données deviennent des variables centrées réduites : moyenne centrée à zéro, et écart-type fixé à 1. Pour cela, la transformation appliquée à la valeur X est une simple soustraction et division telle que décrite dans l'Équation 3.1, avec μ la moyenne initiale de la colonne, et σ son écart-type. On peut cependant noter que ce type de normalisation s'applique surtout pour les variables à distribution normale, ce qui n'est pas le cas de la plupart des colonnes des jeux de données envisagés.

$$X' = \frac{X - \mu}{\sigma}. \quad (3.1)$$

Une autre méthode permet de mettre à l'échelle les valeurs étudiées sans pour autant modifier leur distribution. Ainsi, la normalisation *min-max*, ou normalisation à l'unité, permet de s'assurer que toutes les données de la colonne sélectionnée se situent entre 0 et 1, sans plus d'information sur la moyenne ni l'écart-type. La transformation appliquée est décrite dans l'Équation 3.2, avec X_{min} et X_{max} respectivement le minimum et le maximum de la colonne.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}. \quad (3.2)$$

Nous avons plutôt privilégié la normalisation *max-min* dans la suite des expériences, afin d'avoir une borne fixe de nos colonnes, ce qui permet de mieux cadrer l'entraînement des agents RL. Ce traitement s'applique aux colonnes de données continues.

Encodage *one-hot*

Pour qu'un réseau de neurones puissent traiter les valeurs d'une colonne catégorique, celle-ci doit également faire l'objet d'un pré-traitement. En effet, les modèles ne prennent pas en entrée les chaînes de caractères, tandis que représenter chaque catégorie par un nombre entier risque de biaiser la prédiction, la valeur de ces entiers étant purement arbitraire.

Le codage *one-hot* est donc une technique utilisée pour représenter les variables catégorielles sous forme de vecteurs binaires. Chaque valeur unique de la variable est convertie en une nouvelle colonne binaire. Cette technique permet donc de traiter les types de protocole, adresses IP source et destination, etc. Le codage *one-hot* permet de capturer les relations entre les différentes catégories sans imposer d'ordre ou de hiérarchie artificielle.

Gestion des valeurs manquantes

Enfin, les jeux de données dont les valeurs n'ont pas déjà été retravaillées, comme c'est le cas d'AWID, peuvent contenir de nombreuses valeurs manquantes. Plutôt que d'essayer de remplir ces cases vides, au risque encore une fois d'introduire un biais, nous avons préféré supprimer les colonnes contenant plus de 50% de valeurs manquantes, puis supprimer également les observations contenant plusieurs valeurs manquantes dans la ligne. Bien que cela réduise fortement la taille du jeu de données, cette méthode permet de s'assurer de l'intégrité des données étudiées.

3.2 Implémentation

Après avoir décrit les jeux de données utilisés et mis en évidence les différentes étapes de pré-traitement appliquées, nous pouvons nous intéresser à l'implémentation des modèles de détection d'intrusion : dans cette section, nous décrivons donc plus en détail les algorithmes RL choisis pour la détection d'intrusion, ainsi que les bibliothèques utilisées pour leur implémentation et la mise en place des attaques adverses.

3.2.1 Algorithmes

Pour entraîner un SDI et mener les expériences voulues, nous avons sélectionné deux algorithmes RL très utilisés dans le domaine : DQN et PPO, que nous avons combinés avec une méthode de sélection adverse pour améliorer les performances des agents. Nous développons ici les détails des techniques utilisées.

Motivation

Le choix des algorithmes DQN (Deep Q-Network) et PPO (Proximal Policy Optimization) pour notre recherche est motivé par leurs performances éprouvées et leur adaptabilité aux problèmes complexes tels que la détection d'intrusion.

DQN est un algorithme de Deep RL qui s'est révélé très performant dans diverses tâches d'apprentissage par renforcement. Grâce à son architecture profonde, DQN peut apprendre des représentations complexes des données, ce qui le rend potentiellement efficace pour la détection d'intrusion où les données peuvent être hautement structurées et hétérogènes.

D'autre part, PPO est un algorithme de politique de proximité qui est réputé pour sa stabilité et sa capacité à apprendre des politiques de manière fiable. Il utilise une approche de gradient basée sur une limite de proximité pour garantir des mises à jour de politique sûres et stables.

En combinant ces deux algorithmes, nous souhaitons explorer différentes approches pour la détection d'intrusion par Deep RL. DQN peut être utilisé pour les environnements discrets tandis que PPO peut être appliqué pour les environnements continus, nous permettant ainsi d'aborder une variété de problèmes de détection d'intrusion dans des contextes différents.

En somme, le choix de DQN et PPO repose sur leur capacité à apprendre des politiques efficaces dans des environnements complexes, ainsi que leur adaptabilité à différents types de tâches et de problèmes liés à la détection d'intrusion par RL. Ces algorithmes offrent une combinaison prometteuse pour notre recherche, nous permettant d'explorer différentes approches et de mieux comprendre leur impact sur la détection d'intrusion dans des scénarios réels.

Double Deep Q-Network (DDQN)

L'algorithme DQN est l'un des modèles les plus intuitifs de RL : avec un réseau de neurones profond permettant d'estimer la fonction de valeurs Q, c'est-à-dire le gain attendu pour chaque état, l'algorithme tire parti de l'Équation de Bellman (3.3) pour définir la fonction de perte à optimiser.

$$Q^*(S_t, A_t) = R_{t+1} + \gamma \max_a (Q(S_{t+1}, a)). \quad (3.3)$$

Ainsi, lors de l'entraînement, les poids du réseau Q sont mis à jour pour minimiser la fonction L^{DQN} , qui dépend de la fonction de perte L choisie (ReLU, sigmoïde, etc.).

$$L_t^{DQN} = L(R_{t+1} + \gamma \max_a (Q(S_{t+1}, a)) - Q(S_t, A_t)) \quad (3.4)$$

L'une des particularités de DQN est l'utilisation d'une technique appelée expérience de relecture avec une mémoire de relecture. Cette technique consiste à stocker les transitions passées (état, action, récompense, nouvel état) dans une mémoire tampon, puis à échantillonner aléatoirement un lot d'expériences pour l'apprentissage. Cela permet d'éviter la corrélation temporelle entre les transitions et d'améliorer la stabilité de l'apprentissage.

DDQN est une extension de l'algorithme DQN (*Deep Q-Network*) qui a été proposée pour améliorer la stabilité et la convergence de l'apprentissage par renforcement [38]. Il s'agit simplement d'utiliser deux réseaux de neurones au lieu d'un : le premier, dit réseau principal, est utilisé pour prendre les décisions d'action et mis à jour à chaque pas de temps. Le second, dit réseau cible, est une copie du premier qui n'est mise à jour que rarement et qui sert à approximer la valeur Q à l'état suivant. Ainsi, la fonction à optimiser dans le cas de DDQN serait plutôt telle que définie Équation 3.5

$$L_t^{DDQN} = L(R_{t+1} + \gamma \max_a (Q^{cible}(S_{t+1}, a)) - Q^{principal}(S_t, A_t)) \quad (3.5)$$

Le fonctionnement détaillé de DDQN est développé dans l'Algorithme 1.

Algorithm 1: Double Deep Q-Learning

Initialiser le réseau principal Q_θ , le réseau cible $Q_{\theta'}$, l'état initial s_t et la mémoire D .

Initialiser $seuil$, max_step et $freq$.

$Q_{\theta'} \leftarrow copie(Q_\theta)$

Tant que $|D| < seuil$

 Choisir $a_t = \arg\max(Q_\theta(s_t))$

 Exécuter a_t , observer r_t, s_{t+1}

 Ajouter (s_t, a_t, r_t, s_{t+1}) à la mémoire D

$s_t \leftarrow s_{t+1}$

Pour $0 < i < max_step$:

 Échantillonner $e_t = (s_t, a_t, r_t, s_{t+1})$ dans D

 Calculer la valeur cible :

$Q^* = r_t + \gamma \max_a (Q_\theta(s_t, a))$

 Appliquer la descente de gradient sur $L(Q^*, Q_\theta(s_t))$

Si $i \% freq = 0$:

$Q_{\theta'} \leftarrow copie(Q_\theta)$

Proximal Policy Optimization (PPO)

L'algorithme PPO est le plus couramment utilisé en RL, réputé pour sa stabilité et son efficacité dans l'apprentissage de politiques complexes [41]. Cette méthode est cette fois-ci basée sur la politique, c'est-à-dire que le réseau de neurones cherche à estimer la fonction π . Cependant, elle inclut également une estimation de la fonction de valeur V , ce qui la rend plus stable que les méthodes purement concentrées sur la politique.

Comme dans le cas de DQN, PPO fonctionne avec un tampon de mémoire pour éviter la variance et le biais induit par les aléas de trajectoire. Ensuite, pour optimiser la politique, PPO utilise cet historique pour calculer les gains estimés pour chaque couple état-action, puis les ratios de probabilité entre la nouvelle et l'ancienne politique. Ensuite, la fonction de perte utilisée pour la descente de gradient combine deux termes : le terme de ratio de probabilité, qui encourage les mises à jour de politique dans la bonne direction, et le terme de clip, qui limite les changements de politique à un certain seuil pour assurer la stabilité de l'apprentissage.

En résumé, l'algorithme PPO est une méthode d'optimisation puissante pour entraîner des politiques d'agents dans des environnements de RL. Il utilise une fonction de perte spécifique pour assurer la stabilité et l'efficacité de l'apprentissage. En exploitant les données du tampon de mémoire, et en mettant à jour la politique itérativement, PPO permet aux agents de développer des politiques de haute qualité pour prendre des décisions adaptatives et performantes dans leur environnement.

La contrepartie des performances de PPO est sa complexité, c'est pourquoi nous ne détaillerons pas toutes les étapes de sa mise en place. Si coder à la main cet algorithme serait fastidieux et probablement sous-optimal, il existe en revanche plusieurs bibliothèques Python qui sont d'une aide précieuse dans le domaine.

3.2.2 Bibliothèques et outils

Pour mettre en place les expériences voulues en Python, nous utilisons ainsi plusieurs bibliothèques logicielles couramment utilisées dans le domaine du RL et de l'apprentissage automatique adverse. Ces bibliothèques fournissent des outils et des fonctionnalités essentielles pour la mise en œuvre et l'évaluation de nos algorithmes.

Tout d'abord, la construction de réseaux de neurones complexes et performants est largement facilitée par l'emploi de la bibliothèque d'apprentissage automatique *open source* Pytorch [96]. Sa grande flexibilité et facilité d'utilisation en font un choix populaire pour l'implémentation d'architectures de réseaux de neurones, et elle fait désormais partie de la norme dans le

domaine. PyTorch fournit une interface intuitive pour la construction et l’entraînement de modèles, ainsi que des fonctionnalités avancées telles que le calcul automatique des gradients et la prise en charge des calculs sur GPU. Nous utilisons PyTorch pour mettre en œuvre les réseaux imbriqués dans nos modèles d’apprentissage par renforcement profond.

Stable Baselines3 (SB3) [97] est une bibliothèque de DRL basée sur PyTorch. Elle offre une implémentation simple et efficace de divers algorithmes de RL, y compris PPO et DQN. SB3 facilite l’entraînement des agents et fournit des outils pour l’évaluation et la visualisation des résultats. Nous utilisons SB3 pour implémenter nos algorithmes d’apprentissage par renforcement, en exploitant sa facilité d’utilisation et sa compatibilité avec les environnements de RL standards.

Après avoir construit des modèles RL performants, nous comparons leur réaction face au comportement adverse à l’aide d’*Adversarial Robustness Toolbox* (ART) [98], une bibliothèque dédiée à la sécurité des modèles d’apprentissage automatique. Cette bibliothèque fournit des outils pour générer, évaluer et atténuer les exemples adverses, ainsi que pour analyser la robustesse des modèles face aux attaques. ART offre des implémentations de différentes attaques et défenses adverses, cependant elle est centrée sur les méthodes supervisées, et ne contient donc pas de méthode spécifique au RL. C’est pourquoi nous avons créé notre propre librairie *open source*, AdvRL, qui contient plusieurs approches adverses spécifiquement conçues pour les algorithmes de RL.

Enfin, Weights & Biases (W&B) est une plateforme de suivi et de visualisation des expériences en apprentissage automatique. Elle permet de stocker et d’analyser les résultats expérimentaux, de comparer différentes configurations et de suivre la progression de l’apprentissage. Nous utilisons donc W&B pour enregistrer les métriques, les hyperparamètres et les courbes d’apprentissage de nos modèles.

En résumé, les bibliothèques PyTorch, Stable Baselines3, Adversarial Robustness Toolbox et W&B jouent un rôle crucial dans la mise en œuvre, l’évaluation et l’analyse de nos modèles d’apprentissage par renforcement et de détection d’intrusion. Notre propre bibliothèque, AdvRL, est basée sur PyTorch et SB3 en complément d’ART pour permettre aux futurs travaux de s’appuyer sur des outils communs et libres d’accès.

3.3 Métriques d’évaluation

Dans cette section, nous discutons des métriques que nous utilisons pour évaluer les performances de nos modèles de détection d’intrusion, ainsi que leur robustesse face aux exemples adverses. Nous expliquons en quoi le choix de ces métriques spécifiques est pertinent pour

notre cas d’usage.

3.3.1 Performance des modèles

Dans un premier temps, nous nous intéressons aux performances des différents modèles de détection d’intrusions que nous avons construits, afin de déterminer quels paramètres sont les mieux adaptés et quels algorithmes nous pouvons retenir. Il existe de nombreuses manières d’évaluer ces performances, cependant les métriques retenues doivent être adaptées à notre cas d’utilisation particulier.

Ainsi, bien que nos modèles appartiennent au domaine du RL, les métriques classiques de RL ne sont pas les plus appropriées : la notion d’épisode étant purement arbitraire et leur durée fixée dans les expériences, nous ne nous appuierons pas sur le gain des agents pour évaluer leur performance. En gardant à l’esprit le but initial de la détection d’intrusion, qui est de classer les observations récoltées, nous appliquons plutôt les métriques usuelles dans le cas de modèles supervisés.

Nous avons donc sélectionné trois métriques pertinentes et significatives pour la détection d’intrusion par RL :

- Le Taux de Faux Positifs (TFP) représente la proportion de faux positifs parmi toutes les instances négatives. Dans notre cas, cela revient à la proportion d’instances normales signalées comme des attaques, donc au taux de fausses alertes.
- Le Taux de Faux Négatifs (TFN) représente, à l’inverse, la proportion de faux négatifs parmi toutes les instances positives, i.e. la proportion d’attaques manquées par rapport au nombre total d’attaques dans le jeu de données. Il est donc particulièrement intéressant dans le cas de la détection d’intrusion, et doit être minimal.
- Enfin, le score F1 est l’une des métriques les plus complètes en classification. En effet, il s’agit d’une moyenne harmonique entre la précision et le rappel, c’est-à-dire entre la proportion d’attaques signalées parmi les attaques, et la proportion d’alertes correctes parmi les alertes. Ainsi, un score F1 élevé indique que le modèle est capable d’obtenir un bon équilibre entre la détection précise des intrusions et la minimisation des erreurs de classification.

Ces trois métriques nous permettront d’évaluer la performance de nos modèles de SDI par RL, ainsi que d’analyser leurs forces et leurs faiblesses. Elles nous aideront à prendre des décisions éclairées pour améliorer nos modèles et à les comparer avec les approches existantes dans la littérature.

3.3.2 Évaluation de la robustesse

Dans un second temps, nous devons définir les critères d'évaluation de la robustesse de nos modèles de détection d'intrusion, afin de comparer leur capacité à résister aux attaques adverses et à maintenir leur performance dans des environnements hostiles.

Les méthodes d'exemples adverses envisagées ont pour objectif de faire passer une intrusion pour un comportement normal. Ainsi, une attaque adverse réussie augmente significativement le taux de faux négatifs, i.e. d'attaques manquées, tout en diminuant le taux de faux positifs, ou fausses alertes, car la plupart des instances seront signalées comme normales. Pour mesurer l'impact d'une attaque adverse, il est donc pertinent de s'intéresser à l'évolution du TFN ainsi que du score F1, mais la diminution du TFP n'est pas représentative de l'effet des attaques.

Les critères retenus pour l'évaluation de la robustesse des modèles sont donc :

- L'évolution du TFN : calculer le taux d'augmentation d'intrusions manquées par le modèle permet de quantifier directement l'impact des attaques sur les performances de détection.
- L'évolution du score F1 : le taux de diminution du score F1 est une manière indirecte mais plus globale d'évaluer la résistance du modèle face aux exemples adverses, en prenant en compte l'ensemble des métriques pertinentes.

De plus, nous garderons en tête l'importance de la taille des perturbations dans cette étude : une perturbation plus discrète aura certes moins d'impact mais peut s'avérer tout aussi dangereuse, car plus difficile à mitiger.

Dans ce chapitre, nous avons donc abordé les aspects clé de notre approche, de la préparation des données à l'implémentation des algorithmes, en passant par l'évaluation de nos résultats. À partir de cette méthodologie rigoureuse, nous pouvons donc présenter les solutions développées au cours de notre étude.

CHAPITRE 4 VULÉRABILITÉ DES SYSTÈMES DE DÉTECTION D’INTRUSION PAR RL FACE AUX EXEMPLES ADVERSES

Ce chapitre se concentre sur l’étude de la vulnérabilité des SDI basés sur l’apprentissage par renforcement face aux attaques adverses. Nous avons introduit les concepts et les définitions liés à la détection d’intrusion, au RL et aux attaques adverses dans les sections précédentes. Dans cette partie, nous approfondirons cette problématique en construisant un premier modèle de SDI par RL et en évaluant sa résistance face à des attaques adverses soigneusement sélectionnées.

Dans un premier temps, nous présenterons la construction de notre modèle de SDI basé sur le RL, et en particulier l’adaptation d’algorithmes de RL classiques à la tâche de classification des intrusions. Nous détaillerons également la mise en place d’un environnement adverse, qui permet d’améliorer les performances de notre modèle en l’exposant à des situations plus complexes et réalistes. De plus, nous discuterons des choix d’hyperparamètres essentiels pour garantir une convergence efficace du modèle.

Ensuite, nous analyserons les résultats de l’entraînement de notre modèle de SDI par RL et essayerons de l’élargir aux données cyber-physiques. Nous évaluerons les performances de détection d’intrusion à l’aide des métriques introduites dans la section méthodologie, ce qui nous permettra d’analyser les forces et les limites de notre modèle initial.

Enfin, dans le but d’évaluer la vulnérabilité de notre modèle RL aux attaques adverses, nous mettrons en place des attaques adverses ciblées et évaluerons leur impact sur les performances de l’un de nos meilleurs modèles. Nous sélectionnerons des méthodes d’attaques adverses appropriées pour les SDI basés sur le RL, telles que FGSM et PGD, et conclurons sur la robustesse du RL profond en détection d’intrusion.

En conclusion, ce chapitre fournira une étude approfondie de la vulnérabilité des SDI par RL face aux attaques adverses. Nous mettrons en évidence les résultats obtenus en termes de performances de détection et de résistance aux attaques, soulignant l’importance de développer des mécanismes de défense robustes pour les SDI basés sur le RL. Ces résultats serviront de base pour la discussion des implications pratiques et ouvriront la voie à de futures recherches visant à renforcer la sécurité des systèmes de détection d’intrusion dans des environnements complexes et dynamiques.

4.1 Construction du modèle

4.1.1 Un agent RL pour la classification

Cadre théorique

Afin de construire un modèle de DRL performant dans le contexte de la classification d'instances réseau, nous avons adapté le cadre RL classique à notre problème, en s'inspirant des travaux précédemment réalisés par Lopez-Martin *et al.* [48]. Dans cette optique, nous avons tout d'abord défini le MDP $\Omega = (S, A, P, R, \gamma)$ à étudier pour le cas de données réseau (voir Section 2.1.2) :

- les états S sont les observations réseau, récoltées au sein du jeu de données puis pré-traitées comme décrit dans la Section 3.1.2,
- en guise d'actions A , l'agent décide d'une étiquette (ou catégorie) pour classer l'observation,
- la fonction de transition P suit une loi uniforme parmi toutes les observations du jeu de données - indépendamment de l'action choisie,
- la récompense est égale à 1 si l'étiquette choisie par l'agent correspond à la classe réelle de l'observation et -1 sinon, et
- le facteur γ , qui représente le lien entre deux états consécutifs, est réduit à 0.

Dans cette configuration l'agent agit donc comme un classificateur, et est entraîné comme tel : la Figure 2.2 illustre ce parallèle entre RL et classification. Pourquoi alors choisir d'utiliser le RL, dont l'entraînement peut être long et converger plus difficilement, plutôt qu'un modèle supervisé ?

Tout d'abord, le RL offre une grande flexibilité et adaptabilité aux environnements dynamiques et changeants. Les SDI sont confrontés à des attaques de plus en plus sophistiquées et évolutives, nécessitant une capacité d'adaptation en temps-réel. L'apprentissage par renforcement permet à notre modèle de s'ajuster en continu aux nouvelles formes d'attaques et de modifier sa politique de décision en fonction de l'évolution des menaces. Un autre avantage du RL est sa capacité à explorer de manière autonome l'espace des actions et à découvrir des stratégies de détection efficaces. Contrairement aux algorithmes supervisés, le RL explore différentes actions possibles et évalue leur impact sur les récompenses obtenues. Cela permet au modèle de découvrir des comportements de détection innovants et de s'adapter aux scénarios d'attaques encore jamais rencontrés. Enfin, le RL offre la possibilité de prendre en compte les contraintes de coût et d'efficacité dans la détection d'intrusion. Dans de nombreux systèmes, les ressources sont limitées et la détection d'intrusion doit être effectuée de manière

efficace. Le RL permet d’optimiser la politique de décision en prenant en compte les coûts de détection et les ressources disponibles, permettant ainsi de trouver un équilibre entre la performance de détection et l’efficacité opérationnelle.

En somme, l’utilisation du RL plutôt qu’un algorithme supervisé offre des avantages significatifs en termes d’adaptabilité aux environnements dynamiques, de découverte de nouvelles stratégies de détection et de prise en compte des contraintes de coût. Ces aspects font du RL un choix pertinent et prometteur pour la construction de modèles de SDI robustes et performants face aux attaques adverses.

Premiers résultats

Intéressons-nous tout d’abord aux performances de ce premier modèle. Si l’entraînement s’avère satisfaisant, nous pourrions ensuite affiner certains choix pour améliorer les performances de détection.

Nous avons ainsi construit dans un premier temps deux agents RL à l’aide du cadre défini ci-dessus. Les algorithmes RL choisis sont DQN et PPO, décrits précédemment dans la Section 3.2.1, que nous avons implémentés à l’aide de la bibliothèque Python Stable-Baselines3. Chaque réseau interne est composé de 2 couches de 64 neurones : ce choix est pour l’instant arbitraire, et sera remis en question par la suite.

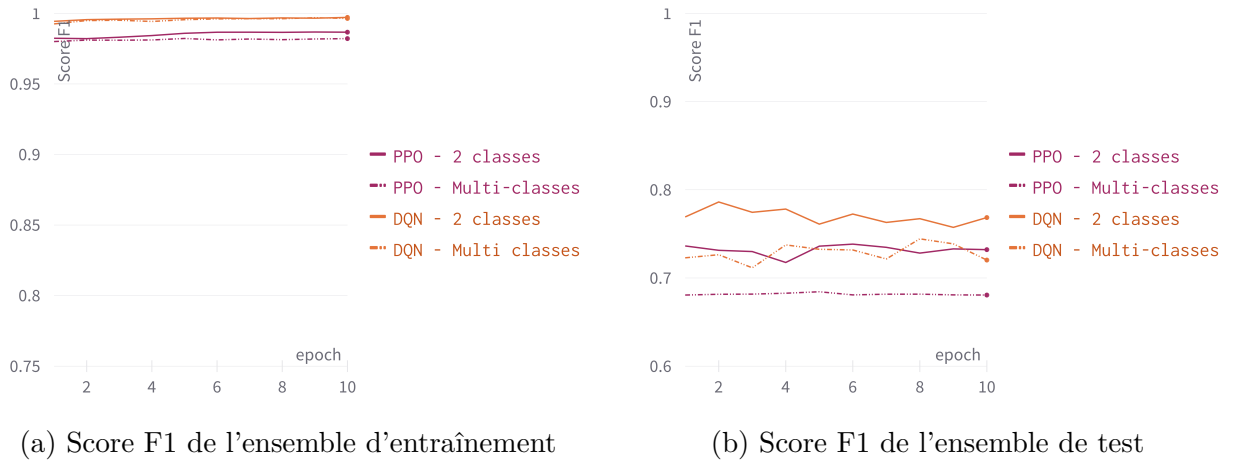


FIGURE 4.1 Performance des agents PPO et DQN à 2 couches et 64 unités au cours de l’entraînement sur NSL-KDD

La Figure 4.1 montre l’évolution du score F1 de ces agents au cours de leur apprentissage, évalué sur l’ensemble d’entraînement et sur l’ensemble de test de NSL-KDD. Nous avons testé deux configurations : dans la première (lignes pleines) nous avons choisi de séparer les

instances du jeu de données en deux classes, "Attaques" et "Normal", plutôt que d'utiliser les sous-classes d'attaques développées dans la Section 2.1.1. Dans la seconde (lignes pointillées), nous avons laissé les différentes classes du jeu de données telles quelles.

Nous pouvons remarquer, pour ces deux jeux de données, que les performances des agents sur l'ensemble d'entraînement sont très satisfaisantes, avec le modèle PPO légèrement dominé par le modèle DQN. Ainsi, à la fin de l'entraînement, les deux agents DQN ont un score F1 de 0.997 et 0.996, tandis que les deux agents PPO ont un score F1 de 0.986 et 0.982. Cependant, il ne suffit pas d'obtenir de bonnes performances sur l'ensemble d'entraînement, car nous souhaitons que les modèles se généralisent aux attaques qu'ils n'ont encore jamais rencontrées. C'est ainsi que les métriques de l'ensemble de test revêtent une importance particulière dans l'évaluation des modèles.

On observe dans le cas de l'ensemble de test que le score F1 est bien inférieur à celui de l'ensemble d'entraînement : 0.768 pour DQN, et 0.732 pour PPO (2 classes) à la fin de l'apprentissage. Si cette baisse est attendue étant donné le caractère nouveau et inconnu des instances de cet ensemble, ces résultats ne sont pas complètement satisfaisants, et restent relativement faibles pour le jeu de données NSL-KDD. De plus, il apparaît que les agents multi-classe peinent à se généraliser à l'ensemble de test, avec un score de 0.720 et 0.680 respectivement pour DQN et PPO (multi-classe). Cela est probablement dû à la nature déséquilibrée du jeu de données, certaines classes d'attaques étant très faiblement représentées.

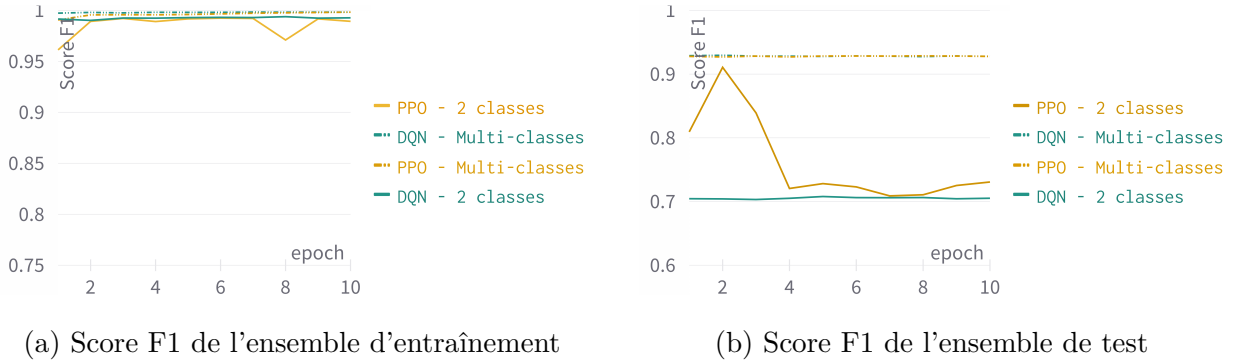


FIGURE 4.2 Performance des agents PPO et DQN à deux couches et 64 unités au cours de l'entraînement sur AWID

De la même manière, la Figure 4.2 présente les résultats de l'entraînement de ces agents PPO et DQN sur le jeu de données AWID, avec encore une fois le cas "2 classes" et le cas "multi-classe". On observe là aussi la réussite de l'apprentissage sur l'ensemble de données d'entraînement (score F1 entre 0.992 et 0.998), ainsi qu'une baisse des performances sur

l'ensemble de test (score F1 entre 0.705 et 0.928).

Si la différence de performance entre PPO et DQN semble moins flagrante que dans le cas de NSL-KDD, on remarque en revanche que le score F1 des modèles multi-classe sur l'ensemble de test est largement supérieur (autour de 0.93) que celui des modèles binaires (environ 0.70). Cela pourrait mener à penser que les modèles multi-classe sont plus performants que les modèles binaires. Il faut cependant garder à l'esprit que dans le cas de données déséquilibrées, comme ici, le score F1 n'est pas toujours représentatif de la qualité du modèle : il peut rester très élevé même si la classe minoritaire n'est jamais détectée. C'est pourquoi nous pouvons nous intéresser au taux de faux positifs, c'est-à-dire au pourcentage d'attaques non détectées par le modèle.

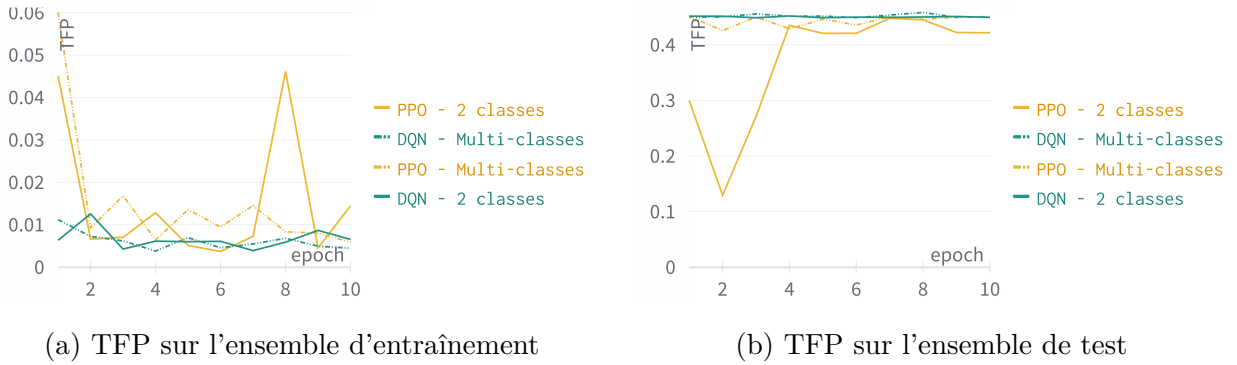


FIGURE 4.3 Évolution du taux de faux positifs des agents PPO et DQN au cours de l'entraînement sur AWID

La Figure 4.3 montre l'évolution du TFP des agents en questions au cours de l'entraînement sur AWID. On remarque qu'il est particulièrement élevé sur l'ensemble de test, que ce soit pour les modèles binaires ou multi-classe : en effet, avec un TFP entre 0.42 et 0.46, cela signifie que plus de 40% des attaques ne sont pas détectées par le modèle, et ce malgré un très bon score F1.

En conclusion, le cadre RL envisagé permet d'obtenir des performances de détection d'intrusion prometteuses, que ce soit dans le cas de NSL-KDD ou dans celui d'AWID. Cependant, les modèles restent limités et peinent à identifier les nouvelles attaques, en particulier dans le cas de jeux de données très déséquilibrés comme AWID et de détection plus fine comme la multi-classe.

4.1.2 Environnement adverse

Bien que les premières expériences aient confirmé la pertinence de l'utilisation du RL pour la détection d'intrusion, il est clair que les performances de nos agents peuvent être améliorées. Nous avons identifié un défi majeur, à savoir le déséquilibre du jeu de données, qui limite la capacité de généralisation des modèles. En effet, lors de l'apprentissage, les agents sont confrontés à un nombre limité d'attaques, ce qui rend d'autant plus difficile l'identification de nouvelles attaques. Ainsi, il est impératif de trouver des solutions pour surmonter ce défi et rendre nos modèles plus performants et capables de détecter efficacement une plus large gamme d'attaques. En abordant cette problématique, nous visons à renforcer la performance et la fiabilité de nos agents de détection d'intrusion afin de mieux prévenir les menaces potentielles dans les systèmes informatiques.

Caminero *et al.* [49] ont introduit une approche novatrice appelée "environnement adverse" - à ne pas confondre cette notion avec les attaques adverses traditionnelles. Dans ce contexte, l'environnement adverse fait référence à une modification contrôlée de l'environnement, plus spécifiquement à la sélection des instances du jeu de données. En reprenant notre MDP précédemment défini, la fonction de transition P était une loi uniforme permettant de sélectionner une instance au hasard parmi l'ensemble des instances disponibles. Avec l'environnement adverse, cette fonction de transition est remplacée par un agent RL dont les actions déterminent la classe de la prochaine instance à sélectionner, et dont les récompenses sont opposées à celles de l'agent principal. L'agent adverse, ou agent sélecteur, est donc entraîné à choisir la classe la plus difficile pour l'agent principal : si l'agent principal se trompe lors de sa prédiction, le sélecteur reçoit une récompense positive, et dans le cas contraire une récompense négative. Cette approche permet de créer un environnement plus complexe et dynamique, et d'améliorer les capacités de détection d'intrusion en rendant ajoutant une difficulté lors de l'apprentissage.

Nous avons donc lancé les mêmes expériences que dans la section précédente en modifiant simplement l'environnement, et nous sommes intéressés à la différence de performance entre l'environnement naïf et l'environnement adverse. Cette fois-ci, nous étudierons simultanément le score F1 et le taux de faux positifs sur l'ensemble de test : en effet, ce sont les métriques les plus représentatives des progrès de nos modèles.

Dans la Figure 4.4, une nette amélioration du score F1 est observée lors de l'utilisation de l'environnement adverse pour NSL-KDD, tant pour le modèle PPO que pour le modèle DQN et que ce soit en mode binaire ou en mode multi-classe. De plus, les modèles utilisant l'environnement adverse présentent des taux de faux positifs considérablement plus faibles (maximum 0,32) par rapport à nos premiers modèles (compris entre 0,35 et 0,39). Cette constatation dé-

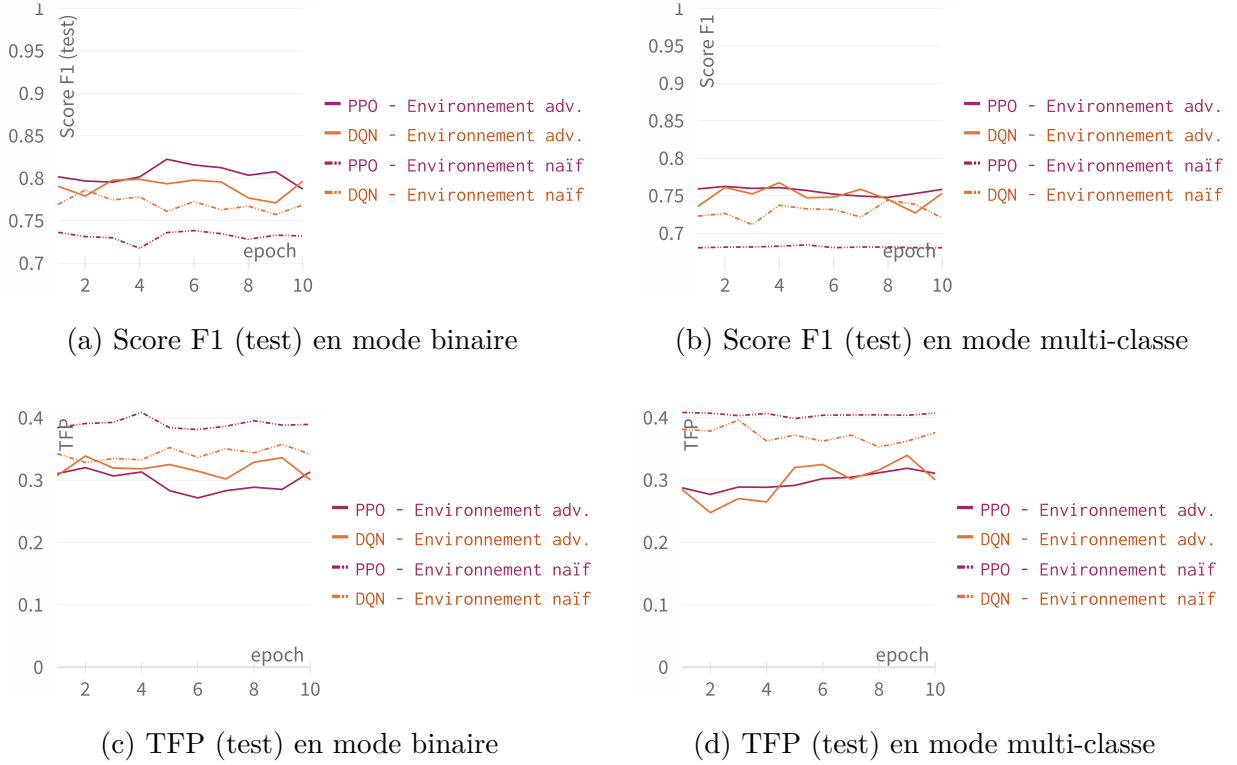


FIGURE 4.4 Évolution du score F1 et du TFP des agents PPO et DQN sur NSL-KDD, avec et sans l'environnement adverse, évalué sur l'ensemble de test

montre l'efficacité de l'environnement adverse pour améliorer la performance des modèles de détection d'intrusion, en réduisant les erreurs de classification et en augmentant la précision globale des prédictions.

Il est également intéressant de noter que l'utilisation de l'environnement adverse permet au modèle PPO de surpasser le modèle DQN, alors que dans un environnement naïf, le modèle PPO était légèrement moins performant. Cela suggère que l'environnement adverse favorise la capacité du modèle PPO à apprendre des stratégies plus robustes et adaptatives face aux attaques.

La Figure 4.5 montre les expériences avec et sans environnement adverse menées sur le jeu de données AWID. Deux observations principales peuvent être déduites de ces courbes : d'une part, dans le cas binaire, l'environnement adverse permet une nette amélioration des performances, et ce en particulier pour l'algorithme PPO (le taux de faux positifs descend à 0.15 avec l'environnement adverse, au lieu de 0.42 sans). D'autre part, dans le cas multi-classe, les différents modes (environnement adverse ou non, choix de l'algorithme) ne semblent pas impacter significativement les performances. Plus encore, le taux particulièrement élevé

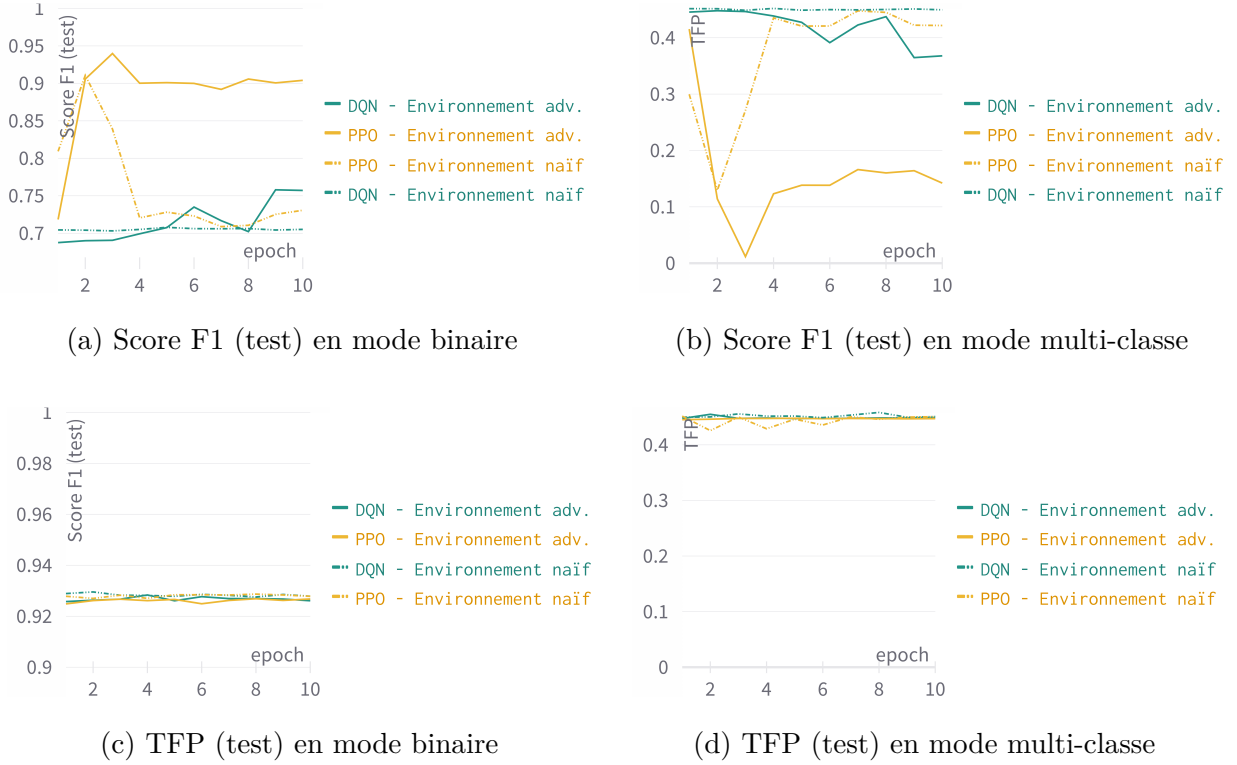


FIGURE 4.5 Évolution du score F1 et du TFP des agents PPO et DQN sur AWID, avec et sans l’environnement adverse, évalué sur l’ensemble de test

de faux positifs (0.45) mène à penser que le modèle peine à détecter les attaques dans une configuration multi-classe, quels que soient les choix d’entraînement. La configuration binaire serait donc mieux adaptée au jeu de données AWID.

En conclusion, l’utilisation de l’environnement adverse s’est révélée être une approche prometteuse pour améliorer la performance des modèles de détection d’intrusion basés sur le RL. Les expériences menées ont montré que l’introduction d’un tel environnement a permis d’obtenir des résultats significativement meilleurs en termes de score F1 et de taux de faux positifs. De plus, l’environnement adverse a favorisé une amélioration des performances du modèle PPO par rapport au modèle DQN. Dans la suite de nos expériences, nous utiliserons donc systématiquement l’environnement adverse pour entraîner nos modèles.

4.1.3 Hyperparamètres

Dans notre étude, nous nous intéressons particulièrement à l’impact de certains hyperparamètres sur l’entraînement, puis sur la vulnérabilité des modèles. Plus précisément, nous

analysons les variations de performances avec :

- **le nombre de classes** - binaire / multi-classe,
- **la profondeur du modèle** - 1, 2 ou 4 couches cachées,
- **la largeur du modèle** - 32, 64 ou 128 unités par couche, et
- **l’algorithme RL** - DQN ou PPO.

Le Tableau 4.1 présente les performances des modèles DQN et PPO avec différentes architectures dans un mode binaire, en utilisant les jeux de données AWID et NSL-KDD. Pour chaque ensemble de données, modèle et profondeur spécifiés, les métriques de performance sont répertoriées. Les valeurs maximales de chaque métrique sont mises en gras afin de les mettre en évidence pour une meilleure lisibilité et comparaison. Ce tableau permet d’évaluer et de comparer les performances des modèles en fonction de leurs différentes configurations.

Après une analyse approfondie du tableau, plusieurs conclusions se dégagent. Tout d’abord, en moyenne, le modèle DQN présente de meilleures performances que le modèle PPO sur le jeu de données NSL-KDD. De plus, l’ajout de couches cachées ou l’augmentation du nombre d’unités par couche ne présente pas d’impact significatif sur les performances de détection sur NSL-KDD. En effet, la configuration la plus performante est obtenue avec une seule couche cachée de 64 neurones, qui atteint un score F1 de 83,2%. Ces résultats suggèrent que la complexité accrue de l’architecture ne conduit pas nécessairement à une amélioration des performances de détection d’intrusion.

En revanche, pour les données AWID, nous observons une tendance différente, où les modèles PPO surpassent largement les modèles DQN : alors que le meilleur score F1 atteint par DQN est de 76%, les modèles PPO affichent en moyenne un score F1 d’environ 87%. Ces résultats suggèrent que les modèles PPO sont mieux adaptés à la détection d’intrusion sur le jeu de données AWID, peut être en raison de la nature plus déséquilibrée de ces données. De plus, l’architecture des modèles semble jouer un rôle crucial, avec des scores F1 plus élevés pour les modèles plus profonds, que ce soit pour DQN ou PPO.

Cette première analyse nous donne un aperçu de l’influence de l’architecture des modèles sur les performances de détection d’intrusion. Concernant le mode multi-classe, nous avons regroupé dans le Tableau 4.2 les métriques associées aux meilleures architectures, pour chaque jeu de données et chaque algorithme RL.

Il apparaît tout d’abord que les performances sur le mode multi-classe sont nettement moins bonnes que celles du mode binaire, ce qui confirme l’hypothèse d’une détection facilitée par un nombre de classes réduit. Cependant, nous ne nous plongeons pas ici dans les performances d’identification des attaques, qui pourraient justifier l’emploi du multi-classe mais ne sont

TABLEAU 4.1 Regroupement des performance des différents modèles étudiés sur AWID et NSL-KDD (mode binaire)

Paramètres				Score F1 (%)	TFP (%)	TFN (%)
Données	Modèle	Profondeur	Unités			
NSL-KDD	DQN	1	32	82.5	26.9	05.5
			64	83.2	27.0	03.3
			128	80.8	28.2	07.6
		2	32	80.4	31.1	03.3
			64	82.5	27.7	03.9
			128	80.1	29.2	07.8
		4	32	78.8	33.7	02.6
			64	76.6	36.7	02.7
			128	76.8	35.2	05.1
	PPO	1	32	77.5	34.4	03.8
			64	79.0	32.8	04.0
			128	77.5	35.0	03.7
		2	32	78.7	32.1	05.9
			64	78.5	31.8	07.2
			128	78.3	33.8	03.5
		4	32	75.7	35.8	06.9
			64	78.3	32.5	06.3
			128	78.2	32.2	07.5
AWID	DQN	1	32	67.7	44.7	01.6
			64	67.4	44.7	01.2
			128	69.0	44.7	00.1
		2	32	69.0	44.8	00.5
			64	69.6	44.7	00.4
			128	70.0	44.8	00.4
		4	32	69.5	44.8	00.5
			64	69.1	44.8	00.6
			128	76.0	35.8	00.5
	PPO	1	32	79.7	30.0	00.7
			64	93.3	01.9	01.5
			128	83.4	23.7	00.7
		2	32	96.3	03.0	00.7
			64	86.3	18.8	00.8
			128	88.8	13.4	01.0
		4	32	70.2	43.2	00.6
			64	93.4	08.9	00.5
			128	97.0	01.4	00.5

pas critiques dans le cadre de notre étude. Nous pouvons de plus observer que les meilleurs

TABLEAU 4.2 Performances des meilleurs modèles multi-classe étudiés, et hyperparamètres associés

Données	Modèle	Meilleure architecture		Score F1 (%)	TFP (%)	TFN (%)
		Profondeur	Unités			
NSL-KDD	DQN	1	32	76.0	33.4	03.6
	PPO	2	64	75.5	31.1	08.1
AWID	DQN	1	128	96.9	16.7	01.1
	PPO	4	64	94.1	37.0	00.5

modèles sur NSL-KDD n'ont que relativement peu d'unités et de couches cachées, seulement une couche de 32 neurones pour DQN et 2 couches de 64 neurones pour PPO. Dans le cas d'AWID, ce sont toutefois les modèles plus profonds qui semblent détecter le plus d'intrusions, comme c'était le cas dans le mode binaire, avec cependant une différence de performance moins nette entre PPO et DQN.

Ainsi, nous disposons d'une première analyse de l'impact de différents paramètres sur les performances des modèles, que ce soit le nombre de classes, l'algorithme RL choisi, ou bien l'architecture du réseau. Il apparaît dans l'ensemble que les agents sont plus performants en mode binaire, que DQN est mieux adapté aux données de NSL-KDD et PPO aux données d'AWID, et que l'impact de l'architecture varie selon les données envisagées et le modèle RL choisi. Nous nous ne nous sommes pour l'instant intéressés qu'aux données réseau pures, et n'avons pas encore vérifié la généralisabilité de ces résultats aux données cyber-physiques.

4.2 Extension aux données cyber-physiques

L'extension des méthodes de détection d'intrusion par RL aux données cyber-physiques constitue l'un des défis de notre projet. Alors que de nombreux travaux se sont concentrés sur l'application de ces techniques aux données issues des réseaux informatiques, l'importance croissante des systèmes cyber-physiques soulève de nouveaux défis en matière de sécurité. Les systèmes cyber-physiques tels que les infrastructures critiques, les véhicules autonomes et les systèmes industriels intègrent des composantes logicielles et matérielles interconnectées, créant ainsi des environnements propices aux attaques malveillantes. Par conséquent, il serait intéressant d'étendre les modèles de détection d'intrusion par RL pour couvrir ces scénarios spécifiques. Dans cette section, nous explorons les particularités de ces données, détaillons les méthodes et les stratégies que nous avons développées pour cette adaptation, et soulignons les défis qui persistent pour ce cas d'usage.

4.2.1 Particularité des données

Pour étendre les méthodes de détection d'intrusion par RL aux données cyber-physiques, plusieurs aspects doivent être pris en considération. Tout d'abord, ces données sont caractérisées par des interactions entre le monde physique et les systèmes informatiques. Cela signifie que nous devons prendre en compte les informations provenant de capteurs physiques, tels que les capteurs de température, de pression ou de vitesse, en plus des données réseau traditionnelles. Ces données physiques obéissent à des contraintes fondamentalement différentes de celles des données réseau, et la combinaison de ces sources de données hétérogènes est un défi pour nos modèles d'apprentissage automatique.

En outre, les données cyber-physiques peuvent présenter des caractéristiques spécifiques, telles que des modèles de comportement complexes ou des variations dynamiques et long-terme. Ces particularités ajoutent une couche de complexité supplémentaire à la détection d'intrusion, car les attaques peuvent être plus difficiles à détecter dans un environnement évolutif. Nous devons donc adapter nos modèles de RL pour prendre en compte ces variations et développer des mécanismes d'apprentissage qui permettent une détection précise et robuste dans des conditions changeantes.

Enfin, les attaques présentes dans le jeu de données SWaT touchent rarement deux fois le même capteur ou actionneur du système : cette situation rend l'entraînement des modèles de détection d'intrusion par RL plus complexe, car il est difficile pour les modèles d'apprendre efficacement à détecter les attaques lorsqu'elles sont sous-représentées dans les données d'entraînement.

Pour illustrer les particularités des données cyber-physique, la figure 4.6 montre les valeurs des capteurs et actionneurs du premier processus de la centrale de traitement d'eau au cours d'une journée de simulation. La partie orange indique les pas de temps pendant lesquels une attaque a lieu sur deux des éléments du système, LT101 et MV101. Les conséquences de cette attaque, en revanche, peuvent se répercuter sur les autres éléments du système bien après la fin de l'intrusion elle-même.

4.2.2 Adaptation du modèle

Pour construire un modèle de détection d'intrusion RL à partir de ces données, nous avons dans un premier temps dû convenir d'une séparation entre données d'entraînement et données de test. Nous avons décidé de préserver la continuité temporelle : toutes les données datées avant le 01 janvier 2016 à 00h00 font partie des données d'entraînement (7 jours de fonctionnement normal, ainsi que 2 jours sous attaque) tandis que les deux derniers jours

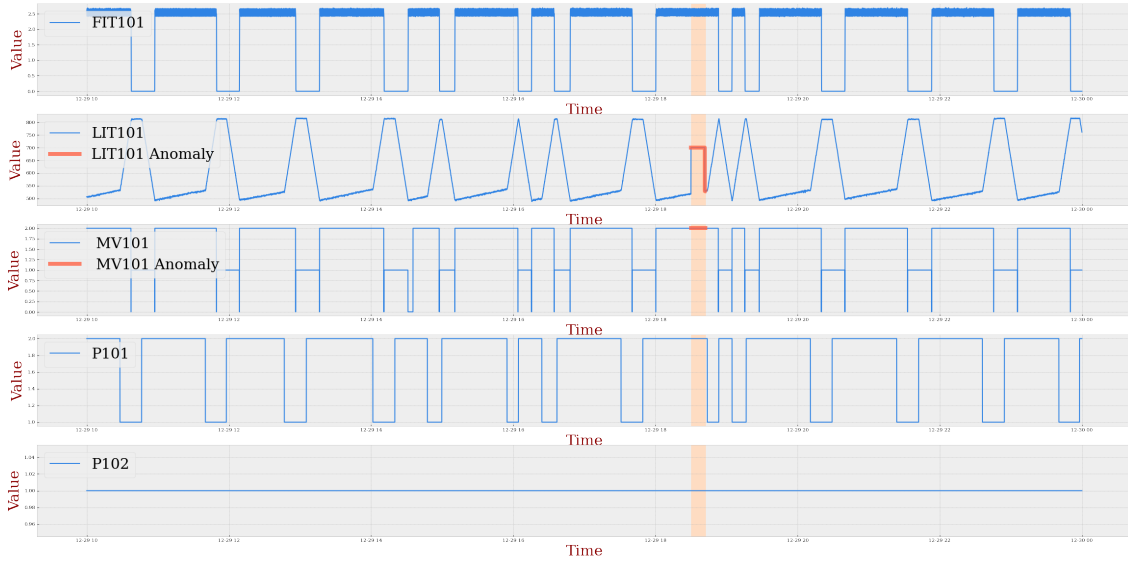


FIGURE 4.6 Valeurs des capteurs du premier processus de SWaT, le 29 décembre 2015 de 10h à 23h59. En orange, une anomalie lié à une attaque simultanée de LIT101 et MV101.

servent comme ensemble de test.

Ensuite, nous avons dans un premier temps traité chaque observation de manière individuelle et indépendante. Bien que cette approche ne tienne pas compte de la corrélation entre plusieurs observations consécutives, elle permet d'obtenir un premier modèle naïf et d'adapter directement la méthode utilisée pour les données informatiques. Le Tableau 4.3 rassemble les métriques de performance des différentes architectures pour DQN et PPO.

On observe que, bien que le score F1 puisse à première vue sembler élevé, le taux de faux positifs est également particulièrement haut. Cela signifie que la plupart des attaques sont détectées, mais qu'il y a de nombreuses fausses alertes qui bruitent le signal et brouillent la détection. La Figure 4.7, montre par exemple les valeurs des capteurs et actionneurs du premier processus de SWaT, du 01-01-2016 à 14h00 au 02-01-2016 à 00h00. Les sections qui apparaissent en orange clair correspondent aux périodes pendant lesquelles le système est attaqué, tandis que les points en rouge correspondent aux pas de temps signalés comme suspects par le modèle PPO à 4 couches et 128 unités.

Il apparaît clairement que, si 3 des 4 attaques principales sont détectées, de nombreuses autres alertes ont lieu lors du fonctionnement normal du processus. Rien ne permet donc a priori de différencier les fausses alertes des vraies attaques, et ces fausses alertes sont trop nombreuses pour pouvoir être chacune examinée par un expert métier. Nous pouvons donc en conclure que cette adaptation n'est pas satisfaisante, et ne permet pas de bonnes performances de détection.

TABLEAU 4.3 Performances des modèles RL appliqués aux données SWaT

Paramètres			Score F1 (%)	TFP (%)	TFN (%)
Modèle	Profondeur	Unités			
DQN	1	32	88.6	73.1	01.3
		64	94.2	75.1	03.2
		128	95.8	71.5	01.1
	2	32	94.1	75.0	03.3
		64	95.4	69.3	01.7
		128	95.7	80.1	00.4
	4	32	95.2	72.8	01.6
		64	94.9	81.0	01.5
		128	95.1	79.6	01.3
PPO	1	32	93.8	85.4	02.8
		64	89.9	78.8	12.0
		128	86.3	82.8	16.4
	2	32	64.9	45.1	48.1
		64	90.1	73.6	10.3
		128	87.1	69.7	11.7
	4	32	84.9	85.1	18.7
		64	76.2	41.4	33.8
		128	71.7	32.7	45.5

4.2.3 Pistes d'amélioration

Ainsi, notre modèle de détection d'intrusion par RL ne s'applique pas de manière simple aux données cyber-physiques. Dans un souci de cohérence avec le reste de notre étude, nous n'avons pas souhaité nous éloigner du modèle initial, cependant nous avons identifié quelques pistes qui pourraient permettre d'obtenir une détection satisfaisante sur les données SWaT. Celles-ci pourront faire l'objet d'études plus poussées dans un cadre spécifiques aux CPS.

Tout d'abord, il est important de noter que la principale différence entre les données informatiques et les données cyber-physiques réside dans la temporalité des instances : ce n'est donc pas surprenant que notre approche, qui traite chaque pas de temps de manière individuelle et indépendante, peine à cerner les motifs récurrents et les schémas longs-termes présents dans les données SWaT.

Afin de prendre en compte cette continuité temporelle, plusieurs méthodes peuvent être envisagées : d'une part, une approche pourrait consister à traiter chaque colonne du jeu de données comme une série temporelle, et ainsi appliquer les méthode statistiques de traitement de série comme ARIMA [99]. Cela nous éloigne en revanche de l'utilisation de l'apprentissage par renforcement, et de l'apprentissage profond en général.

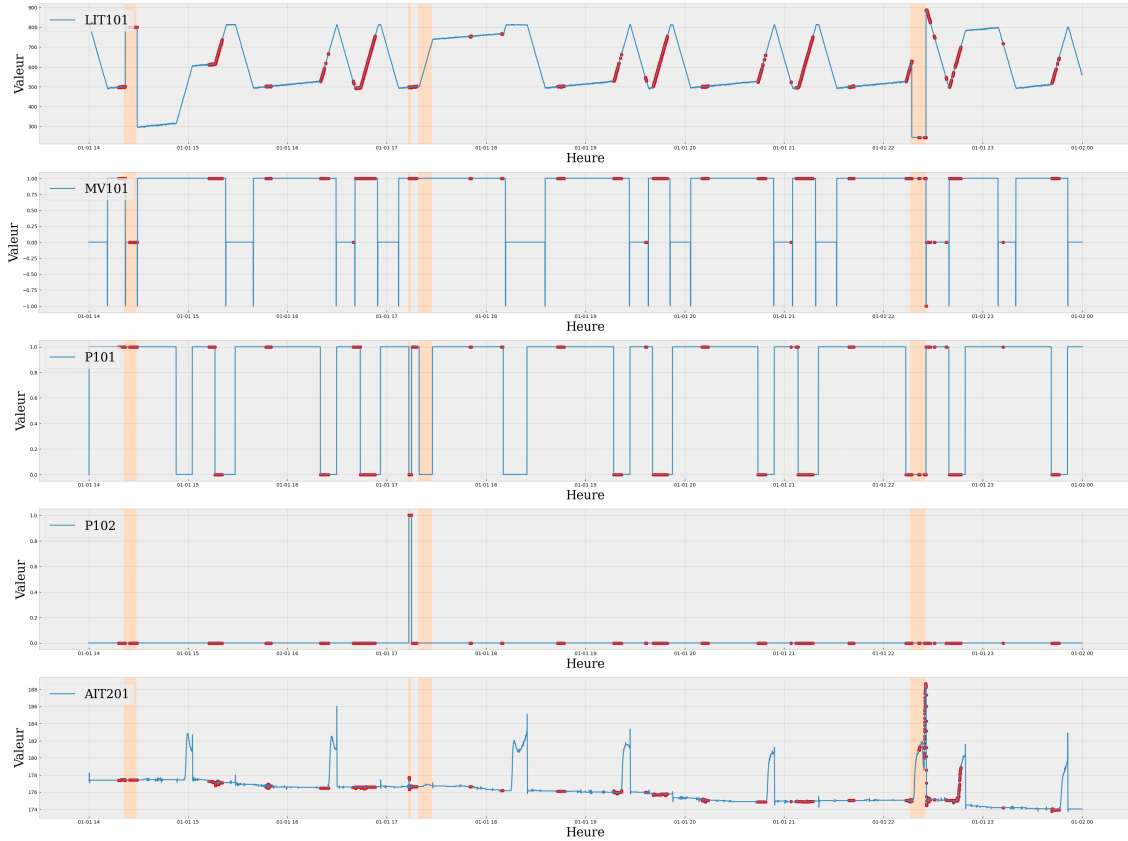


FIGURE 4.7 Attaques véritables (en orange) et détectées (en rouge) sur le premier jour de test de SWaT

Pour revenir dans le cadre de l'intelligence artificielle, un outil en particulier a retenu notre attention : les réseaux convolutifs (CNN, *Convolutional Neural Network*). Spécifiquement conçus pour traiter efficacement les données structurées (images, séries temporelles), ils exploitent des opérations de convolution pour extraire des caractéristiques locales à partir des données, en prenant en compte la relation spatiale entre les différents éléments de la séquence ou de l'image [100]. La Figure 4.8 illustre leur fonctionnement dans le cadre des séquences temporelles. Cette capacité à capturer les motifs locaux est particulièrement adaptée pour la détection d'intrusion dans les données cyber-physiques, où des anomalies peuvent se manifester dans des sous-régions spécifiques des séquences.

Utiliser des réseaux convolutifs pourrait donc permettre d'exploiter les propriétés spécifiques des données cyber-physiques et d'améliorer la capacité des modèles à détecter les intrusions. Ces modèles seraient capables de capturer les relations spatiales et temporelles complexes présentes dans les données et de fournir des prédictions plus précises. Cependant, la complexité de ces couches convolutives rend la convergence de l'entraînement plus délicate à

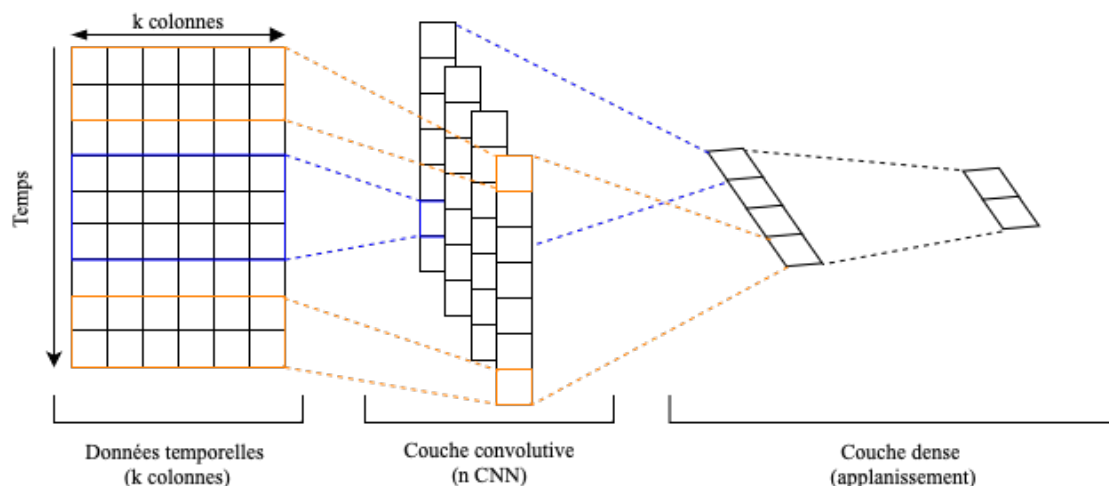


FIGURE 4.8 Schéma du fonctionnement d'un réseau convolutif à 1 dimension

atteindre. Combinée aux aléas liés à l'apprentissage par renforcement, qui n'est pas la manière la plus directe d'apprendre à partir de données étiquetées, cela peut mener à un temps d'entraînement particulièrement long et qui ne converge pas de façon systématique. Ainsi, aucune des expériences que nous avons lancées avec du RL et des CNN n'a pu donner de résultats satisfaisants dans un temps raisonnable : pour utiliser les méthodes convolutives, il semble judicieux de s'intéresser plutôt aux algorithmes supervisés.

En conclusion, l'approche par RL n'a pas produit de résultats satisfaisants dans le traitement de données cyber-physiques, la complexité et l'intrication de ces données se prêtant plutôt à un entraînement plus direct comme l'apprentissage supervisé. Le RL pourrait en revanche être plutôt utilisé comme agent de décision de synthèse de plusieurs modèles de détection, mais cela s'écarte du cadre de la présente étude. Ainsi, bien que certaines pistes, comme les CNN, apparaissent comme prometteuses, nous n'avons pas poursuivi dans le domaine des données CPS, et nous sommes concentrés sur les données informatiques pures.

4.3 Vulnérabilité des modèles

Dans cette section, nous abordons le sujet crucial de la vulnérabilité des modèles de détection d'intrusion par RL face aux attaques adverses. Malgré les performances prometteuses des modèles que nous avons développés jusqu'à présent, il est essentiel de mettre en évidence leurs limites en termes de résistance face à des attaques sophistiquées. Pour cela, nous proposons de réaliser une preuve de concept, en implémentant une attaque simple sur l'un de nos modèles de détection d'intrusion par RL.

L’objectif de cette section est de démontrer expérimentalement que nos modèles peuvent être trompés par des exemples adverses spécifiquement conçus pour contourner leur capacité de détection. Nous souhaitons illustrer comment une petite modification imperceptible des données d’entrée peut suffire à induire nos modèles en erreur et à les amener à prendre de mauvaises décisions. En identifiant cette vulnérabilité, nous cherchons à sensibiliser à l’importance de renforcer la robustesse des modèles de détection d’intrusion par RL et à encourager le développement de techniques de défense adaptées.

Pour cela, nous implémentons ici deux attaques adverses courantes sur l’un de nos modèles de détection d’intrusion par RL. Nous détaillerons les étapes des attaques, et évaluerons l’impact de ces exemples adverses sur les performances du modèle en termes de score F1, faux positifs, et faux négatifs. En mettant en lumière les faiblesses de nos modèles face aux attaques adverses, nous soulignerons la nécessité de développer des mécanismes de défense appropriés pour renforcer leur résistance et leur fiabilité dans des environnements réels.

4.3.1 Implémentation des attaques

Pour mener notre preuve de concept, nous avons sélectionné deux techniques d’attaque adverse couramment utilisées dans la littérature : PIA (ou FGSM) et PGD (voir Section 2.2.3). Ces deux méthodes ont été choisies en raison de leur simplicité et de leur efficacité éprouvée dans la génération d’exemples adverses.

PIA [57] est une adaptation de FGSM [59] qui consiste à exploiter le gradient de la fonction de perte. En propageant ce gradient jusqu’aux données d’entrée, la fonction de perte est alors maximisée, et le modèle est donc trompé. Dans notre cas, nous avons choisi d’utiliser la version ciblée de PIA, qui vise à influencer la classification vers une catégorie particulière (ici, entraîner les instances d’attaques dans la classe normale). L’Équation 4.1 décrit le calcul de la perturbation, avec ∇ le gradient, J_θ la fonction de perte compte tenu des paramètres θ , ϵ la taille de la perturbation et l' la classe visée.

$$x' = x - \epsilon \cdot \text{sign}(\nabla J_\theta(x, l')) \quad (4.1)$$

PGD est la version itérative de PIA : le calcul de la perturbation se fait en plusieurs étapes. L’augmentation du nombre d’itérations produit des perturbations plus fines et peut conduire à des exemples adverses plus subtils. En revanche, le calcul par étapes est généralement plus lent que les méthodes non-itératives, et l’équilibre doit être trouvé entre rapidité du calcul et finesse de la perturbation.

En ce qui concerne l’implémentation, nous avons choisi de montrer l’impact des exemples

adverses sur les agents RL étudiés Section 4.1.2 : un agent DQN et un agent PPO binaires à deux couches et 64 unités, entraînés sur NSL-KDD avec un environnement adverse. Une fois l'agent entraîné, nous calculons les perturbations avec $\epsilon=0.05$, pour toutes les instances de l'ensemble de test. Ensuite, nous calculons les métriques d'évaluation à partir de ces instances modifiées, et analysons les variations de performances.

4.3.2 Résultats

Les résultats de cette première mise à l'épreuve de notre modèle de détection d'intrusion par RL sont significatifs. Nous avons utilisé les techniques d'attaque FGSM et PGD pour générer des exemples adverses spécifiques au modèle choisi, un agent DQN entraîné en binaire sur NSL-KDD. Les perturbations sont limitées à $\epsilon = 0.05$, ce qui signifie que chaque dimension des données d'entrée ne peut varier que de 5% au maximum.

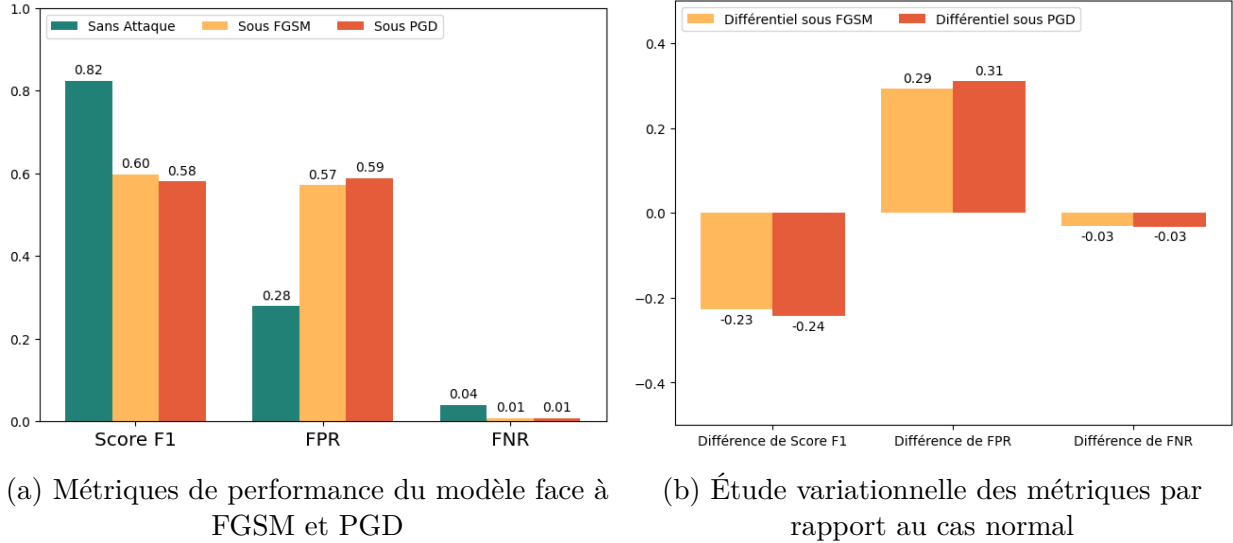


FIGURE 4.9 Performances d'un agent DQN à 2 couches, 64 unités face à une perturbation adverse de $\epsilon=0.05$

La Figure 4.9 permet de comparer les performances du modèle avant et après ces deux types d'attaques adverses. Les résultats révèlent que l'agent est fortement sensible à ces petites variations : ainsi, le score F1 diminue de 23 et 24% respectivement pour FGSM et PGD, tandis que le taux d'attaques non détectées (taux de faux positifs) augmente de 29 et 31%. En revanche, comme l'on pouvait s'y attendre, le nombre de fausses alertes (FNR) diminue en présence d'exemples adverses. En effet, les attaques adverses sont ciblées pour envoyer toutes les instances dans la classe *Normale*, ce qui réduit le nombre d'instances normales

faussement signalées comme des attaques. Enfin, nous pouvons également constater que la méthode PGD est légèrement plus performante que FGSM, mais que les deux approches restent équivalentes en termes d'évasion de la détection.

Les matrices de confusion normalisées de l'agent initial et de celui perturbé par FGSM et par PGD peuvent être retrouvées dans la Figure 4.10. Elles permettent de confirmer l'impact des attaques adverses sur le modèle de détection, avec le taux d'erreur pour la classe "Attaques" qui augmente significativement dans les deux configurations adverses.

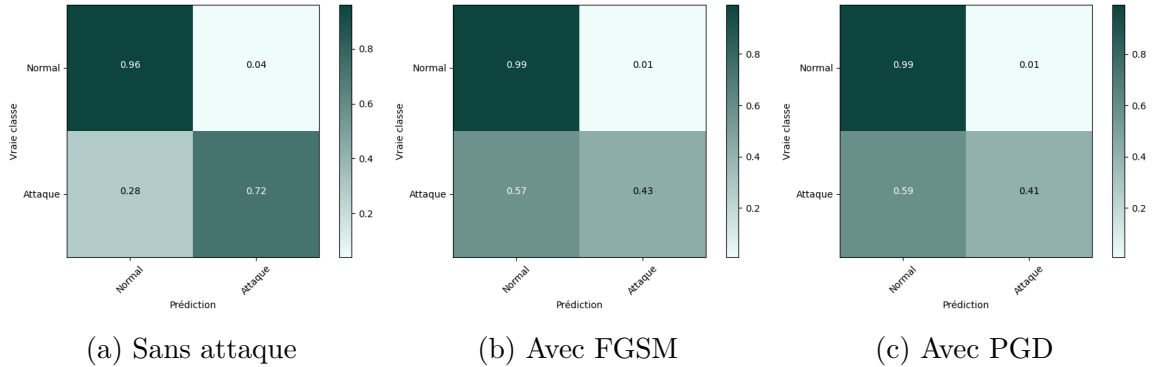


FIGURE 4.10 Matrices de confusion de l'agent soumis ou non aux perturbations adverses

Ces résultats démontrent clairement que nos modèles de détection d'intrusion par RL ne sont pas invulnérables aux attaques adverses et qu'ils peuvent être trompés par des exemples perturbés. Cela soulève de nombreuses préoccupations en termes de sécurité des systèmes de détection d'intrusion, et souligne l'importance capitale de prendre en compte la robustesse des agents pendant leur développement.

4.4 Conclusion

Dans ce chapitre, nous avons donc développé un modèle d'agent RL performant pour la détection d'intrusion. Nous avons affiné notre approche en ajoutant une dimension adverse à l'environnement considéré, ainsi qu'en analysant l'impact des hyperparamètres sur les performances des agents. En revanche, nous avons conclu sur la pertinence de l'utilisation du RL pour les données cyber-physiques : à l'heure actuelle, adapter notre approche aux données des CPS n'a pas fourni de résultats satisfaisants, tandis que des approches supervisées, plus directes, apparaissent comme plus prometteuses.

De plus, nos expériences ont révélé que ces modèles RL, bien que performants, ne sont pas à l'abri des attaques par exemples adverses telles que PGD et FGSM. Ce constat met en

évidence la nécessité de développer des stratégies de défense avancées capables de résister à des attaques sophistiquées et itératives. Ces résultats soulignent également l'importance de prendre en compte la robustesse des modèles de détection d'intrusion par RL lors de leur conception et de leur déploiement. Les systèmes de détection d'intrusion doivent être capables de détecter et de résister aux attaques adverses afin de garantir leur efficacité dans des environnements réels.

Dans la suite de notre étude, nous nous intéresserons aux méthodes de défense et de protection contre les attaques adverses pour renforcer la robustesse de nos modèles.

CHAPITRE 5 PISTES DE DÉFENSE DES SDI PAR RL CONTRE LES ATTAQUES PAR EXEMPLES ADVERSES

Dans ce chapitre, nous abordons le sujet crucial et délicat de la protection des modèles de détection d'intrusion basés sur le RL contre les attaques adverses. Comme nous l'avons vu dans le chapitre précédent, les modèles de RL ont montré des performances prometteuses dans la détection d'intrusions pour les réseaux informatiques. Cependant, ces modèles se sont également montrés vulnérables aux attaques par exemples adverses, ce qui remet en question leur fiabilité et leur utilité dans des conditions réelles. La question de la robustesse des modèles est un vaste sujet dont chaque cas d'usage possède ses propres contraintes et spécificités. Ici, nous cherchons donc à mener une étude expérimentale des pistes les plus prometteuses et les mieux adaptées au cas des SDI.

La motivation principale de ce chapitre est d'améliorer la robustesse des SDI par RL, en comparant l'efficacité de différentes stratégies de protection pour atténuer l'impact des attaques adverses sur les modèles de détection RL. En comprenant les vulnérabilités des modèles existants et en identifiant les faiblesses qui les rendent sensibles aux attaques adverses, nous pouvons concevoir des mécanismes de défense appropriés. Notre objectif est de fournir un cadre formel pour la robustesse ainsi qu'une première approche pour renforcer la résistance des modèles, accompagnés de pistes d'amélioration pour les recherches futures.

Pour ce faire, nous commencerons par poser une définition mathématique de la robustesse. Ensuite, nous explorerons en particulier deux approches distinctes : l'apprentissage robuste et l'entraînement adverse. L'apprentissage robuste vise à améliorer la résistance des modèles en optimisant les hyperparamètres ainsi que l'architecture du modèle, tandis que l'entraînement adverse consiste à utiliser des exemples adverses pendant l'entraînement pour renforcer la capacité de détection du modèle face aux perturbations. Ces deux approches complémentaires offrent des perspectives intéressantes pour renforcer la robustesse des agents de détection RL. Après avoir analysé les résultats obtenus, nous concluons sur la pertinence de ces approches.

En combinant nos connaissances sur les vulnérabilités des modèles RL ainsi que sur les techniques de protection existantes, nous sommes convaincus que ce chapitre fournira des informations précieuses pour renforcer la résistance des modèles de détection d'intrusion basés sur le RL dans des environnements réels confrontés à des menaces constantes.

5.1 Cadre mathématique

5.1.1 Cas général

Avant de se lancer dans une étude expérimentale, il convient de poser formellement les notions mathématiques qui définissent la robustesse en RL de manière générale.

Si l’objectif de robustesse est un concept très répandu, il est spécifique à chaque cas d’utilisation et donc rarement défini de manière précise et formelle. Nous proposons ici une approche qui permet d’englober la notion de robustesse en RL quelle que soit l’application envisagée. Ainsi, la robustesse d’un agent RL peut être posée comme le problème d’optimisation suivant :

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\Omega \sim \Phi(\Omega|\pi)} \mathbb{E}_{\tau \sim \pi^{\Omega}(\tau)} [G(\tau)] \quad (5.1)$$

où Φ correspond à la distribution des environnements Ω auxquels sera confronté l’agent à son déploiement, $\pi^{\Omega}(\tau)$ à la distribution des trajectoires de la politique π avec les dynamiques de Ω , et $G(\tau)$ le gain de τ . Cela revient à trouver la politique π^* qui permette d’obtenir la meilleure espérance de gain quel que soit l’environnement dans lequel l’agent sera déployé.

Construire une architecture RL robuste revient donc à optimiser l’Équation 5.1 en trouvant les hyperparamètres qui permettent d’obtenir un gain satisfaisant sur tout Φ . La deuxième approche envisagée dans notre étude sur la robustesse, l’entraînement adverse, implique un adversaire externe qui vient perturber l’entraînement pour permettre une meilleure généralisation de π .

Nous pouvons en effet introduire un adversaire ξ dans l’environnement d’entraînement, défini comme une fonction pouvant altérer l’une ou plusieurs des composantes de Ω étant donné π (Équation 5.2). Dans le cadre de ce projet, nous ne nous intéresserons qu’aux modifications de l’environnement S ou des observations O , ainsi qu’à la fonction de transition P associée à l’environnement.

$$\xi(\pi, \Omega) = (\xi_S(\pi, S^{\Omega}), \xi_A(\pi, A^{\Omega}), \xi_P(\pi, P^{\Omega}), \xi_R(\pi, R^{\Omega}), \xi_O(O^{\Omega})) \quad (5.2)$$

L’agent évolue alors dans un nouvel environnement altéré, $\xi(\pi, \Omega)$. La politique optimale est donc redéfinie par l’Équation 5.3 pour tenir compte de ces modifications

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\tau \sim \pi^{\xi, \Omega}(\tau)} [G(\tau)] \quad (5.3)$$

où $\pi^{\xi, \Omega}$ est la distribution des trajectoires de la politique π sur le MDP altéré $\xi(\pi, \Omega)$.

Pour ensuite estimer la robustesse de l’agent lors de son déploiement dans Φ , nous définissons une série de perturbations ϕ semblables à ξ mais cette fois-ci inconnues durant l’entraînement. Nous évaluons ainsi les différentes méthodes adverses ξ dans plusieurs instances de MDP issues de la distribution Φ .

5.1.2 Adaptation aux agents RL détecteurs d’intrusions

La section précédente propose une approche rigoureuse de la robustesse dans un cadre très général. Pour ce qui est de notre cas d’étude spécifique, il convient de relier les formules théoriques à leur sens pratique.

Ainsi, dans l’Équation 5.1, la politique π^* représente un agent de détection robuste aux variations du monde réel, tandis que la formule $\mathbb{E}_{\tau \sim \pi^{\Omega}(\tau)}[G(\tau)]$ désigne simplement la performance de détection de l’agent dans l’environnement réel Ω , qui suit la loi de distribution aléatoire Φ .

Dans le Chapitre 4, nous avons évalué la robustesse des agents face aux attaques adverses inconnues lors de l’entraînement : ces attaques par exemples adverses peuvent se traduire plus formellement par les perturbations ϕ évoquées plus haut, qui modélisent la distribution Φ des environnements du monde réel en ne modifiant ici que l’espace des observations. L’objectif de ce chapitre est donc d’améliorer les performances de nos agents face à ces attaques, en explorant notamment la piste des architectures robustes et celle de l’entraînement adverse.

5.2 Apprentissage robuste : optimiser les hyperparamètres pour une architecture plus résiliente

Comprendre l’impact des hyperparamètres sur la robustesse des modèles est essentiel pour concevoir des stratégies de protection efficaces. En identifiant les combinaisons optimales d’hyperparamètres, nous pouvons améliorer la capacité du modèle à détecter les intrusions tout en conservant une bonne robustesse face aux exemples adverses. Ces informations nous permettront de guider la configuration et l’optimisation des modèles de détection RL pour une protection maximale.

L’objectif de cette partie est donc d’explorer comment les réglages de l’entraînement peuvent influencer à la fois les performances et la robustesse des agents de détection d’intrusion basés sur le RL. Après avoir détaillé notre stratégie expérimentale, nous examinerons l’impact de divers hyperparamètres tels que l’architecture du réseau, le nombre de classes, ou l’algorithme

RL utilisé sur les performances de détection d'intrusion soumise aux exemples adverses. En analysant les résultats de nos expérimentations, nous pourrions déterminer les combinaisons d'hyperparamètres les plus appropriées pour maximiser à la fois les performances et la robustesse des modèles de détection d'intrusion basés sur le RL. Nous discuterons des observations clé et des conclusions tirées de ces résultats, mettant en évidence les hyperparamètres qui ont le plus d'impact sur la robustesse des agents.

5.2.1 Approche expérimentale

Nous décrivons ici les étapes clé de nos expériences, avec notamment la sélection des hyperparamètres à étudier et les métriques utilisées pour évaluer la robustesse des modèles.

Afin de définir les hyperparamètres pertinents pour cette étude, en s'appuyant sur nos expériences précédentes ainsi que sur la littérature nous avons identifié plusieurs critères à étudier. Ceux-ci font écho aux paramètres identifiés dans la Section 4.1.3 :

- **Architecture du réseau** : le nombre de couches de neurones, ainsi que d'unités par couche, est probablement l'un des facteurs les plus importants pour diminuer la sensibilité des modèles aux perturbations. En effet, l'intuition et les précédentes études veulent qu'un modèle moins dense ou moins profond soit plus difficile à tromper [101].
- **Format des données** : de même, le choix dans la manière de pré-traiter les données avant d'entraîner le modèle peut avoir une forte influence sur sa robustesse. Par exemple, le fléau de la dimension amène à penser que plus il y a de colonnes dans le jeu de données, moins le modèle de prédiction est stable. Dans notre cas, nous avons donc testé deux jeux de données différents (avec chacun une taille et un nombre de caractéristiques propres) ainsi que deux types de pré-traitements différents : binaire et multi-classe.
- **Algorithme** : enfin, nous avons questionné l'impact du choix de l'algorithme RL utilisé sur la robustesse de l'agent. En gardant les deux modèles employés dans le chapitre précédent, DQN et PPO, nous cherchons donc à conclure sur l'intérêt de chacun dans un contexte adverse.

Pour évaluer l'impact de chaque critère sur la robustesse des agents, nous avons choisi d'utiliser les mêmes attaques que dans le chapitre précédent, c'est-à-dire FGSM et PGD. Nous avons également repris les métriques d'évaluation introduites dans les Chapitres 3 puis 4, notamment le score F1 ainsi que le TFP (taux d'attaques non détectées). Cependant, nous nous intéressons ici en particulier à la différence de performance entre l'agent dans un environnement sain et l'agent soumis aux exemples adverses. Nous introduisons donc deux métriques qui viendront compléter notre analyse, $F1_{diff}$ et TFP_{diff} , décrites Équation 5.5

$$F1_{diff} = F1_{normal} - F1_{attaque} \quad (5.4)$$

$$TFP_{diff} = TFP_{normal} - TFP_{attaque} \quad (5.5)$$

Notons que ces métriques différentielles doivent absolument être combinées avec les métriques fixes afin de prendre en compte simultanément les notions de performances et de robustesse : en effet, si la différence est légère, mais que les performances initiales du modèle étaient déjà faibles, on ne peut pas conclure sur la robustesse de cet agent.

En résumé, notre méthodologie expérimentale est conçue pour évaluer l’impact des hyperparamètres sur la robustesse des agents de détection d’intrusion par RL. En utilisant des ensembles de données réalistes, des métriques appropriées et des attaques adverses soigneusement sélectionnées, nous visons à obtenir des résultats fiables et significatifs pour guider le choix et l’optimisation des hyperparamètres afin d’améliorer la robustesse des agents de détection d’intrusion.

5.2.2 Résultats

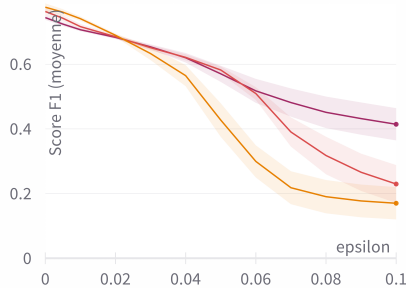
Après avoir calculé les perturbations de FGSM et PGD sur les différents agents RL de détection d’intrusions préalablement entraînés, nous avons recueilli et analysé les données résultantes de nos expériences pour obtenir des informations sur l’influence de l’architecture du réseau, du format des données et de l’algorithme choisi sur la robustesse des modèles.

Impact de l’architecture

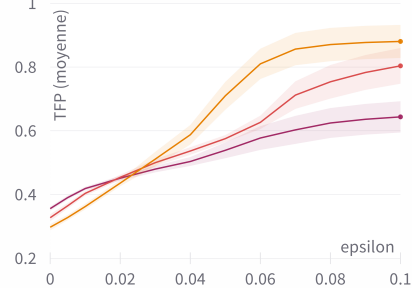
Dans un premier temps, nous pouvons nous intéresser aux choix d’architecture du réseau de neurones dans l’algorithme DRL. En effet, si nous avons vu que ces choix peuvent influencer fortement les performances des modèles, il est fort probable qu’ils influencent également leur robustesse.

Pour évaluer l’impact de l’architecture, nous nous sommes concentrés sur le jeu de données NSL-KDD et avons considéré deux paramètres clé : la profondeur (nombre de couches) et la largeur (nombre d’unités par couche). Dans un premier temps, nous avons donc regroupé les résultats de tous nos agents entraînés sur NSL-KDD en fonction du nombre de couches, en agrégeant les données sur la largeur du réseau, l’algorithme utilisé (PPO ou DQN) et le nombre de classes (binaire ou multiple).

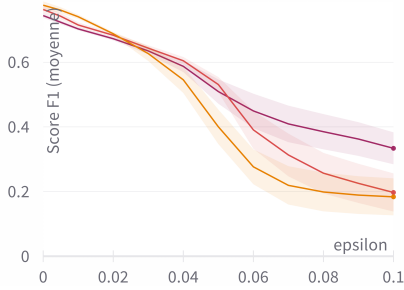
Profondeur : La Figure 5.1 présente la moyenne du score F1 et du taux de faux positifs (TFP) pour différentes valeurs de perturbation ϵ . Nous avons effectué des tests avec des ϵ allant de 0.05 (correspondant à une perturbation maximale de 0,5% de la valeur initiale) à 0.1 (correspondant à une perturbation maximale de 10% de la valeur initiale) pour une analyse exhaustive.



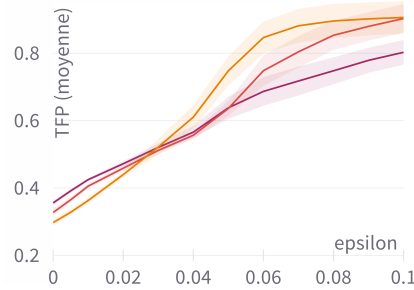
(a) Moyenne du score F1 des agents SDI-RL soumis à FGSM



(b) Moyenne du TFP des agents SDI-RL soumis à FGSM



(c) Moyenne du score F1 des agents SDI-RL soumis à PGD



(d) Moyenne du TFP des agents SDI-RL soumis à PGD

FIGURE 5.1 Évolution du score F1 et du TFP des agents SDI-RL soumis aux attaques par exemples adverses FGSM et PGD, agrégés et regroupés par nombre de couches cachées, en fonction de la taille maximale de perturbation ϵ .

Les résultats présentés dans la Figure 5.1 fournissent une illustration claire de l'impact significatif de la profondeur du modèle sur sa vulnérabilité aux attaques adverses. Indépendamment du type d'attaque considéré, nous observons une différence notable de performance en fonction du nombre de couches du modèle. Par exemple, pour une perturbation $\epsilon = 0.1$, le score F1 des agents à 1 couche soumis à l'attaque FGSM est seulement de 0.17, tandis que celui des agents à 4 couches atteint 0.41. De même, le taux de faux positifs (TFP) des agents à 1 couche sous attaque est nettement plus élevé que celui des agents à 2 couches, qui est lui-même supérieur à celui des agents à 4 couches.

Ces résultats soulèvent une observation intéressante et contre-intuitive : les modèles plus profonds se révèlent en réalité plus robustes face aux exemples adverses que les modèles plus superficiels. Cette constatation contredit notre intuition initiale et peut être expliquée par la difficulté accrue pour l’attaquant de calculer la perturbation optimale dans le cas de modèles profonds. La complexité et la non-linéarité des réseaux profonds rendent la recherche d’une perturbation efficace plus difficile, ce qui compense l’instabilité potentielle liée au nombre de poids du modèle.

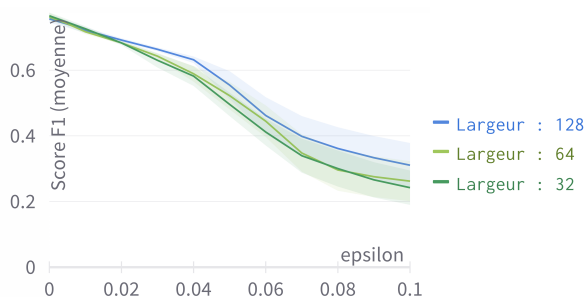
En outre, il est intéressant de noter que cette différence de vulnérabilité entre les modèles en fonction de la profondeur n’est observable que pour des perturbations supérieures à 0.2. Pour des perturbations inférieures à ce seuil, nous observons même une inversion de l’ordre des courbes : les modèles moins profonds affichent de meilleures performances. Cette observation s’explique par les différences de performance initiale entre les modèles. En effet, en l’absence d’attaques, ce sont les modèles moins profonds qui présentent les meilleures performances, il n’est donc pas surprenant qu’ils conservent leur avantage sous de très légères perturbations.

Pour finir, ces résultats soulignent l’importance de trouver un compromis adapté à la situation spécifique du système de détection d’intrusion. Le propriétaire du SDI doit prendre en compte à la fois les performances initiales des modèles sans attaque et leur résilience face aux exemples adverses. La sélection de l’architecture du modèle doit être guidée par une évaluation approfondie des performances et de la robustesse pour déterminer le meilleur équilibre.

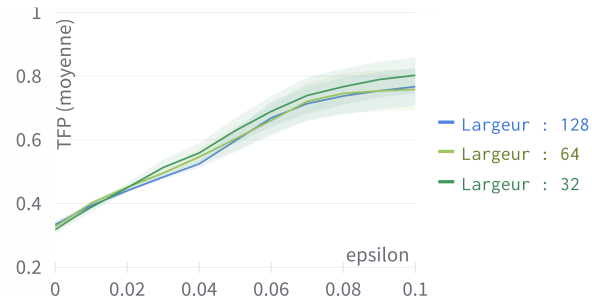
Largeur : Intéressons-nous maintenant à l’influence du nombre d’unités par couches : la largeur du réseau a-t-elle le même impact sur sa vulnérabilité que la profondeur ? La Figure 5.2 présente de nouveau la moyenne du score F1 et du TFP des agents RL pour différentes valeurs de perturbation ϵ , mais ici les courbes sont agrégées et regroupées par largeur du réseau : 32, 64 ou 128 unités par couche cachée.

La Figure 5.2 présente des résultats moins concluants concernant l’impact de la largeur du réseau sur la robustesse des modèles de détection d’intrusion. En effet, les courbes correspondant à chaque groupe sont très proches les unes des autres, et les écart-types ne montrent pas de séparation nette. Bien qu’il y ait une légère différence de performance en fonction du nombre de couches cachées, notamment avec les modèles les plus denses qui semblent être légèrement moins vulnérables, cette différence est trop faible pour conclure de manière catégorique qu’il est plus facile pour un adversaire d’attaquer les modèles moins larges.

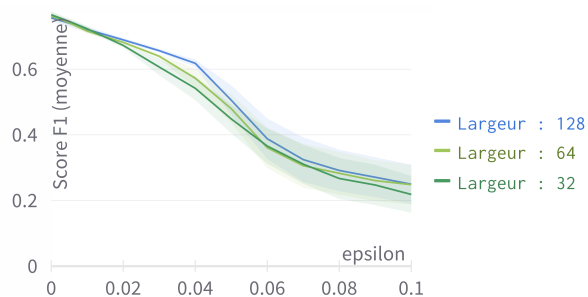
Cette analyse des hyperparamètres permet de mieux comprendre l’influence des choix architecturaux sur la robustesse des modèles de détection d’intrusion, et suggère notamment que



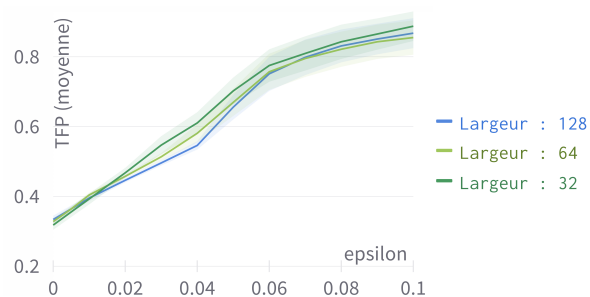
(a) Moyenne du score F1 des agents SDI-RL soumis à FGSM



(b) Moyenne du TFP des agents SDI-RL soumis à FGSM



(c) Moyenne du score F1 des agents SDI-RL soumis à PGD



(d) Moyenne du TFP des agents SDI-RL soumis à PGD

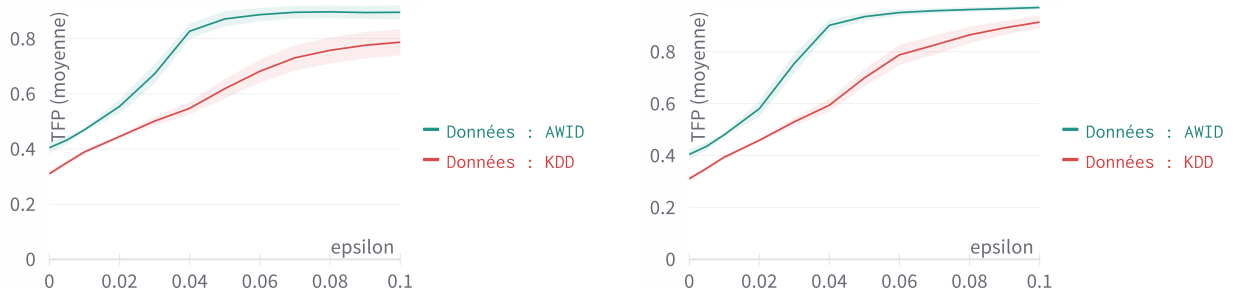
FIGURE 5.2 Évolution du score F1 et du TFP des agents SDI-RL soumis aux attaques par exemples adverses FGSM et PGD, agrégés et regroupés par nombre d'unités par couche, en fonction de la taille maximale de perturbation ϵ .

la largeur du réseau peut avoir un impact plus limité sur la robustesse des agents par rapport à leur profondeur. Elle constitue une étape essentielle dans la conception de mécanismes de défense contre les attaques adverses et fournit des indications précieuses pour orienter les futures recherches en matière d'apprentissage robuste.

Impact du format des données

Outre l'architecture des réseaux de neurones, d'autres facteurs peuvent fortement influencer la vulnérabilité des agents. Nous avons donc analysé ici l'impact du format des données sur la robustesse des modèles de détection d'intrusion. Nous explorons spécifiquement deux aspects : le jeu de données utilisé et le nombre de classes dans le problème de détection d'intrusion.

Jeu de données Le choix du jeu de données est crucial dans la conception d'un modèle de détection d'intrusion, car il détermine la représentativité et la diversité des exemples d'entraînement. Différents jeux de données peuvent présenter des caractéristiques distinctes, telles que la taille, la nature des attaques, le type de trafic, et la complexité des scénarios. Nous évaluons ainsi la robustesse des modèles en utilisant deux jeux de données couramment utilisés dans le domaine de la détection d'intrusion : NSL-KDD et AWID. La Figure 5.3 présente la moyenne du taux de faux positifs sous FGSM et PGD pour chacun de ces deux jeux de données. Comme le score F1 dépend fortement de la proportion d'anomalies dans le jeu de données, il n'est pas pertinent dans la comparaison entre AWID et NSL-KDD.



(a) Moyenne du TFP des agents SDI-RL soumis à FGSM (b) Moyenne du TFP des agents SDI-RL soumis à PGD

FIGURE 5.3 Évolution du TFP des agents SDI-RL soumis aux attaques par exemples adverses FGSM et PGD, agrégés et regroupés par jeu de données, en fonction de la taille maximale de perturbation ϵ

Nous constatons dans la Figure 5.3 que les modèles entraînés sur le jeu de données AWID se révèlent plus sensibles aux exemples adverses que ceux entraînés sur NSL-KDD. En effet, avec une perturbation de seulement $\epsilon = 0.05$ appliquée par l'attaque FGSM, le taux de faux positifs (TFP) moyen des agents AWID atteint en moyenne 0.87, tandis que celui des agents NSL-KDD se situe à 0.61. Cette différence de performance ne peut être expliquée uniquement par les TFP initiaux, qui sont déjà supérieurs pour les agents AWID. Afin de mieux comprendre cette disparité, une étude plus approfondie serait nécessaire pour identifier les caractéristiques spécifiques des ensembles de données responsables de ce comportement.

Nous émettons l'hypothèse que la structure des jeux de données joue un rôle significatif dans cette différence observée. En effet, le jeu de données AWID contient environ dix fois plus d'instances que NSL-KDD, mais la proportion d'attaques y est nettement plus faible. Cette composition peut entraîner une frontière de prédiction plus floue, rendant ainsi les modèles AWID plus vulnérables aux attaques adverses.

Nombre de classes Nous avons vu dans le chapitre précédent qu'entraîner les agents dans un mode binaire, avec deux classes "Normale" et "Attaque", permet d'obtenir de meilleures performances de détection, tout en délaissant l'aspect analyse de l'attaque signalée. Quelles sont les forces et les faiblesses de ce mode binaire comparé au mode multi-classe, face à un adversaire malveillant ?

La Figure 5.4 compare le TFP des agents en mode binaire et multi-classe, soumis à FGSM et PGD avec une perturbation maximale variant entre 0.005 et 0.1. À nouveau, le score F1 n'est pas représenté ici car il est calculé de manière différente selon le nombre de classe, ce qui ne permet pas de comparaison pertinente.



(a) Moyenne du TFP des agents SDI-RL soumis à FGSM (b) Moyenne du TFP des agents SDI-RL soumis à PGD

FIGURE 5.4 Évolution du TFP des agents SDI-RL soumis aux attaques par exemples adverses FGSM et PGD, agrégés et regroupés par mode de prédiction, en fonction de la taille maximale de perturbation *epsilon*

D'après la Figure 5.4, le mode multi-classe semble légèrement plus résistant aux attaques adverses considérées. On observe en particulier que face à FGSM avec $\epsilon = 0.1$, le taux de faux positifs est en moyenne de 0.78 pour le mode multi-classe contre 0.88 pour le mode binaire. Cette différence n'est cependant pas aussi flagrante pour l'attaque PGD, possiblement grâce à la nature itérative de cette méthode qui parvient peut-être à mieux différencier les classes entre elles.

En analysant l'impact du jeu de données et du nombre de classes sur la robustesse des modèles, nous visons à identifier les facteurs extérieurs au modèle qui contribuent à une meilleure résistance aux attaques adverses. De nombreuses autres caractéristiques des données pourraient être prises en compte, comme le nombre de dimensions ou le format des données, que nous laissons à de futures recherches. Ces résultats nous permettent cependant déjà de formuler quelques recommandations pratiques concernant la structure des données d'entraînement à utiliser pour de meilleures performances : dans notre cas, privilégier un jeu de données

équilibré, ainsi que le mode multi-classe semble mieux réussir à limiter les effets des exemples adverses.

Impact de l'algorithme

Pour terminer notre étude sur la robustesse des modèles de détection d'intrusion face aux attaques adverses, nous examinons également l'impact de l'algorithme de RL choisi pour entraîner nos agents. Après avoir analysé l'influence du pré-traitement des données et de l'architecture du réseau, nous nous intéressons à présent à la question de savoir si le mode d'entraînement a également un effet sur la robustesse des modèles.

Nous nous concentrons sur la comparaison des deux algorithmes RL utilisés jusqu'ici dans notre étude : PPO et DQN. Dans cette analyse comparative, nous évaluons la performance moyenne des agents entraînés avec ces deux algorithmes sous les attaques FGSM et PGD. Notre objectif est de déterminer si le mode d'entraînement, c'est-à-dire l'algorithme RL utilisé, influence la capacité des modèles à résister aux exemples adverses.

La Figure 5.5 vise à mettre en évidence la différence de vulnérabilité entre les modèles DQN et PPO face aux attaques adverses FGSM et PGD. Nous observons que les modèles PPO maintiennent un score F1 plus élevé et un TFP plus faible que les modèles DQN, quel que soit le niveau de perturbation appliqué. Cependant, il est important de noter que les performances des modèles sans attaque présentent déjà un écart significatif : par exemple, le score F1 des agents DQN sans attaque atteint en moyenne 0.81, tandis que celui des agents PPO est de 0.93. Avec une perturbation FGSM de 10%, le score F1 des modèles DQN diminue à 0.42, tandis que celui des modèles PPO diminue à 0.50. Ainsi, l'écart entre les deux types de modèles est en fait réduit avec l'intensification de la perturbation, ce qui nous empêche d'affirmer avec certitude que la différence de modèle a un impact sur la robustesse des agents.

Pour confirmer cette hypothèse, il serait intéressant de mener des expériences avec d'autres algorithmes RL tels qu'A3C ou TRPO, afin d'évaluer leur robustesse face aux attaques adverses. De plus, l'étude pourrait être étendue en explorant d'autres méthodes d'attaques adverses. Cependant, malgré cette perspective, il est important de souligner que les résultats obtenus jusqu'à présent indiquent que le choix du modèle a moins d'influence sur la robustesse que d'autres facteurs tels que l'architecture du réseau ou le format des données.

5.2.3 Conclusion

Dans cette section, nous avons donc exploré l'impact des hyperparamètres, du format des données et de l'algorithme d'apprentissage sur la robustesse des modèles de détection d'in-

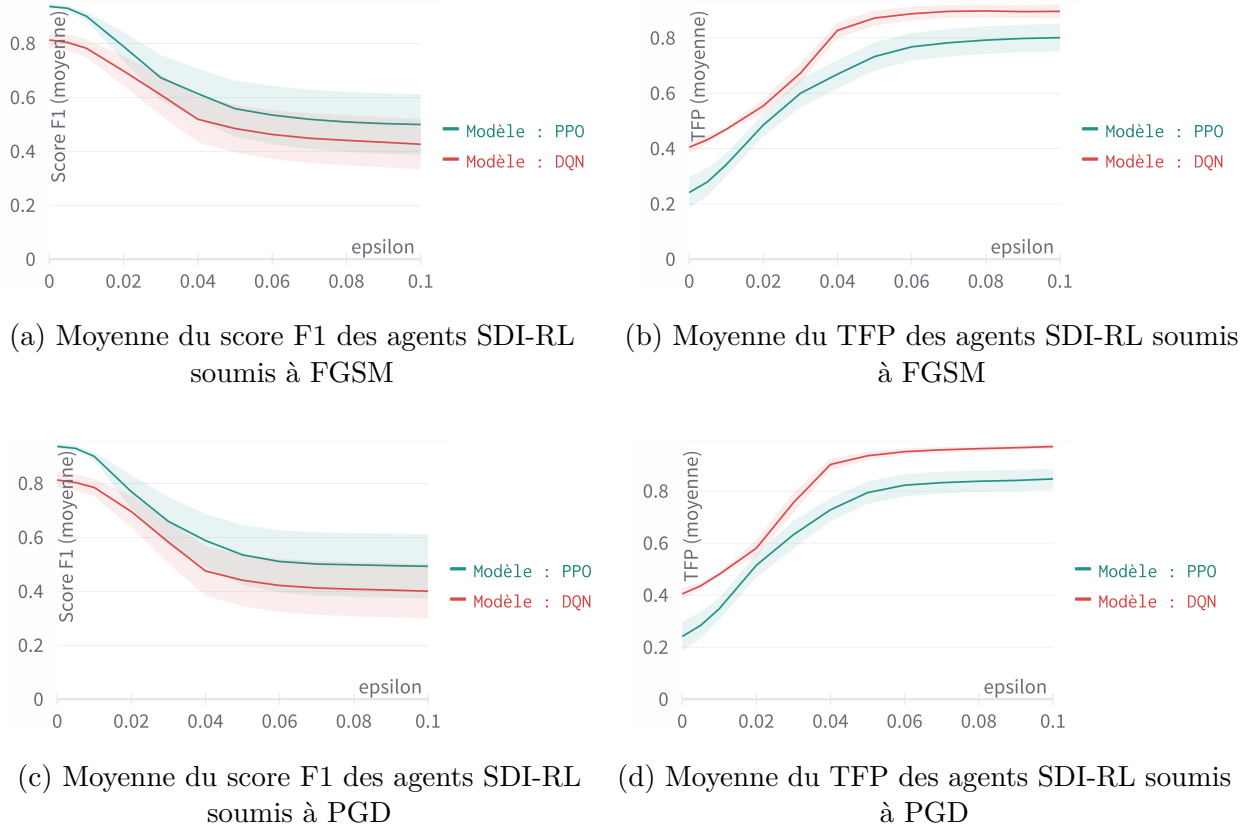


FIGURE 5.5 Évolution du score F1 et du TFP des agents SDI-RL soumis aux attaques par exemples adverses FGSM et PGD, agrégés et regroupés par algorithme RL utilisé, en fonction de la taille maximale de perturbation ϵ .

trusion face aux attaques adverses. Les résultats obtenus ont montré que la profondeur du réseau a un effet significatif sur la capacité du modèle à résister aux exemples adverses, avec des modèles plus profonds démontrant une meilleure robustesse. En revanche, l'impact de la largeur du réseau s'est révélé moins déterminant. De plus, il a été observé que le choix du jeu de données influence également la vulnérabilité des modèles, avec une sensibilité plus élevée aux attaques adverses dans le cas du jeu de données AWID par rapport à NSL-KDD. Enfin, l'analyse comparative entre les modèles DQN et PPO a montré que le choix de l'algorithme d'apprentissage ne semble pas avoir d'impact significatif sur la vulnérabilité des modèles. Toutefois, il convient de noter que ces résultats peuvent être affinés dans des études futures, pour prendre en compte plus de paramètres et étendre les expériences à d'autres méthodes. S'il reste clair que lors de la construction d'un agent SDI, une attention particulière doit être accordée à l'architecture du réseau et au format des données pour s'assurer d'une meilleure robustesse des modèles contre les attaques adverses, d'autres méthodes plus spécifiques existent

pour encore diminuer leur vulnérabilité, comme notamment l’entraînement adverse.

5.3 Entraînement adverse : tirer parti des attaques connues

L’entraînement adverse est une approche prometteuse pour renforcer la robustesse des modèles de détection d’intrusion face aux perturbations. Cette méthode consiste à intégrer des exemples adverses lors de l’entraînement des modèles afin de les rendre plus résistants aux perturbations malveillantes, tout en s’assurant de préserver leurs performances. L’objectif de cette section est d’évaluer l’efficacité de l’entraînement adverse pour protéger nos modèles contre les attaques par exemples adverses, en analysant l’impact sur les performances et la robustesse des agents de détection d’intrusion. Nous examinerons différentes stratégies d’entraînement, notamment l’ajout d’exemples adverses générés par des attaques FGSM et PGD, ainsi que l’exploration de différentes intensités de perturbation. Cette analyse approfondie nous permettra de mieux comprendre comment l’entraînement adverse peut renforcer la résistance de nos modèles aux attaques adverses, tout en maintenant des performances élevées en termes de détection d’intrusion.

5.3.1 Approche expérimentale

Nous souhaitons donc tester la pertinence de l’entraînement adverse dans le cas de la détection d’intrusion par RL. Il convient de noter que l’entraînement adverse offre de multiples possibilités en termes de méthodes de génération des perturbations, d’hyperparamètres et d’approches d’entraînement. Dans le cadre de cette étude, nous avons choisi d’effectuer une preuve de concept afin de démontrer l’utilité potentielle de l’entraînement adverse dans notre domaine spécifique de détection d’intrusion par RL. Cette approche nous permettra de mettre en évidence les avantages et les opportunités offertes par l’entraînement adverse, tout en nous fournissant des indications précieuses pour améliorer la robustesse de nos modèles. Une étude plus exhaustive pourra faire l’objet de futurs travaux.

Nous avons donc choisi de tester l’un des agents les plus performants et représentatifs que nous ayons étudiés, le modèle DQN à 2 couches et 64 unités par couches entraîné en mode binaire. L’objectif est de réentraîner ce modèle en utilisant des exemples adverses générés par l’attaque FGSM, une attaque classique dans le domaine et réputée particulièrement performante. Nous testons de plus différentes tailles de perturbation pour l’entraînement adverse, afin d’obtenir la meilleure robustesse sans pour autant compromettre les performances de détection.

Pour évaluer la résistance de ces modèles nouvellement entraînés, nous appliquons ensuite plusieurs attaques adverses : PGD et DeepFool. Ces attaques ont été choisies car elles utilisent

chacune une méthode de calcul particulière : la nature itérative de PGD et celle géométrique de DeepFool permettent de tester la vulnérabilité des modèles face à une variété d’approches.

Dans l’optique de confirmer si cette approche permet effectivement d’améliorer la résistance des modèles aux exemples adverses tout en maintenant leur précision de détection, nous commençons par mesurer la différence de performance entre le modèle initial et les modèles réentraînés. Ensuite, nous comparons l’impact des trois attaques mentionnées ci-dessus sur les modèles réentraînés, par rapport à leur impact sur le modèle initial. Ces résultats nous permettront de tirer une première conclusion sur l’entraînement adverse pour les SDI par RL, et serviront de base pour orienter nos recherches futures dans l’amélioration des performances et de la robustesse de nos modèles.

5.3.2 Résultats

Performance des modèles réentraînés

Dans un premier temps, nous souhaitons vérifier que l’entraînement adverse ne nuit pas aux performances de détection des agents SDI. Comme mentionné précédemment, nous avons choisi l’agent DQN sur KDD à deux couches et 64 unités par couche en mode binaire pour le réentraîner à l’aide d’instances perturbées par FGSM. Nous disposons donc de plusieurs modèles réentraînés, un pour chaque taille de perturbation (ϵ) choisie.

Nous examinons donc ici les performances de ces agents réentraînés pour évaluer leur capacité à détecter les attaques tout en maintenant une précision élevée. Cette analyse approfondie nous permettra de déterminer dans quelle mesure l’entraînement adverse peut contribuer à améliorer la performance des modèles de détection d’intrusion par RL et leur capacité à faire face aux scénarios d’attaques réelles.

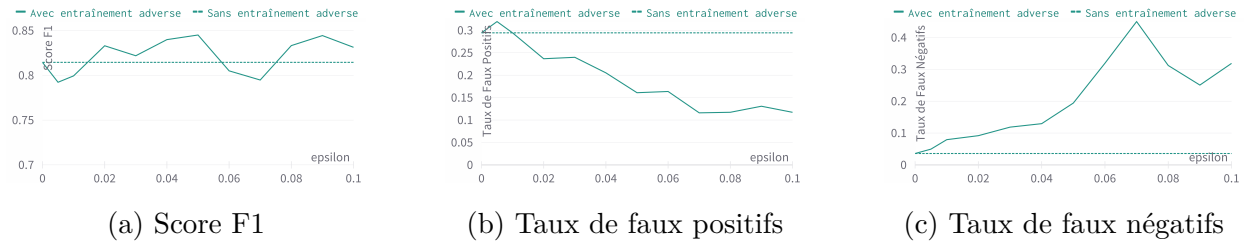


FIGURE 5.6 Évolution des performances l’agent DQN (2 couches, 64 neurones par couches) en fonction de l’amplitude ϵ des perturbations utilisées pour l’entraînement adverse. En pointillés, le seuil de référence des performances de l’agent avant réentraînement.

La Figure 5.6 montre les performances de l’agent sélectionné réentraîné avec des instances

plus ou moins éloignées de celles du jeu de données initiales, ϵ variant de 0.005 à 0.1. On observe, dans l'ensemble, une légère amélioration du score F1 des agents réentraînés, excepté pour les valeurs de ϵ inférieures à 0.02. Il semble donc que l'entraînement adverse puisse améliorer les performances de l'agent, même sans tenir compte de sa robustesse.

En observant ensuite les courbes correspondant au TFP et au TFN, on remarque une tendance marquée : plus la perturbation des exemples utilisés pendant l'entraînement est grande, plus le taux de faux positifs diminue, tandis que le taux de faux négatifs augmente. Ainsi, l'entraînement adverse pourrait permettre de rééquilibrer le TFP par rapport au TFN en gardant un score F1 presque constant : par exemple, en réentraînant les agents avec des perturbation de 5%, on obtient un score F1 de 0.84 au lieu de 0.81, ainsi qu'un TFP de 0.16 au lieu de 0.3. L'entraînement adverse a donc diminué de moitié le taux d'attaques non détectées, cependant le taux de fausses alertes a légèrement augmenté, passant de 0.04 à 0.15.

Choisir la taille de perturbation utilisée lors de l'entraînement adverse revient alors à faire un compromis entre diminuer le nombre d'attaques manquées (TFP) et augmenter le nombre de fausses alertes (TFN). On peut toutefois conclure que, vis-à-vis du score F1, les performances de l'agent ne sont pas diminuées par l'entraînement adverse, ce qui ne vient donc pas contre-indiquer son utilisation pour la défense contre les attaques adverses.

Résistance aux attaques adverses

Après avoir observé l'impact de l'entraînement adverse sur les performances brutes du modèle et nous être assurés que cette technique ne compromet pas la détection d'intrusion, nous pouvons nous intéresser à la robustesse des modèles réentraînés.

Pour ce faire, nous avons testé les différents agents réentraînés face à deux types d'exemples adverses générés par PGD et DeepFool, avec une taille de perturbation de 0.1.

La Figure 5.7 présente une comparaison des performances des agents réentraînés face à l'attaque par exemples adverses PGD. On peut observer que les agents réentraînés sont globalement moins vulnérables à cette attaque que l'agent original, puisqu'ils affichent un score F1 plus élevé et un taux de faux positifs (TFP) plus faible. De plus, il est intéressant de noter que la robustesse des agents face à PGD semble augmenter avec la taille de perturbation utilisée pendant l'entraînement adverse. Cela suggère que l'entraînement adverse a effectivement contribué à améliorer la résistance des agents contre les perturbations malveillantes.

Cependant, malgré le réentraînement, les agents restent sensibles à l'attaque PGD, avec un score F1 inférieur à celui de l'agent original non attaqué, ce qui met en doute l'efficacité de l'entraînement adverse pour fournir une protection complète contre cette attaque. De plus, il

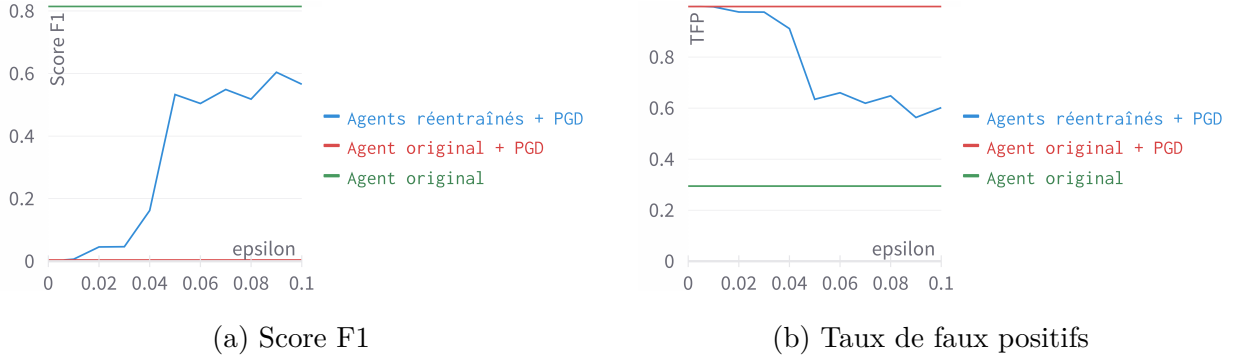


FIGURE 5.7 Performance des agents réentraînés face à l'attaque PGD, en fonction de la taille de la perturbation ϵ utilisée pour l'entraînement adverse (bleu), comparée à l'agent original attaqué (rouge) et l'agent original sain (vert).

est important de noter que les mécanismes de calcul de PGD reposent sur des itérations de la méthode FGSM, ce qui crée un lien étroit entre ces deux approches d'attaques adverses. Ainsi, il devient pertinent d'explorer la robustesse des agents réentraînés face à un autre type d'attaque, comme DeepFool, dont les perturbations sont générées de manière géométrique à partir de projections plutôt que par une simple descente de gradient. Cette investigation nous permettra de mieux comprendre l'efficacité de l'entraînement adverse et d'identifier les types d'attaques pour lesquels il offre une protection optimale.

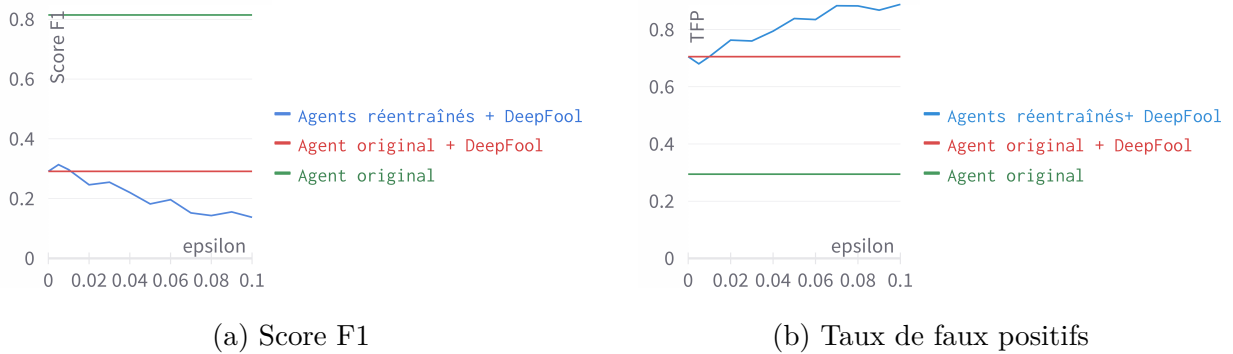


FIGURE 5.8 Performance des agents réentraînés face à l'attaque DeepFool en fonction de la taille de la perturbation ϵ utilisée pour l'entraînement adverse (bleu), comparée à l'agent original attaqué (rouge) et l'agent original sain (vert).

Les résultats observés dans la Figure 5.8 présentent une tendance opposée à ceux de la Figure 5.7. En effet, les agents réentraînés avec des exemples adverses générés par FGSM semblent plus vulnérables à l'attaque de type DeepFool que l'agent initial. De plus, nous constatons que

la robustesse des agents réentraînés est inversement proportionnelle à la taille de perturbation utilisée pendant l’entraînement adverse. En d’autres termes, plus la perturbation utilisée pour l’entraînement adverse est importante, plus les agents réentraînés deviennent sensibles à l’attaque DeepFool. Ces résultats soulèvent une question importante sur l’efficacité de l’entraînement adverse comme méthode de défense contre ce type d’attaque. Il semblerait que dans ce cas spécifique, l’entraînement adverse avec des exemples générés par FGSM n’ait pas réussi à conférer aux agents une meilleure robustesse face à l’attaque DeepFool.

5.3.3 Conclusion

Ces expériences montrent que l’entraînement adverse pour les agents RL de détection d’intrusion peut s’avérer prometteur, mais a également des limites à prendre en compte.

Les résultats obtenus ont mis en évidence que l’entraînement adverse par FGSM permet d’accroître la résistance des agents RL aux attaques FGSM et PGD, deux méthodes basées sur une descente de gradient. Les agents réentraînés ont montré des performances supérieures en termes de score F1 et de taux de faux positifs face à ces attaques, démontrant ainsi l’efficacité de cette approche pour améliorer la robustesse.

Cependant, il est important de noter que malgré l’entraînement adverse, les agents restent sensibles à certaines attaques comme ici DeepFool, et que leur performance n’atteint pas toujours le niveau de l’agent original non attaqué. Cela souligne les limites de l’entraînement adverse en tant que solution universelle pour la protection contre toutes les formes d’attaques adverses.

De plus, notre étude a également révélé que l’efficacité de l’entraînement adverse peut varier en fonction du type d’attaque utilisé pendant l’entraînement et de la taille de perturbation appliquée. Certains types d’attaques peuvent se révéler plus difficiles à contrer que d’autres, ce qui souligne l’importance de développer des méthodes de défense adaptées à chaque type d’attaque.

En conclusion, l’entraînement adverse représente une approche prometteuse pour renforcer la robustesse des agents RL dans les systèmes de détection d’intrusion face aux exemples adverses. Cependant, pour une protection complète, il est essentiel de considérer différentes méthodes d’attaques et de développer des stratégies de défense spécifiques à chaque scénario. Cette recherche contribue à notre compréhension des mécanismes de défense contre les attaques adverses dans le contexte des agents RL pour les SDI, ouvrant la voie à de futures améliorations pour garantir la sécurité et la fiabilité des systèmes cyber-physiques.

CHAPITRE 6 CONCLUSION

La conclusion de ce mémoire marque la fin de notre parcours d'étude sur la robustesse des modèles de détection d'intrusion par RL dans un contexte adverse. Au cours de ce travail, nous avons exploré différentes approches pour développer des agents RL capables de détecter efficacement les attaques malveillantes dans des environnements complexes et dynamiques.

Dans la première partie de cette conclusion, nous synthétiserons les principales contributions de notre recherche, en mettant en évidence les résultats marquants obtenus tout au long de ce mémoire. Nous reviendrons sur l'intégration du RL dans les systèmes de détection d'intrusion, l'étude de la vulnérabilité de ces agents face aux attaques adverses, ainsi que l'efficacité de l'entraînement adverse comme méthode de défense pour renforcer leur robustesse.

Dans la deuxième partie, nous discuterons des limites de notre étude. Nous soulignerons les défis rencontrés lors de l'utilisation du RL dans des environnements cyber-physiques complexes, notamment en ce qui concerne la gestion du déséquilibre du jeu de données et la sensibilité aux perturbations adverses. En identifiant ces limitations, nous visons à fournir une vision plus globale et réaliste des performances de nos modèles.

Enfin, dans la troisième partie, nous explorerons les pistes d'amélioration futures pour renforcer davantage l'efficacité et la robustesse des agents RL dans les SDI. Nous aborderons des questions telles que l'utilisation de nouvelles architectures de réseaux neuronaux, l'optimisation des hyperparamètres, l'intégration d'autres approches de défense avancées, ainsi que l'approche métier spécialisée.

Ce chapitre final est donc l'occasion de récapituler nos découvertes et de proposer des perspectives pour les futures recherches en matière de détection d'intrusion par RL. Nous espérons que ce mémoire contribuera à approfondir les connaissances dans ce domaine critique et permettra de renforcer la sécurité et la fiabilité des réseaux informatiques dans un contexte de plus en plus complexe et menaçant.

6.1 Synthèse des travaux

Les travaux réalisés dans ce mémoire portent sur l'intégration du RL dans les systèmes de détection d'intrusion, et plus particulièrement sur sa pertinence dans un contexte adverse. Nous avons étudié la performance des agents RL en utilisant deux jeux de données de réseaux informatiques, AWID et NSL-KDD, ainsi qu'un jeu de données cyber-physiques. Nous avons alors pu conclure que l'approche RL n'est pas encore adaptée aux données CPS, mais présente

des performances prometteuses pour les données réseau. Cependant, nous avons également mis en évidence la vulnérabilité des modèles RL face aux attaques adverses, ce qui remet en question leur fiabilité dans des environnements réels.

Nous avons poursuivi en proposant des mécanismes pour améliorer les performances de nos agents. En particulier, l'utilisation d'un environnement adverse pendant l'entraînement permet d'améliorer fortement les capacités de détection des modèles. De plus, nous avons constaté que les choix d'architecture du réseau ainsi que le format des données peuvent influencer considérablement la robustesse des agents, ce qui appelle à une attention particulière lors du développement de solutions de détection d'intrusion basées sur le RL.

Enfin, nous nous sommes intéressés à une approche défensive spécifique pour protéger les agents contre les attaques par exemples adverses : l'entraînement adverse, qui consiste à utiliser des instances modifiées pendant l'entraînement. Il est apparu que cette méthode de défense permet en effet de rendre les agents robustes face à certains types d'exemples adverses, mais ne se généralise pas à toutes les perturbations. Nous avons donc montré que l'entraînement adverse est une solution de défense pertinente pour les agents SDI par RL, mais nécessite une étude poussée de ses conséquences sur les performances des agents, et n'est pas auto-suffisante pour assurer leur robustesse.

Au cours de ce travail de recherche, nous avons en parallèle produit des outils d'analyse en libre accès destiné à aider de futurs projets sur le sujet : tout d'abord, nous avons fourni une taxonomie détaillée des méthodes d'exemples adverses dans le contexte du RL. De plus, nous avons construit une bibliothèque Python permettant d'implémenter facilement ces attaques avec les principales bibliothèques RL modernes.

En conclusion, ce mémoire permet de contribuer à une meilleure compréhension de l'utilisation du RL dans les SDI et à l'identification de pistes d'amélioration pour renforcer la robustesse de ces modèles.

6.2 Limitations

Malgré les avancées réalisées dans ce mémoire, plusieurs limitations doivent être prises en compte. Tout d'abord, les performances des modèles RL dépendent étroitement des choix d'architecture du réseau et des hyperparamètres. Bien que nous ayons exploré différentes configurations, il est possible que des architectures plus complexes ou des hyperparamètres optimisés encore plus finement puissent améliorer davantage la performance des modèles. Ces optimisations nécessitent cependant une expertise et un temps d'étude qui peuvent s'avérer coûteux dans le cadre d'une application industrielle.

Une autre limitation réside dans la nature des jeux de données utilisés. Bien que nous ayons utilisé des jeux de données couramment utilisés dans le domaine de la détection d'intrusion, ils peuvent ne pas être représentatifs de tous les scénarios réels. Les SDI sont confrontés à des environnements de réseau complexes et en constante évolution, ce qui peut nécessiter des jeux de données plus diversifiés et spécifiques aux contextes réels. Nous soulignons donc ici le besoin de jeux de données modernes et complets pour l'avancée de la recherche en détection d'intrusion.

De plus, notre approche d'entraînement adverse s'est avérée efficace face à certains exemples adverses, mais elle n'a pas permis d'améliorer la robustesse de nos agents contre toutes les attaques. L'entraînement adverse est une approche complexe et il existe de nombreuses manières de le mettre en œuvre. Des recherches supplémentaires sont nécessaires pour explorer d'autres méthodes d'entraînement adverse, ou encore combiner les combiner avec d'autres défenses connues.

Enfin, bien que nous ayons exploré l'utilisation du RL dans les SDI, il est important de noter que cette approche n'est pas dépourvue de défis et de limites propres au RL. Les techniques de RL peuvent être sensibles à l'initialisation des poids du modèle, aux variations de l'environnement et aux stratégies d'exploration. Nous avons ainsi noté que dans le cas des données cyber-physiques, le RL n'est pas la méthode la plus adaptée pour traiter les données, l'apprentissage ne parvenant pas à faire converger l'agent vers une solution satisfaisante. Les études futures devront tenir compte de ces défis et les surmonter pour rendre les modèles RL encore plus fiables et efficaces dans des environnements de détection d'intrusion réels.

6.3 Améliorations futures

Plusieurs pistes d'améliorations peuvent être envisagées pour poursuivre nos travaux sur la robustesse des modèles RL dans le contexte des SDI. Tout d'abord, une approche prometteuse serait de combiner différentes méthodes de défense pour obtenir une meilleure protection globale. Par exemple, l'entraînement adverse pourrait être utilisé en complément d'autres techniques de défense, telles que la régularisation, la distillation de modèle, ou l'utilisation de mécanismes de détection d'anomalies pour identifier les instances potentiellement malveillantes. Si la combinaison de défenses n'est pas toujours une solution idéale, dans ce cas d'application particulier la diversité des facteurs à prendre en compte pour surveiller un réseau informatique permet de justifier une parallélisation de méthodes défensives.

Une autre piste d'amélioration consiste à envisager une approche hybride combinant les algorithmes RL avec une approche métier plus analytique, par exemple avec des signatures.

Les signatures sont des règles préétablies qui permettent de détecter des attaques spécifiques et bien connues. En utilisant une approche hybride, les modèles RL pourraient bénéficier de la puissance d'apprentissage automatique du RL tout en exploitant les connaissances expertes fournies par les signatures pour détecter certaines attaques connues de manière plus fiable.

Par ailleurs, l'expérimentation avec d'autres algorithmes RL pourrait être envisagée pour comparer leur robustesse et leurs performances dans le contexte des SDI. Des algorithmes tels qu'A3C ou TRPO pourraient être testés pour évaluer leur capacité à détecter et à se défendre contre les exemples adverses.

Enfin, il serait également intéressant de développer des approches de génération d'exemples adverses spécifiquement conçues pour les environnements de détection d'intrusion. Actuellement, les méthodes de génération d'exemples adverses sont généralement conçues pour des tâches de classification d'images, et leur application aux SDI peut être limitée. Des recherches supplémentaires pourraient être entreprises pour développer des méthodes d'attaques plus ciblées et adaptées aux spécificités des données cyber-physiques, et ainsi construire une stratégie de défense mieux adaptée.

En conclusion, ce mémoire a ouvert la voie à de nouvelles perspectives passionnantes dans le domaine de la détection d'intrusion par RL. Les améliorations futures envisagées, telles que la combinaison de méthodes de défense, l'approche hybride avec des signatures, l'utilisation d'autres algorithmes RL, et le développement d'approches d'attaques plus spécifiques, permettront de renforcer la robustesse des modèles RL et de les rendre encore plus performants dans des environnements réels de détection d'intrusion. Ces avancées pourront contribuer à accroître la sécurité des systèmes cyber-physiques face aux attaques malveillantes.

RÉFÉRENCES

- [1] J. Goh, S. Adepu, K. N. Junejo et A. Mathur, “A Dataset to Support Research in the Design of Secure Water Treatment Systems,” dans *Critical Information Infrastructures Security*, ser. Lecture Notes in Computer Science, G. Havarneanu, R. Setola, H. Nas-sopoulos et S. Wolthusen, édit. Cham : Springer International Publishing, 2017, p. 88–99.
- [2] P. Fraga-Lamas, T. M. Fernández-Caramés, M. Suárez-Albela, L. Castedo et M. González-López, “A Review on Internet of Things for Defense and Public Safety,” *Sensors*, vol. 16, n°. 10, p. 1644, oct. 2016, number : 10 Publisher : Multidisciplinary Digital Publishing Institute. [En ligne]. Disponible : <https://www.mdpi.com/1424-8220/16/10/1644>
- [3] gmcDouga, “Check Point Research Reports a 38% Increase in 2022 Global Cyberattacks,” janv. 2023. [En ligne]. Disponible : <https://blog.checkpoint.com/2023/01/05/38-increase-in-2022-global-cyberattacks/>
- [4] Ajay Jotwani, “Why Cybersecurity Should Be Top Of Mind In 2023,” 2023. [En ligne]. Disponible : <https://www.forbes.com/sites/forbestechcouncil/2023/01/06/why-cybersecurity-should-be-top-of-mind-in-2023/?sh=1654d131235c>
- [5] “Stuxnet : Dissecting a Cyberwarfare Weapon | IEEE Journals & Magazine | IEEE Xplore.” [En ligne]. Disponible : <https://ieeexplore.ieee.org/document/5772960>
- [6] R. A. Kemmerer et G. Vigna, “Intrusion Detection : A Brief History and Overview (Supplement to Computer Magazine),” *Computer*, vol. 35, n°. 04, p. 27–30, avr. 2002, publisher : IEEE Computer Society. [En ligne]. Disponible : <https://www.computer.org/csdl/magazine/co/2002/04/r4s27/13rRUIJcWgL>
- [7] V. Kumar et D. O. P. Sangwan, “Signature Based Intrusion Detection System Using SNORT,” *International Journal of Computer Applications*, 2012.
- [8] H. Debar, M. Becker et D. Siboni, “A neural network component for an intrusion detection system,” dans *Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, mai 1992, p. 240–250.
- [9] A. Khraisat, I. Gondal, P. Vamplew et J. Kamruzzaman, “Survey of intrusion detection systems : techniques, datasets and challenges,” *Cybersecurity*, vol. 2, n°. 1, p. 20, juill. 2019. [En ligne]. Disponible : <https://doi.org/10.1186/s42400-019-0038-7>

- [10] S. Levine, C. Finn, T. Darrell et P. Abbeel, “End-to-End Training of Deep Visuomotor Policies,” avr. 2016, arXiv :1504.00702 [cs]. [En ligne]. Disponible : <http://arxiv.org/abs/1504.00702>
- [11] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani et P. Pérez, “Deep Reinforcement Learning for Autonomous Driving : A Survey,” janv. 2021, arXiv :2002.00444 [cs]. [En ligne]. Disponible : <http://arxiv.org/abs/2002.00444>
- [12] A. M. Pasikhani, J. A. Clark et P. Gope, “Adversarial RL-Based IDS for Evolving Data Environment in 6LoWPAN,” *IEEE Transactions on Information Forensics and Security*, vol. 17, p. 3831–3846, 2022, conference Name : IEEE Transactions on Information Forensics and Security.
- [13] R. S. Sutton et A. G. Barto, *Reinforcement Learning, second edition : An Introduction*. MIT Press, nov. 2018, google-Books-ID : uWV0DwAAQBAJ.
- [14] S. Huang, N. Papernot, I. Goodfellow, Y. Duan et P. Abbeel, “Adversarial Attacks on Neural Network Policies,” févr. 2017, arXiv :1702.02284 [cs, stat]. [En ligne]. Disponible : <http://arxiv.org/abs/1702.02284>
- [15] N. Papernot, P. McDaniel, X. Wu, S. Jha et A. Swami, “Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks,” mars 2016, arXiv :1511.04508 [cs, stat]. [En ligne]. Disponible : <http://arxiv.org/abs/1511.04508>
- [16] Y.-C. Lin, M.-Y. Liu, M. Sun et J.-B. Huang, “Detecting Adversarial Attacks on Neural Network Policies with Visual Foresight,” oct. 2017, arXiv :1710.00814 [cs]. [En ligne]. Disponible : <http://arxiv.org/abs/1710.00814>
- [17] Z. Lin, Y. Shi et Z. Xue, “IDSGAN : Generative Adversarial Networks for Attack Generation Against Intrusion Detection,” dans *Advances in Knowledge Discovery and Data Mining*, ser. Lecture Notes in Computer Science, J. Gama, T. Li, Y. Yu, E. Chen, Y. Zheng et F. Teng, édit. Cham : Springer International Publishing, 2022, p. 79–91.
- [18] H. Qiu, T. Dong, T. Zhang, J. Lu, G. Memmi et M. Qiu, “Adversarial Attacks Against Network Intrusion Detection in IoT Systems,” *IEEE Internet of Things Journal*, vol. 8, n°. 13, p. 10 327–10 335, juill. 2021, conference Name : IEEE Internet of Things Journal.
- [19] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel et D. Hassabis, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, n°. 7587, p. 484–489, janv. 2016, number : 7587 Publisher : Nature Publishing Group. [En ligne]. Disponible : <https://www.nature.com/articles/nature16961>

- [20] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps et D. Silver, “Grandmaster level in StarCraft II using multi-agent reinforcement learning,” *Nature*, vol. 575, n°. 7782, p. 350–354, nov. 2019, number : 7782 Publisher : Nature Publishing Group. [En ligne]. Disponible : <https://www.nature.com/articles/s41586-019-1724-z>
- [21] T. T. Wang, A. Gleave, N. Belrose, T. Tseng, J. Miller, M. D. Dennis, Y. Duan, V. Pogrebniak, S. Levine et S. Russell, “Adversarial Policies Beat Professional-Level Go AIs,” oct. 2022, arXiv :2211.00241 [cs, stat]. [En ligne]. Disponible : <http://arxiv.org/abs/2211.00241>
- [22] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin et K.-Y. Tung, “Intrusion detection system : A comprehensive review,” *Journal of Network and Computer Applications*, vol. 36, n°. 1, p. 16–24, janv. 2013. [En ligne]. Disponible : <https://www.sciencedirect.com/science/article/pii/S1084804512001944>
- [23] B. Gibson, S. Townes, D. Lewis et S. Bhunia, “Vulnerability in massive api scraping : 2021 linkedin data breach,” *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, p. 777–782, 2021. [En ligne]. Disponible : https://ieeexplore.ieee.org/abstract/document/9799221/?casa_token=ypzFdddgKegAAAAA:eCGIxUNiGYYHXk6BYu7KRVs3iFUx07aLhVf7HMChRhHNFE03ODzmfaksR5Deo7yeCdZIVqsVrF
- [24] C. Kreibich et J. Crowcroft, “Honeycomb : creating intrusion detection signatures using honeypots,” *ACM SIGCOMM Computer Communication Review*, vol. 34, n°. 1, p. 51–56, janv. 2004. [En ligne]. Disponible : <https://dl.acm.org/doi/10.1145/972374.972384>
- [25] A. Sung et S. Mukkamala, “Identifying important features for intrusion detection using support vector machines and neural networks,” dans *2003 Symposium on Applications and the Internet, 2003. Proceedings.*, janv. 2003, p. 209–216.
- [26] M. Ring, S. Wunderlich, D. Scheuring, D. Landes et A. Hotho, “A survey of network-based intrusion detection data sets,” *Computers & Security*, vol. 86, p. 147–167, sept. 2019. [En ligne]. Disponible : <https://www.sciencedirect.com/science/article/pii/S016740481930118X>
- [27] A. Tanenbaum et D. Wetherall, *Computer Networks*, 5^e éd. Boston : Pearson, sept. 2010.

- [28] B. Claise, “Cisco Systems NetFlow Services Export Version 9,” Internet Engineering Task Force, Request for Comments RFC 3954, oct. 2004, num Pages : 33. [En ligne]. Disponible : <https://datatracker.ietf.org/doc/rfc3954>
- [29] M. Tavallae, E. Bagheri, W. Lu et A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” dans *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, juill. 2009, p. 1–6, iSSN : 2329-6275.
- [30] A. Thakkar et R. Lohiya, “A survey on intrusion detection system : feature selection, model, performance measures, application perspective, challenges, and future research directions,” *Artificial Intelligence Review*, vol. 55, n°. 1, p. 453–563, janv. 2022. [En ligne]. Disponible : <https://doi.org/10.1007/s10462-021-10037-9>
- [31] C. Koliass, G. Kambourakis, A. Stavrou et S. Gritzalis, “Intrusion Detection in 802.11 Networks : Empirical Evaluation of Threats and a Public Dataset,” *IEEE Communications Surveys & Tutorials*, vol. 18, n°. 1, p. 184–208, 2016, conference Name : IEEE Communications Surveys & Tutorials.
- [32] A. P. Mathur et N. O. Tippenhauer, “SWaT : a water treatment testbed for research and training on ICS security,” dans *2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*, avr. 2016, p. 31–36.
- [33] S. Chebrolu, A. Abraham et J. P. Thomas, “Feature deduction and ensemble design of intrusion detection systems,” *Computers & Security*, vol. 24, n°. 4, p. 295–307, juin 2005. [En ligne]. Disponible : <https://www.sciencedirect.com/science/article/pii/S016740480400238X>
- [34] K. Bajaj et 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*, vol. 76, p. 5–11, août 2013.
- [35] C. Annachhatre, T. H. Austin et M. Stamp, “Hidden Markov models for malware classification,” *Journal of Computer Virology and Hacking Techniques*, vol. 11, n°. 2, p. 59–73, mai 2015. [En ligne]. Disponible : <https://doi.org/10.1007/s11416-014-0215-x>
- [36] A. L. Buczak et E. Guven, “A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection,” *IEEE Communications Surveys & Tutorials*, vol. 18, n°. 2, p. 1153–1176, 2016, conference Name : IEEE Communications Surveys & Tutorials.
- [37] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra et M. Riedmiller, “Playing Atari with Deep Reinforcement Learning,” déc. 2013, arXiv :1312.5602 [cs]. [En ligne]. Disponible : <http://arxiv.org/abs/1312.5602>

- [38] H. van Hasselt, A. Guez et D. Silver, “Deep Reinforcement Learning with Double Q-learning,” déc. 2015, arXiv :1509.06461 [cs]. [En ligne]. Disponible : <http://arxiv.org/abs/1509.06461>
- [39] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine Learning*, vol. 8, n°. 3, p. 229–256, mai 1992. [En ligne]. Disponible : <https://doi.org/10.1007/BF00992696>
- [40] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver et K. Kavukcuoglu, “Asynchronous Methods for Deep Reinforcement Learning,” juin 2016, arXiv :1602.01783 [cs]. [En ligne]. Disponible : <http://arxiv.org/abs/1602.01783>
- [41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford et O. Klimov, “Proximal Policy Optimization Algorithms,” août 2017, arXiv :1707.06347 [cs]. [En ligne]. Disponible : <http://arxiv.org/abs/1707.06347>
- [42] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver et D. Wierstra, “Continuous control with deep reinforcement learning,” juill. 2019, arXiv :1509.02971 [cs, stat]. [En ligne]. Disponible : <http://arxiv.org/abs/1509.02971>
- [43] A. Servin, “Towards Traffic Anomaly Detection via Reinforcement Learning and Data Flow,” 2007. [En ligne]. Disponible : <https://www.semanticscholar.org/paper/Towards-Traffic-Anomaly-Detection-via-Reinforcement-Servin/149c832a5a5d0eadfb9f23aafb035e14b618fb0c>
- [44] X. Xu et T. Xie, “A Reinforcement Learning Approach for Host-Based Intrusion Detection Using Sequences of System Calls,” dans *Advances in Intelligent Computing*, ser. Lecture Notes in Computer Science, D.-S. Huang, X.-P. Zhang et G.-B. Huang, édit. Berlin, Heidelberg : Springer, 2005, p. 995–1003.
- [45] R. S. Sutton, “Learning to predict by the methods of temporal differences,” *Machine Learning*, vol. 3, n°. 1, p. 9–44, août 1988. [En ligne]. Disponible : <https://doi.org/10.1007/BF00115009>
- [46] A. Servin et D. Kudenko, “Multi-agent Reinforcement Learning for Intrusion Detection,” dans *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*, ser. Lecture Notes in Computer Science, K. Tuyls, A. Nowe, Z. Guessoum et D. Kudenko, édit. Berlin, Heidelberg : Springer, 2008, p. 211–223.
- [47] B. Deokar et A. Hazarnis, “Intrusion Detection System using Log Files and Reinforcement Learning,” *International Journal of Computer Applications*, vol. 45, 2012.
- [48] M. Lopez-Martin, B. Carro et A. Sanchez-Esguevillas, “Application of deep reinforcement learning to intrusion detection for supervised problems,” *Expert*

- Systems with Applications*, vol. 141, p. 112963, mars 2020. [En ligne]. Disponible : <https://linkinghub.elsevier.com/retrieve/pii/S0957417419306815>
- [49] G. Caminero, M. Lopez-Martin et B. Carro, “Adversarial environment reinforcement learning algorithm for intrusion detection,” *Computer Networks*, vol. 159, p. 96–109, août 2019. [En ligne]. Disponible : <https://www.sciencedirect.com/science/article/pii/S1389128618311216>
- [50] S. Tharewal, M. W. Ashfaq, S. S. Banu, P. Uma, S. M. Hassen et M. Shabaz, “Intrusion Detection System for Industrial Internet of Things Based on Deep Reinforcement Learning,” *Wireless Communications and Mobile Computing*, vol. 2022, p. e9023719, mars 2022, publisher : Hindawi. [En ligne]. Disponible : <https://www.hindawi.com/journals/wcmc/2022/9023719/>
- [51] V. Praveena, A. Vijayaraj, C. Ponnusamy, A. Ihsan, R. Alroobaea, S. Yahya et M. Raza, “Optimal Deep Reinforcement Learning for Intrusion Detection in UAVs,” *Computers, Materials and Continua*, vol. 70, p. 2639–2653, sept. 2021.
- [52] E. Tabassi, K. Burns, M. Hadjimichael, A. Molina-Markham et J. Sexton, “A Taxonomy and Terminology of Adversarial Machine Learning,” National Institute of Standards and Technology, Rapport technique NIST Internal or Interagency Report (NISTIR) 8269 (Draft), oct. 2019. [En ligne]. Disponible : <https://csrc.nist.gov/publications/detail/nistir/8269/draft>
- [53] S. Dahmen-Lhuissier, “ETSI - Best Security Standards | ETSI Security Standards.” [En ligne]. Disponible : <https://www.etsi.org/technologies/securing-artificial-intelligence>
- [54] L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein et J. D. Tygar, “Adversarial Machine Learning.”
- [55] M. Barreno, B. Nelson, A. D. Joseph et J. D. Tygar, “The security of machine learning,” *Machine Learning*, vol. 81, n°. 2, p. 121–148, nov. 2010. [En ligne]. Disponible : <https://doi.org/10.1007/s10994-010-5188-5>
- [56] X. Yuan, P. He, Q. Zhu et X. Li, “Adversarial Examples : Attacks and Defenses for Deep Learning,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, n°. 9, p. 2805–2824, sept. 2019. [En ligne]. Disponible : <https://ieeexplore.ieee.org/document/8611298/>
- [57] V. Behzadan et A. Munir, “Vulnerability of Deep Reinforcement Learning to Policy Induction Attacks,” dans *Machine Learning and Data Mining in Pattern Recognition*, ser. Lecture Notes in Computer Science, P. Perner, édit. Cham : Springer International Publishing, 2017, p. 262–275.

- [58] I. Ilahi, M. Usama, J. Qadir, M. U. Janjua, A. Al-Fuqaha, D. T. Hoang et D. Niyato, “Challenges and Countermeasures for Adversarial Attacks on Deep Reinforcement Learning,” *IEEE Transactions on Artificial Intelligence*, vol. 3, n^o. 2, p. 90–109, avr. 2022, conference Name : IEEE Transactions on Artificial Intelligence.
- [59] A. Kurakin, I. Goodfellow et S. Bengio, “Adversarial examples in the physical world,” févr. 2017, arXiv :1607.02533 [cs, stat]. [En ligne]. Disponible : <http://arxiv.org/abs/1607.02533>
- [60] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan et G. Chowdhary, “Robust Deep Reinforcement Learning with Adversarial Attacks,” déc. 2017, arXiv :1712.03632 [cs]. [En ligne]. Disponible : <http://arxiv.org/abs/1712.03632>
- [61] A. Madry, A. Makelov, L. Schmidt, D. Tsipras et A. Vladu, “Towards Deep Learning Models Resistant to Adversarial Attacks,” sept. 2019, arXiv :1706.06083 [cs, stat]. [En ligne]. Disponible : <http://arxiv.org/abs/1706.06083>
- [62] N. Carlini et D. Wagner, “Towards Evaluating the Robustness of Neural Networks,” dans *2017 IEEE Symposium on Security and Privacy (SP)*, mai 2017, p. 39–57, iSSN : 2375-1207.
- [63] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik et A. Swami, “The Limitations of Deep Learning in Adversarial Settings,” dans *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, mars 2016, p. 372–387.
- [64] V. Behzadan et W. Hsu, “Adversarial Exploitation of Policy Imitation,” juin 2019, arXiv :1906.01121 [cs, stat]. [En ligne]. Disponible : <http://arxiv.org/abs/1906.01121>
- [65] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, G. Dulac-Arnold, J. Agapiou, J. Z. Leibo et A. Gruslys, “Deep Q-learning from demonstrations,” dans *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI’18/IAAI’18/EAAI’18. New Orleans, Louisiana, USA : AAAI Press, 2018, p. 3223–3230.
- [66] J. Lin, K. Dzeparoska, S. Q. Zhang, A. Leon-Garcia et N. Papernot, “On the Robustness of Cooperative Multi-Agent Reinforcement Learning,” mars 2020, arXiv :2003.03722 [cs, stat]. [En ligne]. Disponible : <http://arxiv.org/abs/2003.03722>
- [67] H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. Boning et C.-J. Hsieh, “Robust Deep Reinforcement Learning against Adversarial Perturbations on State Observations,” dans *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates,

- Inc., 2020, p. 21 024–21 037. [En ligne]. Disponible : <https://proceedings.neurips.cc/paper/2020/hash/f0eb6568ea114ba6e293f903c34d7488-Abstract.html>
- [68] L. Hussenot, M. Geist et O. Pietquin, “CopyCAT : Taking Control of Neural Policies with Constant Attacks,” janv. 2020, arXiv :1905.12282 [cs, stat]. [En ligne]. Disponible : <http://arxiv.org/abs/1905.12282>
- [69] B. G. A. Tekgul, S. Wang, S. Marchal et N. Asokan, “Real-time Adversarial Perturbations against Deep Reinforcement Learning Policies : Attacks and Defenses,” sept. 2022, arXiv :2106.08746 [cs]. [En ligne]. Disponible : <http://arxiv.org/abs/2106.08746>
- [70] K. Mo, W. Tang, J. Li et X. Yuan, “Attacking Deep Reinforcement Learning with Decoupled Adversarial Policy,” *IEEE Transactions on Dependable and Secure Computing*, p. 1–1, 2022, conference Name : IEEE Transactions on Dependable and Secure Computing.
- [71] E. Tretschk, S. J. Oh et M. Fritz, “Sequential Attacks on Agents for Long-Term Adversarial Goals,” juill. 2018, arXiv :1805.12487 [cs, stat]. [En ligne]. Disponible : <http://arxiv.org/abs/1805.12487>
- [72] S. Baluja et I. Fischer, “Adversarial Transformation Networks : Learning to Generate Adversarial Examples,” mars 2017, arXiv :1703.09387 [cs]. [En ligne]. Disponible : <http://arxiv.org/abs/1703.09387>
- [73] A. Russo et A. Proutiere, “Optimal Attacks on Reinforcement Learning Policies,” juill. 2019, arXiv :1907.13548 [cs, stat]. [En ligne]. Disponible : <http://arxiv.org/abs/1907.13548>
- [74] —, “Towards Optimal Attacks on Reinforcement Learning Policies,” dans *2021 American Control Conference (ACC)*, mai 2021, p. 4561–4567, iSSN : 2378-5861.
- [75] L. Pinto, J. Davidson, R. Sukthankar et A. Gupta, “Robust Adversarial Reinforcement Learning,” dans *Proceedings of the 34th International Conference on Machine Learning*. PMLR, juill. 2017, p. 2817–2826, iSSN : 2640-3498. [En ligne]. Disponible : <https://proceedings.mlr.press/v70/pinto17a.html>
- [76] X. Ma, K. Driggs-Campbell et M. J. Kochenderfer, “Improved Robustness and Safety for Autonomous Vehicle Control with Adversarial Reinforcement Learning,” dans *2018 IEEE Intelligent Vehicles Symposium (IV)*, juin 2018, p. 1665–1671, iSSN : 1931-0587.
- [77] X. Pan, D. Seita, Y. Gao et J. Canny, “Risk Averse Robust Adversarial Reinforcement Learning,” dans *2019 International Conference on Robotics and Automation (ICRA)*, mai 2019, p. 8522–8528, iSSN : 2577-087X.

- [78] M. Pawlicki, M. Choraś et R. Kozik, “Defending network intrusion detection systems against adversarial evasion attacks,” *Future Generation Computer Systems*, vol. 110, p. 148–154, sept. 2020. [En ligne]. Disponible : <https://www.sciencedirect.com/science/article/pii/S0167739X20303368>
- [79] N. Carlini et D. Wagner, “Adversarial Examples Are Not Easily Detected : Bypassing Ten Detection Methods,” nov. 2017, arXiv :1705.07263 [cs]. [En ligne]. Disponible : <http://arxiv.org/abs/1705.07263>
- [80] M. J. Hashemi et E. Keller, “Enhancing Robustness Against Adversarial Examples in Network Intrusion Detection Systems,” dans *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, nov. 2020, p. 37–43.
- [81] G. Katz, C. Barrett, D. L. Dill, K. Julian et M. J. Kochenderfer, “Reluplex : An Efficient SMT Solver for Verifying Deep Neural Networks,” dans *Computer Aided Verification*, ser. Lecture Notes in Computer Science, R. Majumdar et V. Kunčák, édit. Cham : Springer International Publishing, 2017, p. 97–117.
- [82] —, “Towards Proving the Adversarial Robustness of Deep Neural Networks,” *Electronic Proceedings in Theoretical Computer Science*, vol. 257, p. 19–26, sept. 2017, arXiv :1709.02802 [cs, stat]. [En ligne]. Disponible : <http://arxiv.org/abs/1709.02802>
- [83] G. Hinton, O. Vinyals et J. Dean, “Distilling the Knowledge in a Neural Network,” mars 2015, arXiv :1503.02531 [cs, stat]. [En ligne]. Disponible : <http://arxiv.org/abs/1503.02531>
- [84] J. Bradshaw, A. G. d. G. Matthews et Z. Ghahramani, “Adversarial Examples, Uncertainty, and Transfer Testing Robustness in Gaussian Process Hybrid Deep Networks,” juill. 2017, arXiv :1707.02476 [stat]. [En ligne]. Disponible : <http://arxiv.org/abs/1707.02476>
- [85] I. J. Goodfellow, J. Shlens et C. Szegedy, “Explaining and Harnessing Adversarial Examples,” mars 2015, arXiv :1412.6572 [cs, stat]. [En ligne]. Disponible : <http://arxiv.org/abs/1412.6572>
- [86] R. Huang, B. Xu, D. Schuurmans et C. Szepesvari, “Learning with a Strong Adversary,” janv. 2016, arXiv :1511.03034 [cs]. [En ligne]. Disponible : <http://arxiv.org/abs/1511.03034>
- [87] Y. Wu, D. Bamman et S. Russell, “Adversarial Training for Relation Extraction,” dans *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark : Association for Computational Linguistics, sept. 2017, p. 1778–1783. [En ligne]. Disponible : <https://aclanthology.org/D17-1187>

- [88] R. A. Khamis et A. Matrawy, “Evaluation of Adversarial Training on Different Types of Neural Networks in Deep Learning-based IDSs,” dans *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, oct. 2020, p. 1–6.
- [89] I. Debicha, T. Debatty, J.-M. Dricot et W. Mees, “Adversarial Training for Deep Learning-based Intrusion Detection Systems,” avr. 2021, arXiv :2104.09852 [cs]. [En ligne]. Disponible : <http://arxiv.org/abs/2104.09852>
- [90] M. M. Rashid, J. Kamruzzaman, M. Mehedi Hassan, T. Imam, S. Wibowo, S. Gordon et G. Fortino, “Adversarial training for deep learning-based cyberattack detection in IoT-based smart city applications,” *Computers & Security*, vol. 120, p. 102783, sept. 2022. [En ligne]. Disponible : <https://www.sciencedirect.com/science/article/pii/S016740482200178X>
- [91] A. J. Havens, Z. Jiang et S. Sarkar, “Online Robust Policy Learning in the Presence of Unknown Adversaries,” juill. 2018, arXiv :1807.06064 [cs, stat]. [En ligne]. Disponible : <http://arxiv.org/abs/1807.06064>
- [92] Q. Xinghua, Y. Ong, A. Gupta et Z. Sun, *Defending Adversarial Attacks without Adversarial Attacks in Deep Reinforcement Learning*, août 2020.
- [93] A. Kumar, “Enhancing performance of reinforcement learning models in the presence of noisy rewards,” Thesis, avr. 2019, accepted : 2019-09-09T18 :24 :46Z. [En ligne]. Disponible : <https://repositories.lib.utexas.edu/handle/2152/75758>
- [94] M. Fischer, M. Mirman, S. Stalder et M. Vechev, “Online Robustness Training for Deep Reinforcement Learning,” nov. 2019, arXiv :1911.00887 [cs, stat]. [En ligne]. Disponible : <http://arxiv.org/abs/1911.00887>
- [95] K. Ding, “Real-time Intrusion Detection of Cyber Physical Systems,” 2020. [En ligne]. Disponible : <https://repository.tudelft.nl/islandora/object/uuid%3Aee28b63a-4850-45ba-8e6b-545f8e841646>
- [96] S. Yegulalp, “Facebook brings GPU-powered machine learning to Python,” janv. 2017. [En ligne]. Disponible : <https://www.infoworld.com/article/3159120/facebook-brings-gpu-powered-machine-learning-to-python.html>
- [97] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus et N. Dormann, “Stable-baselines3 : reliable reinforcement learning implementations,” *The Journal of Machine Learning Research*, vol. 22, n^o. 1, p. 268 :12 348–268 :12 355, janv. 2021.
- [98] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. M. Molloy et B. Edwards, “Adversarial Robustness Toolbox v1.0.0,” nov. 2019, arXiv :1807.01069 [cs, stat]. [En ligne]. Disponible : <http://arxiv.org/abs/1807.01069>

- [99] J. D. Hamilton, *Time Series Analysis*, 2020. [En ligne]. Disponible : https://books.google.fr/books?hl=fr&lr=&id=BeryDwAAQBAJ&oi=fnd&pg=PP1&dq=time+Series+Analysis,&ots=BgEV5_aUfl&sig=g_rsgYmzMA1iqV84mD1SfQ0Bhds&redir_esc=y#v=onepage&q=time%20Series%20Analysis%2C&f=false
- [100] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai et T. Chen, “Recent advances in convolutional neural networks,” *Pattern Recognition*, vol. 77, p. 354–377, mai 2018. [En ligne]. Disponible : <https://www.sciencedirect.com/science/article/pii/S0031320317304120>
- [101] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu et R. Hadsell, “Policy Distillation,” janv. 2016, arXiv :1511.06295 [cs]. [En ligne]. Disponible : <http://arxiv.org/abs/1511.06295>