UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

Conception de mécanisme de détection d'intrusion adaptatif basé sur les techniques d'apprentissage par renforcement

MÉMOIRE PRÉSENTÉ

COMME EXIGENCE PARTIELLE DE LA

MAÎTRISE EN INFORMATIQUE

PAR

Ahmed BENIDIR

AVRIL 2026

Abstract

In the face of increasing sophistication in cyber threats and the exponential growth of network traffic, traditional Intrusion Detection Systems (IDS) struggle to maintain an optimal balance between accuracy, processing speed, and decision explainability. The high dimensionality of modern data introduces significant noise and computational latency that compromise real-time detection, particularly in resource-constrained environments such as the Internet of Things (IoT). This thesis proposes an innovative hybrid architecture aimed at optimizing cybersecurity through the synergy between Deep Reinforcement Learning (DRL) and Explainable Artificial Intelligence.

Our approach is based on the development of two interdependent agents, rigorously evaluated on the benchmark CIC-IDS2017 dataset. The first module, named NEMESIS, represents our initial contribution ; leveraging Deep Q-Learning (DQN), it handles the classification of malicious activities by optimizing decision-making processes in the face of intrusions. To enhance the efficiency of this system, we introduce a second module, ULYSSE, a dynamic feature selection mechanism based on the Proximal Policy Optimization (PPO) algorithm. Guided by a reward function incorporating SHAP value theory, this agent learns to isolate the most discriminating variables for each network flow in real-time. It thus acts as an intelligent filter upstream of NEMESIS, reducing dimensionality while preserving interpretability and computational velocity.

Experimental results demonstrate that the hybrid architecture manages to reduce dimensionality by more than 60%, stabilizing selection around 24 essential features instead of the initial 68. Despite this drastic compression, the system maintains a precision of 99.19% and an F1-score of 90.87%. Beyond detection efficiency, our solution shows a 10.97% gain in inference speed compared to a conventional model, while maintaining an extremely low memory footprint of 0.08 MB. This work validates the relevance of using reinforcement learning as a knowledge distillation mechanism to design detection systems that are high-performing, fast, and transparent.

Keywords : Cybersecurity, Intrusion Detection System (IDS), Reinforcement Learning, Random Forest, Feature Selection, NSL-KDD, CICIDS2017.

Résumé

Face à la sophistication croissante des cybermenaces et à l'augmentation exponentielle du trafic réseau, les systèmes de détection d'intrusion (IDS) traditionnels peinent à maintenir un équilibre optimal entre la précision, la rapidité de traitement et l'explicabilité des décisions. La haute dimensionnalité des données modernes introduit un bruit significatif et une latence de calcul qui compromettent la détection en temps réel, particulièrement au sein d'environnements aux ressources limitées, comme l'Internet des objets (IoT). Ce mémoire propose une architecture hybride innovante visant à optimiser la cybersécurité par la synergie entre l'apprentissage par renforcement profond et l'explicabilité de l'intelligence artificielle.

Notre approche repose sur le développement de deux agents interdépendants, évalués rigoureusement sur le jeu de données de référence CIC-IDS2017. Le premier module, nommé NEMESIS, constitue notre contribution initiale ; s'appuyant sur le Deep Q-Learning (DQN), il assure la classification des activités malveillantes en optimisant la prise de décision face aux intrusions. Pour renforcer l'efficacité de ce système, nous introduisons un second module, ULYSSE, un mécanisme de sélection dynamique de caractéristiques basé sur l'algorithme Proximal Policy Optimization (PPO). Guidé par une fonction de récompense intégrant la théorie des valeurs de SHAP, cet agent apprend à isoler en temps réel les variables les plus discriminantes pour chaque flux réseau, agissant ainsi comme un filtre intelligent en amont de NEMESIS pour réduire la dimensionnalité tout en préservant l'interprétabilité.

Les résultats expérimentaux démontrent que l'architecture hybride parvient à réduire le nombre de caractéristiques de plus de 60%, stabilisant la sélection autour de 24 caractéristiques essentielles au lieu des 68 initiales. Malgré cette compression drastique, le système maintient une précision de 99,19% et un F1-score de 90,87%. Au-delà de l'efficacité de détection, notre solution affiche un gain de rapidité d'inférence de 10,97% par rapport à un modèle classique, tout en conservant une empreinte mémoire extrêmement faible de 0,08 Mo. Ces travaux valident la pertinence de l'utilisation de l'apprentissage par renforcement comme mécanisme de distillation de connaissances pour concevoir des systèmes de détection d'intrusion à la fois performants, véloces et transparents.

Mots-clés : Cybersécurité, Système de Détection d'Intrusion (IDS), Apprentissage par Renforcement, Random Forest, Sélection de caractéristiques, NSL-KDD, CICIDS2017.

Remerciements

L'aboutissement de ce travail de recherche est le fruit d'un effort soutenu et de précieuses collaborations. Je tiens à exprimer ma profonde gratitude à toutes les personnes et institutions qui ont contribué à la réalisation de ce projet.

Je tiens tout d'abord à remercier chaleureusement ma directrice de recherche, **Professeure Hakima Ould-Slimane**. Sa rigueur scientifique, sa disponibilité constante et ses conseils avisés ont été les piliers de ce travail. Je la remercie pour la confiance qu'elle m'a témoignée et pour l'autonomie qu'elle m'a accordée tout au long de ce parcours doctoral. Je tiens également à exprimer ma profonde gratitude au **Professeur Nadjia Kara** pour l'intérêt qu'elle a porté à mes travaux. Ses critiques constructives, son expertise technique et ses réflexions enrichissantes ont grandement contribué à affiner la perspective de cette recherche. Sa bienveillance et son soutien académique tout au long de ce parcours ont été pour moi une source de motivation supplémentaire.

Mes sincères remerciements s'adressent également au **ministère de la Défense nationale (MDN) du Canada**. Le soutien apporté a été essentiel à la réussite de ce projet, permettant d'orienter nos recherches vers des problématiques de cybersécurité stratégiques et concrètes.

Sur un plan plus personnel, je souhaite exprimer ma reconnaissance infinie à **mes parents**. Leurs sacrifices, leur éducation et leurs encouragements indéfectibles depuis mon plus jeune âge sont le fondement de toutes mes réussites. Ce travail est autant le leur que le mien. Je tiens également à exprimer toute mon affection à **mon frère et à ma sœur**. Leur présence à mes côtés, leurs encouragements sincères et la complicité qui nous lie ont été une source d'équilibre indispensable durant ces années de travail intensif.

À **mon épouse**, je dédie une pensée toute particulière. Je la remercie pour sa patience exemplaire, son soutien moral et sa présence réconfortante lors des moments de doute. Sa compréhension face aux exigences de la recherche a été ma plus grande force et ma motivation quotidienne.

Enfin, je remercie l'ensemble de **ma famille** et de mes amis qui, de près ou de loin, m'ont soutenu par leur affection et leur bienveillance durant ces années de travail.

Dédicace

À mes parents,
pour leur soutien indéfectible.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

INTRODUCTION GÉNÉRALE

1.1 Contexte et Problématique

À l'ère de la transformation numérique, l'interconnexion croissante des écosystèmes numériques a érigé la sécurité des réseaux au rang de priorité absolue. Face à l'évolution constante des cybermenaces, qui deviennent de plus en plus sophistiquées et automatisées, les mécanismes de défense traditionnels montrent aujourd'hui leurs limites. Historiquement, la première ligne de défense repose sur les Systèmes de Détection d'Intrusion (IDS), classés traditionnellement en deux catégories : ceux basés sur les signatures et ceux basés sur les anomalies. Or, ces approches classiques peinent à répondre efficacement aux défis actuels. Les méthodes basées sur les signatures, bien qu'efficaces contre les attaques connues, se révèlent impuissantes face aux attaques « Zero-Day » ou aux variantes de malwares non répertoriées. Parallèlement, les méthodes basées sur les anomalies, si elles sont capables de détecter des comportements inédits, souffrent souvent d'un taux élevé de faux positifs, ce qui nécessite une intervention humaine constante pour le réglage des seuils de détection.

Dans des environnements réseau dynamiques et changeants, ces approches statiques ne parviennent pas à s'adapter en temps réel. Pour pallier ces lacunes, l'intégration de l'intelligence artificielle, plus spécifiquement de l'apprentissage par renforcement (Reinforcement Learning - RL), offre une voie prometteuse. Cette approche permet de concevoir des agents autonomes capables d'apprendre de manière dynamique en interagissant avec leur environnement réseau, s'adaptant ainsi aux nouvelles stratégies des attaquants, nécessitant une supervision humaine minimale. Cependant, l'application du RL seul se heurte encore à des défis majeurs de complexité computationnelle et de précision, particulièrement lorsqu'il est déployé dans des environnements caractérisés par un grand nombre de caractéristiques.

1.2 Objectifs et Contributions

Face à la sophistication croissante des cyberattaques et à l'explosion du volume de données réseau, ce mémoire poursuit des objectifs fondamentaux visant à transformer la détection d'intrusions. Le premier défi consiste à réduire le nombre de caractéristiques en sélectionnant les plus pertinentes afin d'éliminer le bruit et la redondance. Cette simplification est essentielle pour atteindre notre deuxième objectif : l'optimisation du temps de détection. En réduisant la latence d'inférence, nous visons à garantir une réactivité compatible avec les exigences du temps réel. Cela permet de faire face aux attaques polymorphes comme des *RANSOMWARE*, dont les mutations constantes visent à contourner les signatures statiques — tout en fournissant

une explication sémantique aux alertes générées.

Dans ce contexte, ce mémoire s'articule autour de deux contributions majeures visant à améliorer la robustesse et l'efficacité de la détection d'intrusions. La première contribution porte sur la conception et la validation de NEMESIS, un système de détection d'intrusion hybride qui a fait l'objet d'une publication récente [2]. Ce modèle combine la capacité d'adaptation du Deep Q-Learning (DQL) avec la précision d'un algorithme d'apprentissage supervisé, le Random Forest. L'architecture proposée repose sur un processus décisionnel en deux étapes, où un agent DQL agit d'abord comme un filtre rapide pour distinguer le trafic bénin du trafic suspect, suivi par le classificateur Random Forest, qui intervient uniquement sur les cas suspects pour identifier précisément le type d'attaque. Cette approche hybride permet non seulement d'améliorer la précision de la détection multiclasse sur des jeux de données complexes tels que CSE-CICIDS2017 et NSL-KDD, mais aussi de réduire la charge de calcul en focalisant l'analyse fine sur les menaces potentielles.

Notre deuxième contribution, baptisée ULYSSE (qui est en cours de soumission), constitue notre axe de recherche actuel et se concentre sur l'optimisation de ce système via la sélection de caractéristiques (feature selection). En effet, l'efficacité des modèles d'apprentissage est intrinsèquement liée à la qualité et à la dimensionnalité des données d'entrée. Le trafic réseau moderne génère une quantité massive d'attributs, dont certains peuvent être redondants ou non pertinents, introduisant du bruit qui ralentit l'apprentissage de l'agent RL. Avec ULYSSE, nous travaillons donc à identifier et à extraire les sous-ensembles de caractéristiques les plus discriminants. L'objectif est double : alléger la complexité computationnelle du modèle pour faciliter son déploiement en temps réel et améliorer sa capacité de généralisation face à des scénarios d'attaques variés, y compris les attaques zero-day telles que l'exploitation de failles inconnues et les attaques polymorphes comme le malware Emotet.

1.3 Organisation du mémoire

La structure de ce mémoire suit l'évolution de notre démarche de recherche. Le premier chapitre présentera les concepts fondamentaux liés aux systèmes de détection d'intrusion, aux différentes formes d'attaques, ainsi qu'aux métriques d'évaluation. Le deuxième chapitre offrira un état de l'art exhaustif des IDS et des techniques d'apprentissage automatique qui leur sont associées, en mettant en évidence les limites des approches existantes. Le troisième chapitre exposera de façon détaillée l'architecture de notre modèle hybride NEMESIS, en décrivant son fonctionnement interne et en montrant en quoi ses performances surpassent celles des méthodes classiques. Par la suite, ce chapitre s'attachera à analyser le rôle déterminant de la sélection de caractéristiques et présentera la méthodologie multi-agents ULYSSE, que nous proposons pour optimiser l'espace d'états. Enfin, nous terminerons ce document en récapitulant les principaux résultats obtenus et en ouvrant des pistes de réflexion pour de futurs travaux dans le champ de la détection d'intrusion adaptative.

Chapitre 2

Notions et Concepts préliminaires

2.1 Introduction

Les systèmes de détection d'intrusion regroupent différentes notions importantes. Ce chapitre a pour objectif de présenter les notions fondamentales nécessaires à la compréhension des IDS. Afin de poser un cadre conceptuel solide pour les chapitres suivants, ce chapitre introduit les concepts essentiels liés à la détection d'intrusion, aux types de menaces, ainsi qu'aux principes généraux de fonctionnement des IDS. Il présente également les différentes catégories d'IDS, leurs architectures, ainsi que les approches de détection les plus couramment utilisées, telles que la détection par signatures et la détection par anomalies.

Enfin, ces notions préliminaires permettent d'établir un vocabulaire commun et une base théorique indispensable à l'analyse, à la conception et à l'évaluation des solutions IDS étudiées dans ce mémoire, en particulier celles reposant sur des techniques d'apprentissage automatique et de renforcement.

2.2 Cybersécurité et Systèmes de Détection d'Intrusion

La sécurité des systèmes d'information est confrontée à des menaces qui visent la confidentialité, l'intégrité et la disponibilité des données [3]. Dans une architecture de défense en profondeur, on trouve de multiples dispositifs qui assurent la sécurité, à savoir principalement des pare-feu. Cette technologie ne suffit plus ; c'est pour cette raison que les Systèmes de Détection d'Intrusion (IDS) agissent comme une couche de surveillance critique, surveillant le trafic réseau pour identifier les menaces qui échappent aux règles de filtrage statique du pare-feu. Un Système de Détection d'Intrusion (IDS) est une technologie centrale de la cyberdéfense, destinée à surveiller le trafic réseau ou l'activité des systèmes pour repérer des comportements suspects ou des violations des politiques de sécurité. Son objectif principal est de classer le trafic en deux catégories : bénin (normal) ou malveillant. Il joue un rôle crucial pour garantir la confidentialité, l'intégrité et la disponibilité des systèmes d'information. Les IDS peuvent être implémentés dans deux endroits différents : au niveau du réseau pour surveiller le flux global (Network IDS) ou au niveau de l'hôte (Host IDS) pour surveiller le flux propre à la machine uniquement.

2.3 Approches de détection

Dans la littérature, on distingue deux grandes familles d'IDS selon leurs méthodes d'analyse, chacune représentant des avantages et des limites par rapport à l'autre ;

2.3.1 Détection basée sur les signatures (SIDS)

Cette méthode fonctionne par la correspondance des signatures, comme l'illustre l'exemple 2.3.1, pour trouver une attaque connue. Lorsque la signature d'une intrusion correspond à celle d'une intrusion déjà existante dans la base de données des signatures, un signal d'alarme est déclenché. Dans la littérature, les SIDS sont également appelés "détection basée sur des connaissances" ou "détection des abus". Les SIDS sont efficaces pour détecter les menaces déjà répertoriées, avec un faible taux de faux positifs. Cependant, leur limite majeure est leur incapacité à détecter les nouvelles attaques inconnues ou les menaces ayant changé de forme pour lesquelles aucune signature n'existe encore.

Pour illustrer de manière détaillée ce qu'est la signature d'un virus, prenons un exemple concret à partir du tableau 2.1. ***Melissa*** était un ver informatique qui s'est propagé en 1999 via des pièces jointes envoyées par e-mail. Il ciblait spécifiquement les ordinateurs utilisant Microsoft Outlook et se diffusait en expédiant automatiquement des messages infectés aux contacts présents dans le carnet d'adresses de l'utilisateur. Ce virus a fait l'objet d'une forte couverture médiatique à l'époque, en raison de sa capacité à se répandre très rapidement. La séquence d'octets du virus est la suivante :

Table 2.1 – Exemple de signature de malware

Nom	*Signature*
Accom.1280	89C3 B440 8A2E 2004 8A0E 2104 BA00 05CD 21E8 D500 BF50 04CD
Die.448	B440 B9E8 0133 D2CD 2172 1126 8955 15B4 40B9 0500 BA5A 01CD
Xany.979	8B96 0906 B000 E85C FF8B D5B9 D303 E864 FFC6 8602 0401 F8C3
Melissa	A1 00 10 8B C1

```
A1 00 10 8B C1
```

Listing 2.1 – Exemple de séquence de bytes du virus Melissa

La signification en ASCII de chaque couple de caractères est la suivante ;

```
'A1' est l'opcode pour l'instruction MOV.
'00 10' represente l'adresse memoire source et destination.
'8B' est l'opcode pour l'instruction ADD.
'C1' represente le registre cible.
```

Échec des modèles SIDS face aux attaques Zero-day

Une attaque inconnue, souvent appelée *zero-day*, exploite une vulnérabilité logicielle méconnue des développeurs et des fournisseurs de sécurité. Par définition, il n'existe aucun correctif

(patch) disponible ni aucune signature virale correspondante au moment de l'attaque [4]. Cette caractéristique rend les systèmes de détection basés sur les signatures inefficaces [3], les SIDS reposent sur la comparaison de paquets avec une base de données de menaces connues. Face à une attaque Zero-Day, cette base est muette, créant une "fenêtre de vulnérabilité" critique qui persiste jusqu'à ce que l'attaquant soit identifié et que la signature soit extraite, analysée et distribuée.

2.3.2 Détection basée sur les anomalies (AIDS)

Pour pallier les limites des SIDS, les AIDS modélisent le comportement "*normal*" du trafic réseau ; toute différence significative entre le comportement observé et le modèle est perçue comme une anomalie, pouvant être interprétée comme une intrusion. L'avantage qu'offrent les AIDS est l'identification des attaques *zero-day* et *polymorphes*, mais leurs limites résident dans le fait qu'elles génèrent fréquemment des faux signalements (faux positifs) et, dans certains cas, que les intrusions ne sont pas signalées (faux négatifs) ; elles sont également généralement complexes à mettre en œuvre. Le développement d'un AIDS nécessite un ensemble de données afin de construire le modèle ; bien sûr, il existe des ensembles de données disponibles et libres de droits que nous allons détailler dans la section des ensembles de données. Les AIDS sont distingués en trois grandes catégories, chacune comportant différentes techniques. La taxonomie proposée dans l'état de l'art est illustrée comme suit dans la figure 2.1.

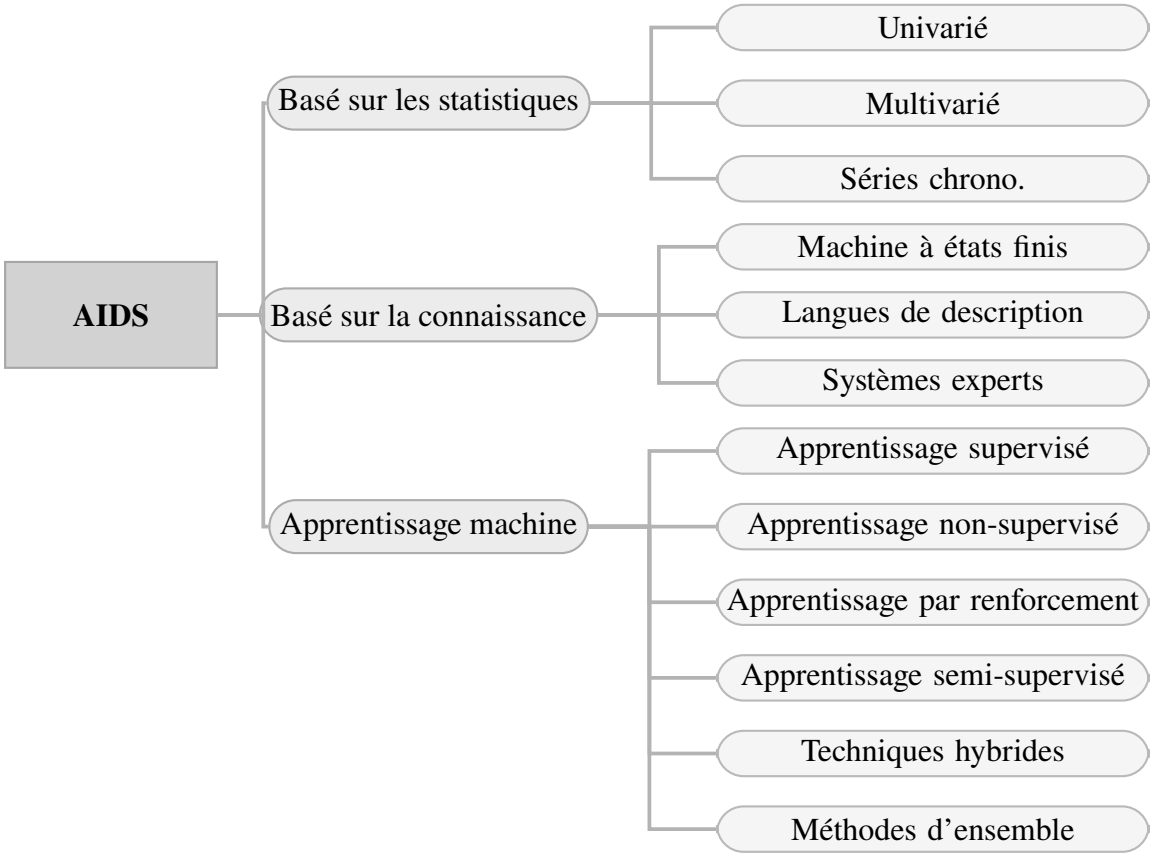


FIGURE 2.1 – Classification des méthodes AIDS

AIDS basé sur les statistiques

L'approche basée sur les statistiques consiste à collecter et à examiner chaque enregistrement des données dans un ensemble d'éléments afin de construire le modèle statistique du comportement normal d'un utilisateur. Elle prend en compte les mesures statistiques telles que la médiane, la moyenne et l'écart type des paquets. Cette approche utilise l'un des modèles suivants ;

- **Univarié** : ce qui signifie que la donnée est formée sur une seule variable uniquement. Cette technique est utilisée quand un profil utilisateur normal est créé pour une seule mesure de comportement dans le système. Il analyse individuellement les variables ou les caractéristiques pour détecter les anomalies indiquant une activité malveillante au sein d'un système ou d'un réseau.
- **Multivarié** : Cette technique repose sur les corrélations entre deux ou plusieurs mesures afin de comprendre les relations entre les variables. Contrairement à la technique Univarié, cette approche considère les interactions et les dépendances entre plusieurs variables afin d'effectuer une analyse plus approfondie. Cette méthode pourrait être utile lorsque les caractéristiques individuelles ne fournissent pas suffisamment d'informations pour détecter des intrusions, mais que les relations entre ses caractéristiques peuvent révéler des modèles significatifs. Elle nécessite souvent des méthodes statistiques avancées pour traiter efficacement les données à haute dimension.
- **Modèle de séries chronologiques** : Cette technique se présente comme chaîne d'observations effectuées sur un certain intervalle de temps, une nouvelle observation est anormale si la probabilité de se produire à ce moment-là est trop faible.

AIDS basé sur la connaissance

Aussi appelées les *méthodes des systèmes experts*, cette approche nécessite la création d'une base de connaissances qui reflète le profil du trafic légitime ; ainsi ; les actions qui diffèrent du comportement standard sont considérées comme une intrusion. Contrairement aux autres techniques des IDS, le modèle de profil standard est normalement créé sur la base de connaissances humaines, généralement par des experts dans le domaine. Cet ensemble de règles tente de définir l'activité normale du système. L'avantage des systèmes basés sur la connaissance est de réduire la détection des faux positifs. Dans un environnement en évolution dynamique, cette catégorie d'IDS nécessite une mise à jour régulière des connaissances sur le comportement. Il'est également fastidieux de traiter tous les cas de comportement et de rassembler toutes les informations sur les comportements normaux. Néanmoins, l'existence de plusieurs approches et techniques que nous pouvons citer comme suit ;

- **Automate à Etats Finis (FSM)** : Aussi appelé en anglais finite state automaton, ce qui signifie automate à états finis, cette technique est un modèle de calcul utilisé pour représenter et contrôler le flux d'exécution. En général, le modèle est représenté comme un ensemble d'états et un ensemble de transitions, qui sont utiles pour déterminer les états actuels. L'ensemble des états vérifie les données historiques, toute variation de l'entrée est notée, par suite en fonction de cette variation détectée, une transition se produit.
- **Langage de Description** : Cette technique définit la syntaxe des règles qui peuvent définir les caractéristiques d'une attaque. Chaque attaque comporte son langage de description différent des autres attaques. Ses règles peuvent être construites à l'aide de langages de description comme les N-grammars ou UML.

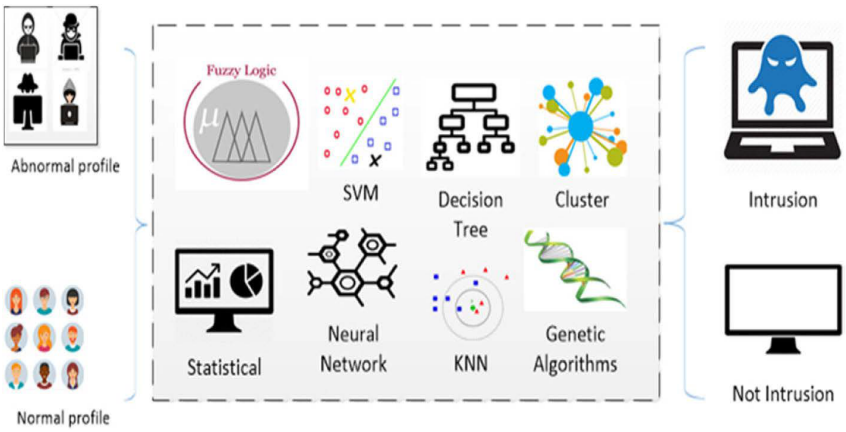


FIGURE 2.2 – Fonctionnement conceptuel des approches AIDS basées sur l'apprentissage automatique

- **Systèmes Experts** : Les systèmes experts comportent un ensemble de règles qui définissent les attaques ; les règles sont généralement définies manuellement.

AIDS basé sur l'apprentissage machine

L'apprentissage automatique constitue un sous-domaine de l'intelligence artificielle qui extrait des connaissances à partir de données d'entraînement durant la phase d'apprentissage. Les méthodes d'apprentissage automatique ont été largement exploitées dans le contexte des systèmes de détection d'intrusion pour extraire des connaissances à partir de jeux de données d'intrusions [3]. De nombreux AIDs ont ainsi été conçus en s'appuyant sur l'apprentissage automatique, comme le montre la figure 2.2. Le recours à ces techniques vise à développer des systèmes de détection d'intrusion plus précis, tout en réduisant la dépendance à l'expertise humaine. Ces dernières années, le nombre d'IDS fondés sur des approches d'apprentissage automatique a fortement augmenté. L'un des buts majeurs de la recherche sur les IDS basés sur l'apprentissage automatique est d'identifier des schémas récurrents et de construire un système de détection d'intrusion adapté à partir de l'ensemble de données. En général, il existe deux types de méthodes d'apprentissage automatique : supervisées et non supervisées [3]. Cependant, certaines lectures proposent davantage de techniques, à savoir l'apprentissage non-supervisé et l'apprentissage par renforcement [5], l'apprentissage semi-supervisé, les techniques hybrides et les méthodes d'ensemble [3].

1. **Apprentissage supervisé :** L'apprentissage supervisé est la conception d'un modèle entraîné sur des données étiquetées. Principalement, elle compte deux phases ou étapes : la phase d'entraînement, où le modèle est entraîné sur des données fournies et la phase de test, qui permet de tester son apprentissage. Il existe plusieurs algorithmes d'apprentissage supervisé [5] qui sont listés comme suit ;
   - ◇ **Artificial Neural Network :** Les ANN sont des graphes avec des nœuds et arêtes comportant des poids sur chacune des arêtes, ils se distinguent principalement par leur flexibilité et rapidité au traitement mais sont limités par leur capacité a traiter des masses volumineuses de données.
   - ◇ **Nearest Neighbor algorithm :** Les NN sont des méthodes spécifiques où un point de données est classé en fonction de l'étiquette de son voisin le plus proche dans l'espace des caractéristiques. Leur avantage est la Simplicité et ne nécessite pas

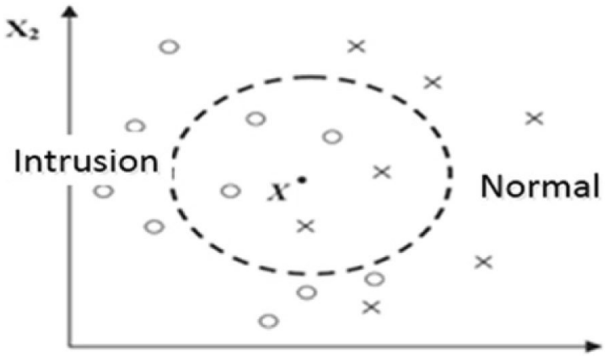


FIGURE 2.3 – Exemple de classification par KNN pour k = 5

d'entrainnement ainsi que la capabilité de capturer des structures complexes, mais elles se présentent souvent avec le problème d'overfitting et ne sont pas adaptées aux données de grande dimension.

- ◇ **Support Vector Machine :** Les SVM sont des classificateurs discriminatifs dont le principe repose sur la recherche d'un hyperplan de séparation optimal. Cet hyperplan est défini dans l'espace des caractéristiques de dimension N, et vise à maximiser la marge entre les vecteurs de support des différentes classes. Les SVM sont bien connus pour leur capacité de généralisation et sont principalement efficaces lorsque le nombre d'attributs est grand et le nombre de points de données est petit.
- ◇ **Hidden Markov Model :** Le modèle HMM est un modèle statistique de Markov dans lequel le système modélisé est supposé être un processus de Markov avec des données inédites. dans [3], des recherches antérieures ont montré que l'analyse HMM peut être appliquée pour identifier des types particuliers de logiciels malveillants. Sa modélisation séquentielle contribue efficacement à la modélisation séquentielle, ses limites sont la complexité et elle est sensible aux paramètres.
- ◇ **Decision Trees (C4.5, ID3, CART, Random Forrest) :** Les DT sont les algorithmes de classification, le but est de créer un classificateur pour prédire la valeur d'une classe cible pour une instance de test non vue, basée sur plusieurs instances déjà connues. À travers une séquence de décisions, une instance de test non vue est classifiée par un arbre de décision. Les DT sont considérées comme autonomes car il peut être implémenté en tant que classificateur seul. Les avantages des DT se basent sur leur interprétation, car ils représentent visuellement des décisions simples et claires, de plus ils peuvent gérer à la fois des données numériques et catégorielles mais, ils ont tendance à être sensibles aux variations mineures dans les données d'entraînement
- ◇ **K-nearest neighbors :** KNN est une méthode non paramétrique basée sur les instances dont les fondements reposent sur sa fonction de distance, qui calcule les corrélations ou les différences entre les paires d'instances ou de points. Cette méthode est très facile à comprendre et à mettre en œuvre, mais nécessite un calcul intensif et n'est pas adaptée aux données de grande dimension. La figure 2.3 illustre un classificateur KNN où k = 5. Le point X représente une instance de date non étiquetée qui doit être classifiée. Parmi les cinq voisins les plus proches de X, il y a trois modèles similaires de la classe Intrusion et deux de la classe Normal. Le vote à la majorité permet d'attribuer X à la classe Intrusion [3].
- ◇ **Naive Bayes :** Cette approche repose sur l'application du principe de Bayes avec des

hypothèses d'indépendance solides entre les attributs. Les Bayes naïfs répondent à des questions telles que "quelle est la probabilité qu'un type particulier d'attaque se produise, compte tenu des activités observées du système" en appliquant des formules de probabilité conditionnelle [6]

- ◇ **Régression logistique :** La régression logistique est une technique de modélisation statistique utilisée pour prédire la probabilité qu'une variable binaire dépendante prenne l'une des deux valeurs possibles en fonction d'un ensemble de variables indépendantes
- ◇ **Logique floue :** La logique floue est une approche permettant de gérer l'incertitude et l'imprécision inhérentes aux données de trafic réseau. Contrairement à la logique binaire traditionnelle (0 ou 1), elle repose sur un modèle d'inférence floue qui permet de représenter la gradation entre ces deux états via des fonctions d'appartenance. Dans le cadre de ce système, elle est intégrée au modèle de prétraitement pour transformer des variables continues en variables linguistiques, facilitant ainsi la capture de comportements d'intrusion complexes qui ne répondent pas à des seuils rigides.
- ◇ **Algorithmes génétiques :** Les algorithmes génétiques sont une approche heuristique de l'optimisation, basée sur les principes de l'évolution. Une population est générée durent chaque époque portant des chromosomes. Les gènes individuels ayant une évolution considérable durent cette époque sont transmis à la population successeur.

Il existe de nombreuses méthodes de classification. Chaque technique utilise une méthode d'apprentissage pour construire un modèle de classification. Cependant, une approche de classification appropriée ne doit pas seulement traiter les données d'apprentissage, mais aussi identifier avec précision la classe d'enregistrements qu'elle n'a jamais vue auparavant. La création de modèles de classification dotés d'une capacité de généralisation fiable est une tâche importante mais aussi très complexe pour l'algorithme d'apprentissage.

2. **Apprentissage non-supervisé :** L'apprentissage non-supervisé est un paradigme dans l'apprentissage automatique. il repose sur la construction d'un modèle à partir des données non étiquetées, en regroupant les données en différentes classes selon leur contenu au fil de l'apprentissage. on distingue trois classes selon [3]
   - ◇ **K-means :** La technique K-means est l'une des techniques les plus courantes d'analyse par regroupement qui vise à séparer « n » objets de données en « k » groupes dans lesquels chaque objet de données est inséré dans le groupe ayant la moyenne la plus proche. Il s'agit d'une technique de regroupement basée sur la distance qui ne nécessite pas de calculer les distances entre toutes les combinaisons d'enregistrements. Elle applique une métrique euclidienne comme mesure de similarité. Le nombre de clusters est déterminé à l'avance par l'utilisateur. En général, plusieurs solutions sont testées avant d'accepter la plus appropriée. Annachhatre et al. ont utilisé l'algorithme de clustering K-means pour identifier différents profils de comportement des hôtes [7]. Ils ont proposé de nouvelles mesures de distance qui peuvent être utilisées dans l'algorithme k-means pour relier étroitement les clusters. Ils ont regroupé les données en plusieurs clusters et les ont associées à des comportements connus à des fins d'évaluation. Leurs résultats ont révélé que le clustering k-means est une meilleure approche pour classer les données à l'aide de méthodes non supervisées pour la détection des intrusions lorsque plusieurs types d'ensembles de données sont disponibles. Le regroupement pourrait être utilisé dans les IDS pour réduire

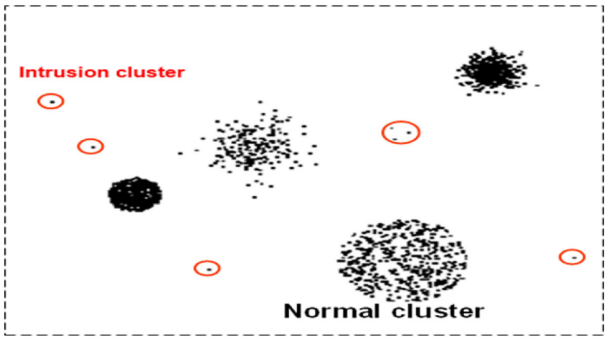


FIGURE 2.4 – Utilisation du regroupement pour la détection des intrusions

les signatures d'intrusion, générer une signature de haute qualité ou regrouper des intrusions similaires.

- ◇ **Regroupement hiérarchique :** Il s'agit d'une technique de regroupement qui vise à créer une hiérarchie de clusters. Les approches de regroupement hiérarchique sont généralement classées en deux catégories :
  (a) Techniques de regroupement agglomératif ascendant dans lesquelles les clusters ont des sous-clusters, qui à leur tour ont des sous-clusters et les paires de clusters sont combinées à mesure que l'on remonte dans la hiérarchie.
  (b) Divisifs : algorithmes de regroupement hiérarchique dans lesquels, de façon itérative, le cluster présentant le plus grand diamètre dans l'espace des caractéristiques est choisi et scindé en sous-clusters binaires de portée plus réduite.
- ◇ **Redundancy-based resilience :** Cette forme de résilience repose sur l'exploitation des corrélations intrinsèques et des dépendances statistiques au sein des flux réseau. Dans un cadre non supervisé, le système utilise la redondance des informations pour maintenir une détection cohérente des anomalies, même lorsque certaines caractéristiques sont bruitées ou absentes, en s'appuyant sur la distribution globale des données pour reconstruire le profil du trafic normal [8].

Comme le montre la figure 2.4, une fois les enregistrements regroupés, tous les cas qui apparaissent dans de petits groupes sont étiquetés comme une intrusion, car les occurrences normales devraient produire des groupes plus importants que les anomalies. De plus, les intrusions malveillantes et les instances normales sont différentes, elles ne tombent donc pas dans le même groupe.

3. **Apprentissage semi-supervisé :** L'apprentissage semi-supervisé est une méthode assez complexe dans l'apprentissage automatique ; elle utilise à la fois des données étiquetées et non-étiquetées afin de construire son modèle. L'idée derrière l'apprentissage semi-supervisé est de tirer des informations supplémentaires des données non étiquetées afin de mieux généraliser. L'avantage de l'apprentissage semi-supervisé est qu'il nécessite moins d'efforts humains et offre une plus grande précision. La figure 2.5 illustre les taxonomies des techniques d'apprentissage semi-supervisé.

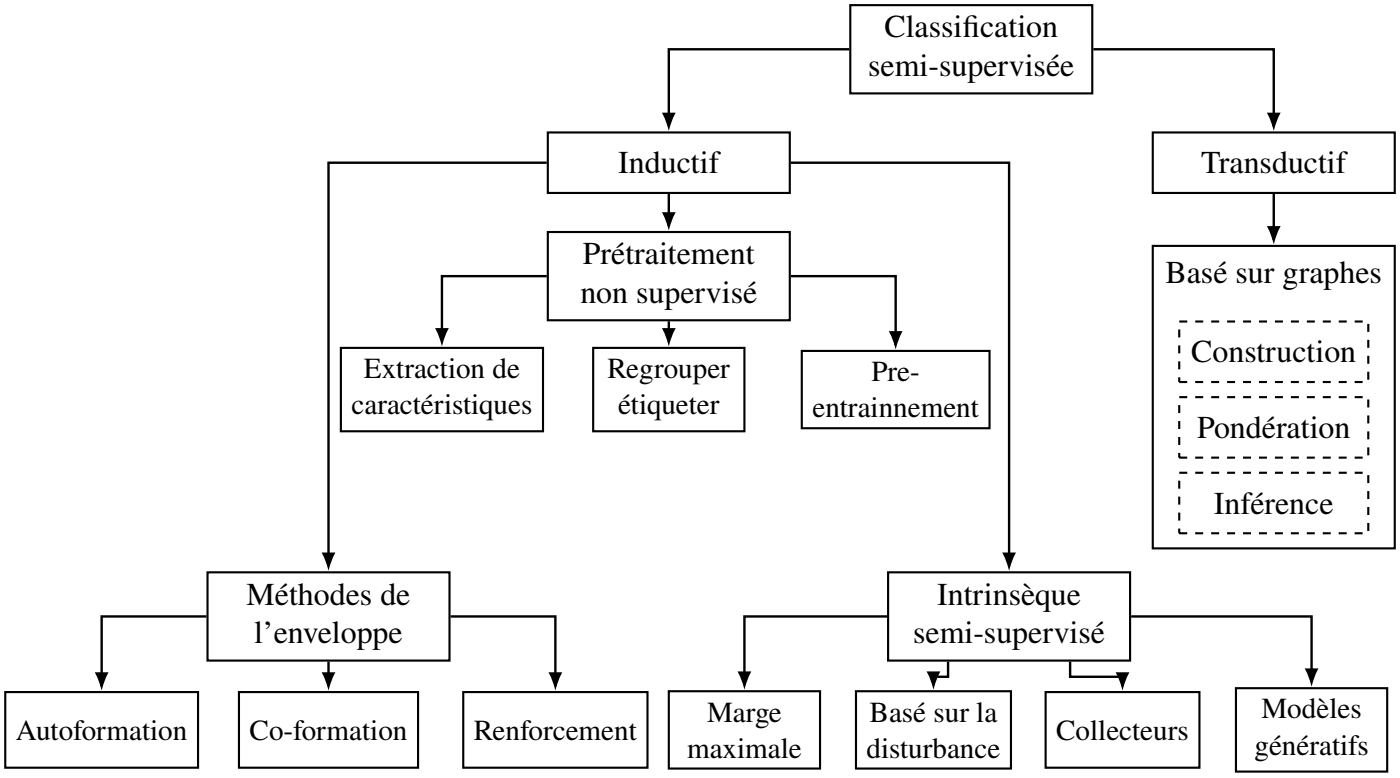


FIGURE 2.5 – Graphique des techniques d'apprentissage semi-supervisé.

4. **Méthodes ensemblistes :** Les méthodes d'ensemble, comme leur nom l'indique, sont l'association de deux algorithmes ou davantage pour améliorer considérablement les performances de prédiction des systèmes de détection. Il existe trois méthodes principales
   - ◇ **Boosting :** Le Boosting représente un groupe d'algorithmes qui ont la capacité de convertir des modèles d'apprentissage faibles en des modèles forts.
   - ◇ **Bagging :** Bagging consiste à entraîner le même classificateur sur différents sous-ensembles du même ensemble de données.
   - ◇ **Empilement :** Stacking combine différents classificateurs via un métaclassificateur.
5. **Apprentissage par renforcement :** Une alternative prometteuse à l'apprentissage automatique classique est l'Apprentissage par Renforcement (RL). Le RL permet de concevoir des agents adaptatifs capables d'apprendre de manière autonome (self-learning) et d'identifier des intrusions sans supervision directe. Le problème est modélisé comme un problème de processus de décision markovien (MDP), défini par des états, des actions, des récompenses, un agent et un environnement.
   - **Environnement**
     En apprentissage par renforcement, l'environnement constitue le composant central du processus. Il englobe des méthodologies telles que les fonctions *step* et *reward*. Dans cette étude, l'environnement repose sur le jeu de données utilisé, qui a fait l'objet d'un prétraitement, d'une normalisation et d'un rééchantillonnage. Les caractéristiques de ce jeu de données (colonnes) représentent l'état du DQN. À la suite du prétraitement, l'ensemble des caractéristiques est utilisé pour représenter l'état, tandis qu'une étiquette unique est exploitée pour le calcul du vecteur de récompense, fondé sur les résultats prédictifs du modèle.

- **Agent**
  Un agent de type *Deep Q-Network* se caractérise par une approche basée sur la valeur. Au cours de l'apprentissage, l'agent cherche à optimiser un DQN en interagissant avec son environnement. Il observe l'état courant et sélectionne une action à partir de l'espace des actions. Par la suite, l'environnement fournit une récompense en comparaison avec l'étiquette réelle de l'état, cette sélection d'actions est régie par une stratégie spécifique appelée *politique* (*policy*). Initialement, durant la phase d'exploration, l'agent teste l'ensemble des actions possibles en appliquant une stratégie d'exploration telle que l' *epsilon-greedy*, une politique stochastique ou toute autre variante existante. Une fois cette phase initiale d'exploration terminée, l'agent apprend progressivement à réduire la sélection aléatoire d'actions. Cette évolution est rendue possible par la politique, qui guide l'agent vers des choix plus *gourmands* (*greedy*) en fonction de la fonction de valeur, optimisant ainsi la probabilité de sélection d'action à un niveau de $1 - \epsilon$, où ϵ est la probabilité que l'agent choisit une action au hasard parmi toutes les actions disponibles.

- **États**
  Dans le cadre du DQN, les états représentent les données fournies par l'environnement à l'agent. Ces entrées constituent la base essentielle à partir de laquelle l'agent prend ses décisions concernant les actions à entreprendre. Les caractéristiques du jeux de données sont transformées en un *vecteur d'état*, qui est ensuite fourni en entrée au DQN. Par ailleurs, la caractéristique *label* qui représente l'étiquette réel de l'échantillon est utilisée de manière constante, tant lors de la phase d'apprentissage que pour l'évaluation des prédictions.

- **Actions**
  Une action est une décision prise par l'agent à partir de son expérience cumulative acquise lors de l'exploration de l'environnement. À chaque changement d'état, l'agent génère une liste d'actions possibles. La valeur Q est alors utilisée pour déterminer si une attaque a été détectée. Le vecteur d'état, dimensionné selon la taille du mini-lot (*mini-batch*), est transmis au DQN courant. L'agent évalue ensuite la sortie du DQN, applique des seuils de décision pour interpréter les valeurs Q et détermine la valeur seuil Q utilisée pour la classification des différentes catégories d'attaques.

- **Récompenses**
  En apprentissage par renforcement profond, la récompense représente le retour de l'environnement suite à une action de l'agent. Elle peut être exprimée sous la forme d'un vecteur, déterminé à partir des valeurs de sortie du DQN et de la taille du mini-lot. L'agent reçoit une récompense positive lorsque le DQN classe correctement les données en fonction des étiquettes réelles du jeu de données et une récompense négative dans le cas contraire. La valeur de la récompense peut également être influencée par les probabilités de prédiction du classificateur. L'ajustement de cette valeur en fonction des valeurs Q contribue à améliorer les performances globales du classificateur.

l'IDS est considéré comme un agent, à travers l'interaction avec son environnement il prend des actions (ex. classer un profil comme normal ou intrusion) dans un environnement (le réseau) pour maximiser une récompense (par exemple, la détection efficace des

intrusions) tout en minimisant les coûts ou les conséquences indésirables (comme les faux positifs). L'agent apprend par essais et erreurs, en recevant des rétroactions positives ou négatives sur ses actions. Les algorithmes RL se séparent en deux catégories **off-policy** et **on-policy** [9, 10], la différence entre les deux réside dans la manière dont la politique (stratégie) est apprise à partir des données. Les méthodes on-policy mettent à jour la politique en utilisant uniquement les expériences générées par cette même politique lors de l'interaction avec l'environnement, ce qui garantit une cohérence entre exploration et apprentissage, mais limite la réutilisation des données. À l'inverse, les méthodes off-policy apprennent une politique cible à partir d'expériences collectées par une politique différente, ce qui permet de réutiliser des données passées ou externes et d'améliorer l'efficacité de l'apprentissage, au prix d'une complexité et d'une instabilité potentielles accrues. Il existe plusieurs algorithmes RL que nous pouvons lister comme suit [11] ;

TABLE 2.2 – Classification des algorithmes RL : Off-Policy vs On-Policy

Type	**Algorithme**	**Description**
	Off-Policy	
Value-Based	**Q-Learning**	Classique, tabulaire (espaces d'états discrets).
	DQN	Q-Learning avec réseaux de neurones.
	Double DQN (DDQN)	Réduit la surestimation des valeurs Q.
Actor-Critic	**DDPG**	Gradient déterministe profond de la politique (Espace d'état continu).
	TD3	DDPG double différé (plus stable que DDPG).
	SAC	Acteur-Critique doux (maximise l'entropie, SOTA).
	On-Policy	
Value-Based	**SARSA**	State-Action-Reward-State-Action.
Policy Gradient	**REINFORCE**	Monte Carlo Policy Gradient basique.
	A2C	Advantage Actor-Critic (Synchrone).
	TRPO	Trust Region Policy Optimization (complexe).
	PPO	Proximal Policy Optimization (standard actuel), espace d'action continu.

— **Q-Learning**
Le Q-learning est reconnu comme la technique d'apprentissage par renforcement la plus utilisée [12]. L'agent exploite l'espace des actions et des états pour construire une table de consultation appelée *Q-table*. Le contenu de cette matrice est mis à jour à l'aide de la *Q-function*, présentée dans l'équation 2.1 :

$$Q(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \,\middle|\, s_t = s,\ a_t = a\right] \tag{2.1}$$

où $\mathbb{E}$ est l'espérance mathématique, et $Q(s,a)$ représente la récompense cumulative espérée en partant de l'état s et en exécutant l'action a [13]. Afin de maximiser cette valeur, l'estimation est mise à jour de manière itérative selon l'algorithme des différences temporelles présenté dans l'équation 2.2 :

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[\underbrace{r + \gamma \max_{a'} Q(s',a')}_{\text{Cible TD}} - Q(s,a) \right] \tag{2.2}$$

Dans cette expression :

— α désigne le taux d'apprentissage ($0 < \alpha \leq 1$) ;
— r est la récompense immédiate ;
— γ est le facteur d'actualisation (*facteur d'actualisation*) ;
— $\max_{a'} Q(s',a')$ est l'estimation du gain futur maximum.

Cette approche par table devient toutefois impraticable lorsque l'espace des états est de grande dimension ou continu. Pour résoudre ce problème de dimensionnalité, des méthodes de *Deep Q-Learning* (DQN) sont utilisées, remplaçant la table par un réseau de neurones profond agissant comme un approximateur de fonction.

— **Deep Reinforcement Learning (DRL)**
L'utilisation du *Q-learning* tabulaire comme solution à notre problème de détection n'est pas viable, car l'espace d'états (les données réseaux) est de trop grande dimension. L'*apprentissage par renforcement profond* (*Deep Reinforcement Learning*, DRL) contrairement au Q-learning classique, qui stocke les valeurs dans une table, le DRL estime la fonction de valeur d'action $Q(s,a)$ (tel que définie dans l'équation 2.2) directement à l'aide d'un réseau de neurones profond, ce qui cette approches une des plus utilisées pour ce genre de problématique, notamment à travers la méthode du *Deep Q-Network* (DQN).
La Figure 2.6 [14] illustre le protocole d'interaction entre un agent et son environnement dans le cadre du DQN. Le réseau de neurones profond reçoit en entrée l'ensemble des caractéristiques de l'état courant et fournit en sortie les Q-valeurs correspondant à chaque action possible.

— **Proximal Policy Optimization (PPO)**
Le *Proximal Policy Optimization* (PPO) est une méthode d'apprentissage par renforcement profond appartenant à la famille des algorithmes de *Policy Gradient*. Proposé par Schulman et al. [15], cet algorithme a pour objectif d'accroître la stabilité et l'efficacité de l'apprentissage des agents en régulant l'ampleur des mises à jour appliquées à la politique. Contrairement aux méthodes basées sur la valeur (comme le DQN) qui apprennent une fonction $Q(s,a)$, le PPO optimise directement une politique paramétrée $\pi_\theta(a|s)$ en maximisant une fonction objectif qui contraint les changements de politique entre deux itérations successives.
La fonction de perte du PPO est formulée de manière à limiter les variations trop brusques de la politique via un terme de ratio tronqué (*clipped*), défini comme suit :

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{2.3}$$

où $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ représente le ratio de probabilité entre la nouvelle et l'ancienne politique, $\hat{A}_t$ désigne l'avantage estimé à l'instant t, et ϵ est un hyperparamètre de

tolérance permettant de contrôler la plage d'ajustement. En limitant explicitement la différence entre les politiques successives, le PPO parvient à un compromis entre *exploration* et *stabilité*. Cette approche a démontré une performance robuste dans divers environnements continus et discrets, notamment pour les tâches complexes de détection d'intrusions. L'opérateur $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$ agit comme un mécanisme de sécurité sur le ratio de probabilité $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$. Mathématiquement, il force la valeur de ce ratio à rester dans l'intervalle $[1-\epsilon, 1+\epsilon]$, où ϵ est un hyperparamètre (souvent fixé à 0,2). Cette opération remplit deux fonctions critiques :

(a) **Contrôle de la région de confiance :** Elle empêche la nouvelle politique π_θ de s'éloigner radicalement de l'ancienne politique $\pi_{\theta_{old}}$, évitant ainsi des mises à jour catastrophiques qui pourraient dégrader irréversiblement les performances de l'agent.

(b) **Approche pessimiste :** En prenant le minimum (min) entre la valeur originale et la valeur clippée, l'algorithme adopte une borne inférieure de l'objectif. Cela garantit que l'agent ne surestime pas l'amélioration d'une action, même si le gradient suggère une augmentation massive de sa probabilité.

— **Gestion de l'expérience : Le concept de Tampon (Buffer)** Dans le cadre de l'apprentissage par renforcement, le transfert d'informations entre l'agent RL et l'algorithme d'optimisation transite par un **tampon** (*buffer*).

Pour un algorithme *on-policy* tel que PPO, on utilise un *tampon de déploiement (rollout buffer)*. Ce dernier permet de stocker un ensemble de transitions $\langle s, a, r, s' \rangle$ sur un horizon temporel fixe avant d'effectuer une mise à jour. Contrairement au *replay buffer* utilisé dans le DQN, le tampon de PPO est intégralement réinitialisé après chaque phase d'optimisation. Cette approche garantit que les gradients calculés pour ajuster la politique π proviennent exclusivement d'interactions générées par la version la plus récente de l'agent.

— **Les algorithmes Acteur-Critique :** Les méthodes *Acteur-Critique* représentent une classe d'algorithmes d'apprentissage par renforcement exploitant une structure duelle pour stabiliser l'apprentissage. L'**Acteur** est chargé de définir la politique $\pi(a|s)$ en sélectionnant les actions à entreprendre, tandis que le **Critique** estime la fonction de valeur $V(s)$, agissant comme une référence pour évaluer la pertinence des décisions prises.

Dans le cadre de l'algorithme PPO utilisé dans cette étude, cette synergie permet de calculer un signal d'*avantage*. Ce signal indique à l'Acteur si une action spécifique a surpassé l'espérance de gain moyenne, permettant ainsi une mise à jour précise et robuste de la politique de sélection des caractéristiques réseau.

2.4 Ensembles de données

Les ensembles de données sont un moyen d'évaluer un IDS dans une situation proche de la réalité. Ils sont caractérisés par des informations relatives au flux réseau, qui peuvent décrire le comportement du profil durant son utilisation du réseau. Les ensembles de données des AIDS sont différents des bases de données pour les SIDS ; les informations y sont plus intuitives et compréhensibles que les signatures. Au cours de ce mémoire, nous allons nous concentrer sur

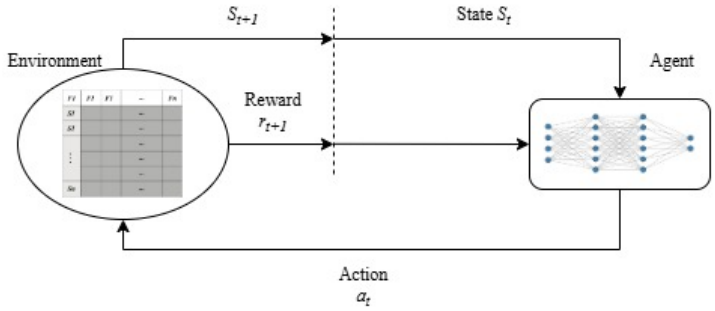


FIGURE 2.6 – Schéma d'interaction agent/environnement

les ensembles de données disponibles pour le AIDS, plus précisément pour les NIDS. Il existe une diversité remarquable dans les ensembles de données publics .

Le dataset *KDDCup99* est l'ancêtre des ensembles de données pour NIDS, créé par la *DARPA* (Defense Advanced Research Projects Agency) en 1999 et comptant approximativement 6 900 000 enregistrements et 41 caractéristiques. il a connu une amélioration en éliminant des échantillons redondants et est devenu alors *NSL-KDD*, qui demeure à ce jour utilisé dans la recherche. Il existe bel et bien un ensemble de données avant *KDDcup99*, qui est *DARPA98*, mais *KDDcup99* est le premier a être utilisable. le tableau 2.3 représente un descriptif des ensembles de données publics les plus populaires et les plus utilisés dans l'état de l'art [3, 16].

2.4.1 Jeux de données utilisés

NSL-KDD

Le jeu de données *NSL-KDD* (Network Security Laboratory - Knowledge Discovery in Databases) est une version améliorée du célèbre *KDD'99*, conçue pour l'évaluation des systèmes de détection d'intrusions. Il corrige plusieurs limitations majeures du KDD99, notamment la redondance excessive des enregistrements, qui biaisait les algorithmes d'apprentissage en leur permettant d'atteindre artificiellement des taux de précision élevés. Le NSL-KDD supprime ces duplications, offrant ainsi un jeu de données plus équilibré et représentatif.
Chaque connexion réseau y est décrite par **41 caractéristiques**, regroupées en quatre catégories d'attaques principales :

- *Denial of Service (DoS)*
- *Probe*
- *Remote to Local (R2L)*
- *User to Root (U2R)*

en plus de la classe *Normal*. Ces caractéristiques incluent à la fois des attributs basés sur le contenu, le trafic et les statistiques de session.
Le jeu de données est distribué gratuitement pour un usage académique et la recherche. Il est disponible sur le site de l'université du Nouveau-Brunswick ou sur Kaggle. Les fichiers sont souvent déjà convertis en CSV avec les en-têtes, ce qui fait gagner du temps.

CSE-CICIDS2017

Le jeu de données *CSE-CICIDS2017* constitue une ressource plus récente et complète pour la recherche en détection d'intrusions. Construit par le *Canadian Institute for Cybersecurity (CIC)*, il comprend un trafic réseau étiqueté, simulant des activités bénignes et malveillantes réalistes. Il contient **79 caractéristiques** décrivant des flux réseau, annotées soit comme *Benign*, soit selon

TABLE 2.3 – Taxonomie des ensembles de données publiques

Ensemble de données	**Année**	**Caractéristiques**	**Classes**
CSE-CICIDS2017	2017	80	DoS Golden Eye, Heartbleed, DoS hulk, DoS Slow http, DoS Slowloris, DDoS, SSH-Patator, FP, Patator, Brute force, XSS, Botnet, infiltration, PortScann, SQL injection.
CSE-CICIDS2018	2018	80	DoS Golden Eye, Bening, DoS hulk, DoS Slow http, DoS Slowloris, DDoS-LOIC HTTP, DDoS-LOIC-UDP, DDoS-HOIC, SSH-Patator, FTP Patator, Brute force, XSS, Botnet, infiltration, SQL injection.
UNSW-NB15	2015	49	Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, Worms
ISCX-2012	2012	80	Normal, Attacker
NSL-KDD	1999	43	Normal, DoS, Remote to Local (R2L), User to Root(U2R), Probing
DARPA	1998	43	DoS, Remote to Local (R2L), User to Root(U2R), Probing
KYOTO	2006-2009	23	Oth, rej, rsto, rstos0, rstr, rstrh, s0, s1, s2, s3, sf, sh, shr
KDD99	1999	43	DoS, Remote to Local (R2L), User to Root(U2R), Probing
AWID	2015	156	Normal, Flooding, Injection, Impersonation
CIDDS-001	2017	14	Normal, Attacker, Victim, Suspicious, Unknown

le type d'attaque. Avant l'entraînement, un prétraitement rigoureux a été appliqué : suppression des colonnes inutiles ou dupliquées (telles que *destination_port* et *fwd_header_length.1*), élimination des valeurs manquantes ou infinies (*flow_bytes/s*, *flow_packets/s*) et normalisation des valeurs numériques. Le jeu de données final comprend 68 caractéristiques utilisées pour représenter l'état du DQN et une étiquette (*label*) utilisée pour le calcul des récompenses.

Outre ses caractéristiques techniques, la pertinence du *CICIDS2017* réside dans sa méthodologie de capture, conçue pour refléter la cyclicité et la diversité d'un environnement de production réel. Les données ont été collectées sur une période continue de cinq jours, du lundi 3 juillet au vendredi 7 juillet 2017, suivant un scénario évolutif précis. La journée du lundi est exclusivement consacrée au trafic bénin, permettant d'établir un profil de comportement normal (*baseline*), tandis que les jours suivants introduisent progressivement des vecteurs d'attaques complexes.

Cette structure temporelle permet d'exposer l'agent RL à une taxonomie d'intrusions variées, regroupées en cinq familles majeures :

— **Brute Force** (FTP et SSH) : Tentatives de devinettes de mots de passe, principalement observées le mardi.

— **Déni de Service (DoS) et DDoS** : Inondation de trafic via des outils comme *Hulk*,

GoldenEye ou *Slowloris*.

— **Attaques Web** : Injections SQL, Cross-Site Scripting (XSS) et Brute Force Web.

— **Infiltration et Botnet** : Scénarios plus furtifs visant à compromettre le réseau de l'intérieur.

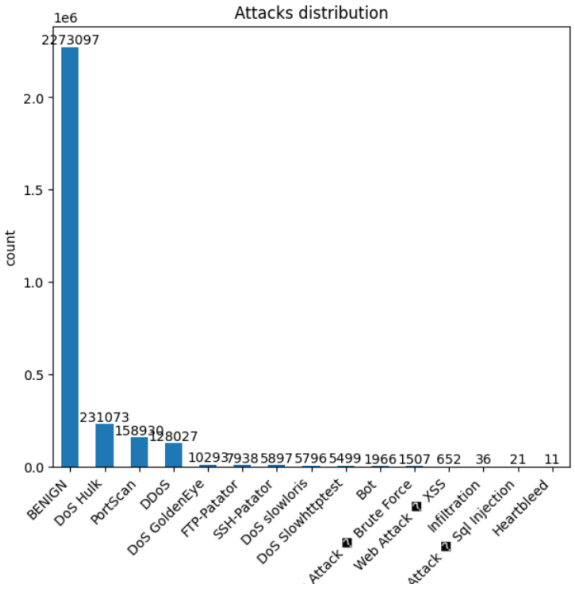
— **Port Scan** : Reconnaissance active des ports ouverts pour identifier les vulnérabilités.

2.4.2 Problématique du déséquilibre des classes et stratégies de résolution

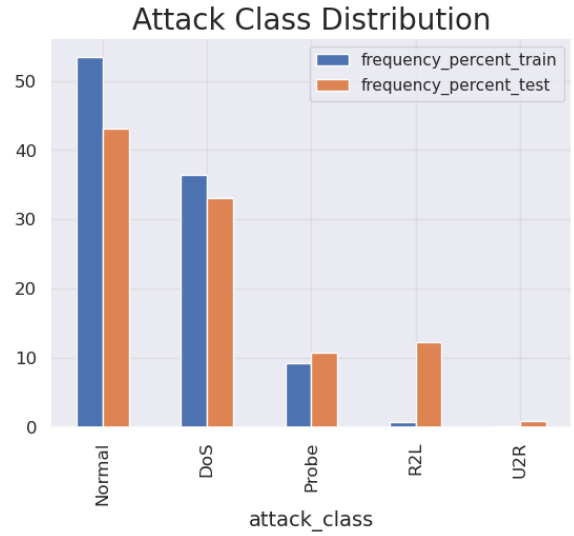
Le déséquilibre flagrant observé dans les tableaux 2.4 et 2.5, ainsi que les distributions dans les figures 2.7a et 2.7b, induisent un biais critique lors de l'entraînement des modèles d'apprentissage automatique. Dans un contexte de détection d'intrusions, les algorithmes standards, qui cherchent souvent à maximiser la précision globale (*accuracy*), tendent à favoriser la classe majoritaire (trafic normal).

Ce phénomène engendre ce que l'on nomme le « paradoxe de l'exactitude » : un modèle peut atteindre une précision de 99% en classant simplement tout le trafic comme *Normal*, échouant ainsi à détecter les attaques rares (U2R, R2L), qui constituent pourtant les menaces les plus critiques. Pour remédier à ce problème, il est impératif d'adopter des stratégies de ré-échantillonnage (*resampling*) visant à modifier la distribution des données avant l'entraînement.

Nous détaillons ci-après les notions préliminaires relatives aux principales techniques de ré-échantillonnage utilisées dans la littérature pour rétablir l'équilibre entre les classes.



(a) Diagramme pour la distribution des données pour CICIDS17



(b) Diagramme pour la distribution des données pour NSL-KDD

Techniques de sous-échantillonnage (Undersampling)

Le sous-échantillonnage consiste à réduire le nombre d'exemples de la classe majoritaire afin d'atteindre un ratio équilibré avec les classes minoritaires. Deux familles d'algorithmes s'offrent a nous ;

1. **Random Undersampling (RUS) :** Il s'agit de la méthode la plus intuitive, consistant à supprimer aléatoirement des échantillons de la classe majoritaire (Normal) jusqu'à obtenir

TABLE 2.4 – Fréquence des données NSL-KDD

Classe	Nombre	Freq. (%)
Normal	77 054	51.88
DoS	53 385	35.95
Probe	14 077	9.48
R2L	3 749	2.52
U2R	252	0.17

TABLE 2.5 – Fréquence des attaques (CI-CIDS2017)

Classe	Nombre	Freq. (%)
BENIGN	2 273 097	80.30
DoS Hulk	231 073	8.16
PortScan	158 930	5.61
DDoS	128 027	4.52
DoS GoldenEye	10 293	0.36
FTP-Patator	7 938	0.28
SSH-Patator	5 897	0.21
DoS slowloris	5 796	0.20
DoS Slowhttptest	5 499	0.19
Bot	1 966	0.07
Web Attack - Brute Force	1 507	0.05
Web Attack - XSS	652	0.02
Infiltration	36	<0.01
Web Attack - Sql Injection	21	<0.01
Heartbleed	11	<0.01

la taille souhaitée. Bien que cette méthode réduise considérablement le temps de calcul et l'espace mémoire requis, elle présente un inconvénient majeur : la perte d'informations potentielles [17]. En éliminant des données de manière arbitraire, le modèle risque de perdre des caractéristiques essentielles permettant de définir correctement la frontière de décision de la classe normale.

2. **NearMiss :** Contrairement au RUS, la famille d'algorithmes NearMiss sélectionne les échantillons de la classe majoritaire de manière déterministe en se basant sur la distance par rapport aux échantillons de la classe minoritaire [18]. L'objectif est de conserver les données majoritaires les plus "difficiles" à classer (celles proches de la frontière de décision) afin de forcer le modèle à mieux discriminer les deux classes.

Techniques de sur-échantillonnage (Oversampling)

À l'inverse, le sur-échantillonnage vise à augmenter la quantité de données de la classe minoritaire.

1. **Random Oversampling (ROS) :** Cette méthode duplique aléatoirement des échantillons existants de la classe minoritaire. Bien qu'elle permette d'équilibrer les classes sans perdre d'information, elle augmente le risque de sur-apprentissage (*overfitting*). En effet, le modèle apprend par cœur les exemples dupliqués plutôt que de généraliser les caractéristiques de l'attaque [17].
2. **SMOTE (Synthetic Minority Over-sampling Technique) :** Afin de contourner le problème de sur-apprentissage du ROS, Chawla et al. ont proposé SMOTE [17]. Au lieu de réaliser une simple réplication, SMOTE génère de nouvelles données synthétiques par interpolation linéaire. Le fonctionnement de l'algorithme est le suivant :
   (a) Pour chaque échantillon x_i de la classe minoritaire, l'algorithme identifie ses k plus proches voisins (kNN) appartenant à la même classe.
   (b) Un voisin x_{zi} est sélectionné aléatoirement parmi ces k voisins.
   (c) Un nouvel échantillon synthétique $x_{nouveau}$ est créé selon la formule :

$$x_{nouveau} = x_i + \lambda \times (x_{zi} - x_i) \tag{2.4}$$

où $\lambda \in [0, 1]$ est un scalaire aléatoire. Géométriquement, cette opération effectue une interpolation linéaire qui place l'échantillon synthétique sur le segment de droite reliant x_i et x_{zi}.

Géométriquement, cela revient à créer un point sur le segment reliant deux instances d'attaques existantes dans l'espace des caractéristiques.

3. **ADASYN (Adaptive Synthetic Sampling) :** ADASYN est une extension de SMOTE qui introduit une pondération adaptative. Alors que SMOTE génère le même nombre d'échantillons synthétiques pour chaque point minoritaire, ADASYN se concentre davantage sur les échantillons « difficiles à apprendre » [19]. L'algorithme calcule une distribution de densité Γ_i pour chaque exemple minoritaire, basée sur le nombre de voisins appartenant à la classe majoritaire. Ainsi, plus un exemple d'attaque est entouré de trafic normal (donc difficile à détecter), plus ADASYN générera de données synthétiques autour de lui. Cela permet de déplacer dynamiquement la frontière de décision vers les zones les plus complexes.

Approches Hybrides

Il est aussi fréquent de combiner sur-échantillonnage et sous-échantillonnage afin d'épurer les données bruitées produites. Par exemple, la combinaison de **SMOTE et des Tomek Links** consiste d'abord à appliquer SMOTE pour accroître la taille de la classe minoritaire, puis à utiliser les liens de Tomek pour éliminer les paires d'échantillons de classes opposées situées à très faible distance l'une de l'autre. Cette procédure contribue à rendre la frontière de décision plus nette en limitant le chevauchement entre les classes normales et les classes d'attaque.

2.5 Sélection des caractéristiques

Les jeux de données récents dédiés à la détection d'intrusions, comme CICIDS2017 ou NSL-KDD [8, 20], présentent une forte dimensionnalité. CICIDS2017, par exemple, contient 79 variables décrivant les flux réseau (durée, taille des paquets, indicateurs TCP, etc.). Toutefois, ces différentes caractéristiques n'apportent pas toutes la même contribution à la distinction entre trafic légitime et trafic malveillant. Certaines caractéristiques sont redondantes (fortement corrélées entre elles), tandis que d'autres sont non pertinentes (bruit) ou même préjudiciables aux performances du modèle.

La sélection de caractéristiques (*Feature Selection*) constitue ainsi une étape essentielle qui consiste à projeter l'espace d'entrée $\mathbb{R}^d$ dans un sous-espace $\mathbb{R}^m$ (avec $m < d$), tout en préservant au mieux l'information à fort pouvoir discriminant.

Dans le cadre particulier de l'apprentissage par renforcement, cette étape est cruciale pour atténuer le « fléau de la dimension » (*Curse of Dimensionality*) [21], qui désigne l'augmentation exponentielle du volume de l'espace des états en fonction du nombre de variables d'entrée. En Apprentissage par Renforcement, ce phénomène rend l'exploration exhaustive de l'environnement impossible et nuit à la capacité de généralisation de l'agent. En pratique, la complexité temporelle de l'entraînement d'un agent RL croît de manière exponentielle avec la taille de l'espace d'états. Une réduction pertinente du nombre de caractéristiques permet ainsi non seulement d'accélérer la convergence de l'algorithme (tel que PPO ou DQN), mais aussi de diminuer le risque de sur-apprentissage (*overfitting*) et de renforcer l'interprétabilité des résultats.

Les approches de sélection de caractéristiques se classent généralement en trois catégories principales : les méthodes de filtrage, les méthodes d'enveloppement et les méthodes intégrées [3, 22].

2.5.1 Méthodes de Filtrage (Filter Methods)

Les approches de type filtrage mesurent la pertinence des variables de façon intrinsèque, sans faire intervenir de modèle d'apprentissage spécifique. Elles s'appuient sur des propriétés ou des critères statistiques pour calculer un score associé à chaque caractéristique, puis sélectionnent uniquement celles qui obtiennent les meilleures valeurs. Grâce à leur faible complexité en termes de calcul, ces méthodes se révèlent particulièrement appropriées pour traiter des ensembles de données de grande dimension ou de grande taille, comme ceux exploités dans le cadre de ce mémoire.

1. **Gain d'information (Information Gain) :** Le gain d'information quantifie la diminution de l'entropie (c'est-à-dire de l'incertitude) de la variable cible Y (la classe d'attaque) lorsque l'on connaît la variable X (la caractéristique) [23]. Il se définit par :

$$IG(Y, X) = H(Y) - H(Y|X) \tag{2.5}$$

   où $H(Y)$ désigne l'entropie de la classe et $H(Y|X)$ l'entropie conditionnelle. Une caractéristique associée à un gain d'information important est considérée comme fortement discriminante pour la tâche de classification.

2. **Test du Chi-carré (χ^2) :** Ce test statistique permet de mesurer le degré d'association entre une variable descriptive et la variable cible. Une valeur élevée de la statistique de test χ^2 indique une forte probabilité que la caractéristique soit liée à la classe, ce qui constitue un argument en faveur de son maintien dans l'ensemble des variables explicatives.

3. **Corrélation de Pearson :** Cette approche sert à analyser la colinéarité entre les différentes caractéristiques. Lorsque deux variables X_i et X_j présentent un coefficient de corrélation de Pearson proche de 1 ou de -1, elles sont considérées comme fortement corrélées et véhiculent ainsi une information largement redondante. Dans ce cas, il devient judicieux de ne conserver qu'une seule de ces caractéristiques afin de simplifier le modèle, réduire sa complexité et limiter le risque de surajustement, tout en préservant l'essentiel de l'information pertinente.

2.5.2 Méthodes d'Enveloppement (Wrapper Methods)

À la différence des approches de filtrage, les méthodes d'enveloppement s'appuient directement sur un modèle prédictif donné pour juger de la qualité d'un sous-ensemble de variables. Elles évaluent donc les caractéristiques en fonction de leur impact réel sur la performance du modèle. Le procédé est généralement itératif : on génère ou explore plusieurs combinaisons de caractéristiques, on entraîne le modèle sur chacune d'elles, puis on compare les performances (par exemple à l'aide de la précision ou du F1-score) afin de retenir les sous-ensembles les plus pertinents.

Recursive Feature Elimination (RFE) : Parmi les méthodes d'enveloppement, Recursive Feature Elimination (RFE) est l'une des plus représentatives. Le principe consiste d'abord à ajuster le modèle sur l'ensemble complet des caractéristiques, puis à les classer selon un critère d'importance dérivé du modèle (poids, coefficients ou scores d'importance). À chaque itération, la caractéristique jugée la moins influente est supprimée, puis le modèle est réentraîné sur le nouvel ensemble réduit. Ce processus récursif se poursuit jusqu'à ce que l'on atteigne le nombre de caractéristiques souhaité [24]. RFE peut fournir des sous-ensembles de caractéristiques très efficaces et compacts, mais son principal inconvénient réside dans son coût computationnel :

chaque étape nécessite un nouvel entraînement du modèle, ce qui devient rapidement prohibitif pour des jeux de données volumineux ou de haute dimension[24]. Dans la pratique, on combine souvent RFE avec une étape préliminaire de filtrage afin de réduire l'espace de recherche avant d'appliquer cette méthode plus fine mais plus coûteuse.

2.5.3 Méthodes Intégrées (Embedded Methods)

Les méthodes intégrées réalisent la sélection des caractéristiques au cœur même du processus d'apprentissage du modèle. Autrement dit, la sélection de variables est intégrée à l'algorithme d'entraînement au lieu d'être appliquée en amont (filtrage) ou en aval (enveloppement). Ces approches cherchent à tirer parti de la rapidité et de la simplicité des méthodes de filtrage, tout en conservant la capacité des méthodes d'enveloppement à prendre en compte les interactions avec le modèle et les dépendances entre variables. Elles sont particulièrement appréciées lorsque l'on souhaite un bon compromis entre performance prédictive, complexité de calcul et interprétabilité du modèle.

Importance des caractéristiques via Forêts aléatoires : Les modèles fondés sur des arbres de décision et, en particulier, les forêts aléatoires (Random Forest), fournissent intrinsèquement une estimation de l'importance des variables. Lors de la construction des arbres, l'algorithme mesure, à chaque nœud, dans quelle mesure la variable utilisée pour la scission contribue à réduire l'impureté (par exemple, selon l'indice de Gini ou l'Entropie) [25]. Cette d'impureté est ensuite calculée sur tous les nœuds où la moyenne est calculée sur l'ensemble des arbres de la forêt.. Les caractéristiques qui apparaissent fréquemment près de la racine des arbres et qui entraînent de fortes diminutions d'impureté sont interprétées comme étant les plus influentes pour la prédiction. On obtient ainsi un classement global des variables par importance. Cette approche est largement répandue, car elle est robuste au bruit, gère naturellement les relations non linéaires et les interactions complexes entre caractéristiques et demeure applicable à des ensembles de données de grande taille. Dans le contexte de l'analyse du trafic réseau, l'utilisation de Random Forest pour l'estimation de l'importance des attributs permet de repérer efficacement les variables les plus discriminantes (par exemple, certains champs de paquets, des statistiques de flux ou des indicateurs temporels), ce qui améliore à la fois la performance des modèles de détection et leur interprétabilité pour les analystes de sécurité.

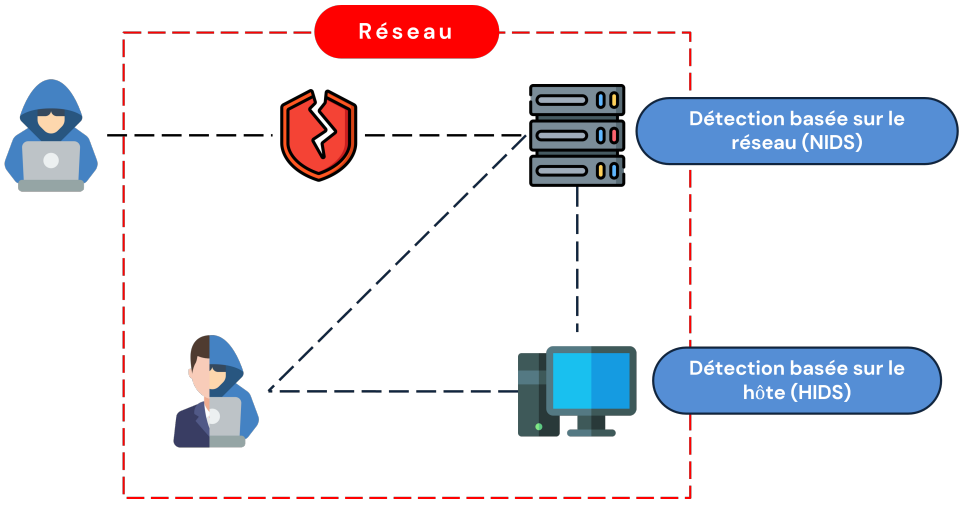


FIGURE 2.8 – Attaque cybernétique

2.6 Taxonomie des attaques et des techniques d'évasion

Une attaque cybernétique est toute tentative d'intrusion illégale ou légale à des fins illégales. En d'autres termes, tout ce qui pourrait nuire à la confidentialité, à l'intégrité ou à la disponibilité de l'information dans un système sera considéré comme une attaque. La montée en sophistication des cybermenaces impose la mise en place d'une classification des intrusions particulièrement structurée et précise. Dans cette optique, cette section présente de manière détaillée les différentes familles d'attaques observées à la fois dans les jeux de données historiques (NSL-KDD) et dans les jeux de données plus récents et représentatifs des menaces actuelles (CICIDS2017). Elle décrit également les principales techniques de dissimulation et de contournement utilisées par les attaquants pour masquer leurs activités, échapper aux mécanismes de détection et rendre l'analyse de ces attaques plus complexe.

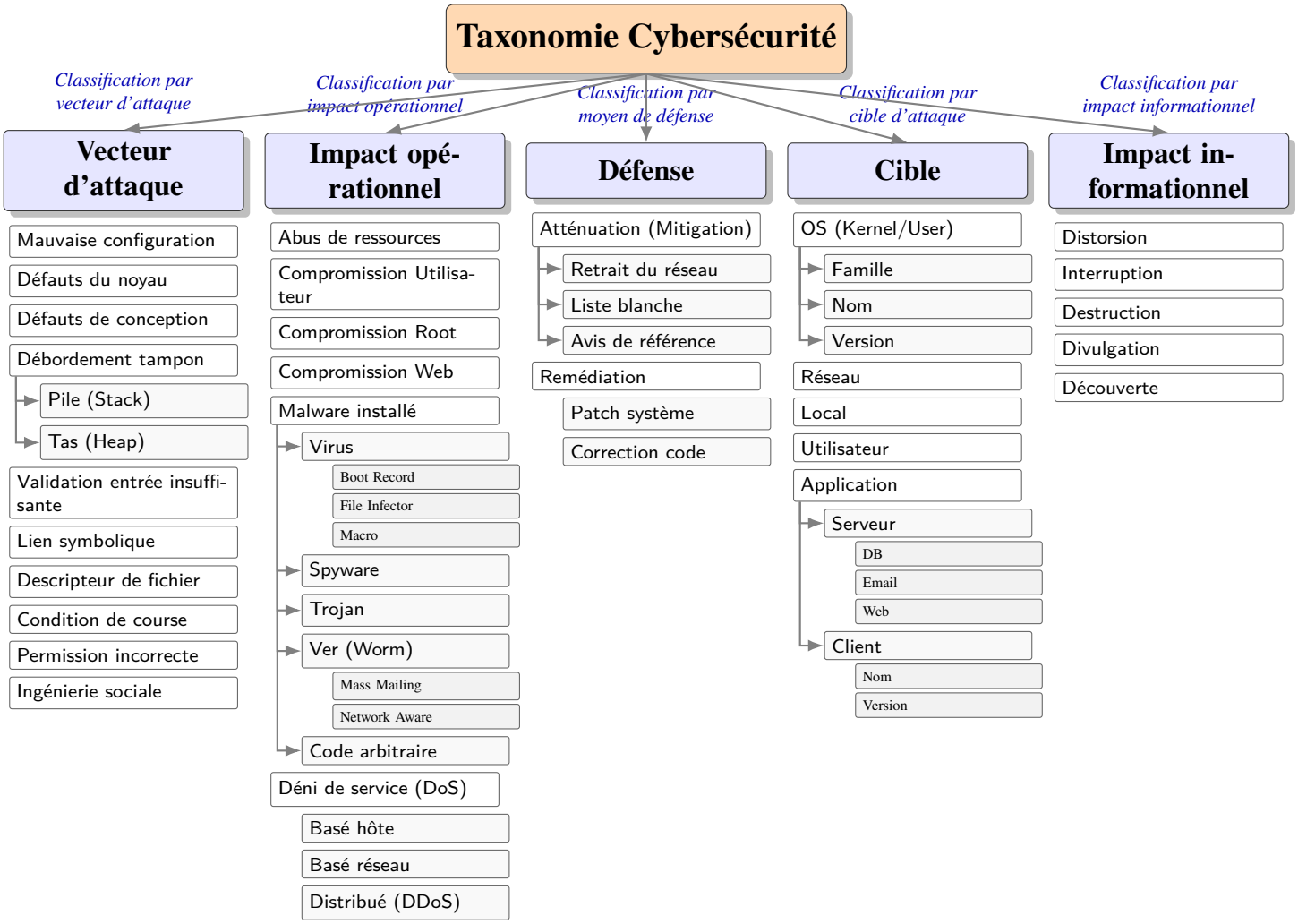


FIGURE 2.9 – Taxonomie des cyberattaques

Comme l'illustre la figure 2.9 [26], une attaque ne se définit pas uniquement par son vecteur technique, mais aussi par son impact opérationnel (déni de service, compromission) et la cible visée (réseau, application, OS). Cette vision multidimensionnelle permet de mieux contextualiser les quatre grandes familles d'attaques historiques que nous détaillons ci-après.

2.6.1 Classification traditionnelle (Modèle DARPA/KDD)

Historiquement, les attaques réseau sont classées en quatre grandes familles, telles que définies dans les travaux de Khraisat et al. [3] :

— **Déni de Service (DoS) :** Tentative de rendre une ressource (serveur, réseau) indisponible. Cela inclut la saturation de la bande passante (ex : *UDP Flood*) ou l'épuisement des ressources système (ex : *TCP SYN Flood*).

— **Sondage (Probe) :** Activités de reconnaissance visant à cartographier le réseau (ex : *PortScan*, *IP Sweep*) pour identifier les vulnérabilités avant une attaque réelle.

— **User-to-Root (U2R) :** Un attaquant disposant d'un compte utilisateur local exploite une vulnérabilité (ex : *Buffer Overflow*) pour obtenir les privilèges d'administrateur (Root).

— **Remote-to-Local (R2L) :** Un attaquant distant tente d'obtenir un accès utilisateur sur une machine cible sans avoir de compte légitime, souvent via des attaques par force brute ou l'exploitation de services mal configurés.

2.6.2 Menaces modernes et spécifiques

Le jeu de données CICIDS2017 met en scène des scénarios d'attaques sophistiqués, conçus pour reproduire de manière réaliste le paysage contemporain des menaces. D'après les travaux de Sharafaldin et al. [8], l'ensemble de ces attaques peut être réparti en plusieurs sous-catégories jugées critiques :

Attaques Web (OWASP Top 10)

Ces attaques visent spécifiquement la couche applicative des systèmes et se révèlent d'autant plus ardues à identifier que les pare-feux traditionnels ne sont généralement pas conçus pour les repérer efficacement.

— **Injection SQL (SQLi) :** L'attaquant insère des requêtes SQL étrangères et malveillantes dans les champs de saisie de l'application (formulaires, paramètres d'URL, cookies, etc.) afin d'altérer le comportement normal du système de gestion de base de données. Cette attaque peut permettre, entre autres, d'extraire des informations sensibles, de modifier ou supprimer des enregistrements, ou encore de contourner les mécanismes d'authentification et de contrôle d'accès [27]. Comme exemple, l'instruction ci-dessous nous permettrait de récupérer l'intégralité des *Id* de l'application, car même si l'utilisateur 105 n'existe pas, l'instruction **1=1** renvoie *vrai*.

```
SELECT * FROM Users WHERE UserId = 105 OR 1=1
```

— **Attaque par script intersite (XSS) :** Insertion de scripts malveillants dans des pages web consultées par d'autres utilisateurs, ce qui permet à un attaquant d'exécuter du code dans le navigateur de la victime et, par exemple, de dérober des cookies de session [27].

— **Force brute :** Méthode consistant à tester de façon exhaustive un grand nombre de combinaisons d'identifiants de connexion, généralement à l'aide de listes de mots (dictionnaires) ou (word list) et d'outils automatisés, afin de trouver les bons paramètres d'accès. Les dictionnaires sont écrits par un système automatisé et peuvent parfois contenir plusieurs mots allant jusqu'à 31 bits, d'où le dictionnaire peut avoir plusieurs Gigaoctets de volume.

Botnets et Infiltration

— **Botnet (Ares) :** Il s'agit d'un ensemble de machines compromises (appelées zombies) pilotées à distance par un serveur de commande et de contrôle (C&C). Dans CICIDS2017, le botnet *Ares* est reproduit afin de modéliser ce type de menace : il exploite différents canaux de communication pour exfiltrer des informations sensibles ou pour orchestrer des attaques par déni de service distribué (DDoS) à grande échelle [8].

— **Infiltration :** Situation dans laquelle l'attaquant tire parti d'une faille logicielle (par exemple via l'ouverture par la victime d'un document piégé) afin d'installer une porte dérobée (*backdoor*) sur le système ciblé, puis d'y exécuter des commandes internes pour explorer l'environnement ou préparer d'autres actions (par exemple lancer un *nmap* en local).

Attaques Cryptographiques

— **Cœur qui saigne (Heartbleed - CVE-2014-0160) :** Une faille majeure affectant la bibliothèque OpenSSL. Elle autorise un attaquant à extraire, à chaque requête *Heartbleed*, jusqu'à 64 Ko de la mémoire vive du serveur, ce qui peut révéler des informations hautement sensibles telles que des clés privées de chiffrement ou des mots de passe. Cette exfiltration de données s'effectue de manière transparente, sans générer d'entrées suspectes dans les journaux classiques de supervision et d'audit [28].

2.6.3 Techniques d'évasion avancées

Afin d'échapper à la détection par les IDS, les attaquants mettent en œuvre diverses techniques, initialement décrites par Ptacek et Newsham [29] et plus tard détaillées et synthétisées par Khraisat [3], qui reprennent et approfondissent ces stratégies d'évasion.

Fragmentation et Désynchronisation

L'attaquant découpe la charge utile malveillante en une multitude de très petits fragments IP. Si l'IDS ne parvient pas à reconstituer ces paquets de manière fidèle – par exemple parce que son délai de réassemblage diffère de celui de la machine visée, ou parce qu'il gère autrement les fragments qui se chevauchent et s'écrasent – il ne reconnaîtra pas la signature de l'attaque. Parallèlement, l'hôte ciblé, lui, réassemblera correctement les fragments et pourra alors exécuter le code malveillant [29].

Obfuscation (Offuscation)

Cette technique vise à masquer la signature statique de l'attaque.

— **Encodage :** Utilisation d'encodages comme Base64, URL-encoding ou Unicode pour transformer les chaînes de caractères (ex : `/bin/sh` devient `%2f%62%69...`) afin qu'elles ne correspondent pas aux règles de l'IDS.

— **Polymorphisme :** Modification dynamique du code de l'attaque à chaque itération (changement de variables, ajout d'instructions inutiles) pour changer son empreinte numérique (hash) sans changer sa fonction [3].

Chiffrement (Encryption)

L'utilisation de tunnels chiffrés (HTTPS, SSH, TLS) rend le contenu des paquets illisible pour l'IDS (L'inspection approfondie des paquets est impossible sans proxy). Les attaquants dissimulent leurs communications C&C ou l'exfiltration de données dans ce trafic chiffré, forçant les IDS à se baser uniquement sur des métadonnées statistiques (analyse de flux) plutôt que sur des signatures [8].

2.6.4 Les attaques Zero-Day

L'essor des malwares contemporains met en évidence les limites des approches de sécurité classiques et constitue un défi majeur pour leur efficacité. Au sein de cet ensemble de menaces, les attaques dites "Zero-Day" représentent l'un des risques les plus sévères pour les infrastructures réseau ainsi que pour les systèmes industriels. On parle d'attaque zero-day lorsqu'une vulnérabilité est activement exploitée dans l'environnement réel (*in the wild*) avant d'être rendue publique ou documentée par les éditeurs et la communauté de sécurité. De manière analogue, une vulnérabilité zero-day désigne une faille qui sert spécifiquement de vecteur dans une attaque zero-day [4]. La figure 2.10 présente de façon schématique le cycle de vie d'une attaque zero-day, depuis l'identification initiale de la vulnérabilité au sein du système, en passant par son exploitation clandestine, jusqu'à sa divulgation publique, suivie de sa prise en compte par les mécanismes de normalisation et de standardisation. Dans la figure 2.11, la chronologie d'une attaque zero-day est représentée. Les différents événements n'apparaissent pas nécessairement dans cet ordre précis, mais certaines contraintes temporelles demeurent : on a $t_a > t_p \geq t_d > t_v$ et $t_0 \geq t_d$, d'où chaque t représente un $temps$. Même si, dans la majorité des situations, il est impossible d'établir avec certitude la relation exacte entre tous ces instants. En outre, pour qu'une attaque soit qualifiée de zero-day, il faut que $t_0 \geq t_e$.

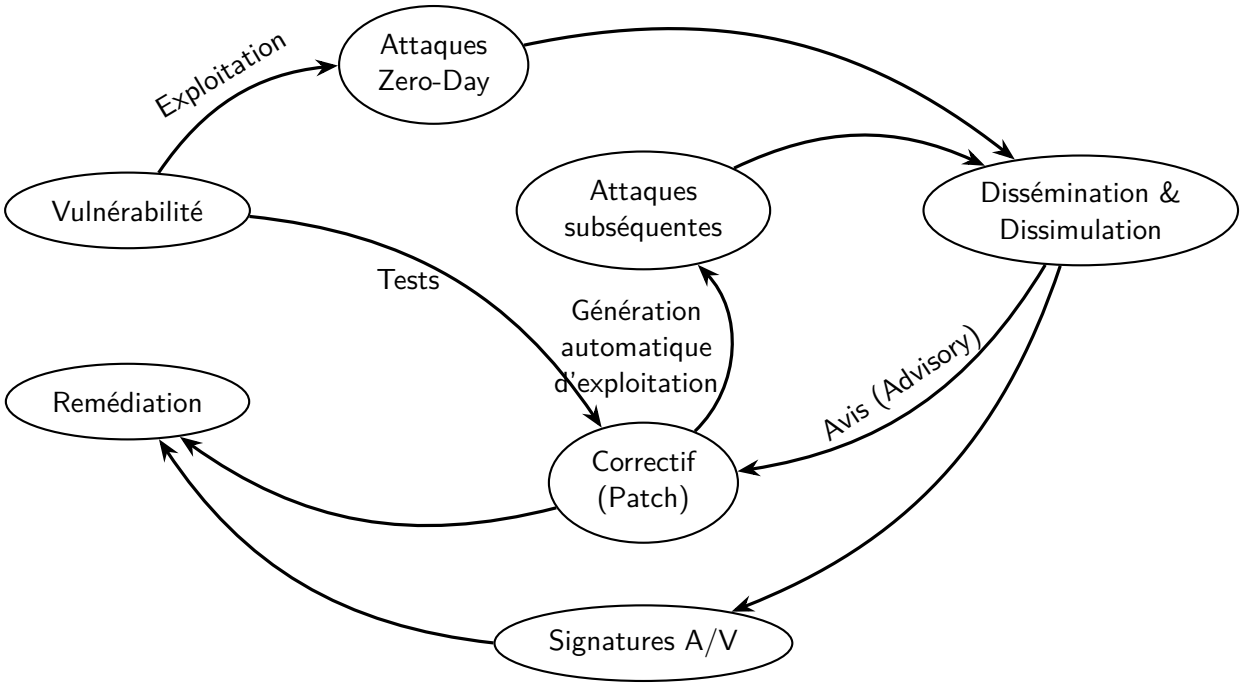


FIGURE 2.10 – Cycle de vie d'une attaque zero-day

- **Introduction de la vulnérabilité** (t_v) : Une vulnérabilité (par exemple une erreur de

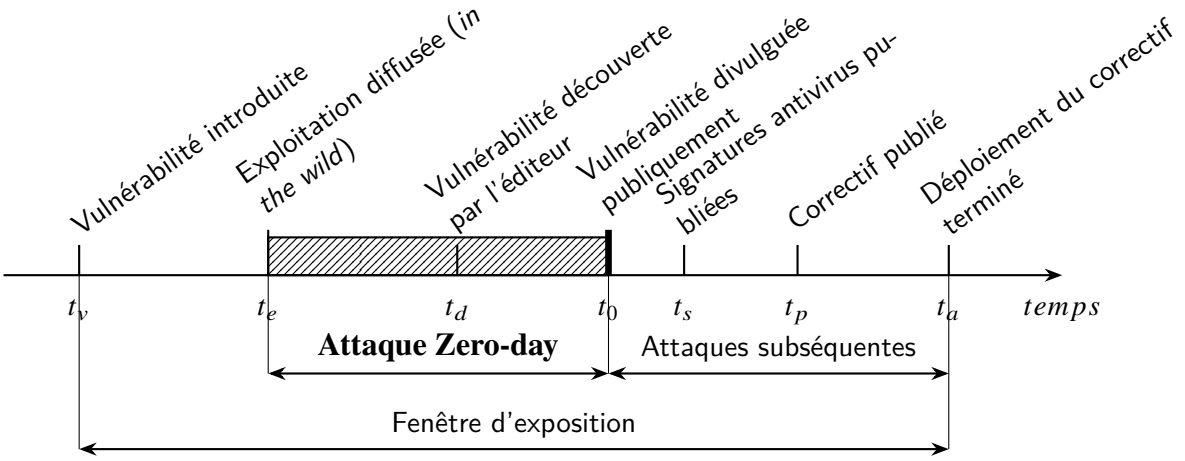


FIGURE 2.11 – Chronologie des attaques zero-day

développement ou une mauvaise gestion de la mémoire) est introduite dans un logiciel, lequel est ensuite distribué et installé sur des hôtes répartis à travers le monde.

- **Exploitation diffusée (in the wild)** (t_e) : Des acteurs malveillants de l'économie souterraine identifient cette vulnérabilité, conçoivent un exploit opérationnel et s'en servent pour conduire des attaques discrètes ciblant des systèmes soigneusement sélectionnés.
- **Vulnérabilité découverte par l'éditeur** (t_d) : L'éditeur du logiciel finit par prendre connaissance de la faille (via ses propres procédures de test ou grâce à un signalement externe), en analyse l'impact et le niveau de critique, fixe un ordre de priorité pour la corriger et lance le développement d'un correctif.
- **Vulnérabilité divulguée publiquement** (t_0)] : La vulnérabilité est rendue publique, soit officiellement par l'éditeur, soit par une publication sur des forums spécialisés, des listes de diffusion ou d'autres canaux ouverts. Un identifiant CVE (par exemple CVE-2010-2568) est alors associé à cette vulnérabilité afin de la référencer de manière standardisée.
- **Signatures antivirus publiées** (t_s) : Après la divulgation publique, les éditeurs d'antivirus conçoivent et diffusent de nouvelles signatures spécifiques aux attaques observées et mettent en place des mécanismes de détection heuristique de l'exploitation. À partir de cette étape, les logiciels A/V mis à jour peuvent repérer et bloquer les tentatives d'exploitation sur les hôtes finaux.
- **Correctif (patch) publié** (t_p) : À la date de divulgation, ou peu de temps après, l'éditeur met à disposition un correctif (patch) corrigeant la vulnérabilité. Dès lors, les systèmes qui appliquent ce correctif ne sont plus exposés à cette forme d'attaque particulière.
- **Déploiement du correctif terminé** (t_a) : Le déploiement du patch est achevé sur l'ensemble des hôtes concernés dans le monde ; tous les systèmes vulnérables ont été mis à jour, ce qui met fin à l'impact pratique de la vulnérabilité.

Ampleur et Impact opérationnel

La fréquence de ces attaques a connu une augmentation exponentielle. Selon le rapport de sécurité de Symantec cité par Khraisat et al. [3], le volume des attaques Zero-Day a substantiellement augmenté, avec des milliards d'incidents de brèches de sécurité recensés. Contrairement aux anciennes générations de malwares qui ciblaient les utilisateurs

finaux de manière opportuniste, les nouvelles variantes sont "plus ambitieuses" et ciblent directement les institutions financières et les infrastructures critiques pour des gains financiers massifs ou du sabotage. L'impact est particulièrement sévère sur les systèmes hérités (Legacy Systems) et les systèmes de contrôle industriel (ICS), où les mises à jour de sécurité sont difficiles, voire impossibles à appliquer.

La nécessité de l'approche comportementale (AIDS)

Face à l'obsolescence des SIDS face à ces menaces, la détection basée sur les anomalies devient la priorité absolue de la recherche. L'avantage majeur de l'AIDS est son indépendance vis-à-vis des bases de signatures : en modélisant le comportement "normal" du système, l'IDS est théoriquement capable de détecter une attaque Zero-Day dès lors qu'elle génère une déviation statistique significative (anomalie). Cependant, cette approche présente un défi inhérent : le taux élevé de faux positifs. Une activité légitime mais inhabituelle (nouveau comportement utilisateur) risque d'être classée comme une attaque, obligeant à un compromis constant entre la sensibilité de détection et la fiabilité des alertes.

2.7 Techniques d'attaque pilotées par l'intelligence artificielle

L'appropriation de l'intelligence artificielle par les cybercriminels constitue une rupture profonde dans la manière de concevoir et de conduire les attaques informatiques. Là où les approches classiques reposaient sur des règles déterministes et figées (de type conditionnel), les offensives fondées sur l'IA (Offensive AI) disposent désormais de mécanismes d'apprentissage et d'adaptation continues à leur environnement opérationnel. Comme le soulignent Guembe et al. [30], ces nouvelles formes d'attaques se distinguent par une vitesse d'exécution de type machine speed et une capacité de déploiement à grande échelle, toutes deux largement supérieures aux facultés de réaction humaines. L'enjeu dépasse donc la simple automatisation de tâches malveillantes : il s'agit d'atteindre un véritable niveau d'autonomie, dans lequel le malware est en mesure de prendre des décisions contextualisées, d'ajuster sa stratégie en temps réel, de sélectionner ses cibles ou ses vecteurs d'intrusion, et ainsi de maximiser l'impact des dommages tout en préservant un haut degré de furtivité et de résilience face aux mécanismes de défense.

2.7.1 Taxonomie des attaques par IA

Une revue systématique de la littérature réalisée par Guembe et al. (2022) [30], portant sur un ensemble de 46 articles, a mis en évidence 19 cas d'usage différents de l'IA à des fins offensives. La mise en perspective de ces techniques au regard de la *Cyber Kill Chain* montre une répartition non uniforme des efforts de recherche et de développement menés par les attaquants.

Comme l'illustre la figure 2.12, la concentration des techniques d'IA dans les phases de Reconnaissance et d'Accès (10 techniques sur 18) souligne l'urgence de déployer des systèmes de détection d'intrusion (IDS) intelligents. Les défenses traditionnelles étant souvent contournées dès ces premières étapes, un IDS basé sur l'apprentissage par renforcement (comme proposé dans ce mémoire) devient essentiel pour détecter les comportements anormaux post-intrusion que les règles statiques ne peuvent pas percevoir.

L'analyse technique des outils offensifs recensés par Guembe et al. (2022) [30] révèle une nette prédominance des modèles d'apprentissage profond (Deep Learning) pour exécuter la

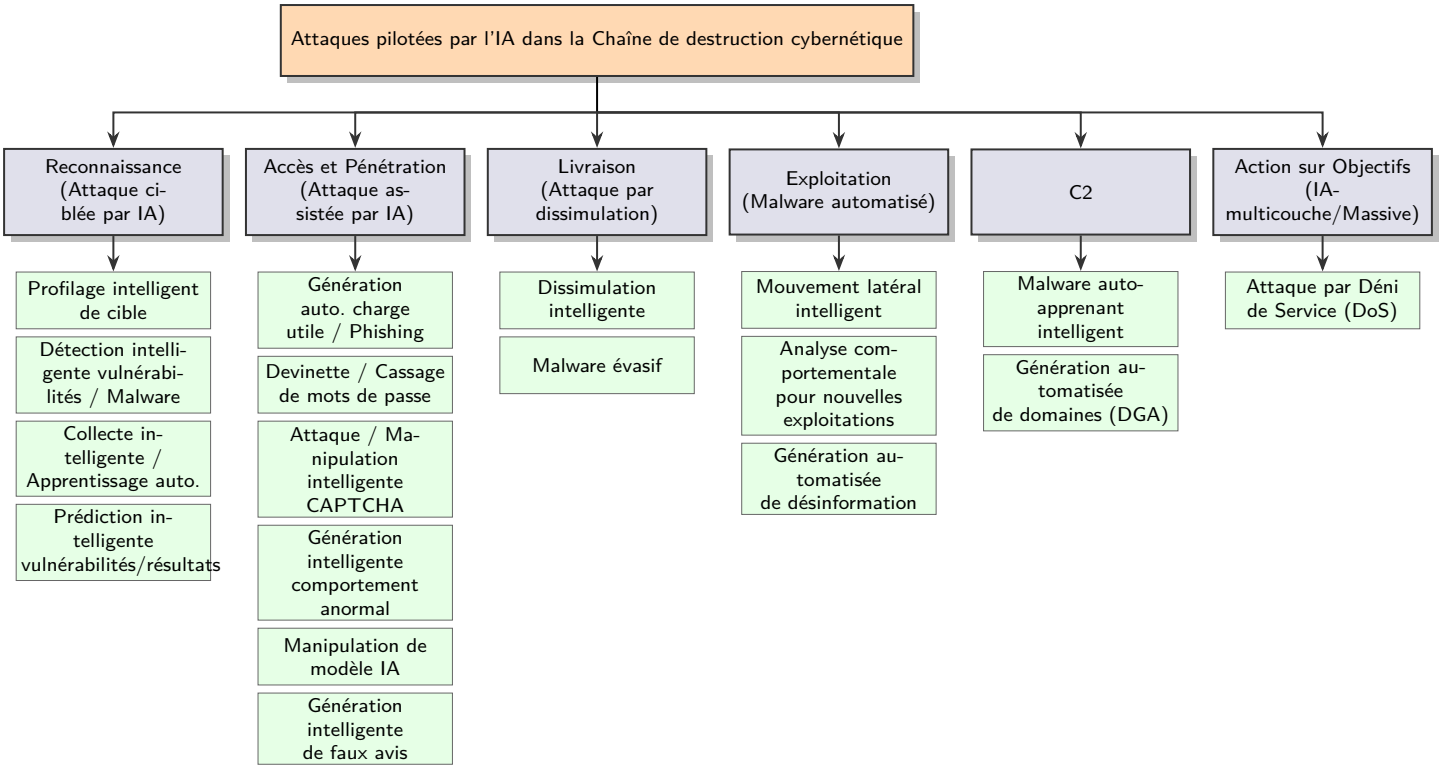


FIGURE 2.12 – Chaîne de cyberattaques modifiée pour les attaques basées sur l'IA [1].

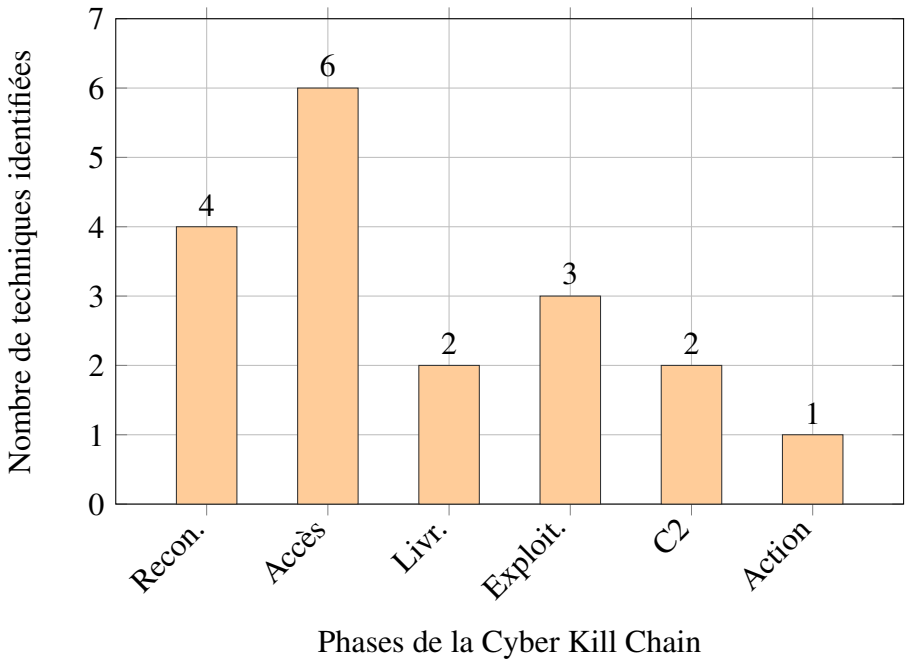


FIGURE 2.13 – Répartition des techniques d'IA Offensive par Phase de la Kill Chain

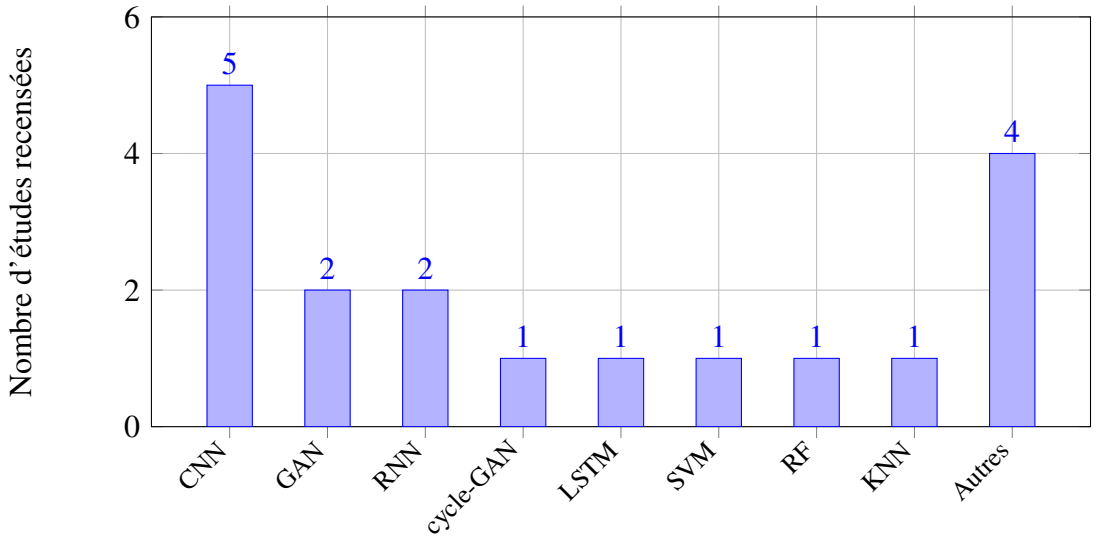


FIGURE 2.14 – Fréquence des types d'IA (Phase d'Accès et Pénétration)

phase critique d'Accès et de Pénétration. Comme l'illustre la Figure 2.14 ci-dessous, les Réseaux de Neurones Convolutifs (CNN) constituent la technique la plus fréquemment employée, suivis par les architectures génératives (GAN) et séquentielles (RNN).

Cette distribution n'est pas fortuite ; elle correspond aux besoins spécifiques des vecteurs d'attaque modernes : la vision par ordinateur pour briser les sécurités visuelles et la génération de séquences pour le cassage de mots de passe.

2.8 Métriques d'évaluation d'un IDS

Plusieurs métriques permettent d'évaluer les performances d'un modèle IDS. Dans une matrice de confusion, les éléments situés sur la diagonale correspondent aux prédictions correctes, tandis que les éléments hors diagonale reflètent les erreurs commises par un classificateur donné [31]. Le tableau 2.6 présente ces différents éléments de la matrice de confusion. Par ailleurs, les principales métriques d'évaluation mobilisées dans les travaux récents sont les suivantes :

Vrais Positifs (TP) : Représentent les instances où le modèle a correctement prédit la classe positive.

Vrais Négatifs (TN) : Représentent les instances où le modèle a correctement prédit la classe négative.

Faux Positifs (FP) : surviennent lorsque le modèle prédit à tort la classe positive (Erreur de Type I).

Faux Négatifs (FN) : surviennent lorsque le modèle prédit à tort la classe négative alors qu'il s'agit d'un cas positif (Erreur de Type II).

- **Précision :** C'est le ratio des Attaques correctement prédites à tous les échantillons prédits comme des Attaques.

$$Precision = \frac{TP}{TP + FP} \tag{2.6}$$

- **Rappel :** C'est un ratio de tous les échantillons correctement classés comme des Attaques à tous les échantillons qui sont effectivement des Attaques. On l'appelle aussi Taux de Détection ou Taux de vrai positif.

$$Rappel(TPR) = \frac{TP}{TP + FN} \tag{2.7}$$

- **Taux de fausses alarmes :** Il est également appelé taux de faux positifs et est défini comme le ratio des échantillons d'Attaque incorrectement prédits à tous les échantillons qui sont Bénins.

$$FPR = \frac{FP}{FP + TN} \tag{2.8}$$

- **Taux de vrais négatifs :** Il est défini comme le ratio du nombre de Bénins correctement classifiés parmi tous les échantillons qui sont Bénins.

$$TNR = \frac{TN}{TN + FP} \tag{2.9}$$

- **Taux de faux négatifs :** On parle de faux négatif lorsqu'un détecteur n'identifie pas une anomalie et la classe comme Bénigne.

$$FNR = \frac{FN}{TN + FP} \tag{2.10}$$

- **Exactitude :** C'est le ratio des instances correctement classifiées sur le nombre total d'instances. On l'appelle également Exactitude de Détection et c'est une mesure de performance utile uniquement lorsque un ensemble de données est équilibré.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.11}$$

- **Moyenne Harmonique :** La moyenne harmonique est utilisée pour calculer le F1-Score afin de fournir une mesure unique de la performance du modèle. Elle est définie par la relation suivante :

$$H = \frac{2}{\frac{1}{Precision} + \frac{1}{Rappel}} \tag{2.12}$$

Dans le cas spécifique de l'évaluation d'un classificateur binaire, le F1-Score représente la moyenne harmonique entre la précision (P) et le rappel (R) :

$$F - Score = 2 \cdot \frac{Precision \times Recall}{Precision + Rappel} \tag{2.13}$$

Cette métrique est privilégiée car elle pénalise fortement les valeurs extrêmes, forçant ainsi le modèle à être performant sur les deux aspects simultanément.

Ces métriques d'évaluation sont calculées en testant les méthodologies proposées à l'aide de jeux de données de référence. La prochaine section discute de l'ensemble de données publiques populaire utilisé pour tester les NIDS.

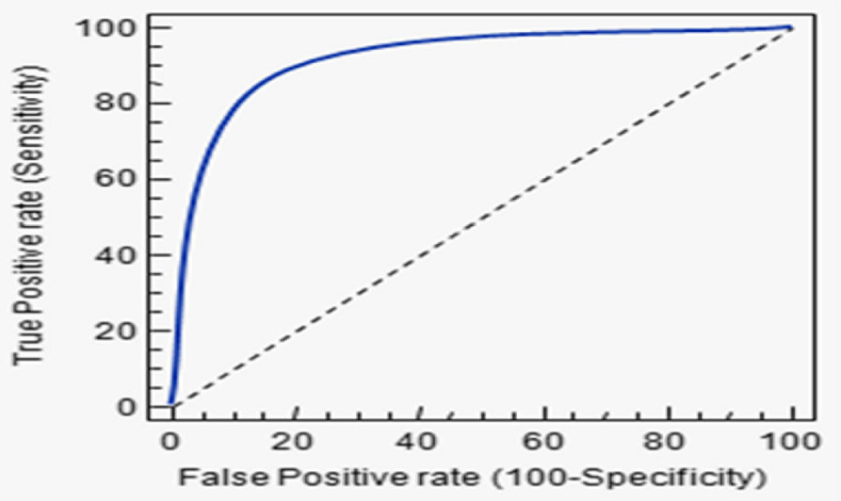


FIGURE 2.15 – Courbe ROC

TABLE 2.6 – Matrice de Confusion

	Classe Prédite	
Classe Réelle	**Attaque**	**Normal**
Attaque	Vrai positif (TP)	Faux négatif (FN)
Normal	Faux négatif (FP)	Vrai négatif (TN)

- **Courbe ROC (Receiver Operating Characteristic) :** la courbe ROC affiche le Taux de faux positifs sur l'axe horizontal et le TPR sur l'axe vertical. Elle montre donc le TPR en fonction du FPR pour une série de seuils de décision. Chaque point de la courbe correspond à une paire (FPR, TPR) associée à un seuil donné. Lorsque le seuil de classification change, on obtient un autre point sur la courbe ROC, caractérisé par un nouveau taux de fausses alarmes (FAR) et un nouveau TPR. Un test présentant une capacité de discrimination parfaite (absence totale de chevauchement entre les deux distributions) donne une courbe ROC qui passe par le coin supérieur gauche (sensibilité de 100%, spécificité de 100%). La courbe ROC est représentée à la figure 2.15 [3] par la courbe bleue.

Chapitre 3

État de l'art

3.1 Introduction

Ce chapitre propose une synthèse des travaux de recherche et des développements existants en lien avec les systèmes de détection d'intrusion basés sur l'apprentissage automatique. Il a pour but principal de positionner ce travail dans un cadre scientifique et technique plus large, de mettre en lumière les bases théoriques et pratiques sur lesquelles il repose et de repérer les lacunes auxquelles il cherche à répondre. Nous commençons par présenter les notions essentielles et les premières méthodes qui ont servi de fondement à la détection d'intrusion. Nous nous intéressons ensuite aux contributions les plus proches de notre première approche, NEMESIS, en insistant particulièrement sur les méthodes employées, les résultats obtenus, ainsi que sur les modèles et les technologies les plus pertinents pour nos objectifs. Une attention spécifique est accordée aux études abordant des problématiques analogues, proposant des solutions comparables ou intervenant dans des domaines d'application voisins de celui d'ULYSSE. En examinant et en confrontant de manière systématique les travaux antérieurs, ce chapitre cherche à montrer comment la recherche proposée s'inscrit dans la continuité des approches existantes, les étend ou s'en distingue. Cette mise en perspective justifie non seulement la pertinence de l'étude actuelle, mais contribue aussi à préciser son originalité et sa contribution potentielle. Les connaissances rassemblées ici serviront de référence pour les choix méthodologiques et les décisions de conception présentés dans les chapitres suivants.

3.2 Travaux connexes

L'évolution de la détection d'intrusions réseau (NIDS) s'inscrit dans une quête constante d'équilibre entre la robustesse statistique, la capacité de généralisation et l'agilité algorithmique. La littérature récente s'articule autour de plusieurs axes méthodologiques que nous détaillons ci-après.

— **Détection basée sur les signatures et approches industrielles :** Les approches de détection basées sur les signatures constituent toujours la référence industrielle pour l'identification des menaces connues. Comme le soulignent Mughal et al. [32], ces solutions effectuent une comparaison bit à bit entre le flux entrant et un référentiel de signatures, offrant une précision quasi-parfaite et un taux de faux positifs réduit pour les attaques classiques. Cependant, cette méthode présente une limite intrinsèque face au polymorphisme. Hussain et al. [33] ajoutent que la gestion de ces bases de signatures devient ardue avec l'augmentation du volume de données, provoquant une diminution des performances

sur les liens à très haut débit. Cette incapacité à appréhender les attaques « Zero-day » légitime l'orientation vers les modèles comportementaux que nous étudions.

— **Paradigme supervisé et apprentissage profond :** Les approches supervisées s'appuient sur des données étiquetées pour atteindre des taux de précision élevés. Egwu et al. [34] ont démontré que les modèles arborescents comme *Forêt aléatoire* conservent une efficacité redoutable sur CIC-IDS2017, offrant une rapidité cruciale pour le temps réel. Toutefois, face à la complexité des infrastructures Cloud et IoT, Neto et al. [35] soulignent que les réseaux de neurones profonds deviennent indispensables pour capturer les motifs d'attaque non linéaires que les modèles classiques ne parviennent plus à isoler.

— **Méthodes non supervisées et détection d'anomalies :** Pour s'affranchir du besoin de signatures, la recherche s'est tournée vers le paradigme non supervisé. Yoo et al. [36] exploitent la structure temporelle du trafic via des réseaux de reconstruction récurrents (RRN), où toute déviation de l'erreur de reconstruction signale une intrusion. Cette approche est complétée par les travaux d'Al-Zewairi et al. [37], qui utilisent le *clustering* dynamique pour regrouper les flux suspects. Bien que protectrices contre les menaces inconnues, ces méthodes génèrent souvent un taux de faux positifs plus élevé, menant au développement d'architectures hybrides.

— **Architectures hybrides, intelligence générative et explicabilité :** L'état de l'art converge vers des systèmes capables d'apprendre et d'expliquer leurs décisions. L'intégration de l'IA générative, illustrée par le cadre AAE-DRL de Kotb et al. [38], permet de simuler des attaques rares pour équilibrer l'apprentissage. Parallèlement, Zhang et al. [39] ont démontré qu'une synergie entre le *clustering*, *Forêt aléatoire* et l'explicabilité par valeurs de SHAP permet d'atteindre une précision de 94,3% tout en offrant une interprétation sémantique de chaque alerte. C'est dans cette lignée que s'inscrivent **NEMESIS** et **ULYSSE**.

— **Apprentissage par renforcement profond pour la détection :** L'intégration du DRL, particulièrement des DQN, permet de renforcer la réactivité des IDS. Tellache et al. [40] utilisent un agent DQN pour repérer les activités malveillantes avant une classification supervisée fine. Malik et Saini [41] proposent un filtrage similaire par DQN pour améliorer la granularité. Par ailleurs, Sethi K. et al. [42] ont introduit un IDS contextuel où une logique centrale gère les récompenses et la localisation de l'attaque, tandis qu'Alavizadeh et al. [14] ont optimisé les hyperparamètres d'un DQN pour la détection multiclasses. Enfin, Caminero et al. [43] ont conçu un agent DQN focalisé sur les attaques rares, atteignant une précision compétitive de 80,16

— **Optimisation de l'espace des caractéristiques (Feature Selection) :** La haute dimensionnalité reste un défi majeur traité selon trois axes :

  — **Méta-heuristiques :** Utilisation d'algorithmes bio-inspirés comme l'optimisation des sauterelles (GOA) [44], l'optimisation de plantes (APO) [45] ou le *Spider Monkey Optimization* (SMO) [46]. Ces méthodes élaguent les données mais restent souvent coûteuses et statiques.

  — **Extraction profonde :** Transformation des données en représentations latentes via des auto-encodeurs empilés [47] ou la conversion en images via VGG16 [48]. Ces approches souffrent toutefois d'une perte d'interprétabilité.

  — **Sélection dynamique par RL :** Approche adoptée par **ULYSSE**, où la sélection est un problème de décision séquentielle. Al-Dwairi et al. [11] ont utilisé le DQN pour identifier un sous-ensemble minimal, tandis que Mohammadi et al. [49] ont

proposé une architecture multi-agents RL. Enfin, Jiang et al. [50] traitent la dérive conceptuelle par un mécanisme de vote basé sur le RL.

3.3 Synthèse et positionnement des travaux

Afin de mettre en perspective les différentes approches explorées et d'identifier clairement les lacunes comblées par nos travaux, le tableau 3.1 synthétise les principales contributions de la littérature. Cette vue d'ensemble permet de comparer les paradigmes de détection, allant des signatures traditionnelles aux méthodes d'apprentissage par renforcement les plus récentes,en mettant en relief leurs forces respectives ainsi que leurs limites intrinsèques en termes de latence, de précision et de capacité d'adaptation. Cette synthèse justifie ainsi le positionnement de nos modèles **NEMESIS** et **ULYSSE**, conçus pour répondre spécifiquement aux défis de la haute dimensionnalité et de l'efficience opérationnelle dans les réseaux modernes.

TABLE 3.1 – Synthèse des travaux connexes et positionnement des contributions NEMESIS et ULYSSE

Approche / Paradigme	Références clés	Algorithmes / Méthodes	Forces (+) et Limites (-)
Signatures based SIDS	Mughal et al. [32] Hussain et al. [33]	Pattern Matching (Bit-à-bit)	(+) Précision quasi-parfaite (attaques connues) (-) Inefficace face au Zero-day et polymorphisme
Supervisé & Profond AIDS	Egwu et al. [34] Neto et al. [35]	Random Forest, J48, DNN CNN (IoT/Cloud)	(+) Haute précision, capture des motifs complexes (-) Latence élevée, dépendance aux étiquettes
Non Supervisé AIDS	Yoo et al. [36] Al-Zewairi et al. [37]	Auto-encodeurs (RRN) Clustering dynamique	(+) Détection d'attaques inconnues (anomalies) (-) Taux de faux positifs souvent plus élevé
Hybride & XAI	Kotb et al. [38] Zhang et al. [39]	AAE + DRL, SHAP	(+) Interprétabilité, gestion des classes rares (-) Complexité architecturale accrue
Hybride RL-Supervisé	Tellache et al. [40] Sethi et al. [42] Benidir et al. [2]	DQN + Classifieurs	(+) Filtrage efficace avant classification fine (+) Résilience accrue et classification robuste (-) Sensibilité au bruit des caractéristiques
Méta-heuristiques (Optimisation)	Alshammari et al. [44] Kanimozhi et Jacob [45]	GOA, APO, SMO (Bio-inspiré)	(+) Réduction efficace de la dimensionnalité (-) Processus statique et coût de recherche élevé
RL Dynamique Sélection des cara.	Al-Dwairi et al. [11] Mohammadi et al. [49]	Deep Q-Learning, PPO Multi-agents	(+) Sélection adaptative en temps réel (-) Instabilité sur très grands espaces d'états
	Contribution actuelle	**ULYSSE (PPO + SHAP)**	**(+) Sélection dynamique, interprétabilité**

3.4 Conclusion

En résumé, ce chapitre a servi à situer notre travail au regard des principales contributions présentées dans l'état de l'art. Les approches existantes ont clairement mis en lumière le rôle central des techniques d'optimisation et ont proposé des solutions efficaces pour réduire l'espace d'état. Néanmoins, malgré leur pertinence, ces travaux souffrent de plusieurs limites, en particulier en ce qui concerne l'interprétabilité des modèles obtenus et leur capacité de généralisation à d'autres contextes ou jeux de données.

Ces observations ont permis de faire émerger plusieurs pistes d'amélioration et de consolider les choix méthodologiques effectués dans le cadre de cette thèse/mémoire. Elles ont notamment guidé la définition de notre problématique et la conception de notre solution. Plus concrètement, notre apport se singularise en introduisant un nouveau système de sélection, conçu pour répondre plus adéquatement aux défis soulevés par la littérature, tant sur le plan de l'explicabilité que sur celui de la robustesse et de l'adaptabilité des résultats.

Le chapitre suivant sera donc dédié à la description approfondie de notre démarche. Il présentera de manière structurée le cadre conceptuel qui sous-tend notre approche, les principes

théoriques sur lesquels elle s'appuie, ainsi que les différents modules qui la composent. Nous détaillerons également les modalités de sa mise en œuvre pratique et les choix techniques qui en découlent. L'ensemble de cette présentation s'articulera directement autour des lacunes, limites et perspectives identifiées dans ce chapitre consacré aux travaux connexes, de façon à montrer explicitement comment notre proposition vient les compléter et y apporter des réponses ciblées.

Chapitre 4

Contributions

4.1 Introduction

Ce chapitre présente la méthodologie développée pour concevoir un système de détection d'intrusions intelligent, interprétable et résilient. Deux propositions complémentaires y sont introduites. La première, nommée NEMESIS, vise à combiner l'apprentissage par renforcement profond et les méthodes d'apprentissage supervisé afin d'améliorer la capacité de détection et de généralisation, ainsi qu'à effectuer une détection précise de la classe d'attaque. La seconde, appelée ULYSSE, prolonge cette approche en intégrant un mécanisme de sélection dynamique des caractéristiques, fondé sur un agent d'apprentissage par renforcement et une supervision interprétable. L'objectif global est d'obtenir un système capable non seulement de détecter efficacement les menaces, mais aussi d'adapter sa représentation des données au contexte et à la variabilité du trafic réseau.

4.2 Sélection du jeu de données et des algorithmes supervisés

Notre sélection, tant pour l'ensemble des données destinées à la validation de nos contributions que pour le choix des algorithmes de classification supervisée, ne repose ni sur le hasard ni sur une simple intuition. Elle résulte d'une analyse empirique approfondie de l'état de l'art, qui nous a conduits à comparer plusieurs approches, globalement équivalentes en termes de performances, avant d'arrêter un choix argumenté. La lecture de l'étude d'Abdallah et al. [51] montre en particulier que les deux jeux de données NSL-KDD et CICIDS17 se prêtent bien à des méthodes de sélection de caractéristiques et qu'un même algorithme de classification, en l'occurrence l'algorithme des *Forêts aléatoires*, peut être appliqué de manière cohérente sur ces deux jeux de données. Leur travail propose une analyse approfondie de la manière dont les systèmes de détection d'intrusion sont classés, de différentes méthodes d'apprentissage supervisé utilisées, ainsi que des principaux types d'attaques en cybersécurité. En nous appuyant sur quatre jeux de données de référence (AWID, NSL-KDD, CICIDS2017 et UNSW-NB15), nous avons passé en revue et synthétisé l'ensemble des contributions majeures de la littérature dans ce domaine. À partir de cette synthèse, une taxonomie pour ces jeux de données a été structurée est élaborée, s'appuyant directement sur les études recensées.

Dans la continuité de cette analyse globale, la pertinence de notre protocole expérimental est spécifiquement corroborée par les travaux récents de [52], qui proposent une évaluation comparative rigoureuse des classificateurs sur ces mêmes jeux de données.

Leurs résultats empiriques mettent en évidence la supériorité de l'algorithme *Random Forest*

par rapport à d'autres méthodes populaires, telles que le *Decision Tree*, le *Naive Bayes* ou encore *XGBoost*. En effet, le RF démontre une stabilité remarquable en atteignant des taux de précision (*accuracy*) de l'ordre de **98,57%** sur NSL-KDD et **98,56%** sur CICIDS2017. Cette constance est cruciale pour notre recherche, car elle prouve que le modèle est capable de généraliser aussi bien sur des signatures d'attaques historiques que sur des vecteurs de menaces modernes et complexes. À l'opposé, des algorithmes plus simples, comme le *Naive Bayes*, montrent leurs limites sur des données hétérogènes, plafonnant, par exemple, à environ 81% de précision sur CICIDS2017 [52]. Ainsi, le choix du *Random Forest* ne s'impose pas seulement par ses performances brutes, mais par sa capacité intrinsèque (grâce au principe de *bagging*) à gérer la haute dimensionnalité et le bruit sans nécessiter de mécanismes de boosting souvent plus coûteux en ressources de calcul.

4.3 Préparation et prétraitement des données

La qualité des données est un facteur déterminant dans la performance des modèles d'apprentissage automatique. Cette section détaille le processus de transformation appliqué aux jeux de données NSL-KDD et CICIDS2017.

4.3.1 Nettoyage des données

Une analyse préliminaire des jeux de données a révélé la présence d'enregistrements redondants et de valeurs aberrantes.

— **Gestion des doublons :** Afin d'éviter que le modèle ne soit biaisé par la répétition fréquente de certains paquets (notamment dans le trafic bénin), nous avons procédé à une suppression rigoureuse des doublons (*duplicates*). Cette étape garantit que les métriques d'évaluation reflètent une réelle capacité de généralisation et non la mémorisation de motifs répétés.

— **Valeurs manquantes et infinies :** Les artefacts de capture générant des valeurs infinies (ex : *Flow Bytes/s*) comportant des virgules allant vers l'infini ou manquantes (NaN - Not a Number) ont été traités par un nettoyage direct pour assurer la stabilité numérique des calculs.

4.3.2 Encodage et normalisation

Les données des réseaux étant hétérogènes, deux transformations majeures ont été appliquées pour les rendre exploitables par les algorithmes numériques (DQN et Random Forest) :

1. **Encodage des variables catégorielles :** Toutes les caractéristiques non numériques (telles que le protocole, les drapeaux TCP ou les identifiants de services) ont été converties en valeurs numériques. Nous avons utilisé la méthode *Label Encoding*, qui assigne un entier unique à chaque modalité.
2. **Normalisation Min-Max :** Les réseaux de neurones profonds étant sensibles à l'échelle des données, nous avons appliqué une normalisation *Min-Max*. Cette transformation projette toutes les caractéristiques (dont les plages de valeurs varient fortement, ex : durée vs octets) dans l'intervalle $[0, 1]$, facilitant ainsi la convergence de l'algorithme d'optimisation (Descente de gradient).

4.3.3 Gestion du déséquilibre des classes

L'un des défis majeurs dans la détection d'intrusions est le déséquilibre des classes (*Class Imbalance*). Dans le jeu de données CICIDS2017, le trafic bénin représente plus de 80% des données. Pour y remédier, nous avons utilisé la bibliothèque Python ***imbalanced-learn*** [53] pour appliquer des stratégies d'échantillonnage adaptées :

Suréchantillonnage synthétique (SMOTE)

Pour le jeu de données complexe CICIDS2017, nous avons utilisé l'algorithme **SMOTE** (*Synthetic Minority Over-sampling Technique*) [17]. Contrairement à la simple duplication, SMOTE génère de nouveaux échantillons synthétiques en interpolant les caractéristiques des instances minoritaires existantes. Cela permet de densifier l'espace de décision des attaques rares (comme les *Web Attacks*) sans provoquer de sur-apprentissage excessif.

Suréchantillonnage aléatoire (Random Over-Sampling)

Pour le NSL-KDD, nous avons opté pour le *Random Over-Sampling* via *imbalanced-learn*. Cette méthode réplique aléatoirement les exemples des classes sous-représentées (U2R, R2L) jusqu'à obtenir une distribution équilibrée, suffisante pour la dimensionnalité de ce jeu de données.

Note méthodologique : Le suréchantillonnage a été appliqué *uniquement* sur l'ensemble d'entraînement. Les ensembles de validation et de test ont conservé leur distribution originale pour garantir une évaluation réaliste.

4.3.4 Distribution des données

Le Tableau 4.1 illustre la distribution binaire utilisée pour l'entraînement de l'agent DQN. Les Tableaux 4.2 et 4.3 détaillent la répartition des classes pour le classificateur Random Forest après l'application de l'équilibrage.

TABLE 4.1 – Distribution binaire des échantillons pour l'entraînement du modèle DQN

Jeu de Données	Classe	Label	Nombre d'échantillons
CICIDS2017	Benign	0	2 271 320
	Attack	1	2 271 320
NSL-KDD	Normal	0	269 372
	Attack	1	269 372

4.3.5 Partitionnement des données

Afin d'assurer une évaluation rigoureuse, les données ont été divisées en sous-ensembles distincts. Pour l'entraînement de l'agent DQN, nous avons opté pour une répartition de **70/15/15** (Entraînement / Validation / Test). Pour le classificateur final Random Forest, une répartition de **80/20** (Entraînement / Test) a été appliquée. Le choix de stratégies de partitionnement distinctes répond aux besoins spécifiques de chaque modèle. Pour l'agent **DQN**, la répartition **70/15/15** introduit un ensemble de validation indispensable à l'optimisation des hyperparamètres

TABLE 4.2 – Distribution équilibrée des classes (CICIDS2017) pour le Random Forest

ID	Type d'Attaque	Échantillons (Après SMOTE)
0	BENIGN	1 816 702
1	Bot	1 816 702
2	DDoS	1 816 702
3	DoS GoldenEye	1 816 702
4	DoS Hulk	1 816 702
5	DoS Slowhttptest	1 816 702
6	DoS slowloris	1 816 702
7	FTP-Patator	1 816 702
8	Heartbleed	1 816 702
9	Infiltration	1 816 702
10	PortScan	1 816 702
11	SSH-Patator	1 816 702
12	Web Attack Brute Force	1 816 702
13	Web Attack Sql Injection	1 816 702
14	Web Attack XSS	1 816 702

TABLE 4.3 – Distribution équilibrée des classes (NSL-KDD) pour le Random Forest

ID	Type d'Attaque	Échantillons (Après Over-Sampling)
0	Normal	67 343
1	DoS	67 343
2	Probe	67 343
3	R2L	67 343
4	U2R	67 343

et à la surveillance de la convergence de la politique, évitant ainsi le surapprentissage. À l'inverse, pour le classificateur **Forêt aléatoire**, une répartition **80/20** a été privilégiée. Cette configuration permet de maximiser la base de connaissances du modèle tout en conservant un échantillon de test suffisant pour valider la robustesse de la classification finale. Cette approche différenciée garantit que l'évaluation de la robustesse face aux attaques (notamment *zero-day* et polymorphes) repose sur des bases statistiquement fiables.

4.4 Approche NEMESIS

4.4.1 Architecture et hyperparamètres

La conception de l'agent intelligent au cœur du système NEMESIS repose sur une architecture de type Deep Q-Network (DQN). Cette approche combine l'apprentissage par renforcement (Q-Learning) avec des réseaux de neurones profonds pour approximer la fonction de valeur $Q(s, a)$ dans des espaces d'états de haute dimension. Le choix de cette architecture et des hyperparamètres associés résulte d'une étude empirique rigoureuse que nous avons menée, visant à maximiser la précision de détection tout en garantissant la stabilité de l'apprentissage.

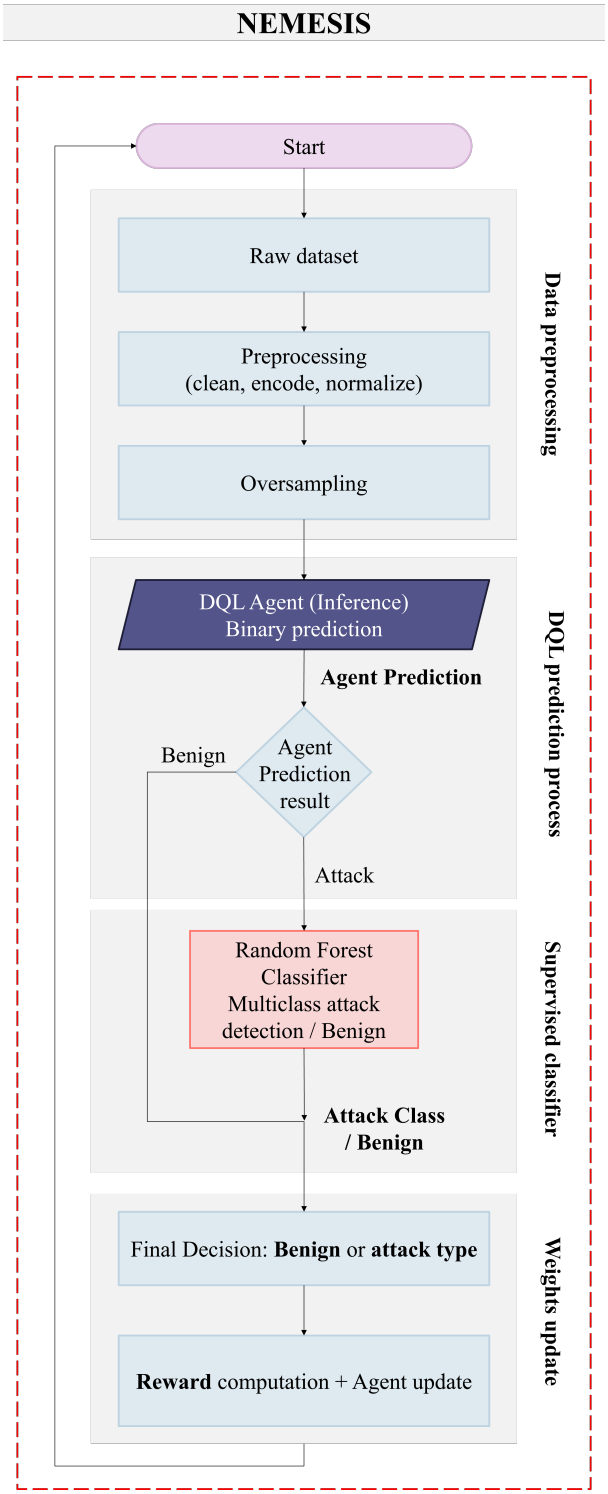


FIGURE 4.1 – Architecture globale du système NEMESIS illustrant l'interaction entre l'agent DQN, le module de prétraitement et le classificateur Random Forest.

4.4.2 Topologie du réseau de neurones (MLP)

Pour l'approximation de la fonction Q 2.1, nous avons opté pour un Perceptron Multicouche (MLP) plutôt que pour des réseaux convolutifs (CNN) ou récurrents (RNN), car les données d'entrée (vecteurs de caractéristiques du réseau) ne présentent pas de corrélation spatiale ou temporelle séquentielle stricte, excepté avec l'étiquette de l'échantillon de données, contrairement

au traitement d'images ou de texte.

L'architecture retenue est composée de trois couches denses (*fully connected*) interconnectées :

- **Couche d'entrée :** Le nombre de neurones correspond à la taille du vecteur d'état post-traitement, soit le nombre de caractéristiques pour chaque jeu de données . Elle reçoit l'état brut normalisé de l'environnement.
- **Couches cachées :**
  - La première couche cachée comporte **128 neurones**. Ce nombre élevé permet de capturer les interactions complexes et non-linéaires entre les différentes métriques réseau (ex : corrélation entre la durée du flux et le volume d'octets).
  - La seconde couche contient **64 neurones** et elle agit comme un goulot d'étranglement (*bottleneck*), forçant le réseau à compresser l'information et à extraire les motifs les plus discriminants.
  - La troisième couche, de **32 neurones**, affine la représentation avant la couche de sortie.
- **Couche de sortie :** Elle correspond à l'espace d'action binaire de l'agent. Elle contient 2 neurones, chacun représentant la Q-valeur estimée pour une action donnée : $Q(s, \text{Benign})$ et $Q(s, \text{Attack})$.

Concernant la fonction d'activation, nous avons systématiquement appliqué la fonction ReLU (*Rectified Linear Unit*) : $f(x) = \max(0, x)$. Contrairement aux fonctions sigmoïdes ou aux tangentes hyperboliques, ReLU permet d'accélérer la convergence de la descente de gradient stochastique en atténuant le problème de disparition du gradient (*vanishing gradient problem*) dans les réseaux profonds.

4.4.3 Justification des hyperparamètres critiques

Le paramétrage d'un agent DQN est une étape délicate qui influence directement la convergence de la politique optimale π^*. Les valeurs finales, présentées dans le Tableau 4.4, ont été fixées après plusieurs cycles d'expérimentation.

Stratégie d'exploration (ϵ-greedy)

L'un des défis majeurs en apprentissage par renforcement est le compromis exploration/exploitation.

- Nous avons implémenté une stratégie ϵ-greedy avec décroissance exponentielle (*exponential decay*).
- Initialement, ϵ est fixé à 1.0, ce qui signifie que l'agent agit de manière totalement aléatoire pour explorer l'environnement et découvrir la diversité des attaques.
- À chaque étape, ϵ est multiplié par un taux de décroissance ($decay\_rate$) jusqu'à atteindre un seuil minimal de 0.01. Cela garantit qu'en fin d'entraînement, l'agent exploite quasi-exclusivement sa politique apprise pour maximiser la récompense.

Le Facteur d'actualisation (γ)

Le paramètre γ (gamma) contrôle l'importance des récompenses futures. Nous avons fixé $\gamma = 0.3$, une valeur relativement basse par rapport aux standards de la littérature (souvent 0.99).

Justification : Dans un contexte de détection d'intrusions, la décision doit être instantanée. Un paquet malveillant doit être détecté immédiatement, indépendamment des paquets qui arriveront 10 étapes plus tard. Une valeur faible de γ force l'agent à être aveugle et à privilégier la récompense immédiate associée à la classification correcte du paquet actuel, ce qui correspond parfaitement aux exigences de réactivité d'un IDS.

Optimisation et apprentissage

L'algorithme d'optimisation Adam (*Adaptive Moment Estimation*) a été choisi pour sa capacité à ajuster le taux d'apprentissage de manière adaptative pour chaque paramètre.

— Taille de lot (Batch Size) : Fixée à 16. Une petite taille de lot introduit un bruit bénéfique dans l'estimation du gradient, aidant le modèle à s'échapper des minima locaux tout en réduisant l'empreinte mémoire sur le GPU.

— Itérations : L'entraînement s'étend sur 300 000 étapes pour CICIDS2017 et 400 000 pour NSL-KDD. Ces seuils ont été déterminés par *Early Stopping* : nous avons arrêté l'entraînement lorsque la courbe de récompense moyenne sur l'ensemble de validation a cessé de croître pendant 50 000 itérations consécutives, signe que l'agent avait convergé vers une politique stable.

TABLE 4.4 – Synthèse détaillée des hyperparamètres de l'agent DQN

Hyperparamètre	**Justification et Rôle**	**Valeur**
Time Steps	Nombre total d'épisodes d'interaction pour assurer la convergence sur l'ensemble du jeux de données.	300k / 400k
Update Frequency	Fréquence de mise à jour du réseau cible pour stabiliser l'apprentissage (Target Network).	4 itérations
Architecture MLP	Structure décroissante pour extraire et compresser les caractéristiques pertinentes.	[128, 64, 32]
Activation	ReLU pour la rapidité de calcul et la non-linéarité.	ReLU
Gamma (γ)	Priorité donnée à la détection immédiate (court terme).	0.3
Optimizer	Adam pour l'ajustement adaptatif des poids.	Adam
Learning Rate	Taux dynamique pour affiner la convergence finale.	$10^{-3} \rightarrow 10^{-5}$
Epsilon (ϵ)	Transition progressive de l'exploration pure vers l'exploitation.	$1.0 \rightarrow 0.01$
Batch Size	Compromis entre stochasticité du gradient et vitesse de calcul.	16

4.4.4 Formalisation des Algorithmes

L'implémentation repose sur deux algorithmes distincts assurant respectivement l'apprentissage hors-ligne (*offline training*) et la détection en temps réel. L'Algorithme 1 décrit la boucle d'interaction Agent-Environnement. On y observe le mécanisme de mémoire de rediffusion

(*Experience Replay*) qui permet de stocker les transitions (s, a, r, s') et de les réutiliser pour briser les corrélations temporelles entre échantillons consécutifs.

L'Algorithme 2 illustre la cascade décisionnelle : le DQN agit comme un filtre binaire haute performance. Cette architecture hybride permet de décharger le classificateur Random Forest (plus coûteux) des flux bénins massifs, réservant son analyse fine uniquement aux cas suspects.

Algorithme 1 : Entraînement de l'agent DQN

Input : jeux de données D, DQN params (episodes, γ, ϵ, batch)
Output : Politique relative aux agents formés

Normalize(D)
Initialize agent parameters
$batch\_size \leftarrow 16$
$States \leftarrow$ sample mini-batches
Initialize model
for $timestep \leftarrow 1$ **to** T **do**
  Reset environment state
  for $iter \leftarrow 1$ **to** num_iter **do**
    `// Step 1: Select action`
    for $i \leftarrow 1$ **to** $batch\_size$ **do**
      if $prob < \epsilon$ **then**
        $A_i \leftarrow$ random action
      else
        $Q_i \leftarrow$ `model`$(state_i)$
        $A_i \leftarrow \arg\max(Q_i)$
      end
    end
    $\epsilon \leftarrow \epsilon \times decay\_rate$
    `// Step 2: Rewards`
    for $i \leftarrow 1$ **to** $batch\_size$ **do**
      $R_i \leftarrow$ get_reward(A_i, label)
    end
    `// Step 3: Update`
    $Q' \leftarrow$ `model`$(next\_st)$
    for $i \leftarrow 1$ **to** $batch\_size$ **do**
      $Tgt[i] \leftarrow R_i + \gamma \max(Q'[i])$
    end
    `model.train`$(state, Tgt)$
    $state \leftarrow next\_state$
  end
end

Algorithme 2 : Interface Hybride : DQN + RF

Input : Ensemble de test, `DQN` entraîné, `RF`
Output : Prédictions & Mises à jour concernant les récompenses

for $i \leftarrow 1$ **to** n **do**
  $x \leftarrow$ `preprocess`(x_i)
  $s \leftarrow$ `extract_state`(x)
  $bin\_pred \leftarrow$ `DQN`(s)
    `// 0=benign, 1=attack`
  if $bin\_pred = 0$ **then**
    $final \leftarrow$ "benign"
  else
    $atk \leftarrow$ `RF`(s)
    $final \leftarrow atk$
  end
  $true\_lbl \leftarrow y_i$
  if $bin = 0$ ***and*** $lbl = benign$ **then**
    $r \leftarrow +2$ `// TN`
  else if $bin = 1$ ***and*** $lbl \neq benign$ **then**
    if $final = lbl$ **then**
      $r \leftarrow +5$ `// Correct`
    else
      $r \leftarrow -2$ `// Wrong type`
  else
    $r \leftarrow -5$ `// FP or FN`
  end
  `DQN.update`(s, r)
end

4.5 L'approche ULYSSE : Optimisation par Sélection Dynamique de Caractéristiques

4.5.1 Limites des NEMESIS et motivation de l'approche ULYSSE

Bien que l'architecture hybride NEMESIS, présentée dans la section précédente, démontre d'excellentes capacités de détection grâce à l'action conjointe du DQN et du classificateur Random Forest, elle se heurte à des défis structurels inhérents à l'analyse des flux réseaux modernes. En effet, le traitement exhaustif de l'ensemble des caractéristiques extraites du trafic soulève plusieurs limites critiques dans un contexte de sécurité en temps réel :

— **Le Fléau de la Dimensionnalité et le Coût Computationnel :** L'évaluation continue de vecteurs d'état (par exemple, 68 caractéristiques pour le jeu de données CICIDS2017) pour chaque flux réseau impose une charge de calcul et de mémoire considérable. Cette complexité dimensionnelle ralentit le temps d'inférence, ce qui s'avère pénalisant pour un IDS dont l'efficacité repose sur la réactivité.

— **Redondance et Bruit dans les Données :** L'intégralité des caractéristiques réseau n'est pas uniformément pertinente pour caractériser une menace. Certaines variables sont fortement corrélées entre elles (redondance d'information), tandis que d'autres n'apportent aucune valeur discriminante (bruit). Maintenir ces données superflues dans le processus d'inférence de NEMESIS risque de complexifier inutilement les frontières de décision et d'augmenter le taux de faux positifs.

— **Rigidité des Filtres Statiques :** Bien que des méthodes de sélection de caractéristiques existent dans la littérature (telles que l'Analyse en Composantes Principales ou l'élimination récursive), celles-ci sont généralement appliquées de manière statique lors de la phase de prétraitement. Elles figent l'espace de représentation et peinent à s'adapter à la nature évolutive et polymorphe des cyberattaques.

Pour pallier ces limites et optimiser les performances globales du système, nous introduisons l'approche **ULYSSE**. Conçu comme un module d'optimisation en amont, ULYSSE déploie un agent PPO dédié exclusivement à la *sélection dynamique de caractéristiques*. Contrairement aux approches statiques, l'agent ULYSSE interagit avec l'environnement de classification pour apprendre à extraire le sous-ensemble minimal d'attributs garantissant une précision maximale. Son objectif est de réduire drastiquement l'espace d'état traité par le classificateur final, allégeant ainsi le coût computationnel tout en isolant les signaux les plus pertinents pour la détection.

L'innovation majeure de l'approche ULYSSE, qui constitue le cœur de notre contribution scientifique, réside dans la conception originale de sa fonction de récompense. Dans les applications standards de l'apprentissage par renforcement, l'agent est souvent récompensé uniquement sur la base de la précision brute (*Accuracy*).

Cependant, notre approche formalise un problème d'optimisation multi-objectifs complexe. La fonction de récompense d'ULYSSE a été spécifiquement conçue pour forcer l'agent à trouver un compromis optimal entre deux forces antagonistes : **maximiser la capacité de détection** des intrusions tout en **minimisant agressivement la dimensionnalité** du vecteur de données.

4.5.2 Modélisation du Processus de Décision Markovien (MDP)

Pour formuler le problème de sélection de caractéristiques sous l'angle de l'apprentissage par renforcement, nous avons modélisé l'environnement d'interaction comme un Processus de Décision Markovien (MDP), défini classiquement par le tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \gamma \rangle$ [54].

Dans notre architecture, baptisée *DynamicGlobalShapEnv*, l'agent interagit avec un classificateur Random Forest pré-entraîné [25] (agissant comme « Juge ») et un profil d'importance SHAP global (agissant comme « Conseiller »).

Espace d'états ($\mathcal{S}$)

À chaque pas de temps t, l'environnement échantillonne aléatoirement une instance de trafic réseau issue du jeu de données d'entraînement. L'état s_t observé par l'agent correspond au vecteur de caractéristiques de cette instance spécifique, préalablement normalisé. Si N représente le nombre total de caractéristiques du jeu de données, l'espace d'états est défini comme un espace continu :

$$s_t = x_i \in [0, 1]^R$$

Où x_i est le i-ème échantillon du jeu de données de taille M ($i \sim \mathcal{U}(1, M)$).

Espace d'actions ($\mathcal{A}$)

Contrairement aux approches de sélection binaire dure (0 ou 1), l'espace d'action de notre agent a été modélisé de manière continue pour permettre une pondération fine (masquage souple). L'action a_t est un vecteur de poids $w \in \mathbb{R}^N$ borné entre 0 et 1 :

$$a_t = w = [w_1, w_2, \ldots, w_N] \quad \text{avec} \quad w_j \in [0, 1]$$

Ce vecteur agit comme un masque multiplicatif sur l'état courant. L'observation masquée $\tilde{s}_t$, qui sera soumise au classificateur Random Forest, est calculée par le produit d'Hadamard d'une manière coordonné :

$$\tilde{s}_t = s_t \odot w$$

Une caractéristique est considérée comme « active » (sélectionnée) si son poids généré par l'agent dépasse un seuil de tolérance fixé empiriquement à 0.1.

Fonction de récompense multi-objectifs ($\mathcal{R}$)

La fonction de récompense est la pierre angulaire de notre contribution. Elle évalue l'action de l'agent selon trois dimensions orthogonales : la justesse de la prédiction, la sparsité du vecteur et la fidélité à l'explicabilité globale. La récompense totale R_t est une combinaison linéaire de ces trois sous-récompenses :

1. **Récompense de précision (R_{acc})** : Le vecteur masqué $\tilde{s}_t$ est évalué par le modèle Random Forest. Si la prédiction $\hat{y}$ correspond à la véritable étiquette y, l'agent est récompensé ; sinon, il est pénalisé.
$$R_{acc} = \begin{cases} 1 & \text{si } \hat{y} = y \\ -1 & \text{sinon} \end{cases}$$
2. **Récompense de sparsité (R_{sparse})** : Pour lutter contre le fléau de la dimensionnalité, l'agent est fortement incité à désactiver un maximum de caractéristiques. La récompense est inversement proportionnelle au ratio de caractéristiques actives (où la fonction indicatrice $\mathbb{I}(w_j > 0.1)$ vaut 1 si le poids dépasse le seuil, et 0 sinon) :
$$R_{sparse} = 1 - \frac{1}{N} \sum_{j=1}^{N} \mathbb{I}(w_j > 0.1)$$

3. **Récompense de similarité SHAP (R_{shap})** : Pour éviter que l'agent ne supprime des variables critiques de manière purement exploratoire, nous guidons son apprentissage grâce à l'intelligence artificielle explicable (XAI), une technique s'apparentant au *Reward Shaping* [55]. Lors de l'initialisation, un profil d'importance globale $\Phi_{global} \in [0,1]^N$ est calculé via les valeurs de Shapley Additive exPlanations (SHAP) [56]. La pertinence de l'action de l'agent est évaluée en mesurant la similarité cosinus entre le vecteur de poids w généré par l'agent et le profil SHAP de référence :

$$R_{shap} = \cos(w, \Phi_{global}) = \frac{w \cdot \Phi_{global}}{\|w\|\|\Phi_{global}\| + \epsilon}$$

Où $\epsilon = 10^{-8}$ assure la stabilité numérique. Cette composante force l'agent à assigner des poids élevés aux caractéristiques statistiquement reconnues comme importantes par l'arbre de décision de SHAP.

Calcul de la récompense finale : Afin d'équilibrer ces trois objectifs, des coefficients de pondération (α) ont été calibrés suite à de multiples entraînements avec des valeurs variées des coefficients :

$$R_{total}(s_t, a_t) = \alpha_{acc} \cdot R_{acc} + \alpha_{sparse} \cdot R_{sparse} + \alpha_{shap} \cdot R_{shap} \tag{4.1}$$

Où :

— R_{acc} évalue la qualité de la prédiction (le verdict rendu par le juge Random Forest [25]).

— R_{sparse} quantifie le taux de compression (la diminution de la dimension du vecteur d'entrée).

— R_{shap} apprécie la cohérence avec les connaissances a priori (le guide fourni par SHAP).

4.5.3 Méthodologie d'ULYSSE

La plupart des travaux connexes qui appliquent l'apprentissage par renforcement à la sélection de caractéristiques reposent sur des architectures discrètes fondées sur les valeurs, en particulier le Deep Q-Network (DQN) ou le Q-Learning tabulaire [57, 58]. Dans ces approches classiques, le problème est formulé de manière binaire : pour chaque caractéristique, l'agent prend une décision binaire (1 pour conserver, 0 pour éliminer). Bien qu'efficace sur des jeux de données de faible dimension, cette stratégie présente des limites majeures que notre méthode cherche précisément à surmonter :

1. **Le mécanisme d'assignation du crédit** L'assignation du crédit est le processus par lequel un agent d'apprentissage par renforcement attribue une valeur de mérite à ses actions passées en fonction des résultats obtenus. Dans le cadre de ce mémoire, ce mécanisme permet de traduire la récompense globale (performance du système de détection d'intrusions) en mises à jour locales de la politique. Ce processus repose sur l'utilisation de fonctions de valeur qui estiment l'apport de chaque décision individuelle (ex : la sélection d'une caractéristique réseau spécifique) à l'atteinte de l'objectif final. Une assignation de crédit efficace est ce qui permet à l'agent de distinguer, au sein d'une trajectoire complexe, les actions déterminantes des décisions superflues.
2. **Explosion combinatoire de l'espace d'actions :** Pour un vecteur d'état de dimension N, représenter la sélection de caractéristiques comme une action globale binaire conduit à un espace d'actions de taille 2^N. Dans le cas du jeu de données CICIDS2017 (N =

68 caractéristiques après prétraitement), cela se traduit par un espace de décision de 2^{68} configurations possibles, ce qui est pratiquement inexplorable d'un point de vue computationnel. Lorsque le problème est formulé de manière séquentielle (l'agent évalue une caractéristique à chaque étape, comme proposé dans [59]), la longueur des épisodes croît fortement avec N, ce qui complique l'apprentissage. Les trajectoires deviennent alors extrêmement longues, rendant le mécanisme d'assignation du crédit (*credit assignment*) peu efficace et ralentissant considérablement la convergence de la politique.

3. **Instabilité du masquage Dur (*Hard Masking*) :** Le schéma de décision binaire impose un masquage dur des caractéristiques. Passer brutalement d'un état « sélectionnée » (1) à « exclue » (0) implique une modification discontinue de la représentation fournie au classificateur. Cette rupture brutale dans l'espace des entrées engendre une forte variabilité dans le signal de récompense observé par l'agent. Une telle variance accentue le biais de surestimation (*Overestimation Bias*) déjà présent dans l'algorithme DQN lorsque l'environnement est complexe et stochastique, comme l'ont montré Van Hasselt et al. [60]. En conséquence, la politique apprise peut devenir instable et avoir tendance à surestimer systématiquement la valeur de certaines actions.
4. **Diminution du niveau de détail de l'information :** En imposant un choix binaire, on suppose implicitement qu'une caractéristique est soit indispensable, soit totalement négligeable. Or, dans le contexte des données réseau, la pertinence d'une variable est généralement graduelle et fortement dépendante du contexte (type de trafic, nature de l'attaque, charge du réseau, etc.). Ce manque de nuance conduit à ignorer les contributions partielles ou conditionnelles de certaines caractéristiques et limite la capacité du modèle à exploiter finement l'information disponible.

Face à ces limitations structurelles, l'approche ULYSSE redéfinit la sélection de caractéristiques non plus comme un choix binaire, mais comme un problème de **pondération continue** (masquage souple ou *Soft Masking*).

Pour traiter ce problème formulé dans un espace d'actions continu, le recours au DQN s'avère mathématiquement inapproprié. Nous avons donc choisi d'utiliser l'algorithme PPO [15]. À la différence du DQN, PPO appartient à la famille des méthodes de gradient de politique (*Policy Gradient*), ce qui lui permet de manipuler directement des espaces d'actions continus et de grande dimension ($w \in [0, 1]^N$), sans recourir à une discrétisation artificielle. L'agent peut ainsi moduler avec finesse le poids associé à chaque caractéristique, de manière progressive et flexible, plutôt que par des sauts brutaux. Par ailleurs, le mécanisme d'écrêtage (*clipping*) propre à PPO limite l'ampleur des mises à jour de la politique entre deux itérations, évitant ainsi les changements trop brusques susceptibles de dégrader les performances. Cette contrainte sur la mise à jour confère à l'apprentissage une robustesse nettement supérieure à celle des approches fondées sur la valeur, en particulier dans des environnements fortement bruités et variables, comme c'est le cas pour le trafic réseau, où la stabilité de la politique est un enjeu central.

4.5.4 Architecture et paramétrage de l'agent ULYSSE

Ayant établi la nécessité d'une approche d'optimisation continue via l'algorithme PPO, cette section détaille la topologie du réseau de neurones sous-jacent et la configuration des hyperparamètres régissant l'apprentissage de l'agent ULYSSE.

Topologie Acteur-Critique (Actor-Critic)

L'algorithme PPO repose sur une architecture bicéphale dite *Actor-Critic*. Contrairement au DQN, qui n'utilise qu'un seul réseau pour estimer la valeur des actions, notre implémentation (reposant sur une politique de type *MlpPolicy*) sépare le processus décisionnel en deux réseaux de neurones profonds distincts qui opèrent conjointement :

— **Le réseau acteur (π)** : C'est le cœur décisionnel de l'agent. Il prend en entrée le vecteur de caractéristiques du flux réseau (l'état s_t) et génère en sortie le vecteur de pondération continu $w \in [0,1]^N$. Son architecture est composée de trois couches denses (MLP) de dimensions décroissantes : **256, 256 et 128 neurones**.

— **Le réseau critique (V)** : Il agit comme un évaluateur. Il observe le même état s_t mais prédit la fonction de valeur $V(s_t)$, c'est-à-dire le rendement espéré depuis cet état. Cette estimation permet de calculer l'avantage (*Advantage function*), un signal crucial pour réduire la variance lors de la mise à jour des poids de l'Acteur. Le Critique possède une architecture miroir, composée également de trois couches denses de **256, 256 et 128 neurones**.

Cette séparation stricte (sans partage de poids entre les premières couches) empêche les interférences entre l'apprentissage de la politique de sélection et l'estimation de la valeur, garantissant une meilleure stabilité.

Configuration des hyperparamètres

La robustesse de PPO repose sur son mécanisme d'écrêtage (*clipping*) de la fonction d'objectif, empêchant des mises à jour de politique trop destructrices d'une itération à l'autre. Les hyperparamètres retenus pour l'entraînement d'ULYSSE, calibrés empiriquement pour maximiser le compromis entre exploration et exploitation, sont synthétisés dans le Tableau 4.5.

TABLE 4.5 – Configuration des hyperparamètres de l'agent PPO (Module ULYSSE)

Paramètre	**Rôle et Justification**	**Valeur**
Policy	Type de politique (Perceptron Multicouche)	`MlpPolicy`
Architecture (π, V)	Dimensions des couches cachées (Acteur et Critique)	[256, 256, 128]
Learning Rate	Pas d'apprentissage initial de l'optimiseur Adam	3×10^{-4}
Horizon (n_steps)	Nombre d'interactions collectées avant la mise à jour	1024
Batch Size	Taille des mini-lots pour la descente de gradient	256
Epochs (n_epochs)	Nombre de passages d'optimisation sur le buffer	10
Clip Range	Rayon de confiance limitant l'amplitude de la mise à jour	0.2
Entropy Coef	Coefficient encourageant la diversité du masquage	0.001
Gamma (γ)	Facteur d'actualisation des récompenses futures	0.99

Deux choix spécifiques de cette configuration méritent d'être soulignés :

1. **L'efficacité d'échantillonnage (*Sample Efficiency*) :** L'agent interagit avec l'environnement pendant un horizon de 1024 pas (*n_steps*) avant de stocker ces expériences. Ce tampon est ensuite divisé en mini-lots de 256 (*batch_size*) pour entraîner les réseaux sur 10 époques (*n_epochs*). Ce ratio permet une exploitation maximale des données collectées tout en lissant le gradient.
2. **Le contrôle de l'entropie (*ent_coef*) :** Bien que paramétré à une valeur faible (0.001), ce coefficient est indispensable pour injecter une stochasticité contrôlée dans les actions de l'Acteur. Il prévient la convergence prématurée de l'agent vers un sous-ensemble de caractéristiques sous-optimal (minimum local).

Initialisation et génération du profil SHAP global

Afin de guider l'exploration de l'agent et d'éviter la suppression accidentelle de caractéristiques critiques, nous construisons un vecteur de référence, appelé *Profil SHAP Global* (Φ_{global}), en amont de la phase d'apprentissage par renforcement. Ce processus, exécuté lors de l'initialisation de l'environnement, se décompose en quatre étapes algorithmiques rigoureuses :

1. **Échantillonnage Représentatif :** Le calcul des valeurs de Shapley exactes sur l'intégralité du jeu de données d'entraînement (contenant des millions de flux) étant computationnellement prohibitif, nous sélectionnons un sous-ensemble représentatif $X_{sub} \subset X_{train}$ de taille $K = 1000$ échantillons via un tirage aléatoire uniforme sans remise.
2. **Calcul des Contributions Locales (TreeExplainer) :** Nous utilisons l'algorithme *TreeExplainer* [56], spécialement optimisé pour les ensembles d'arbres comme le Random Forest. Pour chaque échantillon $x^{(k)} \in X_{sub}$ et chaque caractéristique j, l'explainer calcule la valeur de Shapley $\phi_j(x^{(k)})$, qui représente la contribution marginale de la caractéristique j à la prédiction du modèle pour cet échantillon spécifique.
3. **Agrégation globale et gestion multi-classes :** Les valeurs SHAP locales peuvent être positives (contribuant à la classe attaque) ou négatives (contribuant à la classe bénigne). Pour obtenir une mesure d'importance globale, nous devons considérer la magnitude de l'impact. De plus, dans un contexte multi-classes, une caractéristique peut avoir des impacts différents pour chaque classe c. Nous calculons d'abord la moyenne des valeurs absolues sur l'ensemble des classes, puis nous faisons la moyenne sur l'ensemble des échantillons du sous-ensemble :

$$\psi_j = \frac{1}{K} \sum_{k=1}^{K} \left(\sum_{c \in Classes} |\phi_{j,c}(x^{(k)})| \right) \tag{4.2}$$

   Où ψ_j représente l'importance brute moyenne de la caractéristique j.
4. **Normalisation Min-Max :** L'agent PPO génère des actions (poids) bornées dans l'intervalle $[0, 1]$. Pour que la comparaison via la similarité cosinus soit cohérente, le vecteur d'importance SHAP doit être mis à la même échelle. Nous appliquons donc une normalisation Min-Max pour obtenir le profil final Φ_{global} :

$$\Phi_{global,j} = \frac{\psi_j - \min(\psi)}{\max(\psi) - \min(\psi) + \epsilon} \tag{4.3}$$

Le vecteur résultant $\Phi_{global} \in [0, 1]^N$ agit comme un biais informatif : il indique à l'agent quelles caractéristiques sont statistiquement les plus discriminantes selon le modèle supervisé, accélérant ainsi la convergence de la politique de sélection vers des solutions robustes. L'algorithme d'entrainnement est donc détaillé dans le pseudo-code 3

Algorithme 3 : Processus de sélection de caractéristiques dynamique (ULYSSE)

Entrées : $\mathcal{D}$: Jeu de données d'entraînement (Flux réseaux)
RF : classificateur Random Forest pré-entraîné (le Juge)
K : Nombre d'échantillons pour l'estimation SHAP
T : Horizon d'interaction par épisode
Sortie : π_θ : Politique de l'agent PPO optimisée

`// Phase 1 : Initialisation du conseiller (Profil SHAP)`
$X_{sub} \leftarrow$ Échantillonner K instances aléatoires de $\mathcal{D}$
$\Phi_{raw} \leftarrow$ `TreeExplainer`(RF, X_{sub}) `// Calcul des contributions`
$\psi \leftarrow \frac{1}{K} \sum_k |\Phi_{raw}^{(k)}|$ `// Moyenne des valeurs absolues`
$\Phi_{global} \leftarrow$ `MinMaxNormalize`(ψ) `// Profil de référence` $\in [0,1]^N$
`// Phase 2 : Boucle d'apprentissage PPO`
Initialiser les paramètres de l'Acteur θ et du Critique ϕ
for $episode \leftarrow 1$ **to** $Max\_Episodes$ **do**
  $s_t \leftarrow$ Observer l'état initial (Flux aléatoire)
  for $t \leftarrow 1$ **to** T **do**
    `// A. Action et Masquage Souple`
    $w_t \sim \pi_\theta(s_t)$ `// Poids continus` $w_t \in [0,1]^N$
    $\tilde{s}_t \leftarrow s_t \odot w_t$ `// Produit de Hadamard`
    `// B. Évaluation par le Juge`
    $\hat{y} \leftarrow RF$.`Predict`$(\tilde{s}_t)$
    $R_{acc} \leftarrow (1$ si $\hat{y} = y_{true}$ sinon $-1)$
    `// C. Calcul de la Récompense Multi-Objectifs`
    $R_{sparse} \leftarrow 1 - \frac{1}{N} \sum_{j=1}^{N} \mathbb{I}(w_{t,j} > 0.1)$
    $R_{shap} \leftarrow \text{Cosinus}(w_t, \Phi_{global})$
    $r_{total} \leftarrow \alpha_{acc} R_{acc} + \alpha_{sparse} R_{sparse} + \alpha_{shap} R_{shap}$
    Stocker la transition $(s_t, w_t, r_{total}, s_{t+1})$
    $s_t \leftarrow s_{t+1}$
  end
  `// D. Mise à jour de la Politique`
  Calculer les avantages généralisés
  Optimiser θ et ϕ via la perte PPO-Clip (Surrogate Loss)
end
return π_θ

4.5.5 Mécanisme de masquage souple et filtrage dynamique

L'innovation centrale de l'interface entre l'agent PPO et le classificateur réside dans l'abandon de la sélection binaire stricte au profit d'un mécanisme de **masquage souple** (*Soft Masking*). Cette opération, mathématiquement formalisée par le produit de Hadamard, s'inspire des mécanismes d'attention neuronale [61] et permet une transition fluide de l'espace des caractéristiques, essentielle pour la stabilité du gradient de politique.

Formalisation du produit de hadamard

Soit $x = (x_1, ..., x_n) \in \mathbb{R}^N$ le vecteur de caractéristiques brut normalisé d'un flux réseau, et $w \in [0, 1]^N$ le vecteur d'action généré par l'agent ULYSSE à l'instant t. L'opération de masquage ne supprime pas physiquement les colonnes de la matrice de données, mais applique une pondération multiplicative élément par élément, similaire aux portes de régulation (*gates*) utilisées dans les réseaux Squeeze-and-Excitation [62] :

$$\tilde{x} = x \odot w = \begin{bmatrix} x_1 \cdot w_1 \\ x_2 \cdot w_2 \\ \vdots \\ x_N \cdot w_N \end{bmatrix}$$

Où $\tilde{x}$ représente le vecteur d'état filtré soumis au Random Forest. Cette approche agit comme un **mécanisme d'attention visuelle** transposé aux données tabulaires [63] :

— Si $w_i \approx 1$, la caractéristique x_i est transmise intégralement au classificateur (signal préservé).

— Si $w_i \approx 0$, la valeur de x_i est écrasée vers 0, neutralisant son impact sur les nœuds de décision des arbres du Random Forest (bruit atténué).

— Les valeurs intermédiaires ($0 < w_i < 1$) permettent à l'agent d'indiquer une incertitude ou une importance pondérée, offrant une granularité que la sélection binaire (0 ou 1) ne permet pas.

Dualité : filtrage continu vs sparsité discrète

Il existe une distinction importante dans notre implémentation entre la donnée vue par le classificateur et l'évaluation de la sparsité pour la récompense :

1. **Pour la classification (Vue Continue) :** Le Random Forest reçoit le vecteur $\tilde{x}$ continu. Cela permet de conserver les nuances subtiles des données lors de l'inférence, évitant la perte d'information brutale typique des méthodes de sélection "Wrapper" classiques [64].
2. **Pour la récompense (vue discrète) :** Afin de forcer l'agent à converger vers une véritable réduction de dimensionnalité, nous appliquons un seuil de coupure strict ($\tau = 0.1$) pour le calcul de la pénalité de sparsité R_{sparse}. Une caractéristique j est considérée comme "Active" uniquement si $w_j > \tau$.

Cette dualité permet de bénéficier du meilleur des deux mondes : la stabilité d'apprentissage des espaces continus (pour PPO) et la capacité de décision tranchée requise pour la sélection finale de caractéristiques.

La Figure 4.2 synthétise l'architecture fonctionnelle proposée pour le module ULYSSE. Ce diagramme illustre le cycle d'apprentissage où l'agent PPO, agissant comme un filtre dynamique, module l'importance des caractéristiques via un mécanisme de masquage souple. Le schéma met en évidence l'interaction tripartite entre l'agent, le classificateur Random Forest (le Juge) évaluant la performance, et le profil SHAP global (le Conseiller) guidant l'explicabilité, dont la combinaison forme le signal de récompense multi-objectifs nécessaire à l'optimisation de la politique.

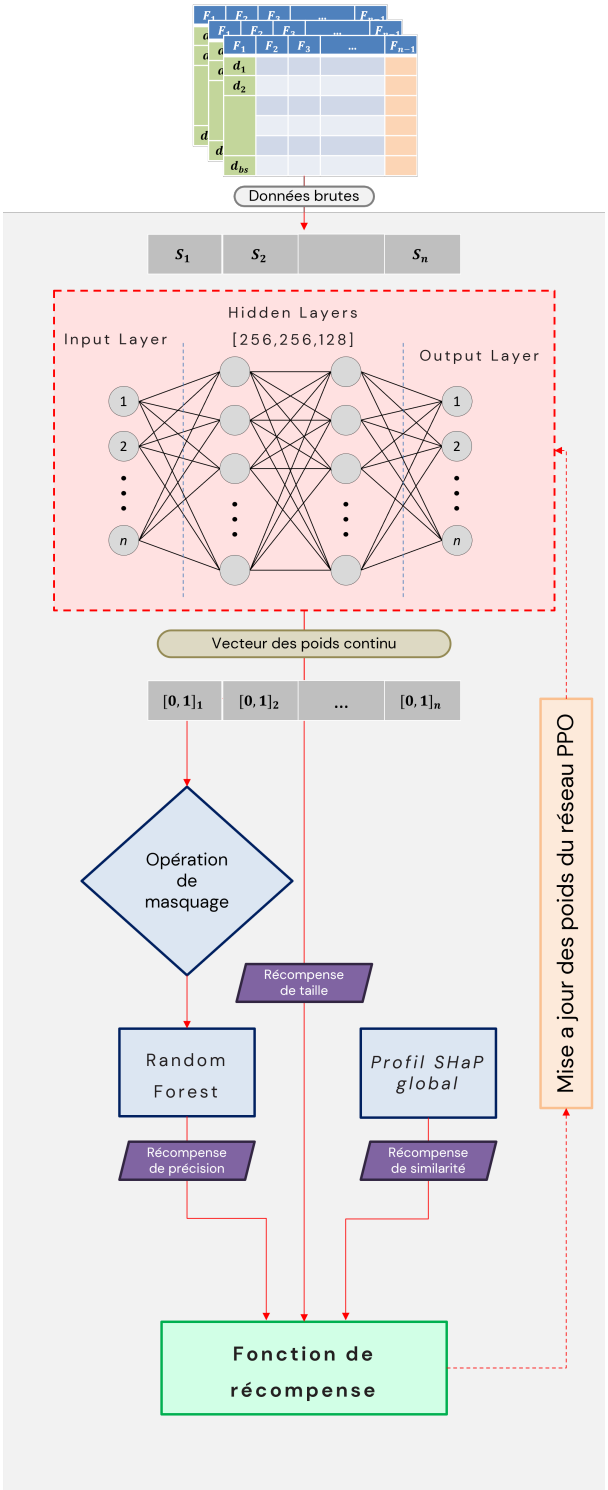


FIGURE 4.2 – Architecture de l'approche de sélection des caractéristiques

4.6 Le Module de détection d'intrusion

Une fois le flux réseau épuré par le masquage dynamique d'ULYSSE, les données pondérées sont transmises au second agent du système : NEMESIS. Contrairement à ULYSSE, qui agit comme un pré-processeur, NEMESIS est un classificateur séquentiel basé sur l'algorithme **DQN** (*Deep Q-Network*) [58]. Son objectif est de associer l'état filtré à une décision binaire : trafic

bénin ou malveillant.

4.6.1 Modélisation de l'environnement de détection

L'environnement, implémenté sous le nom `DQN_DetectionEnv`, simule un analyste de sécurité traitant les paquets réseau un par un. Il est défini par le tuple suivant :

Espace d'états et d'actions

— **État (s_t) :** L'état correspond au vecteur de caractéristiques transformé par les poids optimaux générés par ULYSSE (w_{ppo}). Si x_t est le flux brut à l'instant t, l'entrée de NEMESIS est :

$$s_t = x_t \odot w_{ppo} \tag{4.4}$$

L'agent ne voit donc que les informations jugées pertinentes par le module de sélection, ce qui réduit le bruit et la complexité de la tâche.

— **Action (a_t) :** L'espace d'actions est discret et binaire $\mathcal{A} = \{0, 1\}$, où :

— $a_t = 0$: Le flux est classé comme **Normal**.

— $a_t = 1$: Le flux est classé comme **Attaque**.

La stratégie de récompense asymétrique

La contribution majeure de ce module réside dans sa fonction de récompense $\mathcal{R}(s_t, a_t)$. Dans la détection d'intrusion, les erreurs de classification n'ont pas toutes la même gravité : manquer une intrusion (Faux Négatif) est critique, tandis qu'une fausse alerte (Faux Positif) est une nuisance gérable. Pour intégrer cette contrainte métier, nous avons conçu une matrice de récompenses asymétrique telle que décrite dans le Tableau 4.6, qui pénalise sévèrement les faux négatifs.

TABLE 4.6 – Matrice de Récompenses de l'Agent NEMESIS

Vraie Étiquette	Action de l'Agent	Récompense	Justification Métier
Normal (0)	**0 (Vrai Négatif)**	**+2**	Valorisation de la stabilité sur le trafic sain.
	1 (Faux Positif)	**-3**	Pénalité modérée (bruit pour l'analyste).
Attaque (1)	0 (Faux Négatif)	**-15**	**Pénalité Critique.** L'intrusion passe inaperçue.
	1 (Vrai Positif)	**+5**	Encouragement fort pour la détection d'attaque.

Cette distribution de valeurs force l'agent DQN à adopter un comportement de "paranoïa contrôlée" : il apprend à prioriser le rappel (*Recall*) pour éviter la pénalité de -15, tout en maintenant une précision suffisante pour maximiser l'accumulation des +2 sur le trafic normal majoritaire.

Architecture DQN

L'agent utilise un réseau de neurones profonds pour approximer la fonction $Q(s, a)$. L'architecture comprend des couches denses connectées à l'espace d'observation de dimension réduite (features actives), suivies d'une couche de sortie à 2 neurones (Q-values pour les actions 0

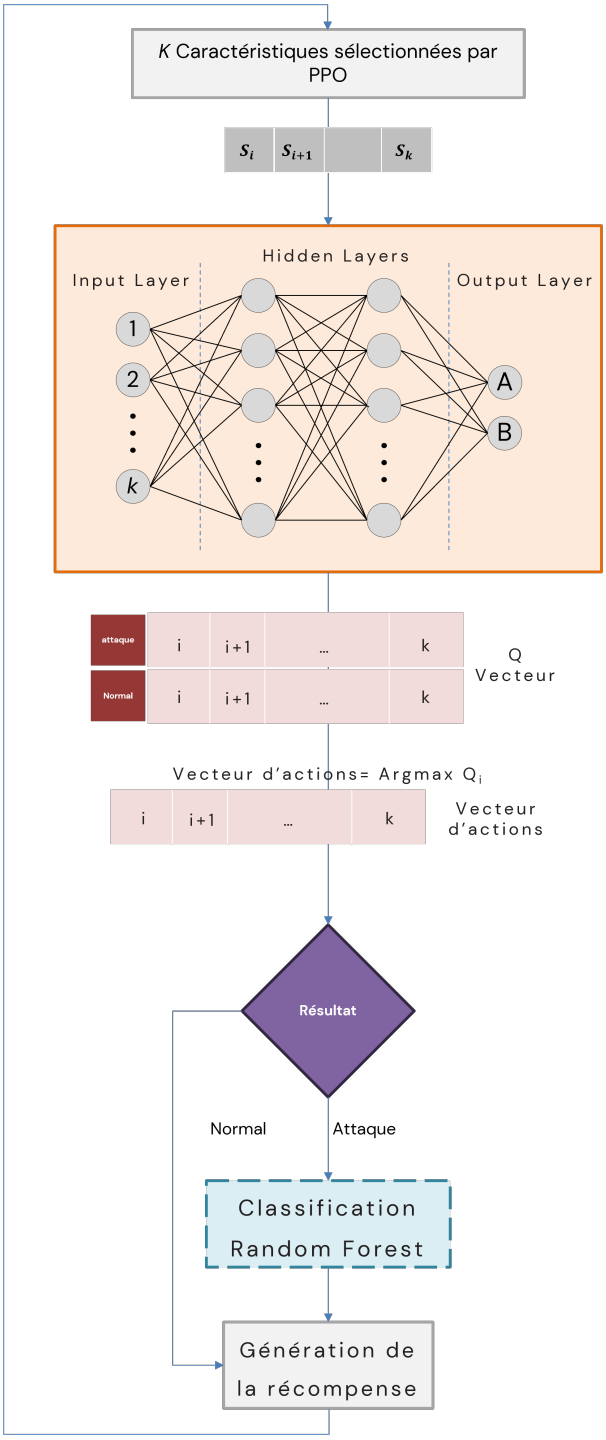


FIGURE 4.3 – Architecture du modèle DQN

et 1). L'apprentissage est stabilisé par l'utilisation d'un *Replay Buffer* et d'un *Target Network*, conformément aux standards de l'algorithme DQN. L'architecture du module de détection de l'approche ULYSSE est globalement identique à celle proposée dans notre première méthode, NEMESIS. La figure 4.3 reprend d'ailleurs une structure très proche de l'architecture précédemment présentée pour NEMESIS, tant au niveau des composants que de leur organisation. Il en va de même pour la procédure d'apprentissage, dont le déroulement illustré dans l'algorithme 4 suit les mêmes étapes et principes que le processus d'entraînement décrit pour NEMESIS.

Algorithme 4 : Entraînement du Détecteur d'Intrusion (Agent NEMESIS - DQN)

Entrées : $\mathcal{D}_{masked}$: Flux réseaux pondérés par ULYSSE ($\tilde{s} = x \odot w_{ppo}$)
$\mathcal{A}$: Espace d'actions binaire {0 : Normal, 1 : Attaque}
$N_{episodes}$: Nombre d'épisodes d'entraînement
ϵ, γ, α : Paramètres (Exploration, Actualisation, Learning Rate)
Sortie : Q_ϕ : Réseau de neurones Profond (Q-Network) optimisé

`// Initialisation`
Initialiser le Q-Network $Q(s, a|\phi)$ et le Target Network $\hat{Q}(s, a|\phi')$
Initialiser le Buffer d'expérience $\mathcal{B} \leftarrow \emptyset$
for $episode \leftarrow 1$ **to** $N_{episodes}$ **do**
  Observer l'état initial pondéré $\tilde{s}_1$ (Sortie d'ULYSSE)
  for $t \leftarrow 1$ **to** T_{max} **do**
    `// 1. Sélection de l'Action (ϵ-Greedy)`
    Générer un nombre aléatoire $u \sim \mathcal{U}(0, 1)$
    if $u < \epsilon$ **then**
      $a_t \leftarrow$ Choix aléatoire dans $\{0, 1\}$ `// Exploration`
    else
      $a_t \leftarrow \arg\max_a Q(\tilde{s}_t, a|\phi)$ `// Exploitation`
    end
    `// 2. Interaction et Récompense Asymétrique`
    Exécuter a_t, observer le label réel y_{true}
    if $y_{true} = Normal$ **then**
      $r_t \leftarrow (a_t = 0 ? +\mathbf{2} : -\mathbf{3})$ `// Priorité Stabilité`
    else `// Attaque`
      $r_t \leftarrow (a_t = 1 ? +\mathbf{5} : -\mathbf{15})$ `// Pénalité Critique`
    end
    Stocker la transition $(\tilde{s}_t, a_t, r_t, \tilde{s}_{t+1})$ dans $\mathcal{B}$
    `// 3. Apprentissage (Experience Replay)`
    Échantillonner un mini-lot $(\tilde{s}_j, a_j, r_j, \tilde{s}_{j+1})$ de $\mathcal{B}$
    Calculer la cible y_j :

$$y_j = r_j + \gamma \max_{a'} \hat{Q}(\tilde{s}_{j+1}, a'|\phi')$$

    Mettre à jour ϕ en minimisant la perte (Huber/MSE) :

$$\mathcal{L}(\phi) = \frac{1}{B} \sum (y_j - Q(\tilde{s}_j, a_j|\phi))^2$$

    Mettre à jour l'état $\tilde{s}_t \leftarrow \tilde{s}_{t+1}$
  end
  `// Mise à jour périodique du Target Network`
  Tous les C pas, mettre à jour $\phi' \leftarrow \phi$
  Réduire ϵ (Decay)
end
return Q_ϕ

Configuration et hyperparamètres de l'agent de détection

L'agent de détection repose sur une architecture de réseau de neurones profonds de type *Perceptron Multicouche*, configurée pour traiter les vecteurs de caractéristiques pondérés provenant du module de sélection. La topologie du réseau adopte une structure pyramidale composée de deux couches cachées de **128 et 64 neurones** avec des fonctions d'activation **ReLU**. Cette architecture permet une extraction progressive des motifs non-linéaires complexes associés aux signatures d'attaques.

Le choix des hyperparamètres d'apprentissage, détaillé dans le Tableau 4.7, reflète la nature spécifique de la détection d'intrusion par rapport aux tâches classiques de renforcement :

— **Facteur d'actualisation ($\gamma = 0.4$) :** Contrairement aux jeux vidéo où la planification à long terme est cruciale ($\gamma \approx 0.99$), la détection d'intrusion est une tâche réactive. Une valeur faible de 0.4 force l'agent à privilégier la récompense immédiate (la classification correcte du paquet actuel) plutôt que d'hypothétiques gains futurs, ce qui accélère la convergence.

— **Exploration (ϵ-greedy) :** Avec une fraction d'exploration de 40% et un ϵ_{final} de 0.05, nous maintenons une capacité de découverte élevée sur une grande partie de l'entraînement, indispensable pour que l'agent rencontre suffisamment d'exemples des classes d'attaques minoritaires (déséquilibre de classes).

TABLE 4.7 – Hyperparamètres de l'algorithme DQN (Module NEMESIS)

Paramètre	**Description / Rôle**	**Valeur**
Architecture	Dimensions des couches cachées (MLP)	[128, 64]
Activation	Fonction d'activation non-linéaire	ReLU
Optimizer	Algorithme d'optimisation des poids	Adam
Learning Rate	Pas d'apprentissage initial (α)	1×10^{-3}
Buffer Size	Taille de la mémoire de réexpérience (*Replay Buffer*)	100, 000 transitions
Batch Size	Nombre d'échantillons par mise à jour de gradient	32
Gamma (γ)	Facteur d'actualisation (Horizon de décision)	0.4
Exploration Fraction	Pourcentage de l'entraînement avec ϵ décroissant	40%
Learning Starts	Pas d'attente avant le début de l'optimisation	1, 000 pas
Target Update	Fréquence de mise à jour du réseau cible	Tous les 1000 pas

4.6.2 Fonction de récompense multiobjective et pilotage de l'agent

Le cœur du mécanisme d'apprentissage d'ULYSSE repose sur la définition de sa fonction de récompense $\mathcal{R}$. À la différence des cadres classiques d'apprentissage par renforcement, généralement centrés sur l'optimisation d'un objectif unique [54], notre contexte est intrinsèquement

multiobjectifs. Plus précisément, nous cherchons, de manière conjointe et équilibrée, à maximiser la performance de détection, à réduire le nombre de caractéristiques exploitées, tout en assurant une forte cohérence avec les connaissances métier mises en évidence par SHAP [56].

La récompense globale R_{total} à l'instant t est ainsi formulée comme une combinaison linéaire pondérée de trois composantes distinctes, chacune correspondant à l'un de ces objectifs définis dans la fonction 5.1

Orientation de l'apprentissage de l'agent

Les coefficients α agissent comme des hyperparamètres de contrôle permettant d'orienter la stratégie de convergence de l'agent. Cette approche s'apparente au *Reward Shaping* [55], permettant de guider l'agent vers des comportements souhaités. En modulant ces valeurs, nous pouvons forcer l'agent à adopter différents comportements :

— **Priorité à la sécurité** ($\alpha_{acc} \gg \alpha_{sparse}$) : L'agent conservera un grand nombre de caractéristiques pour maximiser la détection.

— **Priorité au mimétisme** ($\alpha_{shap} \gg \alpha_{others}$) : L'agent se contentera d'imiter le classement SHAP statique.

— **Priorité à la parcimonie** ($\alpha_{sparse} \gg \alpha_{acc}$) : L'agent tentera de résoudre le problème avec un nombre minimal de variables, comme suggéré dans les approches de sélection de caractéristiques par renforcement [57].

Configuration adoptée

Dans le cadre de cette étude, nous avons paramétré les coefficients de manière à favoriser l'obtention d'une solution à la fois autonome et minimaliste.

Concrètement, nous avons attribué une pondération élevée au terme de sparsité ($\alpha_{sparse} = 1.0$) et une pondération intermédiaire au terme d'explicabilité basé sur SHAP ($\alpha_{shap} = 0.7$). Ce réglage n'est pas arbitraire, il répond à une intention précise :

1. **Priorité à la sparsité :** nous cherchons à inciter l'agent à supprimer de manière énergique les caractéristiques superflues afin de réduire autant que possible le coût de calcul du détecteur final.
2. **Renforcement de l'autonomie décisionnelle :** en restreignant le poids accordé à SHAP ($\alpha_{shap} < 1.0$), nous donnons à l'agent ULYSSE la possibilité de s'« affranchir » partiellement des recommandations du conseiller initial. Ainsi, si au cours de son exploration, l'agent identifie une combinaison de caractéristiques peu ou pas valorisée par SHAP mais offrant une meilleure récompense globale, il est libre de la retenir.

Avec cette configuration, l'agent n'est donc pas cantonné à la simple imitation des solutions préexistantes : il est explicitement encouragé à explorer l'espace des caractéristiques pour faire émerger des sous-ensembles inédits, plus parcimonieux et potentiellement plus performants.

4.7 Conclusion

Ce chapitre a présenté en détail la méthodologie de conception d'un système de détection d'intrusion adaptatif, structuré autour de deux agents aux rôles complémentaires. Dans un premier temps, nous avons formalisé le problème de sélection de caractéristiques sous la forme d'un Processus de Décision Markovien [21]. Ceci est résolu par un agent PPO, nommé ULYSSE,

dont le comportement est piloté par une fonction de récompense à objectifs multiples, permettant d'équilibrer différentes contraintes, telles que la performance de détection, la complexité du modèle et les ressources de calcul mobilisées.

Dans un second temps, nous avons défini l'agent de détection, NEMESIS, implémenté comme un classificateur basé sur DQN. Ce dernier intègre explicitement une matrice de coûts asymétrique, conçue pour refléter le caractère critique des erreurs en détection d'intrusion, où les faux négatifs (intrusions non détectées) peuvent avoir des conséquences bien plus graves que les faux positifs. Cette approche s'inscrit dans la lignée des méthodes sensibles aux coûts [14]. Elle adapte l'algorithme d'apprentissage de manière à privilégier les décisions les plus sûres dans un contexte opérationnel.

Après avoir établi cette architecture conceptuelle et décrit les mécanismes de décision des deux agents, le chapitre suivant sera dédié à l'évaluation expérimentale de ces modèles. Nous procéderons à une validation empirique approfondie sur les jeux de données CIC-IDS2017 et NSL-KDD [8], en analysant leurs performances, leurs limites et leur capacité à généraliser à différents types d'attaques et de conditions réseau.

Chapitre 5

Résultats et évaluation

5.1 Résultats d'évaluation de NEMESIS

5.2 Introduction

Ce chapitre est consacré à la validation empirique de l'architecture proposée. Nous y examinons séparément et conjointement les performances du module de sélection (ULYSSE) et du module de détection (NEMESIS). Il s'agit de montrer que l'intégration de mécanismes d'apprentissage par renforcement profond ne se limite pas à atteindre des taux de détection comparables, voire supérieurs, à ceux des approches de pointe, mais permet également d'améliorer l'explicabilité du processus de décision et de réduire de façon marquée la dimension des données manipulées.

Les expériences ont été réalisées sur les jeux de données de référence CIC-IDS2017 et NSL-KDD, retenus en raison de leur capacité à refléter de manière fidèle et détaillée des menaces contemporaines variées. Dans un premier temps, nous décrirons l'environnement technique et les conditions expérimentales mises en place, avant de présenter et d'analyser en détail les différentes métriques de performance obtenues pour chacun des agents étudiés.

5.3 Ressources matérielles et Environnement logiciel

Afin d'évaluer la flexibilité et les besoins computationnels de notre approche, l'entraînement des modèles a été distribué sur deux architectures matérielles distinctes. Cette dichotomie permet de tester le comportement des agents dans des environnements aux ressources variées (Serveur de calcul vs Station de travail standard).

5.3.1 Spécifications matérielles

Le tableau 5.1 détaille les configurations utilisées pour l'entraînement respectif des agents NEMESIS et ULYSSE. Il est important de noter que le système NEMESIS, nécessitant un échantillonnage intensif et une mémoire d'expérience (*Replay Buffer*) volumineuse, a bénéficié de l'accélération massive du GPU A100. À l'inverse, le système ULYSSE a démontré sa capacité à converger sur une configuration grand public (Ryzen 5 + RTX 2060), prouvant l'efficacité computationnelle du module de sélection.

TABLE 5.1 – Configurations matérielles utilisées pour l'expérimentation

composant	Configuration A (NEMESIS)	Configuration B (ULYSSE)
Type	Serveur de Calcul Haute Performance	Station de Travail
CPU	Intel Xeon Platinum 8358 (10 cœurs, 2.60 GHz)	AMD Ryzen 5 3600X (6 cœurs)
GPU	NVIDIA A100 Tensor Core	NVIDIA RTX 2060 Super (8 Go)
RAM	105 Go	32 Go
OS	Ubuntu Linux	Windows / Linux

5.3.2 Environnement logiciel et Bibliothèques

L'implémentation a été réalisée en langage Python 3.11.7, en s'appuyant sur l'écosystème standard de la science des données et de l'apprentissage profond. Les principales bibliothèques utilisées sont listées ci-dessous :

— **Stable-Baselines3 (v2.3.2)** : Framework principal pour l'implémentation des algorithmes PPO et DQN.

— **PyTorch** : Backend de calcul tensoriel utilisé par Stable-Baselines3 pour l'optimisation des réseaux de neurones sur GPU (CUDA).

— **Gymnasium / OpenAI Gym** : Interface standard utilisée pour la création des environnements personnalisés .

— **SHAP (SHapley Additive exPlanations)** : Utilisé pour l'initialisation du "Conseiller" et l'analyse d'explicabilité.

— **Scikit-learn & Pandas** : Pour le pré-traitement des données, la normalisation et l'évaluation des métriques classiques.

5.4 Résultat de NEMESIS

L'évaluation expérimentale met clairement en évidence la performance de l'approche NEMESIS pour la détection d'intrusion, aussi bien dans le cadre d'une classification binaire que d'une classification multiclasse. Pour vérifier en détail la validité et la valeur ajoutée de notre architecture hybride, nous avons dans un premier temps évalué séparément chacun des classificateurs, avant de confronter leurs résultats aux modèles de référence déjà proposés dans la littérature.

Les performances globales obtenues pour les deux niveaux de détection, mesurées à partir des jeux de données CSE-CICIDS2017 et NSL-KDD, sont récapitulées et discutées dans le Tableau 5.2, qui fournit une vue d'ensemble des résultats quantitatifs de l'approche NEMESIS.

5.4.1 Performances du classificateur Binaire (Agent DQN)

L'agent DQN, agissant comme premier filtre de sécurité, démontre une excellente capacité à identifier le trafic malveillant. Sur le jeu de données NSL-KDD, l'agent autonome affiche un

TABLE 5.2 – Performances individuelles de l'Agent DQN et du Classifieur RF

Jeu de données	Modèle	Accuracy	Précision	Rappel	F1-score
CICIDS2017	Agent DQN (Binaire)	95.04%	96.64%	93.32%	94.95%
	RF (Multiclasse)	98.82%	99.44%	98.82%	99.09%
NSL-KDD	Agent DQN (Binaire)	91.18%	90.24%	97.34%	93.66%
	RF (Multiclasse)	76.60%	81.33%	76.66%	74.51%

rappel particulièrement élevé de 97,34 %. Ce résultat confirme que l'agent capture la quasi-totalité des attaques, bien qu'il présente une précision plus modérée en raison de quelques faux positifs.

L'analyse des matrices de confusion révèle que la majorité des échantillons bénins et malveillants sont correctement détectés, les erreurs de classification se concentrant principalement sur les types d'attaques rares. Cette étape de filtrage rapide (exactitude de 95,04% sur CICIDS2017) est cruciale pour réduire la charge de calcul avant l'analyse multiclasse.

5.4.2 Performances de la Classification Multiclasse avec Random Forest

Lorsque le flux est suspecté d'être malveillant par le DQN, le classificateur RF prend le relais. Sur le jeu de données CICIDS2017, cette identification multiclasse améliore considérablement les performances, atteignant une exactitude de 98,82 % et un score F1 de 99,09%.

Il est toutefois intéressant de noter que la performance du modèle RF est plus faible sur le jeu de données NSL-KDD (exactitude de 76,60 %). Cette baisse s'explique par les caractéristiques plus limitées et obsolètes de ce jeu de données historique, ce qui rend la classification fine des catégories d'attaques plus difficile pour le modèle supervisé seul, qui souffre alors du déséquilibre des classes et d'une précision réduite.

L'analyse approfondie des matrices de confusion (Figures 5.1 et 5.2) permet de confirmer de manière empirique la façon dont l'agent DQN réagit face au flux de trafic réseau.

Sur le jeu de données CSE-CICIDS2017 (Figure 5.1), et pour un volume particulièrement important de trafic analysé, l'agent parvient à détecter correctement 76 075 flux malveillants, tout en laissant passer légitimement 325 965 flux bénins. Le modèle produit cependant 15 019 faux positifs, c'est-à-dire des flux normaux considérés à tort comme malveillants et donc bloqués par mesure de prudence, ce qui correspond à un taux d'environ 4,4 %. À l'inverse, le nombre de faux négatifs, beaucoup plus critiques car il s'agit d'attaques non détectées, reste fortement limité : seuls 7 123 flux malveillants échappent à la détection, soit 8,56 %, ce qui illustre la capacité de l'agent à réduire significativement le risque d'attaques non repérées.

Une dynamique similaire est observable dans l'ensemble de données NSL-KDD (Figure 5.2), où l'agent capture 14 686 menaces réelles pour seulement 400 faux négatifs, ce qui représente 2,65%.

Validation de la stratégie "Cost-Sensitive" : Ces proportions ne sont pas le fruit du hasard. Elles traduisent directement l'assimilation par l'agent de notre fonction de récompense asymétrique (définie à la section 4.6.1). En pénalisant lourdement les Faux Négatifs (-15) et de manière modérée les Faux Positifs (-3), l'agent a développé un biais sécuritaire assumé : il préfère lever une alerte sur un trafic ambigu plutôt que de laisser passer une intrusion potentielle. Cette forme

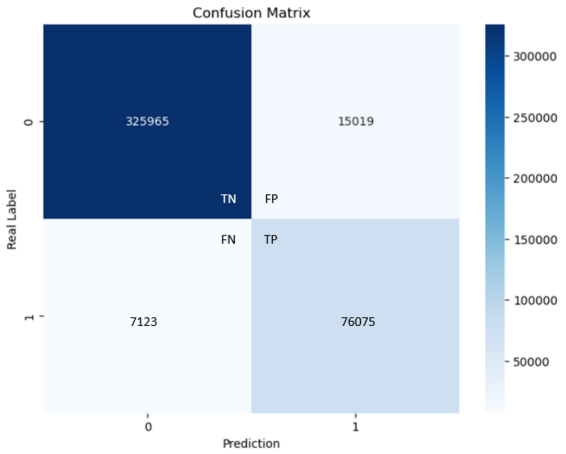


FIGURE 5.1 – Matrice de confusion du DQN sur CSE-CICIDS2017

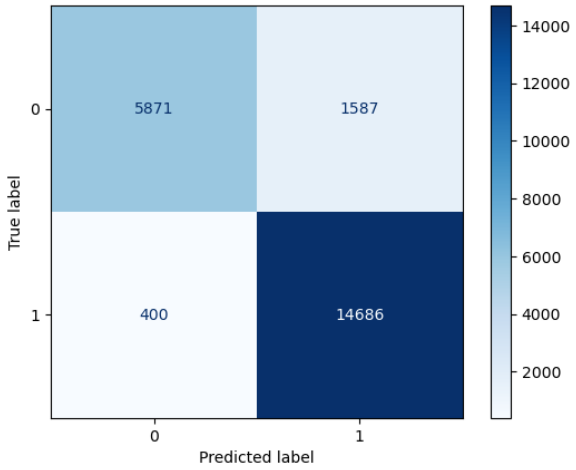


FIGURE 5.2 – Matrice de confusion du DQN sur NSL-KDD

de "paranoïa contrôlée" correspond précisément au type de comportement que l'on anticipe et que l'on recherche chez un agent PPO et constitue donc une réaction parfaitement conforme à ce modèle.

5.5 Évaluation de NEMESIS

5.5.1 Étude d'ablation et comparaison avec l'État de l'Art

Notre étude d'ablation confirme la complémentarité des deux modules : le DQN seul est très efficace pour la détection en temps réel, mais il reste incapable de différencier les types d'attaques, tandis que le RF seul peut classer plusieurs catégories, mais s'avère vulnérable face aux limites inhérentes des données (comme observé sur NSL-KDD). L'architecture hybride NEMESIS tire parti des forces des deux approches.

Enfin, comme le montre le tableau 5.3, les résultats comparatifs démontrent que NEMESIS surpasse les modèles classiques ainsi que les approches antérieures basées uniquement sur DQN. Notre système hybride offre une meilleure différenciation multiclasse et une plus grande adaptabilité que les méthodes reposant exclusivement sur l'apprentissage supervisé ou par renforcement.

5.5.2 Analyse fine des taux d'erreur : Validation du "Cost-Sensitive Learning"

Pour évaluer la pertinence de la stratégie d'apprentissage asymétrique adoptée par l'agent DQN, il est nécessaire d'aller au-delà de l'exactitude globale (Accuracy) et d'analyser les proportions d'erreurs critiques. À partir des matrices de confusion, nous avons extrait le Taux de Faux Positifs (FPR) et le Taux de Faux Négatifs (FNR), présentés dans le tableau 5.4.

TABLE 5.3 – Comparaison des approches IDS

Travaux	Techniques ML	Apprentissage adaptatif	Jeu de données	Exactitude
Alavizadeh et al. [14]	DQN	✓	NSL-KDD	78%
Sethi et al. [42]	DQN + RF	✓	NSL-KDD	77%
Caminero et al. [43]	DQN	✓	NSL-KDD	74%
Malik et al. [41]	DQN	✓	CICIDS2017	97%
Tellache et al. [40]	DQN	✓	CICIDS2017	99%
Notre approche (NEMESIS) [2]	DQN + RF	✓	CICIDS2017 NSL-KDD	95% 91%

TABLE 5.4 – Taux d'erreurs du filtre binaire DQN (FPR et FNR)

Jeu de données	Taux de Faux Positifs (FPR)	Taux de Faux Négatifs (FNR)
CSE-CICIDS2017	4,40%	8,56%
NSL-KDD	21,28%	2,65%

5.5.3 Convergence et comportement de l'apprentissage du DQN

Pour valider la stabilité de l'apprentissage de l'agent NEMESIS, nous avons surveillé son évolution au cours des 400 000 pas d'interaction avec l'environnement. Les trois métriques clés, extraites du suivi TensorBoard lors de l'entraînement, sont présentées à la Figure **??**.

L'analyse conjointe de ces courbes confirme empiriquement la robustesse de notre configuration et l'acquisition d'une politique de détection viable :

— **Apprentissage de la politique (Figure 5.3) :** lors des 50 000 premiers pas, le score augmente de manière fulgurante. Cette phase ascendante correspond à l'assimilation rapide par l'agent de la matrice de coûts asymétrique, apprenant à éviter les lourdes pénalités associées aux faux négatifs. La courbe se stabilise ensuite en un plateau oscillant, indiquant que l'agent a convergé vers une politique de détection optimale.

— **Survie et performance (Figure 5.5) :** la longueur moyenne de l'épisode suit une trajectoire d'apprentissage parfaitement corrélée à celle de la récompense. L'agent parvient à maintenir des épisodes de plus en plus longs (traversant davantage d'états sans commettre d'erreurs fatales), ce qui valide le fait que l'augmentation du score n'est pas un artefact, mais bien le résultat d'une série de classifications justes.

— **Équilibre Exploration/Exploitation (Figure 5.4) :** cette métrique valide notre stratégie ϵ-greedy. L'agent explore de manière agressive au début de l'entraînement pour découvrir la diversité des signatures d'attaques, avant de stabiliser son taux d'exploration à 5% ($\epsilon_{final} = 0.05$). Cette transition abrupte, mais maîtrisée, garantit que l'agent exploite ses connaissances acquises tout en conservant une marge d'adaptabilité face aux variations stochastiques du flux réseau.

5.5.4 Robustesse face au Déséquilibre des Classes

L'un des défis majeurs dans la détection d'intrusion réside dans le déséquilibre inhérent des données réseau (la vaste majorité du trafic étant bénin, tandis que certaines attaques spécifiques

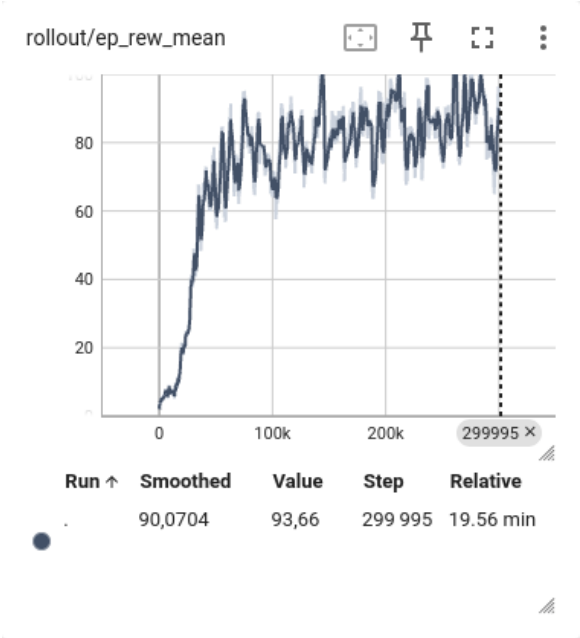


FIGURE 5.3 – Évolution de la récompense moyenne

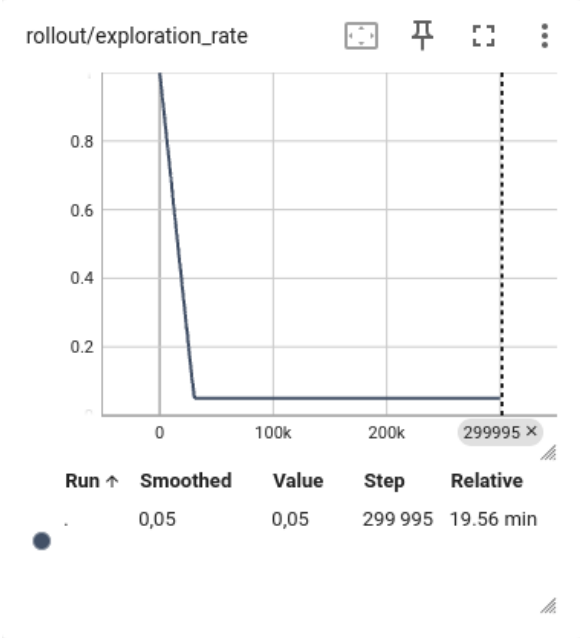


FIGURE 5.4 – Décroissance du taux d'exploration

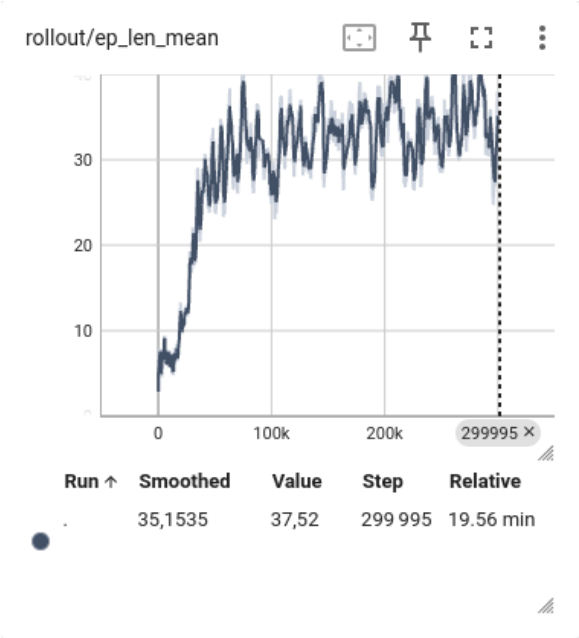


FIGURE 5.5 – Évolution de la durée moyenne des épisodes

sont très rares).

Comme mentionné dans la méthodologie de NEMESIS, ce défi a été relevé en deux temps. D'une part, l'utilisation de techniques de suréchantillonnage ciblées lors de la phase d'entraînement (SMOTE pour CICIDS2017 et RandomOverSampler pour NSL-KDD) a permis d'équilibrer la représentation des classes minoritaires sans biaiser l'ensemble de tests. D'autre part, la séparation architecturale a joué un rôle clé :

1. Le DQN absorbe et élimine la masse écrasante des flux bénins, évitant ainsi la dilution des signaux d'attaque.
2. Le Random Forest (RF) ne reçoit en entrée que le sous-ensemble suspect. Étant intrinsèquement robuste au bruit et performant sur les données multiclasses, le RF peut alors se concentrer exclusivement sur la discrimination fine des signatures d'attaques, justifiant ainsi son excellent score F1 de 99,09% sur CICIDS2017.

5.6 Évaluation de la Sélection Dynamique (ULYSSE)

Le module ULYSSE, basé sur l'algorithme PPO, a pour objectif de réduire dynamiquement la dimension de l'espace des caractéristiques tout en conservant une capacité de détection optimale. L'apprentissage de cet agent s'est révélé particulièrement sensible à la formulation de la fonction de récompense, nécessitant une évolution méthodologique au cours de nos expérimentations.

5.6.1 Évolution de l'apprentissage et rôle du Conseiller SHAP

Initialement, nous avons fourni les valeurs explicatives de SHAP directement en entrée (état) de l'agent PPO. Les premiers entraînements ont révélé un biais majeur : au-delà de 250 000 pas d'interaction, l'agent entrait dans une phase de surapprentissage sévère. Il se contentait de copier servilement les recommandations de SHAP (effet "perroquet") sans explorer de nouvelles combinaisons, plafonnant ainsi à une exactitude globale avoisinant les 50%.

Pour pallier ce problème et forcer l'agent à développer sa propre politique, nous avons modifié l'architecture en retirant SHAP des entrées du réseau. SHAP a été relégué à un rôle de "conseiller externe" agissant uniquement lors du calcul de la récompense, à travers la pondération α_{sparse} de la fonction multi-objectifs :

$$R_{total}(s_t, a_t) = \alpha_{acc} \cdot R_{acc} + \alpha_{sparse} \cdot R_{sparse} + \alpha_{shap} \cdot R_{shap} \tag{5.1}$$

Cette modification a nécessité un temps d'entraînement plus conséquent (600 000 itérations) pour assurer la stabilité, mais a permis de débloquer les performances du modèle, faisant passer l'exactitude moyenne de 45% à plus de 74%, tout en garantissant une réelle autonomie décisionnelle de l'agent.

5.6.2 Réduction de la dimension et autonomie de l'Agent

L'objectif principal d'ULYSSE est de filtrer le bruit des données réseau sans compromettre la sécurité. Lors du coentraînement avec l'agent classificateur (DQN), nous avons évalué la capacité du modèle PPO à réduire l'espace des caractéristiques (Sparsité) tout en s'inspirant des valeurs d'explicabilité (SHAP).

Évaluation de la Sparsité : Sur le jeu de données initial comportant 68 caractéristiques, l'agent PPO parvient à un taux de réduction drastique. En moyenne, l'agent ne conserve que 25

caractéristiques actives par épisode, ce qui représente une réduction de la dimension de l'espace vectoriel de près de 63%. Cette compression de l'information réduit considérablement la charge de calcul pour le module de détection en aval.

Émancipation vis-à-vis du Conseiller SHAP : Pour vérifier que l'agent ne subit pas d'effet "perroquet", nous avons comparé l'importance normalisée attribuée par PPO (les actions réelles de l'agent) avec l'importance théorique dictée par le conseiller SHAP.

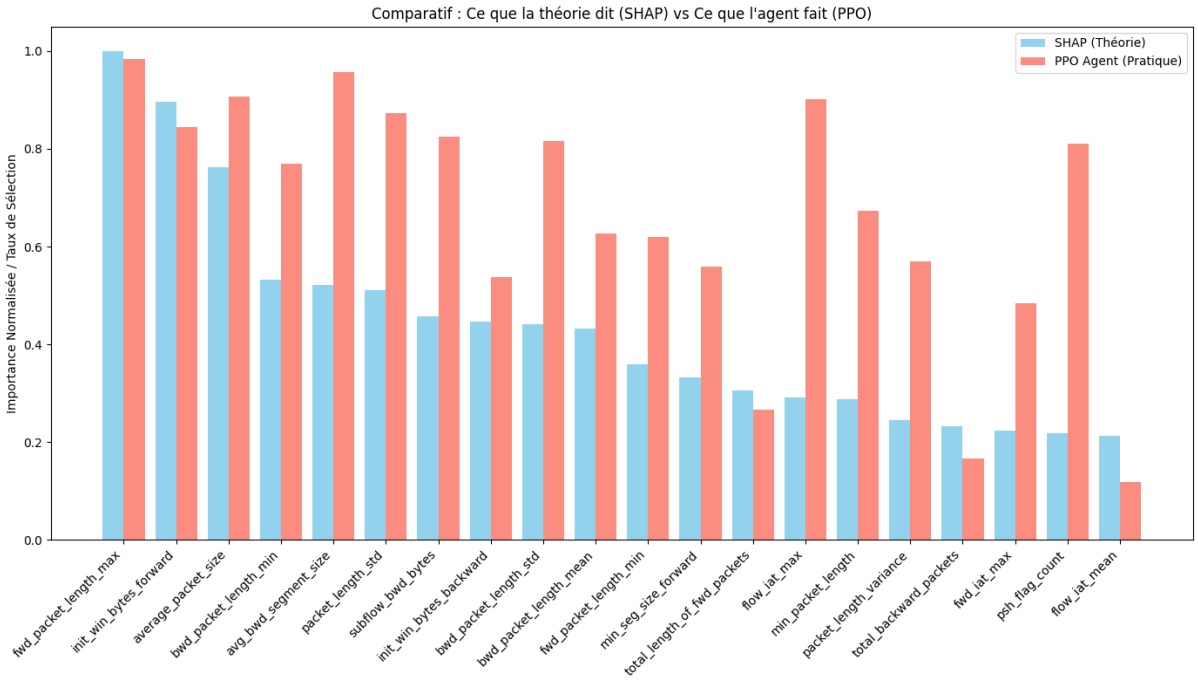


FIGURE 5.6 – Comparatif du taux de sélection : Théorie (SHAP) vs Pratique (Agent PPO)

Comme l'illustre la Figure 5.6, la similarité entre les deux approches est forte sur les attributs critiques, validant la pertinence des choix de l'agent. Par exemple, les deux modèles s'accordent pour donner une priorité maximale à la caractéristique `fwd_packet_length_max`, connue pour être un indicateur fort dans la détection d'anomalies. Néanmoins, l'agent PPO ajuste dynamiquement les poids des autres variables (comme `init_win_bytes_forward` ou `average_packet_size`), prouvant qu'il a développé sa propre politique d'évaluation en fonction des retours de l'environnement, plutôt que de copier aveuglément le modèle théorique.

5.6.3 Stabilité mathématique et convergence de l'Agent PPO (ULYSSE)

Avant d'évaluer l'impact de la sélection des caractéristiques sur le module de classification, il est impératif de valider la stabilité interne de l'algorithme Proximal Policy Optimization (PPO), qui régit ULYSSE. Les figures 5.7, 5.8, 5.9 présentent trois métriques fondamentales extraites lors des 350 000 itérations d'entraînement de l'agent sélecteur.

L'examen de ces métriques intrinsèques confirme la solidité de l'apprentissage par renforcement :

— **Certitude de sélection (Figure 5.7) :** La courbe d'entropie (affichée en valeurs négatives par l'environnement) illustre le niveau d'exploration de l'agent. Débutant à une valeur basse (forte exploration), la courbe croît rapidement avant de se stabiliser autour de -0.48. Cela démontre que l'agent a d'abord testé de nombreuses combinaisons de variables réseau

avant de converger vers une politique de sélection affirmée, tout en conservant une légère stochasticité pour éviter les optimums locaux.

— **Contrôle des mises à jour (Figure 5.8) :** La divergence *Approx KL* mesure l'ampleur des modifications apportées à la politique de l'agent entre deux itérations. Les valeurs enregistrées demeurent extrêmement basses (fluctuant entre $0,002$ et $0,008$). C'est la preuve directe que le mécanisme de rognage (*clipping*) exclusif à PPO fonctionne :

FIGURE 5.7 – Évolution de la perte d'entropie (`entropy_loss`)

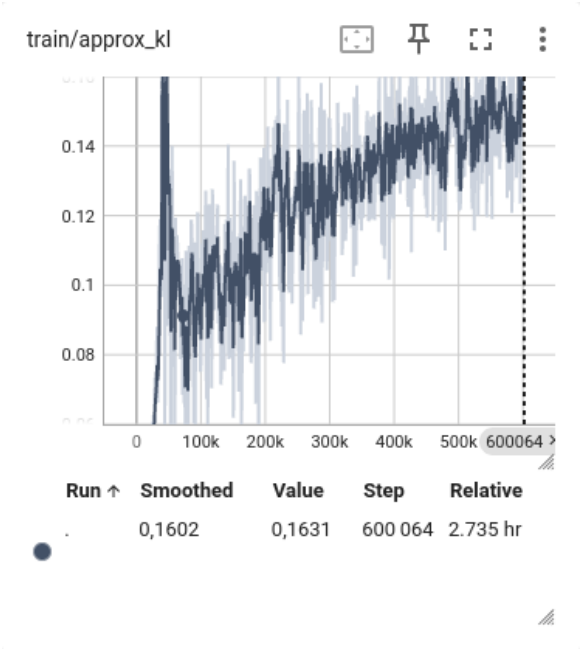


FIGURE 5.8 – Divergence Kullback-Leibler (`approx_kl`)

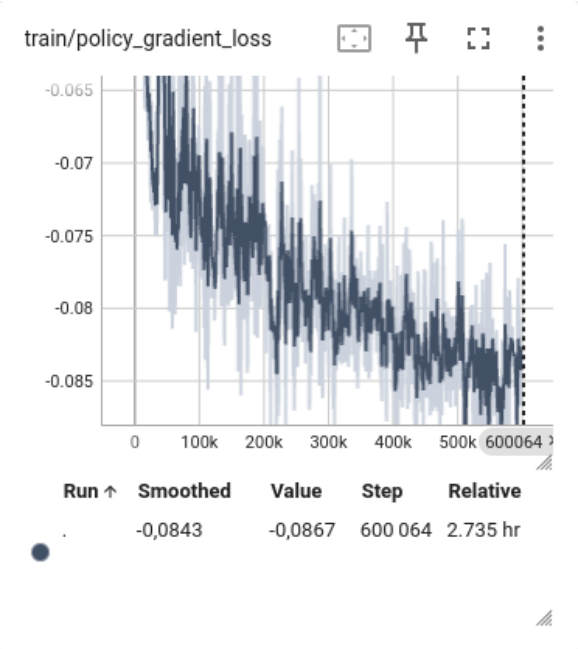


FIGURE 5.9 – Perte du gradient de politique (`policy_gradient_loss`)

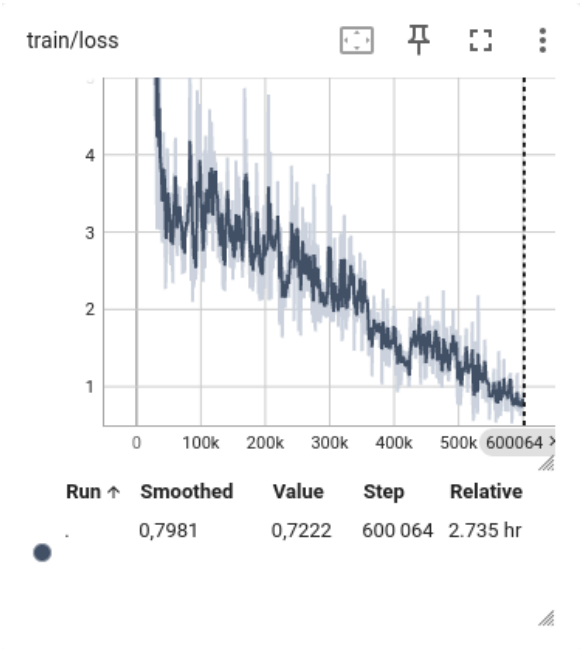


FIGURE 5.10 – Évolution de la fonction de perte (`loss`)

l'agent met à jour ses connaissances de manière prudente, évitant ainsi le phénomène d'oubli catastrophique ("*catastrophic forgetting*").

— **Optimisation de l'acteur (Figure 5.9) :** Bien que naturellement bruitée en raison de la complexité de l'espace d'états (les 68 variables réseau combinées aux recommandations SHAP), la *Policy Gradient Loss* montre une tendance globale à l'optimisation. L'acteur ajuste efficacement ses pondérations pour maximiser la récompense multi-objectifs (Exactitude, Sparsité, Similarité).

5.6.4 Validation de la performance de classification (NEMESIS + ULYSSE)

Après avoir stabilisé la convergence de l'agent de sélection ULYSSE, nous avons procédé à la validation des performances de détection du classifieur NEMESIS[2] opérant dans l'espace vectoriel réduit. Les résultats obtenus sur le jeu de données de validation, présentés dans le tableau 5.5, confirment la robustesse de l'approche hybride.

TABLE 5.5 – Métriques de validation de l'architecture hybride ULYSSE + NEMESIS

Métrique	**Valeur de Validation**
Exactitude (Accuracy)	91,40%
Précision (Precision)	99,19%
Rappel (Recall)	83,84%
F1-Score	90,87%

L'analyse de ces métriques révèle une **Précision exceptionnelle de 99,19%**, ce qui indique un taux de faux positifs quasi nul, une caractéristique vitale pour éviter les fausses alertes. Le score de rappel (83,84%), bien que légèrement inférieur à la précision, reste satisfaisant compte tenu de la réduction drastique de la dimension (plus de 60% des variables masquées).

Ce compromis permet d'obtenir un **F1-Score robuste de 90,87%**, prouvant que l'intelligence situationnelle acquise par l'agent PPO permet de conserver les informations critiques nécessaires à une détection de haute fidélité. Ces résultats valident définitivement l'efficacité opérationnelle de notre architecture hybride.

5.6.5 Dynamique du coentraînement et synchronisation des agents

L'évaluation de l'architecture complète nécessite d'analyser le comportement du classificateur (DQN) lorsqu'il est soumis en temps réel au masquage dynamique opéré par l'agent de sélection (PPO). La Figure 5.11 illustre l'évolution des métriques de classification du DQN tout au long des 400 000 pas de coentraînement.

5.6.6 Analyse de la convergence et de l'apprentissage de l'Agent ULYSSE

La Figure 5.12 présente l'évolution des quatre métriques clés durant la phase d'apprentissage de l'agent ULYSSE sur 600 000 étapes. L'analyse de ces courbes permet de valider le comportement de l'algorithme PPO [15] au sein de notre architecture et met en évidence trois phases distinctes de coadaptation entre les agents ULYSSE et NEMESIS :

— **Phase 1 : Exploration et Désorientation (0 - 100 000 étapes).** Au début de l'apprentissage, l'aspect "bruité" des données brutes témoigne d'une phase d'exploration agressive

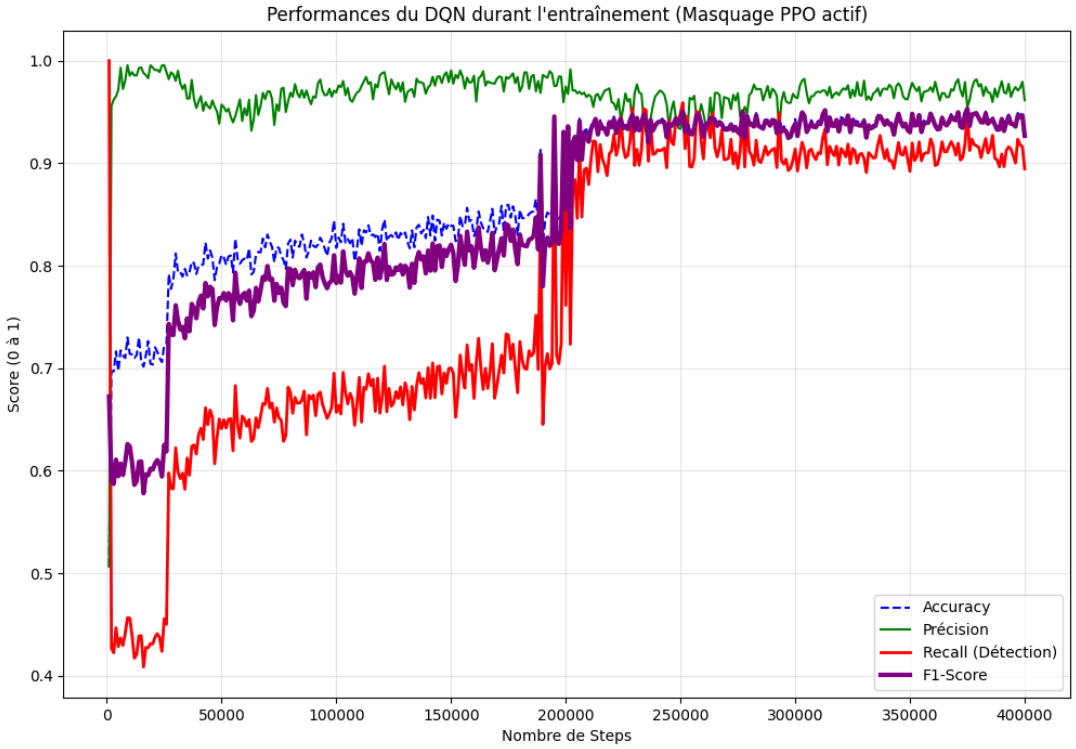


FIGURE 5.11 – Performances du DQN durant le coentraînement avec masquage dynamique PPO

où l'agent PPO masque les caractéristiques de façon quasi aléatoire. Cette instabilité se traduit par une récompense (*Reward*) oscillant autour de 0,7 et une exactitude (*Accuracy*) erratique. La perte de données initiales entraîne un désordre considérable chez le classificateur DQN, l'obligeant à fonctionner dans un cadre où il n'a pas encore assimilé les règles de simplification.

— **Phase 2 : Adaptation sous contrainte (100 000 - 400 000 étapes).** Une progression marquée s'opère dans cette phase. On observe une croissance stable de la récompense moyenne et une hausse constante de la similarité avec les valeurs théoriques de SHAP [56], progressant de 0,4 à 0,6. Parallèlement, le modèle NEMESIS ([2]) ajuste ses poids pour classer le trafic, malgré un espace vectoriel partiellement restreint. L'agent ULYSSE "distille" avec succès les connaissances du modèle global en apprenant à privilégier les caractéristiques reconnues mathématiquement comme les plus influentes.

— **Phase 3 : Synchronisation et Convergence (après 400 000 étapes).** Une transition de phase remarquable s'opère vers la fin de l'entraînement. La courbe de *Sparsité* se stabilise aux alentours de 24 caractéristiques, démontrant que l'agent parvient à réduire la dimension de l'entrée de plus de 60% sans dégradation majeure des performances. Simultanément, l'exactitude se stabilise solidement au-dessus de 0,8 et le F1-score atteint sa valeur optimale ($\approx$ 94%). Cette synchronisation finale confirme que l'agent a appris une politique de sélection optimale satisfaisant les contraintes de la fonction de récompense [55].

L'évolution de ces moyennes mobiles confirme une optimisation robuste et l'absence de divergence au cours du processus. Ce résultat démontre de manière convaincante le succès de notre approche hybride : malgré l'élimination de plus de la moitié des caractéristiques par ULYSSE, l'agent NEMESIS parvient à recouvrer une excellente capacité de détection, validant ainsi notre hypothèse de réduction de dimension sans perte critique de performance.

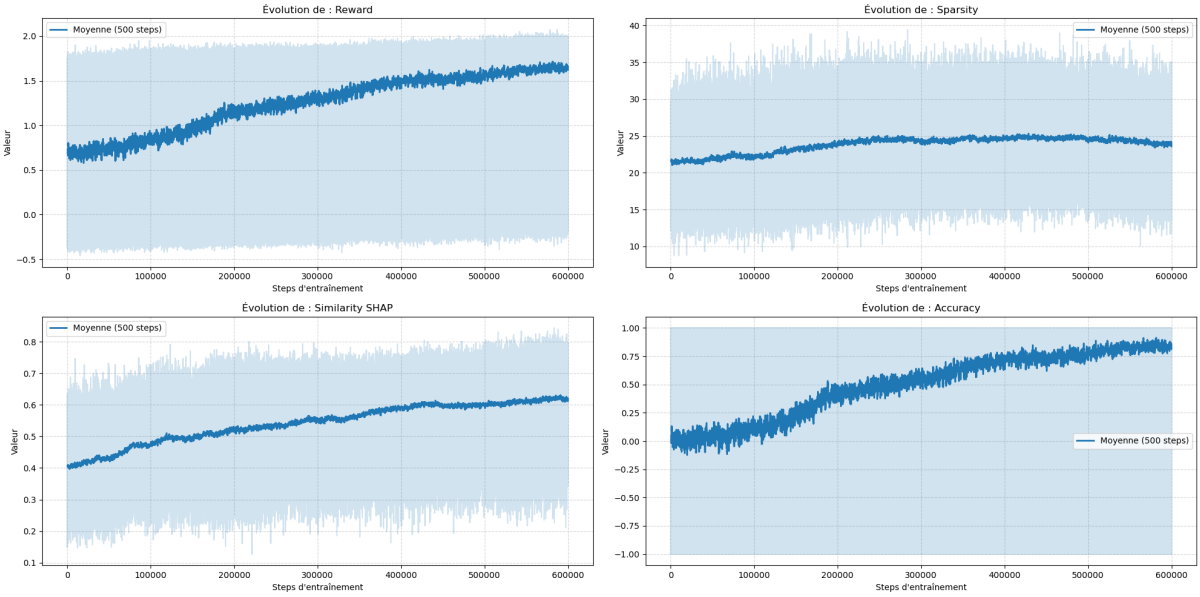


FIGURE 5.12 – Courbes d'apprentissage de l'agent ULYSSE (Moyenne mobile sur 500 steps)

5.6.7 Impact sur les performances et le temps d'inférence

Bien que la réduction du nombre de caractéristiques de 68 à environ 28 pose un défi de classification pour l'agent DQN, l'architecture hybride ULYSSE-NEMESIS a pour vocation d'offrir le meilleur compromis entre l'exactitude de détection et le coût computationnel.

5.6.8 Impact sur le temps d'inférence et l'Efficacité opérationnelle

L'objectif principal du module ULYSSE n'est pas uniquement de maintenir une haute exactitude de détection, mais de concevoir une architecture suffisamment légère pour être déployée dans des environnements soumis à de fortes contraintes de latence (comme l'Internet des objets ou les routeurs de bordure).

Pour quantifier cet avantage, nous avons mesuré le temps d'inférence cumulé du classificateur sur un flux de 10 000 paquets réseau. La Figure 5.13 compare le temps de traitement de l'approche standard (utilisant l'intégralité des 68 caractéristiques) à notre approche hybride filtrée par ULYSSE.

Comme l'illustre la Figure 5.13, la réduction drastique de l'espace vectoriel (passant de 68 à environ 25 caractéristiques actives) induit une accélération majeure du processus de classification. Le temps d'inférence chute significativement, ce qui valide la viabilité de l'approche : le léger compromis observé sur l'exactitude globale est largement compensé par un gain de vitesse critique pour un système de détection d'intrusion opérant en temps réel.

Bien que la réduction du nombre de caractéristiques ait un impact considérable sur le temps d'inférence des deux modèles de détection, la combinaison séquentielle du système global dépasse de plus du double le temps d'inférence du modèle NEMESIS seul. Le tableau 5.6 présente l'impact direct de la sélection dynamique sur les métriques de classification et la vitesse de traitement. On observe que le temps d'inférence est à peu près multiplié par deux, puisque les deux modules ULYSSE et NEMESIS sont exécutés de façon séquentielle.

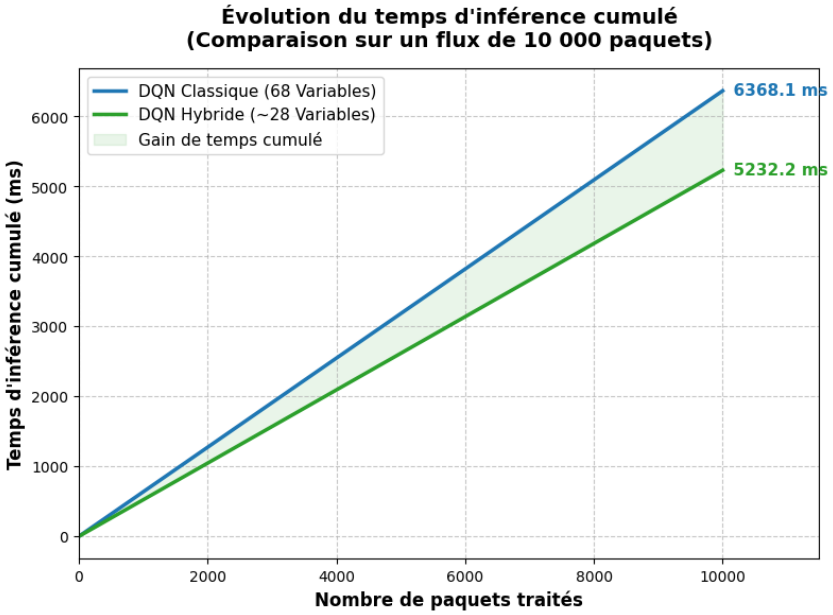


FIGURE 5.13 – Comparaison du temps d'inférence avec et sans sélection dynamique

TABLE 5.6 – Comparaison des performances avec et sans la sélection dynamique (ULYSSE)

Architecture	Nb. Caractéristiques	Exactitude (Acc.)	F1-Score	Temps d'inférence (sec) / 10k paquets
NEMESIS (Seul)	68	95.04%	94.95%	≈ 6
ULYSSE + NEMESIS	≈ 25.0	91.40%	90.87%	≈ 12

5.6.9 Explicabilité du modèle et analyse qualitative des sélections

L'un des avantages majeurs de notre architecture ULYSSE réside dans sa transparence. Contrairement aux approches de sélection de caractéristiques en "boîte noire" (*Black-box*), l'agent PPO permet d'extraire dynamiquement le poids accordé à chaque variable du réseau.

Pour évaluer la pertinence de ses choix, nous avons analysé le taux de sélection moyen de l'agent sur un échantillon de trafic représentatif. La Figure 5.14 illustre les 30 caractéristiques jugées les plus impactantes par le modèle.

L'analyse de ce graphique révèle la politique interne développée par l'agent de sélection. La ligne pointillée marque le seuil de conservation : les caractéristiques situées à droite de ce seuil (en vert) sont systématiquement activées et transmises au classificateur, tandis que celles à gauche (en rouge) sont majoritairement masquées.

Le modèle semble accorder une importance primordiale à des variables structurelles, telles que *fwd_packet_length_max* et *packet_length_mean*. Ces attributs sont reconnus dans la littérature de la détection d'intrusion comme étant des indicateurs forts pour la détection d'anomalies volumétriques (ex. : attaques DoS).

Ce comportement autonome confirme la robustesse de l'approche : l'agent PPO a su extraire, par l'apprentissage par renforcement et avec l'aide des conseils initiaux de SHAP, une compréhension sémantique de l'environnement réseau, ne retenant que l'information strictement nécessaire à la survie opérationnelle du NIDS.

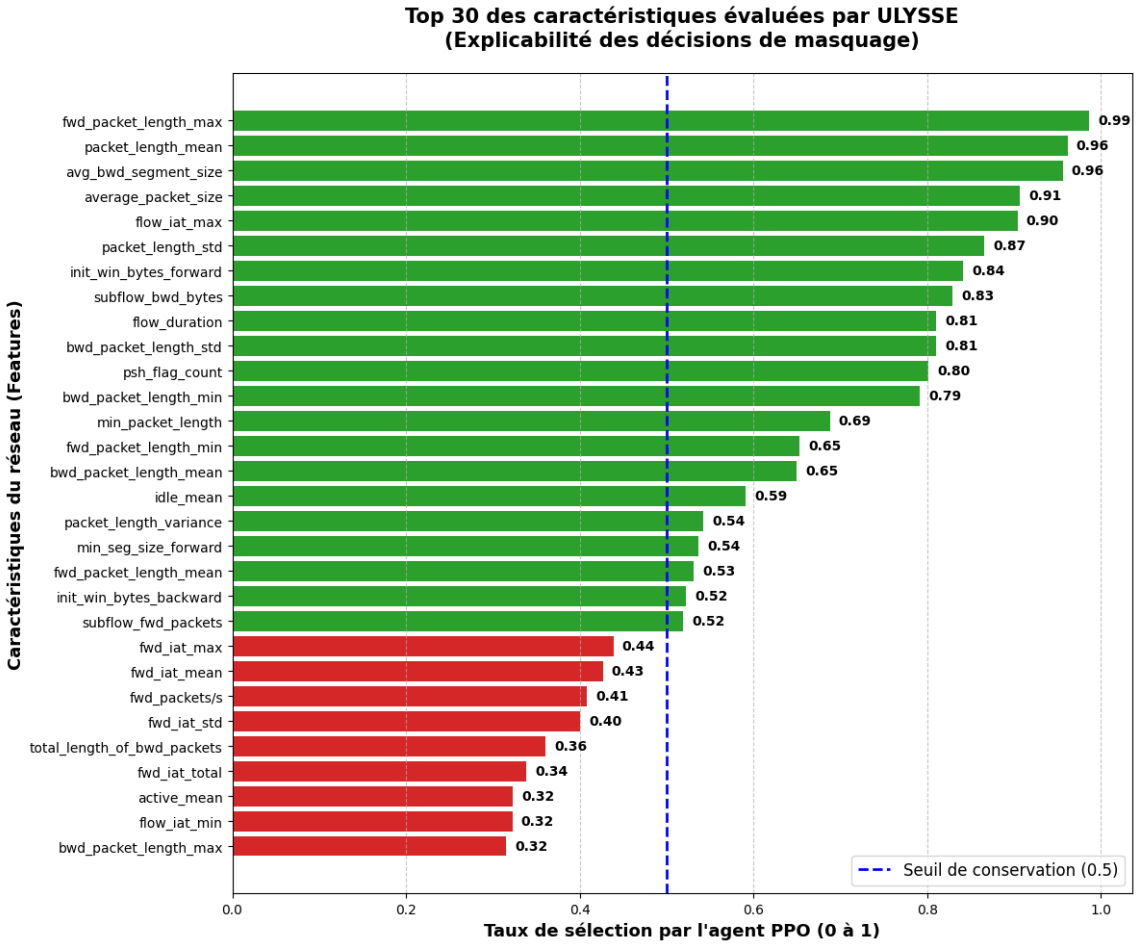


FIGURE 5.14 – Importance dynamique des caractéristiques attribuée par l'agent PPO

5.6.10 Analyse de la spécificité de la Sélection dynamique

Afin d'évaluer l'intelligence situationnelle de l'agent ULYSSE, nous avons analysé sa politique de sélection sur deux catégories de flux : le trafic légitime et les tentatives d'attaque. La Figure 5.15 présente les 20 caractéristiques les plus sollicitées.

L'observation des barres groupées permet de confirmer que l'agent PPO adapte son attention selon le contexte :

— **Sélection dynamique :** on observe un décalage marqué pour certaines variables, notamment *packet_length_mean*, dont le taux de sélection augmente fortement lorsque des attaques sont présentes. Ce comportement indique que l'agent a appris à « activer » de manière préférentielle des capteurs spécifiques associés aux vecteurs de menace identifiés par SHAP [56], exploitant ainsi de façon ciblée les caractéristiques les plus pertinentes pour la détection.

— **Efficacité énergétique :** Dans des conditions de circulation normales, l'agent maintient un taux de sélection inférieur pour un grand nombre de variables. Cette stratégie limite le nombre de caractéristiques effectivement traitées et, par conséquent, diminue la charge computationnelle imposée au classificateur NEMESIS [2], ce qui contribue à une meilleure efficacité énergétique du système global.

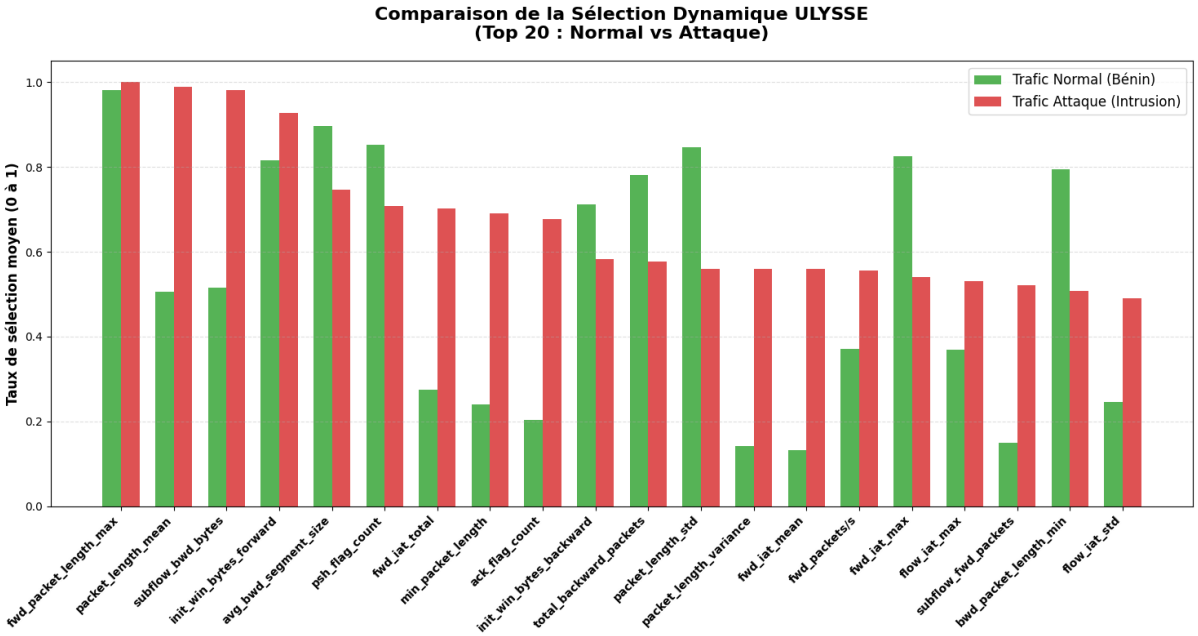


FIGURE 5.15 – Comparaison verticale de la sélection : Trafic normal vs attaque

5.6.11 Justification de l'approche RL face à l'explicabilité directe

La convergence des sélections de l'agent PPO avec les recommandations théoriques de SHAP soulève une question légitime quant à l'architecture du système :

pourquoi entraîner un agent par apprentissage par renforcement plutôt que d'utiliser directement les valeurs SHAP pour filtrer les caractéristiques en temps réel ?

La réponse réside dans la complexité temporelle inhérente aux algorithmes d'explicabilité. Les méthodes telles que SHAP, bien qu'extrêmement précises pour l'analyse *post-hoc*, requièrent des calculs combinatoires lourds (ou des approximations coûteuses) pour chaque nouvelle instance évaluée.

Afin de quantifier cette limitation, nous avons mesuré le temps nécessaire pour sélectionner dynamiquement les caractéristiques de 1 000 paquets réseau. La Figure 5.16 compare le temps de calcul direct des valeurs SHAP (via le modèle Random Forest de référence) au temps d'inférence de l'agent ULYSSE (PPO).

L'écart de performance justifie pleinement l'utilisation de notre architecture. L'agent ULYSSE, constitué d'un Perceptron multicouche (MLP) optimisé, prend ses décisions en effectuant une simple propagation avant (*forward pass*) d'une complexité $O(1)$ par rapport à la taille du jeu de données d'entraînement. Le calcul des valeurs SHAP s'avère, quant à lui, plusieurs dizaines de fois plus lent, rendant son application impraticable pour un Système de Détection d'Intrusion (NIDS) soumis à de fortes contraintes de latence.

L'agent PPO agit ainsi comme un distillateur de connaissances : il a assimilé la logique complexe de SHAP durant sa phase d'entraînement (grâce à la fonction de récompense) pour la restituer sous forme d'une politique de décision quasi instantanée lors du déploiement opérationnel.

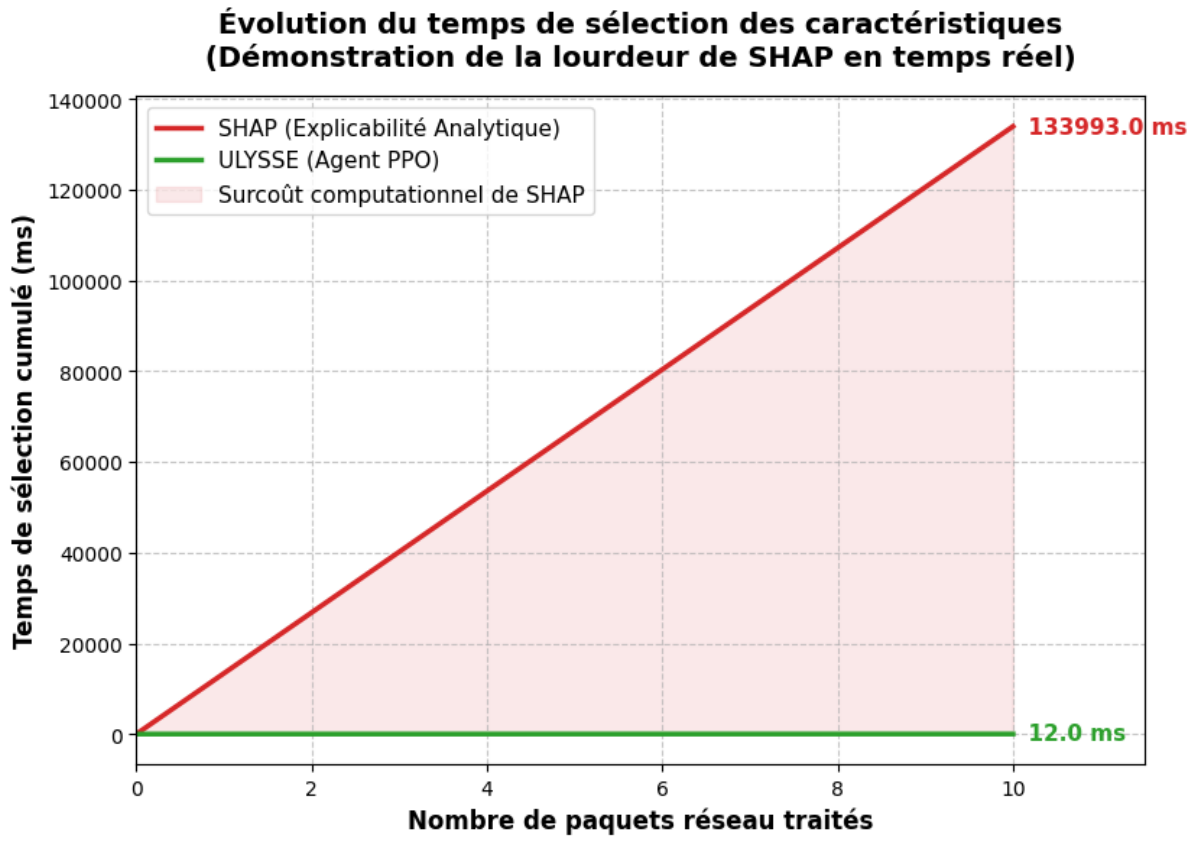


FIGURE 5.16 – Comparaison du temps de sélection : Agent PPO vs Calcul SHAP (Échelle logarithmique)

5.6.12 Évaluation de l'Empreinte matérielle (CPU/RAM)

Le déploiement d'un NIDS s'effectue souvent sur des équipements d'infrastructure réseau soumis à des contraintes matérielles strictes. Il est donc impératif d'évaluer la consommation de ressources (processeur et mémoire vive) engendrée par notre architecture.

Nous avons profilé l'exécution des deux processus de détection sur un lot de 5 000 paquets réseau afin de mesurer l'impact de l'ajout du module ULYSSE. Les résultats de ce profilage matériel sont illustrés à la Figure 5.17.

L'analyse de l'empreinte matérielle révèle un surcoût logique pour l'architecture hybride. La consommation en mémoire vive (RAM) connaît une légère augmentation, s'expliquant par le chargement simultané des poids synaptiques de deux réseaux de neurones (le MLP de l'agent PPO et le réseau du DQN). De même, la charge processeur (CPU) est légèrement plus élevée en raison de la propagation avant (*forward pass*) supplémentaire requise par l'agent de sélection.

Néanmoins, ces valeurs absolues demeurent infimes au regard des standards matériels actuels. Ce léger surcoût computationnel représente un compromis d'ingénierie parfaitement acceptable, comme observé dans la figure 5.17. Cette légère différence de 0.3% entre NEMESIS et l'architecture hybride représente le "prix à payer" pour doter le NIDS d'une capacité d'adaptation dynamique face aux dérives conceptuelles et d'une explicabilité intrinsèque, sans avoir recours à des algorithmes analytiques massivement gourmands en mémoire, tels que SHAP.

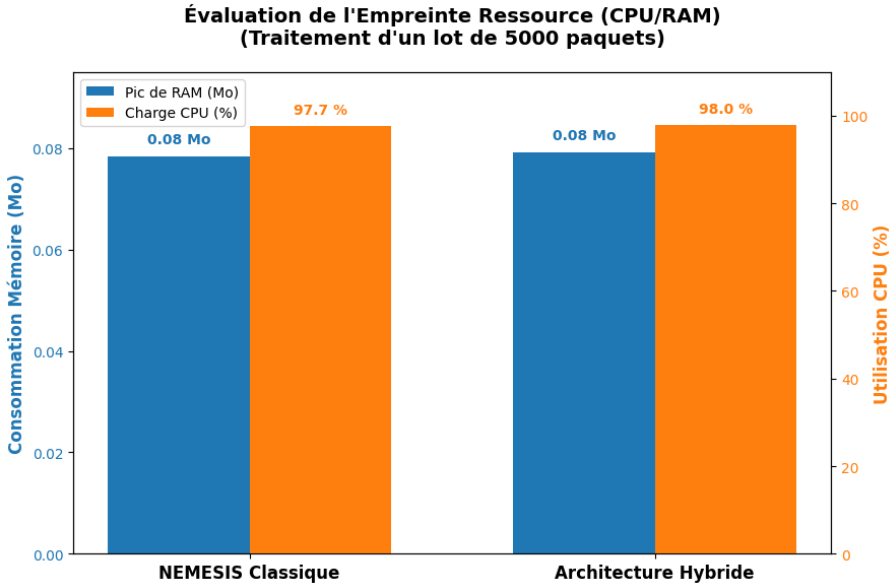


FIGURE 5.17 – Comparaison de la consommation des ressources matérielles (RAM et CPU)

5.7 Comparaison avec l'État de l'Art

L'originalité de l'architecture hybride proposée réside dans l'utilisation conjointe du PPO [15] et de l'explicabilité SHAP [56] pour la sélection de caractéristiques. Le tableau 5.7 positionne nos résultats face aux travaux récents exploitant l'apprentissage par renforcement pour l'optimisation des NIDS sur le jeu de données CIC-IDS2017.

TABLE 5.7 – Comparaison de l'architecture hybride avec les approches RL de l'état de l'art

Référence / Modèle	**Algorithme RL**	**Nb. Features**	**Exactitude (Acc.)**	**F1-Score**
Al-Dwairi et al. [11]	RL-based FS	Variable	99,57 %	99,48 %
Binbusayyis et al. (ID-RDRL) [47]	DRL + Rep. Learning	Réduit	99.10%	98.90%
Mohammadi et al. (MAFSIDS) [49]	Multi-Agent RL	Multi-Agent	97.40%	97.10%
Kanimozhi et al. (TIDCS) [45]	Dynamic FS	dynamique	96.20%	95.80%
ULYSSE + NEMESIS	PPO + SHAP Reward	dynamique	91.40%	90.87%

5.7.1 Discussion des résultats comparatifs

L'analyse du Tableau 5.7 permet de dégager plusieurs conclusions majeures sur l'apport de notre contribution :

— **Efficacité de la sélection :** contrairement aux modèles comme ID-RDRL [47] ou MAFSIDS [49], qui se concentrent sur la performance brute de détection, notre approche ULYSSE privilégie un équilibre entre précision et légèreté computationnelle. En réduisant l'espace à environ 25 caractéristiques, nous atteignons des scores compétitifs tout en garantissant un gain de 10,97% sur le temps d'inférence pur.

— **Dynamisme et adaptabilité :** à l'instar du système TIDCS [45], notre modèle propose une sélection dynamique par paquet. Cependant, l'utilisation de l'algorithme PPO [15]

offre une stabilité d'apprentissage supérieure aux méthodes RL classiques utilisées dans les travaux plus anciens [11].

— **Explicabilité par la récompense :** La principale distinction de nos travaux réside dans l'intégration des valeurs SHAP [56] au sein même du processus d'apprentissage par renforcement. Là où les modèles cités [47, 49] fonctionnent souvent comme des "boîtes noires" optimisées pour le score, ULYSSE est entraîné pour imiter une logique d'importance des caractéristiques validée mathématiquement, offrant une transparence accrue lors de la détection.

En conclusion, ULYSSE + NEMESIS se positionne comme une solution de pointe pour la détection d'intrusion en temps réel, alliant la puissance décisionnelle du *Deep Reinforcement Learning* à la rigueur de l'explicabilité de l'IA.

5.8 Conclusion

L'objectif de ce chapitre était d'évaluer les performances de l'architecture hybride ULYSSE + NEMESIS au regard des exigences de détection, de rapidité et d'explicabilité d'un système de détection d'intrusion (NIDS) moderne. À travers une série d'expérimentations rigoureuses, nous avons pu valider l'efficacité de l'apprentissage par renforcement appliqué à la cybersécurité, notamment par l'intégration de l'agent de sélection ULYSSE qui a permis une réduction drastique de la dimension de l'espace des caractéristiques de 68 à environ 24 variables actives. Cette réduction dynamique démontre que l'agent PPO, guidé par la théorie des valeurs de SHAP durant son entraînement, parvient à identifier de manière autonome les vecteurs d'attaque les plus discriminants tout en éliminant le bruit informationnel inhérent aux flux réseau massifs. Sur le plan computationnel, le modèle hybride traite les flux 10,97% plus rapidement que le modèle NEMESIS classique, confirmant que la réduction de dimension allège effectivement la charge de calcul matriciel. Bien que l'architecture sollicite intensivement le processeur avec une charge avoisinant 98 %, l'empreinte mémoire reste particulièrement faible, à environ 0,08 Mo, validant la viabilité du système pour des équipements d'infrastructure aux ressources limitées comme l'IoT. En somme, ces résultats confirment la robustesse d'une architecture offrant une adaptabilité et une transparence inédites, jetant ainsi les bases nécessaires pour envisager l'extension de ce modèle à la détection d'attaques adversariales dans des travaux futurs.

Chapitre 6

Conclusion générale et perspective

Ce mémoire avait pour objectif de concevoir une architecture hybride de détection d'intrusions réseau, visant à concilier les impératifs de haute précision, de rapidité d'exécution et de transparence décisionnelle. Pour faire face à l'augmentation de la dimensionnalité des données réseau modernes, nous proposons deux approches innovantes utilisant l'apprentissage par renforcement profond : une pour la classification et l'autre comme un mécanisme intelligent de sélection dynamique de caractéristiques, toutes deux développées pour améliorer la fiabilité, la précision et la robustesse des mécanismes de détection dans des environnements complexes et déséquilibrés.

La première contribution, NEMESIS [2], propose une intégration structurée entre un agent DQN chargé de la classification binaire du trafic et un classificateur Random Forest chargé de l'identification multiclasses. Cette architecture, fondée sur un pipeline séquentiel, offre une amélioration notable des performances par rapport aux modèles DQN standards. Les expériences menées sur les ensembles de données NSL-KDD et CSE-CICIDS2017 ont révélé une excellente performance de filtrage de la méthode DQN et une classification en classes multiples efficace grâce à l'utilisation de l'algorithme Random Forest. NEMESIS surpasse plusieurs approches de l'état de l'art, notamment en termes de précision et de rappel concernant les attaques, confirmant la pertinence du couplage RL-ML dans un cadre de détection.

Dans une deuxième contribution, ce mémoire introduit ULYSSE, une extension conceptuelle et méthodologique visant à surmonter l'un des défis majeurs des IDS : la dépendance à un ensemble de caractéristiques statiques. ULYSSE repose sur un agent Proximal Policy Optimization capable d'attribuer dynamiquement des scores continus à chaque caractéristique du trafic, en interaction directe avec les valeurs d'importance générées par SHAP et avec les performances du DQN. La fonction de récompense développée constitue l'une des contributions centrales de cette approche, car elle combine performance de classification, cohérence explicative et contrainte de parcimonie. Grâce à ce mécanisme, ULYSSE acquiert la capacité d'adapter en temps réel la dimension de l'espace de représentation du trafic, offrant ainsi une flexibilité et une capacité d'adaptation supérieures à celles des modèles hybrides classiques.

Les travaux expérimentaux menés dans ce mémoire ont permis de valider la méthodologie de NEMESIS et de préparer la phase d'évaluation approfondie d'ULYSSE. Les résultats observés durant l'entraînement, notamment la stabilité des métriques internes du DQN et la cohérence des courbes PPO, démontrent la solidité de l'approche et constituent une base solide pour la mise en œuvre complète d'ULYSSE.

Au-delà des performances obtenues, ce travail contribue également à une meilleure compréhension de l'intérêt des méthodes d'apprentissage par renforcement pour la cybersécurité. Il montre que les approches hybrides RL–ML offrent non seulement un gain de performance,

mais également une plus grande transparence et une adaptabilité appréciable face à la nature non stationnaire des flux réseau modernes.

Plusieurs perspectives se dégagent à l'issue de ce mémoire. Un premier axe réside dans l'évaluation complète et quantitative d'ULYSSE sur différents jeux de données, afin de mesurer précisément les bénéfices de la sélection dynamique de caractéristiques. Un deuxième axe concerne l'exploration de méthodes d'apprentissage adaptatif en ligne (online learning) pour permettre au système de suivre les variations temporelles des attaques. Enfin, l'intégration de défenses robustes contre les attaques adverses constitue un prolongement naturel de ces travaux, afin de renforcer la résilience des IDS basées sur le RL face aux tentatives d'évasion.

En conclusion, ce mémoire propose des contributions méthodologiques et expérimentales significatives dans le domaine de la détection d'intrusions. NEMESIS et ULYSSE démontrent le potentiel des approches hybrides et adaptatives, ouvrant la voie à des systèmes de cybersécurité plus intelligents, explicables et capables d'évoluer au rythme des menaces contemporaines.

Bibliographie

[1] Nektaria Kaloudi and Jingyue Li. The ai-based cyber threat landscape : A survey. *ACM Computing Surveys (CSUR)*, 53(1) :1–34, 2020.

[2] Ahmed Benidir, Hakima Ould-Slimane, and Nadjia Kara. Nemesis : An enhanced hybrid intrusion detection system leveraging deep q-learning and random forest. *Procedia Computer Science*, 272 :202–209, 2025.

[3] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems : techniques, datasets and challenges. *Cybersecurity*, 2(1), December 2019.

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

[5] Nutan Farah Haq, Abdur Rahman Onik, Md. Avishek Khan Hridoy, Musharrat Rafni, Faisal Muhammad Shah, and Dewan Md. Farid. Application of machine learning approaches in intrusion detection system : A survey. *International Journal of Advanced Research in Artificial Intelligence*, 4, 2015.

[6] Saurabh Mukherjee and Neelam Sharma. Intrusion detection using naive bayes classifier with feature reduction. *Procedia Technology*, 4 :119–128, 2012.

[7] Chinmayee Annachhatre, Thomas H Austin, and Mark Stamp. Hidden markov models for malware classification. *Journal of Computer Virology and Hacking Techniques*, 11(2) :59–73, 2015.

[8] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, pages 108–116, 2018.

[9] Amrin Maria Khan Adawadkar and Nilima Kulkarni. Cyber-security and reinforcement learning — a brief survey. *Engineering Applications of Artificial Intelligence*, 114 :105116, 2022.

[10] Guillermo Caminero, Manuel Lopez-Martin, and Belen Carro. Adversarial environment reinforcement learning algorithm for intrusion detection. *Computer Networks*, 159 :96–109, 2019.

[11] Mahmoud Al-Dwairi, Ahmed S Shatnawi, Yaqoub Al-Khassawneh, and Zaid Al-Qudah. Feature selection for malware detection based on reinforcement learning. *IEEE Access*, 11 :12345–12356, 2023.

[12] Thanh Thi Nguyen and Vijay Janapa Reddi. Deep reinforcement learning for cyber security. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8) :3779–3795, 2021.

[13] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning : A brief survey. *IEEE signal processing magazine*, 34(6) :26–38, 2017.

[14] Hooman Alavizadeh, Hootan Alavizadeh, and Julian Jang-Jaccard. Deep q-learning based reinforcement learning approach for network intrusion detection. *Computers*, 11(3) :41, 2022.

[15] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv :1707.06347*, 2017.

[16] Ilhan Firat Kilincer, Fatih Ertam, and Abdulkadir Sengur. Machine learning methods for cyber security intrusion detection : Datasets and comparative study. *Computer Networks*, 188 :107840, 2021.

[17] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote : synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16 :321–357, 2002.

[18] Jianping Zhang and Inderjeet Mani. knn approach to unbalanced data distributions : a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets (ICML)*, volume 126, pages 1–7, 2003.

[19] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn : Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328. IEEE, 2008.

[20] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. IEEE, 2009.

[21] Richard Bellman. *Dynamic programming*. Princeton University Press, 1957.

[22] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar) :1157–1182, 2003.

[23] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3) :379–423, 1948.

[24] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1) :389–422, 2002.

[25] Leo Breiman. Random forests. *Machine learning*, 45(1) :5–32, 2001.

[26] Chris Simmons, Charles Ellis, Sajjan Shiva, Dipankar Dasgupta, and Qishi Wu. Avoidit : A cyber attack taxonomy. *University of Memphis, Technical Report CS-09-003*, 2009.

[27] Foundation OWASP. Owasp top 10-2017 : The ten most critical web application security risks. *OWASP Foundation*, 2017.

[28] Zakir Durumeric, James Kasten, David Adrian, J Alex Halderman, Michael Bailey, Frank Li, Nicolas Weaver, Johanna Amann, Jethro Beekman, Mathias Payer, et al. The matter of heartbleed. In *Proceedings of the 2014 conference on internet measurement conference*, pages 475–488, 2014.

[29] Thomas H Ptacek and Timothy N Newsham. Insertion, evasion, and denial of service : Eluding network intrusion detection. *Secure Networks, Inc.*, 1998.

[30] Blessing Guembe, Ambrose Azeta, Sanjay Misra, Victor Chukwudi Osamor, Luis Fernandez-Sanz, and Vera Pospelova. The emerging threat of ai-driven cyber attacks : A review. *Applied Artificial Intelligence*, 36(1) :2037254, 2022.

[31] Zeeshan Ahmad, Adnan Shahid Khan, Cheah Wai Shiang, Johari Abdullah, and Farhan Ahmad. Network intrusion detection system : A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1) :e4150, 2021.

[32] M. Mughal et al. A comprehensive review of signature-based intrusion detection systems : Challenges and future directions. *Journal of Network Security and Data Mining*, 2024.

[33] S. Hussain et al. Signature-based vs. anomaly-based ids : A performance evaluation in high-speed networks. *International Journal of Information Security*, 22(3) :589–605, 2023.

[34] N. Egwu et al. A comparative study of machine learning algorithms for network intrusion detection using cic-ids2017 dataset. *Journal of Cybersecurity and Information Management*, 2024.

[35] A. J. Neto et al. Deep learning for intrusion detection in emerging technologies : Iot, cloud, and beyond. *IEEE Access*, 12 :10245–10260, 2024.

[36] M. Yoo et al. Recurrent reconstruction network for unsupervised network anomaly detection. *Applied Sciences*, 13(4) :2134, 2023.

[37] M. Al-Zewairi et al. Agile network intrusion detection system based on unsupervised clustering. *Procedia Computer Science*, 171 :2455–2464, 2020.

[38] H. Kotb et al. Generative ai and deep reinforcement learning for robust intrusion detection. *Cybersecurity Review*, 2024.

[39] L. Zhang et al. Interpretable hybrid ids : Combining clustering, random forest and shap for network security. *Sensors*, 24(2) :456, 2024.

[40] Amine Tellache, Amdjed Mokhtari, Abdelaziz Amara Korba, and Yacine Ghamri-Doudane. Multi-agent reinforcement learning-based network intrusion detection system. In *NOMS 2024-2024 IEEE Network Operations and Management Symposium*, pages 1–9. IEEE, 2024.

[41] Malika Malik and Kamaljit Singh Saini. Network intrusion detection system using reinforcement learning techniques. In *2023 International Conference on Circuit Power and Computing Technologies (ICCPCT)*, pages 1642–1649. IEEE, 2023.

[42] Kamalakanta Sethi, E Sai Rupesh, Rahul Kumar, Padmalochan Bera, and Y Venu Madhav. A context-aware robust intrusion detection system : a reinforcement learning-based approach. *International Journal of Information Security*, 19 :657–678, 2020.

[43] Guillermo Caminero, Manuel Lopez-Martin, and Belen Carro. Adversarial environment reinforcement learning algorithm for intrusion detection. *Computer Networks*, 159 :96–109, 2019.

[44] Mashael Alshammari and Others. Efficient grasshopper optimization algorithm for feature selection in intrusion detection system. *Journal of Cybersecurity and Privacy*, 1(1) :11, 2021.

[45] V Kanimozhi and T Prem Jacob. Tidcs : A dynamic intrusion detection and classification system based feature selection in cloud computing. *ICT Express*, 7(2) :190–196, 2021.

[46] Safa Otoum and Ahmad Al-Rubaie. Blockchain-based federated learning for securing iot networks. *Computers & Security*, 136 :103567, 2024.

[47] Abdulmohsen Binbusayyis and Thanjai Vaiyapuri. Id-rdrl : Intrusion detection mechanism based on representation learning and deep reinforcement learning. *Telecommunication Systems*, 80(4) :1–13, 2022.

[48] Isra Al-Turaiki and Nora Al-Twairesh. Reinforcement learning with deep features : A dynamic approach for intrusion detection in iot networks. *IEEE Access*, 11 :56789–56800, 2023.

[49] S. Mohammadi and Others. Mafsids : Multi-agent feature selection for intrusion detection system. *Expert Systems with Applications*, 215 :119312, 2023.

[50] H. Jiang, Z. He, and X. Ye. Reinforcement learning-based voting for feature drift-aware intrusion detection : An incremental learning framework. *Future Generation Computer Systems*, 150 :200–212, 2024.

[51] Emad E Abdallah, Ahmed Fawzi Otoom, et al. Intrusion detection systems using supervised machine learning techniques : a survey. *Procedia Computer Science*, 201 :205–212, 2022.

[52] Widad K Mohammed, Mohammed A Taha, and Saleh M Mohammed. A novel hybrid fusion model for intrusion detection systems using benchmark checklist comparisons. *Mesopotamian Journal of CyberSecurity*, 4(3) :216–232, 2024.

[53] Guillaume Lemaître, Fernando Nogueira, and ChristosK Aridas. Imbalanced-learn : A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17) :1–5, 2017.

[54] Richard S Sutton and Andrew G Barto. *Reinforcement learning : An introduction*. MIT press, 2018.

[55] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations : Theory and application to reward shaping. In *In Proceedings of the Sixteenth International Conference on Machine Learning*, pages 278–287. Morgan Kaufmann, 1999.

[56] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, volume 30, 2017.

[57] Xia Liu, Jianxun Yin, Xiaocheng Feng, et al. Automated feature selection : A reinforcement learning perspective. *IEEE Transactions on Knowledge and Data Engineering*, 33(11) :3472–3486, 2020.

[58] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540) :529–533, 2015.

[59] Romaric Gaudel and Michele Sebag. Feature selection as a one-player game. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 359–366, 2010.

[60] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[61] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv :1409.0473*, 2014.

[62] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[63] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell : Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[64] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2) :273–324, 1997.

Annexe A

Article NEMESIS : Publication originale

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 272 (2025) 202–209

Procedia Computer Science

www.elsevier.com/locate/procedia

The 16th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2025)
October 28-30, 2025, Istanbul, Türkiye

NEMESIS: An Enhanced Hybrid Intrusion Detection System Leveraging Deep Q-Learning and Random Forest

Ahmed BENIDIR[a,*], Hakima OULD-SLIMANE[a], Nadjia KARA[b]

[a]*Dept. of Math. & Comp. Sci., Univ. of Quebec at Trois-Rivières, 3351 Blvd des Forges, Trois-Rivières, QC G8Z 4M3, Canada*
[b]*Dept. of Software Eng. & IT, ÉTS (École de technologie supérieure), 1100 Notre-Dame W., Montreal, QC H3C 1K3, Canada*

Abstract

Network intrusion detection systems (NIDS) face the growing difficulty posed by increasingly sophisticated and unseen attacks, which represent a dangerous threat due to their ability to exploit vulnerabilities that have not yet been identified. These attacks are inherently difficult to detect with conventional NIDS because such systems typically are built on known threat patterns or signatures, which are absent in unseen scenarios. Consequently, this greatly limits their efficacy in mitigating advanced threats, making networks susceptible to potential security breaches. To address these challenges, recent years have witnessed the emergence of various Reinforcement Learning (RL) approaches aimed at enhancing the automatic detection of network intrusions. These systems are equipped with autonomous agents that acquire the ability to learn independently and make decisions without requiring direct input or knowledge of human experts. In this paper, we propose a network intrusion detection mechanism that integrates a Deep Q Network-based model (DQN) with a supervised machine learning algorithm specifically designed for attack classification. Our model is characterized by meticulous fine-tuning of hyperparameters to optimize the performance of detection. Extensive experimental evaluations that take advantage of the NSL-KDD and CSE-CICIDS2017 datasets demonstrate that our hybrid approach significantly improves detection accuracy across various types of attack and outperforms other existing state-of-the-art solutions designed for similar purposes.

Keywords: Network Intrusion Detection; Reinforcement Learning; Supervised learning; CSE-CICIDS2017; NSL-KDD;

1. Introduction

Network intrusion detection systems (IDS) are a core technology in cyber defense Kilincer et al. [11], Venter and Eloff [20], classifying network traffic as benign or malicious. Signature-based IDS (SIDS) efficiently detects known

* Corresponding author. Tel.: +1-514-216-1109.
E-mail address: ahmed.benidir@uqtr.ca

1877-0509 © 2025 The Authors. Published by Elsevier B.V.

10.1016/j.procs.2025.10.197

attacks with low false positives but fails against novel or polymorphic threats. Anomaly-based IDS (AIDS), often based on machine learning, address this limitation by modeling normal traffic behavior [10]. However, such models, including CNNs, remain prone to high false positive rates and misclassifications and require retraining when new attacks emerge.

Reinforcement learning (RL) offers an alternative by enabling adaptive agents capable of self-learning and identifying unseen intrusions without supervision [17]. Among RL methods, Q-learning is widely used [1], but its scalability issues in large state spaces have motivated the adoption of Deep Q-Networks (DQN) [15]. Several studies [13, 2] have explored DQN for intrusion detection, but most emphasize detection accuracy in outdated datasets such as NSL-KDD, without detailed implementation strategies or hyperparameter tuning.

Unlike prior hybrid approaches, our proposed NEMESIS framework explicitly separates detection tasks into two sequential steps: a DQN agent rapidly filters benign traffic, while a Random Forest (RF) classifier analyzes only potentially malicious samples. This design reduces computational overhead and enhances multiclass detection accuracy by focusing supervised learning on nonbenign traffic.

The main contributions are as follows.

- A hybrid IDS that combines a DQN agent with a Random Forest classifier.
- A detailed DQN architecture specification.
- Experimental validation in NSL-KDD and CSE-CICIDS2017 datasets.

The remainder of this paper is organized as follows. Section 2 reviews related works; Section 3 presents background concepts; Section 4 details NEMESIS; Section 5 discusses results; and Section 6 concludes the article.

2. Related Work

Several studies have combined deep reinforcement learning (DRL), notably Deep Q-Networks (DQNs), with supervised models for network intrusion detection. For example, [19, 13] use a DQN agent for initial filtering, followed by a supervised classifier for refined detection. A context-sensitive DQN-based IDS was proposed in [17], tested on NSL-KDD, AWID, and UNSW-NB15, showing that ensembles of supervised models can further improve performance. Similarly, [2] introduced a DQN with self-learning and hyperparameter tuning, achieving competitive multiclass detection in NSL-KDD. The work of [5] evaluated an adversarial DQN model on NSL-KDD and AWID, reaching an accuracy close to traditional SVM classifiers.

In general, these approaches demonstrate the potential of hybrid DQN–supervised IDSs, but often without a clear separation of roles between modules, leading to computational overhead and limited adaptability in dynamic environments.

3. Background

3.1. Network-based intrusion detection system

Intrusions threaten the confidentiality, integrity, and availability of information systems. Network-based IDS (NIDS) monitor traffic and complement firewalls by detecting threats beyond signature rules [12]. They are mainly divided into signature-based IDS (SIDS), effective for known attacks but blind to novel threats, and anomaly-based IDS (AIDS), which rely on machine learning or statistical models to detect deviations from normal traffic [10]. Both approaches face challenges such as false positives, evolving attack strategies, and frequent model updates.

3.2. Supervised learning in IDS

Supervised IDS train classifiers on labeled data after preprocessing and feature selection [10]. Algorithms such as decision trees, SVM, KNN, neural networks, or random forests predict whether new traffic is benign or malicious. The main challenge is to achieve strong generalization to unseen attacks while minimizing false alarms.

3.3. Reinforcement learning

Reinforcement learning (RL) is a reward-based paradigm where an agent interacts with an environment and learns policies that map states to actions to maximize cumulative rewards. RL problems are typically modeled as Markov Decision Processes (MDPs) defined by states, actions, rewards, agent, and environment.

Q-Learning

Q-learning is a widely used RL algorithm [15], where the agent learns the expected reward for each state–action pair (Eq. 1). However, it struggles with scalability in high-dimensional spaces, motivating the use of deep neural approximations.

$$\begin{aligned} Q(s,a) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0{=}s, a_0{=}a\right], \\ Q(s,a) &\leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right). \end{aligned} \tag{1}$$

Deep Q-Network

Deep Q-Networks (DQNs) extend Q-learning by approximating the Q-function with neural networks, enabling learning in large continuous state spaces [14]. In intrusion detection, DQNs process network traffic features as input and output action values that guide the agent's decisions. Figure 1 illustrates the agent–environment interaction.

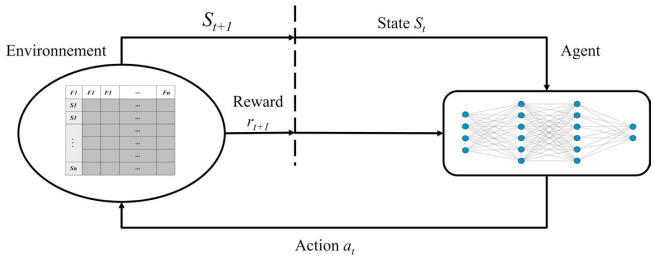


Fig. 1. DQN model based on agent–environment interaction [2]

4. Our proposed Approach: NEMESIS

In the following subsection, we describe the NEMESIS approach, including the DQN model and the datasets used.

Although DQN models alone have shown promising results in network intrusion detection tasks, they typically focus on binary classification (benign vs. attack) and are often limited in their ability to differentiate between multiple types of attacks, as demonstrated in [2] by testing a DQN agent on the NSL-KDD dataset. Our hybrid approach, NEMESIS, addresses this limitation by introducing a two-stage decision process.

1. **Stage 1 – DQN Agent (Binary Classification):** The DQN agent acts as a lightweight and fast filter, determining whether the incoming network traffic is benign or potentially malicious. This reduces the computational load and limits the scope of more complex analysis to relevant samples only.
2. **Stage 2 – Random Forest Classifier (Multiclass Classification):** When the DQN agent flags a sample as non benign, the Random Forest classifier is then activated to perform a fine-grained multiclass classification, identifying the exact attack type (up to 15 classes in CICIDS2017).

This architectural separation brings several benefits.

- Improved precision in multiclass detection: The RF classifier excels at handling unbalanced multiclass data. Benefits from focusing on only suspected attacks, avoiding dilution from benign samples.
- Efficient learning feedback loop: The RF classifier provides precise labels to the DQN agent, which are used to calculate more accurate rewards during training. This tight reward feedback loop improves the agent's learning stability and convergence.
- Better generalization across datasets: The hybrid model shows stronger generalization capacity when tested on both the NSL-KDD and CICIDS2017 datasets, outperforming standalone DQN models in accuracy and recall, particularly for rare attack types.

In contrast, standalone DQN models often struggle with:

- High false positive rates in multiclass settings.
- Slow convergence due to sparse and less informative rewards.
- Inflexibility in handling non-stationary attack distributions or class imbalance.

The choice to combine a DQN agent with a Random Forest classifier is not a simple intuition, but rather a structured experimental approach. First, we evaluated different supervised models (SVM, KNN, Decision Tree and RF) on our datasets, taking into account the criteria of accuracy, robustness to class imbalance, and computation time. Random Forest stood out for several reasons: It is known for its ability to efficiently handle unbalanced multiclass data [4], tolerates noise well and exhibits good generalization, while remaining fast to train and infer. Our own comparative tests showed that RF offered the best performance/speed compromise in our data.

NEMESIS comprises a DQN agent with a variety of components and a supervised classifier. The primary function of this classifier is to assign the *state_vector* to its corresponding class. This classification process plays a crucial role by providing accurate feedback to the environment through correctly labeled classes, thereby facilitating a mechanism for rewarding or penalizing the agent. The ultimate goal is for the agent to refine its learning process to minimize instances of misclassification. The details concerning the various DQN components are described in the following sections.

1. **Environment** The environment provides the agent with states and rewards. Here, it is built from the NSL-KDD and CSE-CICIDS2017 datasets after preprocessing, normalization, and resampling. The features (40 for NSL-KDD, 68 for CICIDS2017) form the state space, while labels are used to compute the rewards.
2. **Agent** The agent is a deep Q-Network (DQN) that learns a value-based policy by interacting with the environment. Observes states, selects actions, and receives rewards. Training starts with exploration (e.g. ϵ-greedy), then gradually shifts toward greedy action selection as the policy improves.
3. **States** States are feature vectors provided by the environment (40 or 68 features depending on the dataset). These vectors represent network traffic and are the input to the DQN during training and evaluation.
4. **Actions** Actions are agent decisions, derived from Q-values on the state vector. The agent evaluates these outputs against thresholds to classify the traffic as benign or malicious.
5. **Rewards** Rewards are feedback signals: positive for correct classifications and negative otherwise. Their values depend on the true labels and prediction probabilities, guiding the agent to improve detection accuracy.

4.1. Datasets

We used two benchmark datasets for intrusion detection: *NSL-KDD* and *CSE-CICIDS2017*. NSL-KDD is an improved version of KDD'99, with redundant records removed to avoid biased learning. It contains 41 features per connection and labels: Normal, DoS, Probe, R2L, and U2R. CICIDS2017 provides richer traffic with 79 features labeled as *Benign* or various *Attack types* [8, 16]. The preprocessing included removal of duplicates, NaN and infinite values, label encoding of categorical attributes [2], Min–Max scaling, and binary relabeling (Benign vs. Attack) for the DQN experiments. The splits were 70/15/15 for DQN and 80/20 for Random Forest. Both data sets are highly imbalanced.

Table 1. Agent and neural network parameters

Parameters	Description	Values
num_timesteps	Number of timesteps to train DQN	300k/400k
num_iter	Number of iteration to improve Q-values in DQN	4
hidden_layers	Number of hidden layers: Setting weights, producing outputs, based on activation function	3
num_units	Number of hidden unit to improve the accuracy of prediction and training	128, 64, 32
activation function	Non-linear activation function	ReLU
gamma γ	Discount factor for target prediction	0.3
Learning rate	The learning rate controls the size of weight updates	Dynamic
epsilon ϵ	Degree of randomness for performing actions	Dynamic
batch-size (bs)	A batch of dataset's records fetched for processing	16

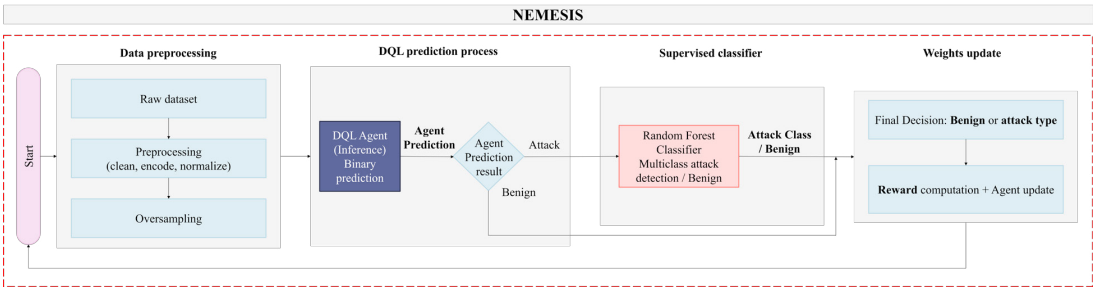


Fig. 2. NEMESIS architecture

To improve rare attack detection, we applied oversampling only in training / validation: *SMOTE* for CICIDS2017 [6] and *RandomOverSampler* for NSL-KDD. This choice balances classes without biasing the test set [7, 18]. After balancing, Random Forest training used equalized class distributions (1,816,702 samples/class for CICIDS2017; 67,343/class for NSL-KDD), while DQN was trained on binary data.

The DQN agent architecture was empirically tuned for accuracy and efficiency. We used a three-layer MLP (128–64–32) with ReLU activations, batch size 16, and a dynamic learning rate ensuring stable convergence. The discount factor was set to $\gamma = 0.3$, and the training was carried out in 300k (CICIDS2017) and 400k (NSL-KDD) time steps. The binary action space (benign / suspicious) and the reward function emphasize accurate detection while minimizing false alarms, enabling adaptation to evolving threats. The final hyperparameters are summarized in Table 1.

Training follows a ϵ-greedy policy [3, 15], where actions are selected, rewards calculated, and Q-values are updated iteratively. During inference, the agent first performs binary classification: benign traffic is discarded, while suspicious samples are forwarded to the Random Forest for multiclass identification. This two-stage process reduces computational cost and improves detection accuracy. The workflow is shown in Figure 4.1, and the procedures are described in Algorithms 1 and 2.

5. Experimental results

Experiments were carried out on a system with 105GB RAM, a 10-core Intel Xeon Platinum 8358 CPU (2.60 GHz) and an NVIDIA A100 GPU, running Ubuntu. The implementation used Python 3.11.7 and Stable-Baselines3 v2.3.2. The results were evaluated using standard machine learning metrics, and the evaluation metrics include accuracy, precision, recall, and F1-score.

Algorithm 1: Training of Deep Q-Learning Agent in Custom Environment

Input : Preprocessed dataset D, DQN parameters (episodes, γ, ϵ, batch size, etc.)
Output: Trained agent policy for binary intrusion detection

```
Normalize(D)
Initialize agent parameters
batch_size ← 16
States ← sample mini-batches from D
Initialize model
for timestep ← 1 to T do
    Reset environment state
    for iter ← 1 to num_iterations do
        // Step 1: Select action
        for i ← 1 to batch_size do
            if with probability ϵ then
                A_i ← random action from action space
            else
                Q_i ← model.predict(state_i)
                A_i ← arg max(Q_i)
            end
        end
        ϵ ← ϵ × decay_rate
        // Step 2: Compute rewards
        for i ← 1 to batch_size do
            R_i ← compute reward(A_i, true label)
        end
        // Step 3: Q-value update
        Q' ← model.predict(next_state)
        for i ← 1 to batch_size do
            Q_target[i] ← R_i + γ · max(Q'[i])
        end
        // Step 4: Train model
        model.train(state, Q_target)
        Update state ← next_state
    end
end
```

Algorithm 2: Hybrid Inference: DQN (binary) + RF (attack class)

Input : Test set $\{(x_i, y_i)\}_{i=1}^{n}$, trained `DQN`, trained `RF`
Output: Final predictions {benign or attack class} and DQN reward updates

```
for i ← 1 to n do
    x ← preprocess_sample(x_i)
    state ← extract_state_vector(x)
    binary_pred ← DQN.predict(state)   // 0 = benign, 1 = attack
    if binary_pred = 0 then
        final_pred ← "benign"
    else
        attack_class ← RF.predict(state)
        final_pred ← attack_class
    end
    true_label ← y_i
    if binary_pred = 0 and true_label = "benign" then
        reward ← +2        // True negative
    else if binary_pred = 1 and true_label ≠ "benign" then
        if final_pred = true_label then
            reward ← +5          // Correct classification
        else
            reward ← −2          // Attack detected, wrong type
    else
        reward ← −5 // False positive or false negative
    end
    DQN.update(state, reward)
end
```

5.1. Performances: individual classifiers

The classifiers (DQN and RF) are evaluated separately and both are compared to similar models in the state of the art.
The prediction of the DQN model and the results of the random forest classification are presented in table 2.

The experimental evaluation highlights the effectiveness of NEMESIS for both binary and multiclass intrusion detection. The standalone DQN agent shows strong recall (97.3% on NSL-KDD), capturing nearly all attacks but with moderate precision due to false positives. When combined with the RF classifier, the multiclass identification

Table 2. DQN Agent and RF Classifier Results

Dataset	Model	Accuracy	Precision	Recall	F1-score
CICIDS17	Agent	95.04%	96.64%	93.32%	94.95%
	RF	98.82%	99.44%	98.82%	99.09%
NSLKDD	Agent	91.18%	90.24%	97.34%	93.66%
	RF	76.60%	81.33%	76.66%	74.51%

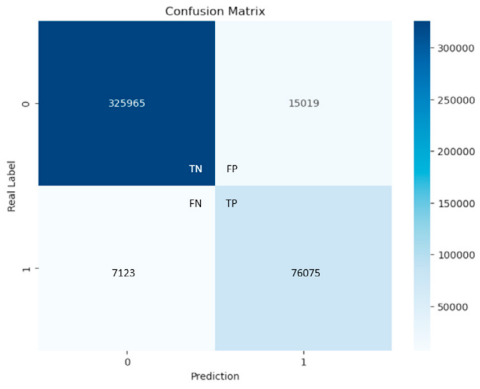


Fig. 3. DQN confusion matrix for CSE-CICIDS2017

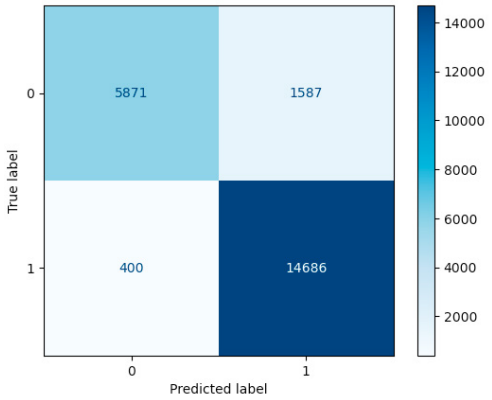


Fig. 4. DQN confusion matrix for NSL-KDD

Table 3. Comparison of IDS approaches

Work	ML Techniques	Adaptive-Learning	Dataset	Accuracy
[2]	DQN	✓	NSL-KDD	78%
[17]	DQN + RF	✓	NSL-KDD	77%
[5]	DQN	✓	NSL-KDD	74%
[13]	DQN	✓	CICIDS2017	97%
[19]	DQN	✓	CICIDS2017	99%
Our approach	DQN + RF	✓	NSL-KDD	91%
			CICIDS2017	95%

improves significantly, achieving 98.8% accuracy on CICIDS2017, while the lower performance of NSL-KDD reflects its limited and outdated features.

The ablation study confirms this complementarity: DQN alone is effective for real-time detection (94.2% on CICIDS2017) but cannot differentiate attack types, while RF alone can classify multiple classes (98.8% on CICIDS2017) but suffers from imbalance and reduced precision, especially on NSL-KDD. The hybrid NEMESIS design takes advantage of both strengths: DQN filters benign traffic, and RF focuses on harder multiclass cases, achieving the best overall F1 score and accuracy across benchmarks.

Confusion matrices (Figures 3–4) show that most benign and attack samples are correctly detected, with errors mainly from rare attack types. Comparative results (Table 3) further demonstrate that NEMESIS outperforms classical models (SVM, Naive Bayes, CNN-BiLSTM) and prior DQN approaches on NSL-KDD (91% accuracy) [2, 9], and achieves competitive performance on CICIDS2017 (94.2%), offering better multiclass differentiation and adaptability than RL-only or supervised methods [19, 13].

6. Conclusion

In this paper, we proposed NEMESIS, a hybrid intrusion detection approach that combines deep Q-network reinforcement learning with a supervised random forest classifier. Our method efficiently detects and classifies network

intrusions, including unseen attack types, leveraging DQN for initial binary filtering and RF for multiclass attack identification. We detailed the core architectural components, training strategies, and hyperparameter selection. The experimental results show that NEMESIS outperforms most existing methods in benchmark data sets. Future work will focus on extending NEMESIS to counter adversarial and distributed attack scenarios.

References

[1] Adawadkar, A.M.K., Kulkarni, N., 2022. Cyber-security and reinforcement learning—a brief survey. Engineering Applications of Artificial Intelligence 114, 105116.

[2] Alavizadeh, H., Alavizadeh, H., Jang-Jaccard, J., 2022. Deep q-learning based reinforcement learning approach for network intrusion detection. Computers 11, 41.

[3] Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A., 2017. Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine 34, 26–38.

[4] Breiman, L., 2001. Random forests. Machine learning 45, 5–32.

[5] Caminero, G., Lopez-Martin, M., Carro, B., 2019. Adversarial environment reinforcement learning algorithm for intrusion detection. Computer Networks 159, 96–109.

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

[7] Dhanabal, L., Shantharajah, S., 2015. A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. International journal of advanced research in computer and communication engineering 4, 446–452.

[8] Haroon, M., Ali, H., 2023. Ensemble adversarial training based defense against adversarial attacks for machine learning-based intrusion detection system. Neural Network World 317, 336.

[9] Jiang, K., Wang, W., Wang, A., Wu, H., 2020. La détection d'intrusion dans le réseau combine l'échantillonnage hybride avec un réseau hiérarchique profond. IEEE access 8, 32464–32476.

[10] Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J., 2019. Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity 2, 1–22.

[11] Kilincer, I.F., Ertam, F., Sengur, A., 2021. Machine learning methods for cyber security intrusion detection: Datasets and comparative study. Computer Networks 188, 107840.

[12] Liao, H.J., Lin, C.H.R., Lin, Y.C., Tung, K.Y., 2013. Intrusion detection system: A comprehensive review. Journal of Network and Computer Applications 36, 16–24.

[13] Malik, M., Saini, K.S., 2023. Network intrusion detection system using reinforcement learning techniques, in: 2023 International Conference on Circuit Power and Computing Technologies (ICCPCT), IEEE. pp. 1642–1649.

[14] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al., 2015. Human-level control through deep reinforcement learning. nature 518, 529–533.

[15] Nguyen, T.T., Reddi, V.J., 2021. Deep reinforcement learning for cyber security. IEEE Transactions on Neural Networks and Learning Systems 34, 3779–3795.

[16] Phulre, A.K., Jain, S., Jain, G., 2024. Evaluating security enhancement through machine learning approaches for anomaly based intrusion detection systems, in: 2024 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), IEEE. pp. 1–5.

[17] Sethi, K., Sai Rupesh, E., Kumar, R., Bera, P., Venu Madhav, Y., 2020. A context-aware robust intrusion detection system: a reinforcement learning-based approach. International Journal of Information Security 19, 657–678.

[18] Tan, X., Su, S., Huang, Z., Guo, X., Zuo, Z., Sun, X., Li, L., 2019. Wireless sensor networks intrusion detection based on smote and the random forest algorithm. Sensors 19, 203.

[19] Tellache, A., Mokhtari, A., Korba, A.A., Ghamri-Doudane, Y., 2024. Multi-agent reinforcement learning-based network intrusion detection system, in: NOMS 2024-2024 IEEE Network Operations and Management Symposium, IEEE. pp. 1–9.

[20] Venter, H., Eloff, J.H., 2003. A taxonomy for information security technologies. Computers & Security 22, 299–307.