UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE
APPLIQUÉES

PAR
BERDANE ILHEM

Renforcement des Systèmes de Détection d'Intrusions : Approche basée sur les techniques Avancées d'Apprentissage Automatique

Mars 2025

Résumé

Les systèmes de détection d'intrusion (Intrusion Detection System IDS) sont des éléments vitaux des cadres d'applications de cybersécurité actuelle dans le monde, conçues pour contrôler les trafics réseau pour détecter les activités malfaisantes. Grâce à l'apprentissage automatique, les IDS ont la capacité d'analyser un énorme volume de données pour identifier les anomalies et les menaces. Cependant, ces systèmes souffrent de certaines lacunes, notamment la fragilité aux attaques contradictoires et les distributions déséquilibrées des données. Les attaques contradictoires ajoutent des perturbations subtiles qui induisent les modèles en erreur, réduisant de ce fait leur efficacité à détecter précisément ces intrusions. En outre, les données déséquilibrées, qui sont typiques dans les problèmes de détection d'intrusion, limitent la capacité des modèles à identifier les menaces rares, mais critiques. Dans ce projet, nous avons travaillé à l'aide du jeu de données CIC IDS 2018, un jeu de données complet et connu de la communauté de référence pour évaluer la performance des IDS. Nous avons abordé ces défis en utilisant une méthodologie sophistiquée combinant des techniques d'apprentissage automatique modernes. Après avoir prétraité et équilibré les données à l'aide de la méthodologie SMOTE, nous avons testé plusieurs algorithmes, y compris la forêt aléatoire, arbre de décision, les réseaux convolutifs, les réseaux à mémoire à long terme, et des réseaux siamois combinant CNN et LSTM. Pour augmenter la robustesse des modèles, nous avons utilisé la méthode FGSM pour dériver les exemples contradictoires utilisés pour entraîner et tester les modèles pour leur résilience face à ces adversités. Nous avons observé que, bien que les exemples classiques, notamment forêt aléatoire et arbre de décision, soient généralement stables, les modèles avancés, en particulier les réseaux siamois combinant CNN et LSTM, affichent des performances de plus de 98.9 %, ce qui indique l'efficience des combinaisons dans l'apprentissage de modèles complexes sans compromettre leur robustesse face à des adversités.

Abstract

Modern cybersecurity frameworks for networks around the world include the use of Intrusion Detection Systems (IDS) to monitor network traffic and identify malicious activity. By using machine learning, the IDS analyzes large amounts of data for detecting anomalies and threats. However, such systems are highly challenged, particularly as they are subject to adversarial attacks and imbalanced data distributions. Adversarial attacks add small perturbations to deceive models, thus compromising their efficiency in accurately detecting intrusions. Also, in intrusion detection problems, the data are usually imbalanced, which makes any system trained on such data fail to detect a rare but important attack.

In our study, we used the CIC IDS 2018 dataset, which is a known and well-acknowledged benchmark in the reference community for intrusion detection system evaluation. We mitigated these issues by applying an advanced approach that integrated contemporary machine learning methods. Hence, after preprocessing and balancing the data using the SMOTE methodology, we tested various algorithms such as Random Forest, Decision Tree, CNNs, LSTM neural networks, and Siamese Networks (a combination of CNNs and LSTMs). We used the FGSM method to create adversarial examples and then trained/tested the models for robustness against adversities.

We found that basic models like Random Forest and Decision Tree provide stable results, but more complex models like Siamese Networks, combining CNN and LSTM, show results over 98.9% performance. This demonstrates the advantage of such combinations in achieving robustness under adversarial conditions while learning to do so quite effectively.

.

Remerciements

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui m'ont aidé tout au long de ce travail. Mes sincères remerciements vont en premier lieu à ma directrice de recherche OULD SLIMANE HAKIMA, pour son précieuse aide, leur soutien et leurs conseils avisés qui m'ont permis de mener à bien ce travail de recherche.

Je n'oublie pas mes amis et ma famille pour leur soutien indéfectible tout au long de mes études et de la rédaction de ce mémoire. Je suis particulièrement reconnaissante envers mes parents : ma mère NASSIMA FETTAL et mon père BERDANE SIDALI, qui m'ont toujours encouragé à poursuivre mes rêves et à ne jamais abandonner.

Je tiens aussi à remercier chaleureusement HAMZA MEKID et IMENE HENNI MANSOUR pour leur précieuse aide et leurs conseils tout au long de mes expérimentations. Leurs connaissances techniques et leur disponibilité ont grandement contribué à la réussite de mon mémoire. Je suis reconnaissante de leur soutien et de leur présence.

Mes remerciements s'adressent également à tous les enseignants et le personnel administratif de l'université du Québec à Trois-Rivières pour leur aide et leur disponibilité tout au long de mes études.

Table des matières

Contents

Liste des Abréviations

IDS : Intrusion Detection System

NIDS: Network Intrusion Detection System

HIDS: Host Intrusion Detection System

TCP: Transmission Control Protocol

ML: Machine Learning

CNN: Convolutionnel Neural Network

FGSM: Fast Gradient Sign Method

RNN: Recurrent Neural Network

ANN: Artificial Neural Network

LSTM: Long short term memory

Liste des Tables

Liste des Figures

Chapitre 1 : INTRODUCTION

1.1 Introduction

Avec le développement d'Internet, les technologies de l'information et de la communication nous ont dotés de fonctionnalités essentielles, telles que l'apprentissage à distance, l'achat et le paiement en ligne, la messagerie instantanée et la vidéoconférence, ainsi que l'émergence des distributeurs automatiques, des voitures autonomes et de nombreuses autres téchnologies . Cependant, à mesure que ces outils numériques inondent le marché, de nouvelles vulnérabilités en matière de sécurité sont apparues. Les réseaux et systèmes d'information interconnectés d'aujourd'hui sont également exposés à de réelles menaces ou incidents. Il semble que chaque jour un nouvel article soit publié sur les défis de la cybersécurité : les intrusions dans la vie privée par les réseaux sociaux, la fraude par carte de crédit, l'espionnage économique, les attaques d'infection de systèmes informatiques provoquées par un déni de service et bien d'autres cybermenaces causeront des problèmes majeurs dans les années à venir. Cette réalité se confirme en 2024, où le nombre de cyberattaques a considérablement augmenté par rapport à l'année précédente : en moyenne, plus de 1 600 attaques par semaine contre des entreprises dans le monde. Cela représente une augmentation de 30%.

La cybersécurité est la combinaison de technologies et de pratiques visant à assurer la sécurité de tous les aspects liés aux technologies de l'information, telle que l'accès, le stockage et le traitement des données, des informations, ainsi que la protection, la transmission et les connexions de ces dernières. Également connu sous le nom de sécurité de l'information. Pour répondre à ces exigences, des systèmes de détection d'intrusion ont été développés. En outre, il vise à prévenir toute menace directe à la sécurité politique, au réseau et aux systèmes de traitement de l'opération, telle que le refus d'itinéraire. C'est le cœur de la technologie que tous les outils de cybersécurité devraient utiliser. Le module d'identification d'intrusion analyse les données collectées jusqu'à présent sur l'objet pour déterminer s'il s'agit d'une menace ou d'un flux légitime. Il est important de savoir que la sophistication toujours croissante des cyberattaques en termes de temps et de densité, ainsi que de fréquence et de vitesse, ainsi que l'observation des méthodes traditionnelles avec un regard non fondé, a conduit à l'utilisation de nouvelles technologies de protection avancées.

En conséquence, les entreprises de sécurité ciblées choisissent de déployer l'apprentissage automatique (Machine Learning) dans leurs nouveaux produits. Après tout, c'est une branche de l'intelligence artificielle (IA) qui a été utilisée avec succès pour la reconnaissance d'images, la

recherche et la prise de décision, et il est tout à fait juste d'utiliser cette technologie unique pour résoudre les problèmes qui affectent tout outil de détection de cyberattaques. Qu'il s'agisse d'une intrusion ou d'une cyberattaque maliciel, même pour la sécurité des infrastructures critiques, telles que la sécurité électrique, les systèmes de contrôle industriel, la surveillance des sites énergétiques.

L'apprentissage profond est une sous-discipline de l'apprentissage automatique très prometteuse en matière de cybersécurité, surtout maintenant qu'une grande quantité de données proviennent du cyberespace. Les méthodes classiques d'apprentissage automatique reposent sur l'ingénierie et la sélection de caractéristiques du domaine de recherche, dont la mise en œuvre peut nécessiter une certaine expertise. Dans cette étude, nous discuterons conjointement de plusieurs méthodes de l'apprentissage automatique et profond testées dans le domaine de la détection d'intrusion, qui posent un nouveau défi : la sécurité des réseaux.

1.2 Problématique

Malgré les avancées significatives dans les techniques de détection des intrusions basées sur l'apprentissage automatique, ces systèmes restent vulnérables aux attaques contradictoires. Ces attaques exploitent les failles des modèles en introduisant des perturbations subtiles, mais efficaces, conduisant à des erreurs de classification et compromettant ainsi leur capacité à détecter correctement les activités malveillantes.

Un autre défi majeur réside dans le déséquilibre des ensembles de données utilisés pour entraîner ces modèles. Ce déséquilibre peut fortement impacter leur performance, notamment en réduisant leur capacité à détecter les intrusions rares mais critiques. En conséquence, le taux de faux positifs et de faux négatifs peut être significativement élevé, limitant l'efficacité des systèmes de détection et augmentant les risques d'intrusions non détectées ou d'alertes excessives.

Dans ce contexte, il est essentiel de développer des approches permettant d'améliorer la robustesse des modèles face aux attaques contradictoires tout en optimisant leur capacité à gérer des ensembles de données déséquilibrés.

1.3 Objectif

L'objectif de cette étude est de développer des approches permettant d'améliorer la robustesse des systèmes de détection d'intrusion basés sur l'apprentissage automatique et profond face aux attaques contradictoires. Pour cela, nous visons à concevoir des modèles capables de résister aux perturbations plus précisément, il s'agit de :

- **Renforcer la résilience des modèles** face aux attaques contradictoires en intégrant des techniques de défense adaptées.
- **Améliorer la détection des intrusions** en appliquant des stratégies de rééquilibrage des données, telles que le sur-échantillonnage et le sous-échantillonnage.
- **Réduire le taux de faux positifs et de faux négatifs** afin d'augmenter la fiabilité du système et minimiser les risques liés aux intrusions non détectées ou aux alertes excessives.

En combinant ces approches, cette recherche vise à proposer une solution plus robuste et efficace pour renforcer la sécurité des systèmes d'information contre les attaques.

1.4 Organisation du mémoire

Ce mémoire se compose de cinq chapitres. Le premier chapitre est consacré à l'introduction générale. Nous dévoilerons également à travers ce chapitre nos problématiques et objectifs de recherche.

Le deuxième chapitre comprend des concepts préliminaires reliés à notre contexte.

Le troisième chapitre porte sur l'état de l'art. Nous débuterons par une brève introduction pour rappeler le contexte, nous discuterons de plusieurs études qui ont été faites sur la détection d'intrusion, ainsi que les jeux de données utilisés dans ce contexte. Nous exposerons par la suite des solutions existantes et leurs limitations. Nous terminerons ce chapitre par une conclusion.

Dans le quatrième chapitre, nous discuterons la méthodologie de recherche suivie dans cette étude, nous débuterons par les questions de recherche que nous avons établies. Nous présenterons le choix des jeux de données lors des expérimentations. Nous allons examiner en détail comment le modèle a été créé, c'est-à-dire le prétraitement des données, les différents algorithmes d'apprentissage automatique et profond utilisés dans le modèle original, et nous présenterons aussi les résultats et en discuterons.

Le cinquième chapitre achève par une conclusion et des perspectives.

Chapitre 2 : Concepts préliminaires

2.1 Introduction

La cybersécurité est devenue un enjeu majeur de la vie d'aujourd'hui : les systèmes d'information sont confrontés à des menaces de plus en plus sophistiquées. Par conséquent, afin de prévenir et de détecter toutes ces attaques, il est important de disposer d'un système d'alerte efficace. Dans un système, les détecteurs d'intrusion (IDS) jouent un rôle clé dans la surveillance des réseaux et des hôtes à la recherche d'activités malveillantes ou anormales.

Dans le domaine des IDS, avec le développement de l'apprentissage automatique, ces systèmes ont évolué pour être plus adaptatifs et prédictifs. Contrairement aux IDS traditionnels qui s'appuient sur des règles statiques, les IDS basés sur l'apprentissage automatique sont capables de détecter les attaques inconnues en étudiant les caractéristiques des comportements normaux et anormaux fournies par les données passées. Ces IDS autonomes utilisent un ensemble complet d'algorithmes tel que : arbres de décision, forêts aléatoires, réseaux de neurones, pour une meilleure détection et protection.

Cependant, pour garantir l'efficacité de ces modèles, il est nécessaire d'envisager des stratégies permettant d'améliorer les performances et la robustesse des modèles d'apprentissage automatique. Cela inclut des techniques telles que l'équilibrage des données pour résoudre le problème des classes déséquilibrées, ainsi que des mécanismes de défense contre les attaques contradictoires conçus pour rendre les modèles plus résistants au sabotage.

Dans ce chapitre, nous explorerons les mécanismes de détection pour la sécurité des systèmes, les IDS basés sur l'apprentissage automatique et différentes stratégies pour améliorer la robustesse et les performances des modèles d'apprentissage automatique utilisés dans ce contexte.

2.2 Mécanismes de détection en cyber sécurité

La cybersécurité est devenue un enjeu essentiel pour la protection des systèmes informatiques contre les menaces et les attaques. Pour garantir la sécurité des réseaux et des données, plusieurs méthodes et outils sont mis en œuvre. Parmi eux, les IDS qui jouent un rôle essentiel en identifiant les activités suspectes et en permettant une réaction appropriée.

2.2.1 Système de détection d'intrusion

Les IDS sont utilisés pour surveiller l'activité d'un réseau ou d'un hôte afin de détecter toute tentative d'intrusion et, si possible, réagir à cette tentative. La détection d'intrusion consiste à suivre et analyser les événements qui surviennent dans un système ou un réseau informatique pour identifier des signes d'intrusion, tels que des tentatives de compromettre la confidentialité, l'intégrité, la disponibilité, ou de contourner les mécanismes de sécurité.

Les intrusions peuvent être causées par des attaquants externes cherchant à accéder au système via Internet, ou par des utilisateurs autorisés essayant d'obtenir des privilèges supplémentaires ou abusant de ceux qui leur sont accordés [1, 2].

Dans le monde réel, un IDS est comparable à un système d'alarme anti-intrusion d'une maison. Le dispositif de contrôle d'accès à l'entrée est comme un pare-feu, permettant ou refusant l'accès en utilisant une carte d'identité et un mot de passe. Toutefois, un voleur peut contourner ce dispositif en brisant une fenêtre ou en usurpant l'identité, ce qui montre les limites du contrôle d'accès.

Tout comme les systèmes d'alarme se déclenchent pour détecter une effraction et alerter les personnes concernées, les IDS génèrent une alerte en cas de tentative d'intrusion. Ces systèmes permettent de limiter les dommages en enregistrant les événements et en fournissant des informations essentielles pour les enquêtes et les éventuelles actions en justice.

La détection d'intrusion, comme les systèmes d'alarme, vise à réduire les risques, aider à l'enquête et rapporter les incidents au niveau de gestion, contribuant ainsi à améliorer la sécurité globale.

2.2.2 Le modèle de base d'un système de détection d'intrusion :

Depuis la publication des premiers articles, de nombreux systèmes de détection d'intrusion ont été développés et mis à jour. Par conséquent, il existe une grande variété de systèmes dans ce domaine, permettant d'établir une vision générale d'un système de détection d'intrusion "typique" ainsi que de ses composants principaux. Le groupe de travail IDWG (Intrusion Detection Exchange Format) de l'IETF (Internet Engineering Task Force) a décrit un modèle architectural général pour un système de détection d'intrusion. La figure suivante illustre un tel système [3].

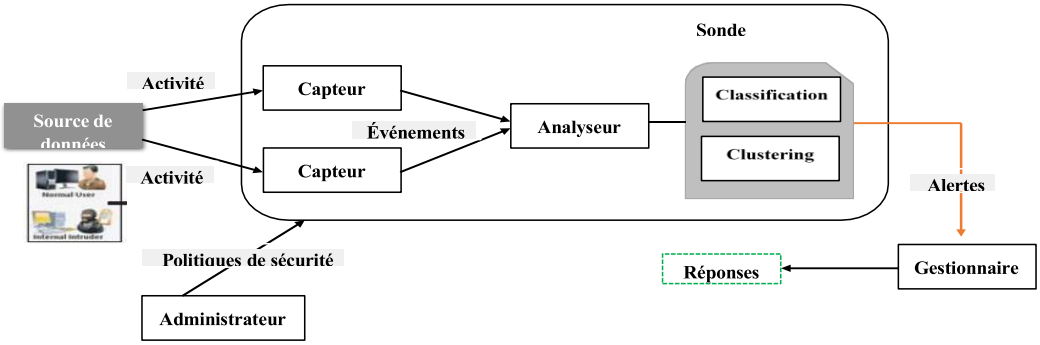

Figure 2.1: Architecture d'un IDS[3]

Comme le montre la figure, ce système est constitué de plusieurs outils, chacun ayant sa propre tâche : détecter les intrusions et informer l'administrateur d'une potentielle intrusion.

L'administrateur est chargé de mettre en place la politique de sécurité de l'entreprise, déployer et configurer les composants de l'IDS, ainsi que de définir la liste des actions autorisées sur le réseau ou sur certains hôtes en fonction des besoins du système d'information.

Les différentes parties du système surveillé peuvent servir de sources de données d'audit, telles que les entrées clavier, les commandes exécutées ou les applications utilisées. Les données d'audit sont généralement classées en deux catégories : celles provenant d'un réseau et celles d'un hôte, bien qu'elles puissent être collectées simultanément. Ces données sont souvent stockées pour une longue période à des fins de référence future, ou temporairement en attente de traitement. Ce stockage est crucial pour tout système de détection d'intrusion, c'est pourquoi certains chercheurs considèrent la détection d'intrusion comme un problème de réduction des données d'audit. Notre travail de thèse vise à minimiser la quantité de données à traiter, en définissant un graphe de flux d'information pour obtenir un IDS performant et efficace.

Le capteur et l'analyseur sont au cœur du système de détection d'intrusion. C'est à ce niveau que des algorithmes sont appliqués pour identifier des intrusions ou des comportements suspects
Dans les données d'audit. Deux approches principales ont été développées : l'approche basée sur les anomalies et celle basée sur les signatures.

Le capteur collecte les données brutes sur les actions en cours et les transmet à l'analyseur sous forme d'événements. L'analyseur examine ces événements pour détecter tout comportement suspect ou indésirable, et en informer le responsable de la sécurité. Dans la majorité des systèmes IDS, le capteur et l'analyseur sont intégrés dans un même composant, souvent appelé "sonde".

L'analyseur classe et filtre les événements. S'il identifie un incident lié à la sécurité, une alerte est générée, par exemple lors d'ouvertures inattendues de sessions Telnet, de tentatives d'accès à des fichiers non autorisés, ou de connexions échouées répétées.

Le gestionnaire reçoit les alertes et événements générés par l'analyseur, et les transmet à un opérateur humain via une console de supervision ou par e-mail. Avant la transmission, les événements sont convertis en un format d'alerte approprié. Le gestionnaire est essentiel pour contrôler les différentes parties du système : configuration des capteurs et des analyseurs, gestion des notifications, agrégation des données, et création de rapports font partie de ses responsabilités.

La réponse aux intrusions : est la partie du système qui gère les actions à prendre lorsqu'une activité suspecte est détectée. Elle peut inclure la notification d'un agent de sécurité ou être entièrement automatisée. Les actions possibles incluent la notification de l'opérateur, l'enregistrement des activités, la collecte de données brutes, la fin de session pour un utilisateur ou une application, ou la modification des contrôles d'accès au réseau ou au système.
Les IDS ne sont pas toujours constitués de composants distincts, comme illustré dans la figure 2.1. Certains systèmes combinent ces composants en un seul module, tandis que d'autres s'appuient sur des applications externes pour compléter leurs capacités de détection d'intrusion [4, 5].

2.2.3 Taxonomie des IDS :

Il existe différents types de technologie des systèmes de détection d'intrusions, caractérisés par différentes approches de l'architecture du système, l'environnement de déploiement, les techniques surveillances et les stratégies de détection. Il y a plusieurs taxonomies des IDS proposé dans la littérature. Une nouvelle perspective de ces taxonomies des IDS a été introduite par Liao et al. (2013) [6] et adoptée selon différents critères basés sur des termes largement acceptés. Elle est présentée par quatre critères principaux de classification, tels qu'illustré dans la figure 2 :

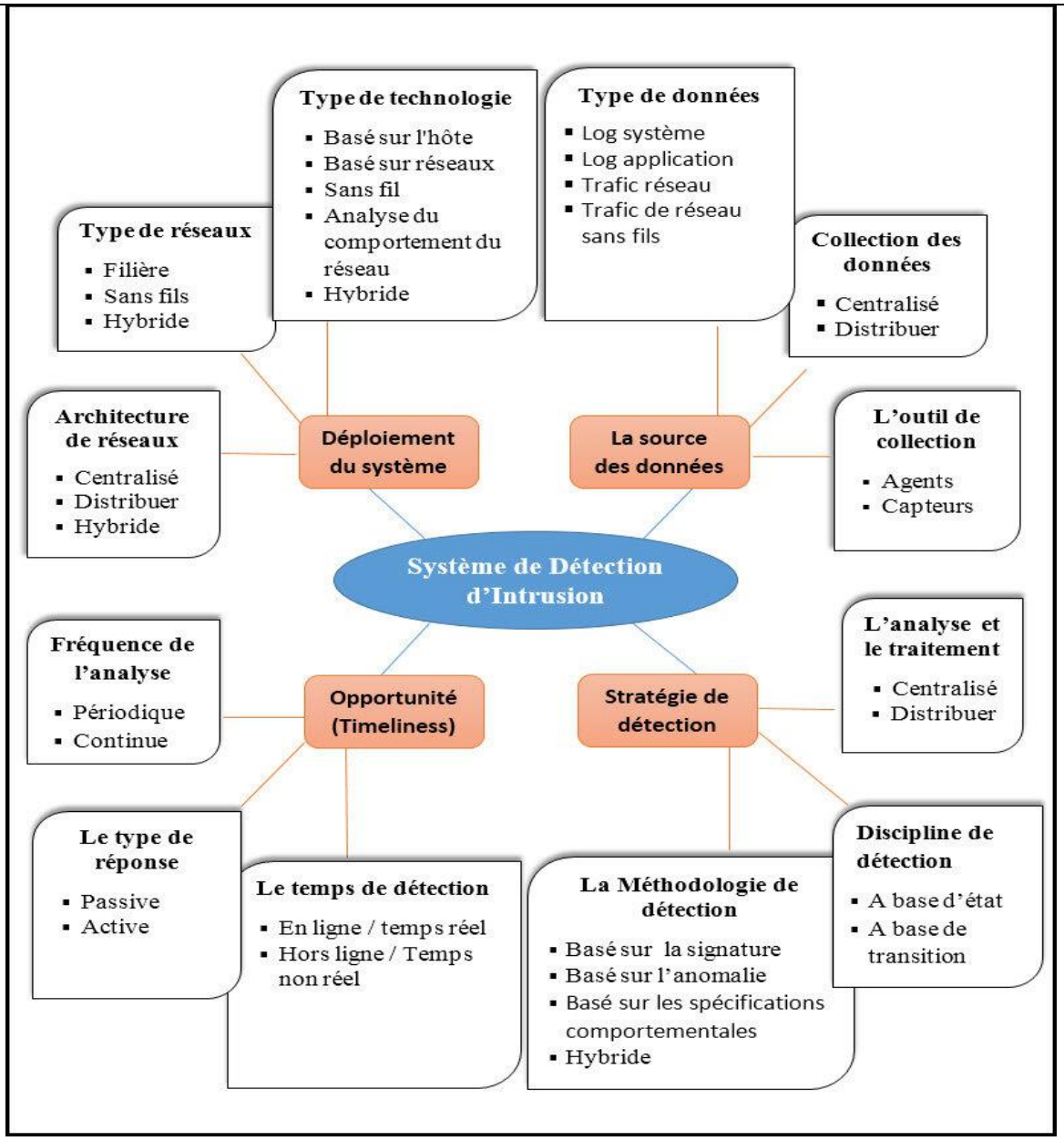

Figure 2.2 : Taxonomie des systèmes de détection d'intrusion proposé par Liao et al[6]

1. Le type de données : Les informations ou les données d'entrées à analyser par le système sont des caractéristiques essentielles des IDS avant d'entamer le processus de détection, les données proviennent des trafic réseau filière ou sans fils, les IDS se basent sur ces données sont appelés Network-based IDS (NIDS). Elles peuvent aussi être des données des machines hôtes comme les logs générées par le système d'exploitation ou par les applications, les IDS basés sur ces données sont appelés Host-based IDS (HIDS). Un autre type d'IDS apparaît récemment c'est le Cloud-based IDS basé sur les données des cloud computing [7].

2. La collecte des données : Deux architectures de systèmes différentes :

• Centralisée : la source de données et le système de détection sont tous ensemble

• Distribuée : les données sont collectées à l'aide de plusieurs capteurs situent en différents emplacements

3. L'outil de collecte: Leur rôle est d'accéder aux données brutes, et les filtrer pour ne renvoyer que les informations intéressantes à un analyseur d'IDS. Cet outil peut être un capteur extérieur du système, ou un agent logiciel qui fonctionne sur le système.

2.2.4 Types des systèmes de détection d'intrusion :

Les systèmes de détection d'intrusion ont pour objectif d'identifier toute tentative d'intrusion en générant des alertes, en s'appuyant sur l'analyse des données. Ces données peuvent provenir soit d'un hôte ou d'une application, auquel cas on parle de systèmes de détection d'intrusion basés sur les applications (IDS), soit d'un réseau, auquel cas il s'agit de systèmes de détection d'intrusion réseau (NIDS).

2.2.4.1 Système de détection d'intrusion hôte HIDS :

Les systèmes de détection d'intrusion basés sur l'hôte, ou HIDS (Host Intrusion Detection Systems) comme illustré dans la figure 2.3, collectent uniquement les informations relatives à l'hôte qu'ils protègent. Ce type de système permet d'identifier de manière précise et fiable les utilisateurs ou processus impliqués dans une attaque. Contrairement aux NIDS (Network Intrusion Detection Systems), les HIDS ont accès direct aux fichiers de données et aux processus, ce qui leur permet de contrôler précisément les cibles habituelles des attaques [8].

Les HIDS utilisent généralement des traces d'audit du système d'exploitation ou des fichiers log pour surveiller l'activité de la machine. Les fichiers log fournissent des informations basiques sur les événements du système, tandis que les traces d'audit sont plus détaillées et offrent des données complètes sur chaque action. Bien que les logs soient plus petits et donc plus faciles à analyser, les traces d'audit permettent une détection plus précise, notamment pour certaines attaques qui nécessitent une analyse approfondie [8].

Les HIDS offrent de nombreux avantages, notamment la capacité de constater immédiatement les effets d'une attaque, ce qui permet une réaction plus rapide et mieux ciblée. Grâce à la quantité de données collectées, il est possible d'observer avec précision l'activité sur l'hôte et d'adapter le système en conséquence. Par ailleurs, les HIDS sont particulièrement efficaces pour détecter des attaques, telles que les chevaux de Troie, qui sont difficiles à identifier avec un NIDS, surtout si ces attaques font partie du trafic chiffré.

Cependant, les HIDS présentent certaines limitations. Ils sont vulnérables aux attaques par déni de service (DoS), qui peuvent surcharger les systèmes en générant un grand volume de traces d'audit. De plus, la gestion des fichiers de rapports d'alertes volumineux peut être contraignante pour les administrateurs de sécurité, d'autant plus que ces fichiers peuvent atteindre plusieurs mégaoctets. La forte consommation de ressources processeur nécessaire pour traiter ces données peut également affecter les performances de la machine surveillée.

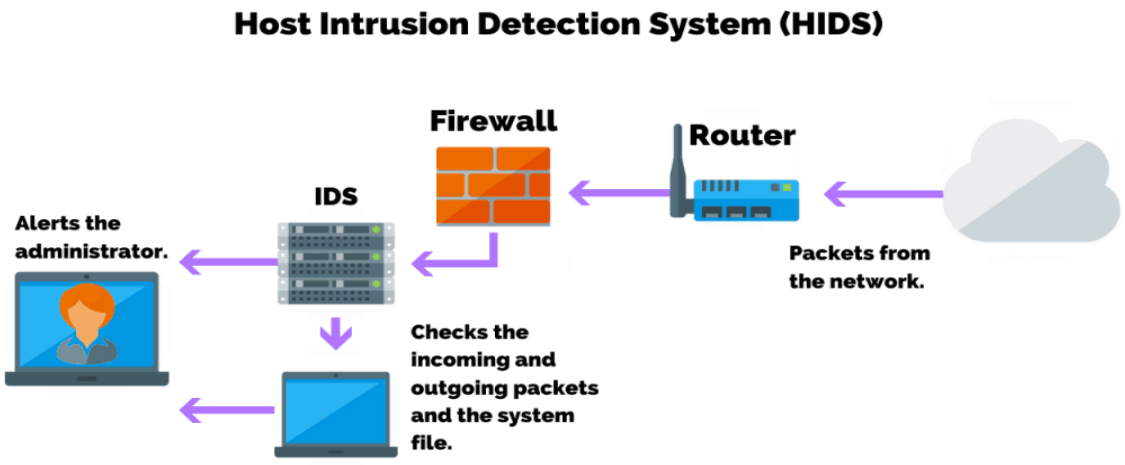

Figure 2.3 : Système de détection d'intrusion hôte HIDS[9]

2.2.4.2 Système de détection d'intrusion réseau NIDS :

Les systèmes de détection d'intrusion les plus couramment utilisés dans le commerce sont les systèmes basés sur le réseau. Ces systèmes, appelés NIDS (Network Intrusion Detection Systems), détectent les attaques en capturant et en analysant les paquets de données sur un segment de réseau ou un commutateur. Ils effectuent cette analyse en comparant les paquets à une base de signatures d'attaques connues ou en analysant les protocoles pour repérer des anomalies, comme l`illustre la figure 2.4.

Les NIDS peuvent déclencher des alertes et interrompre des connexions en temps réel dès qu'une activité suspecte est détectée. Ils surveillent chaque paquet de données qui transite sur le segment réseau local [8].

Bien que très utilisés et indispensables pour la sécurité des réseaux, les NIDS présentent plusieurs faiblesses. Ils ont un taux élevé de faux négatifs, ce qui signifie que certaines attaques peuvent ne pas être détectées. Le manque de données d'expérience et la rapidité avec laquelle les techniques de contournement évoluent les rendent vulnérables. Certains NIDS éprouvent des difficultés à traiter les fragments de paquets, ce qui peut entraîner des dysfonctionnements. De plus, ils peinent à gérer

des volumes élevés de trafic et ne peuvent pas analyser le contenu chiffré. Leur utilisation requiert également un personnel extrêmement vigilant.

Contrairement aux HIDS (Host Intrusion Detection Systems), les NIDS ne sont pas en mesure de percevoir directement l'impact des attaques sur le système cible. Cependant, la distinction entre les NIDS et les HIDS tend à diminuer, car les HIDS intègrent désormais certaines des fonctionnalités de base des NIDS.

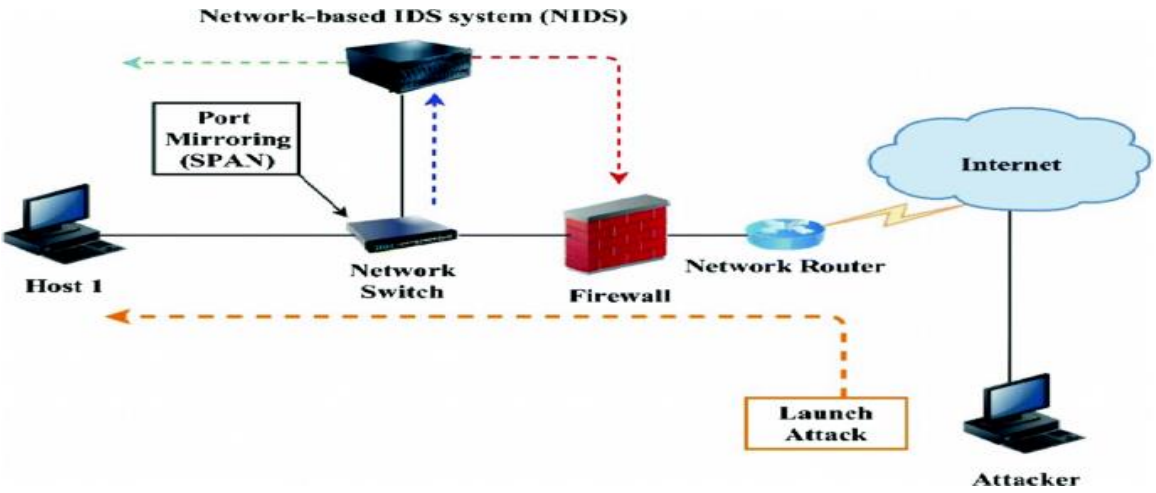

Figure 2.4: Système de détection d'intrusion réseau NIDS [10]

2.2.4.3 Système de détection d'intrusion basé sur une application

Les IDS basés sur les applications (AIDS) constituent une sous-catégorie des IDS hôtes. Ils surveillent l'interaction entre un utilisateur et un programme en générant des fichiers de log, ce qui permet de fournir des informations détaillées sur les activités d'une application spécifique. Un IDS de ce type se concentre sur la communication entre l'utilisateur et l'application surveillée.

L'un des principaux avantages de ces IDS est leur capacité à détecter et bloquer des commandes spécifiques que l'utilisateur pourrait exécuter via l'application, et de surveiller chaque transaction entre l'utilisateur et le programme. De plus, les données sont analysées dans un contexte connu, ce qui permet une interprétation plus fine et précise.

Cependant, ces IDS ne fonctionnant pas au niveau du noyau, ils offrent une sécurité moins robuste, en particulier contre des attaques comme les chevaux de Troie. De plus, les fichiers de log générés par ce type d'IDS sont plus vulnérables aux manipulations par des attaquants, contrairement aux traces d'audit du système qui sont généralement plus sécurisées. Ces IDS sont souvent utilisés pour surveiller des applications très sensibles, mais leur utilisation est souvent couplée à un HIDS. Il est également important de surveiller la charge CPU générée par ces IDS afin de ne pas altérer les performances globales du système [11].

2.2.4.4 Système de détection d'intrusion hybride :

Pour compenser les limitations des HIDS et NIDS mentionnées précédemment, un autre type de système de détection d'intrusion a été développé : le système de détection d'intrusion hybride. Les systèmes hybrides combinent les caractéristiques de différents types d'IDS, permettant ainsi de surveiller à la fois le réseau et l'hôte.

Ces systèmes utilisent des agents mobiles pour collecter des informations provenant de diverses sondes placées sur le réseau et sur les hôtes. Les agents mobiles, du côté HIDS, parcourent chaque hôte pour vérifier les fichiers journaux et détecter les activités suspectes. En parallèle, d'autres agents, appelés agents centraux, parcourent l'ensemble du réseau pour détecter les anomalies dans le trafic.

Prélude est un exemple connu de système IDS hybride dans le monde open-source. Cet IDS collecte les alertes de différents systèmes au sein d'une base de données centralisée. Il combine l'IDS Snort pour analyser le trafic réseau et le logiciel Samhain en tant que HIDS [12], permettant une visualisation centralisée des attaques en utilisant des outils puissants.

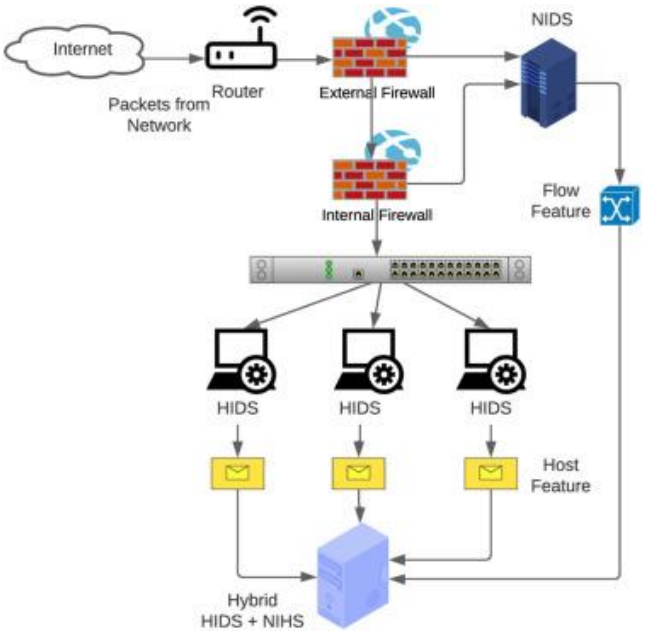

Figure 2.5 : Système de détection d'intrusion hydride[12]

2.2.5 Les approches de détection d'intrusion

L'utilisation d'outils de détection et de réaction face aux différentes attaques est essentielle pour garantir la sécurité des données des systèmes d'information. Cela inclut notamment les systèmes de détection d'intrusion (IDS). En raison de leur utilité pratique, les IDS ont fait l'objet de nombreuses recherches au cours des 30 dernières années, dans le but d'améliorer leur efficacité. Ces études ont

abouti à la création de diverses classes d'IDS, reposant sur des techniques de détection différentes, adaptées à des contextes spécifiques.

On distingue principalement deux approches de détection : l'approche comportementale, qui se base sur la connaissance des entités surveillées, et l'approche par signature, qui repose sur la connaissance des attaques elles-mêmes. Ainsi, un IDS basé sur les signatures peut identifier des comportements connus comme étant anormaux, tandis qu'un IDS basé sur l'analyse comportementale peut détecter des activités déviant du comportement habituel.

2.2.5.1 Approche comportementale (ou détection d'anomalie)

Cette approche a d'abord été proposée par Anderson [13], qui a défini une méthode reposant sur la comparaison de deux comportements : celui des utilisateurs et un comportement de référence appelé "profil". Un profil est un ensemble de mesures empiriques caractérisant un comportement "normal" pouvant concerner un utilisateur, un service, une application ou un système [14]. Ainsi, on établit un comportement normal de l'entité surveillée, et toute déviation par rapport à ce profil est considérée comme potentiellement intrusive. Denning a ensuite implémenté cette approche comportementale [15].

La construction du profil est réalisée lors d'une phase d'apprentissage, pendant laquelle les IDS observent le fonctionnement normal des éléments surveillés et recueillent les informations nécessaires. Pour un HIDS, cette détection peut s'appuyer sur des paramètres tels que le taux d'utilisation du processeur, l'activité sur le disque, les heures de connexion, ou l'utilisation de certains fichiers (heures de bureau).

L'approche proposée par Anderson repose sur l'hypothèse que l'exploitation d'une vulnérabilité implique un usage anormal du système. Ainsi, une intrusion se manifeste par une déviation du comportement habituel. Voici quelques exemples illustrant cette hypothèse :

- L'intrusion d'un utilisateur non autorisé entraîne un taux élevé de mots de passe incorrects.
- Un attaquant se connecte à des heures inhabituelles, utilise fréquemment des commandes pour changer de répertoire, et n'utilise pas les outils habituels de l'utilisateur légitime.
- Un utilisateur cherchant à contourner la politique de sécurité se connectera la nuit, exécutera des programmes inhabituels, générant un plus grand volume de données d'audit et utilisant des ressources non courantes (comme une imprimante différente).
- Un cheval de Troie diffère du programme légitime qu'il remplace en ce qui concerne l'utilisation des ressources d'entrée/sortie.

- Une attaque par déni de service se traduit par une consommation anormalement élevée de certaines ressources.

Les phénomènes décrits peuvent résulter d'une attaque, comme un changement de fonction de l'utilisateur au sein de l'entreprise. L'objectif est donc de trouver des méthodes avec un fort taux de discrimination (taux de détection élevé et faible taux de fausses alertes). On utilise un seuil au-delà duquel le comportement est considéré comme intrusif.

Cette approche, qui pose la question "le comportement actuel de l'utilisateur est-il cohérent avec son comportement passé ?", est appelée "approche comportementale". La méthode la plus courante pour modéliser le comportement normal d'un utilisateur consiste à utiliser des techniques statistiques, mais il est également possible de recourir à des systèmes experts ou à des réseaux de neurones. Nous présenterons les principales approches dans la suite de ce paragraphe [15].

2.2.5.1.1 Les avantages de l'approche comportementale

- L'approche comportementale présente des avantages majeurs, car elle ne cherche pas à caractériser les intrusions inconnues mais le comportement attendu du système.

- Cette détection permet de réagir rapidement aux menaces afin de réduire le temps passé par les intrus dans l'environnement et les dommages qu'ils pourraient alors causer.

2.2.5.1.2 Les lacunes de l'approche comportementale

- Le choix des différents paramètres du modèle statistique est assez complexe et soumis à l'expérience de l'équipe de sécurité.

- Le choix judicieux des mesures à prendre pour un système cible donné.

- Il est difficile de déterminer si les observations faites pour un utilisateur particulier correspondent à des activités que l'on souhaiterait interdire.

- Pour un utilisateur ayant un comportement erratique, toute activité est normale.

- Il ne faut pas prendre en considération les tentatives de collusion entre utilisateurs.

-En cas de profonde modification de l'environnement du système cible, déclenchement d'un flot d'alarmes ininterrompu.

- Utilisateur pouvant modifier lentement son comportement dans le but d'habituer le système à un comportement intrusif.

2.2.5.2 Approche par signature (détection par malveillance) :

Dans cette approche, il est essentiel de collecter des scénarios d'attaques pour alimenter une base de données dédiée. Les techniques de cette classe consistent à utiliser une base de signatures, qui contient des descriptions des scénarios d'attaques (appelées signatures d'attaques). Ces scénarios exploitent les vulnérabilités du système. La question clé posée par cette approche est : « Le comportement de l'utilisateur correspond-il à un scénario d'attaque connu ? ». Par conséquent, toute attaque non présente dans la base de signatures est indétectable, ce qui rend ce type d'IDS purement réactif, ne détectant que les attaques pour lesquelles il possède une signature, nécessitant donc des mises à jour fréquentes [6].

L'efficacité de ce système de détection dépend de la précision de sa base de signatures. Cependant, les attaquants utilisent des techniques d'évasion pour contourner ces systèmes, en modifiant les signatures de leurs attaques afin de ne pas être détectés par l'IDS. Il est possible de développer des signatures plus génériques pour détecter les variantes d'une attaque, mais cela demande une solide connaissance des attaques réseau et système.

Les vendeurs de produits IDS proposent régulièrement des mises à jour des signatures pour s'adapter aux nouvelles attaques découvertes. Toutefois, plus il y a de signatures à vérifier, plus le temps de traitement sera long. L'utilisation de signatures plus élaborées peut améliorer l'efficacité du système tout en réduisant le temps de traitement [2].

Cette méthode permet de détecter des comportements précis des attaquants, mais elle présente des défis tels que :

- La nécessité d'une base de règles bien construite, ce qui peut s'avérer complexe.

- L'incapacité à détecter les attaques inconnues.

- Les systèmes experts sont limités par la connaissance de l'expert humain qui a fourni les règles. En pratique, les responsables de la sécurité ont souvent une connaissance limitée en matière de détection d'intrusion, car l'analyse des volumes importants de fichiers d'audit peut être décourageante.

Ainsi, il est recommandé de combiner l'approche par signature avec l'approche comportementale afin de bénéficier des avantages des deux méthodes.

2.2.6 Types de réponse des systèmes de détection d'intrusion

Une autre façon de classer les systèmes de détection d'intrusions consiste à les classer par type de réaction lorsqu'une attaque est détectée :

2.2.6.1 Réponse passive

Les IDS passifs sont des systèmes configurés spécifiquement pour surveiller et analyser le trafic réseau, et pour alerter un opérateur des éventuelles vulnérabilités ou attaques. Ils ne sont pas en mesure d'entreprendre des actions de protection ou de correction de manière autonome. Lorsqu'une attaque est détectée, ces systèmes génèrent une alerte et informent l'administrateur système par e-mail, message sur une console, voire même via un bipeur. Il revient alors à l'administrateur de prendre les mesures appropriées. Les principaux avantages des IDS passifs résident dans leur déploiement rapide et facile.

Bien que les réponses passives ne fournissent pas de protection en temps réel contre les intrusions, elles permettent aux équipes de sécurité de prendre des décisions informées et de mettre en œuvre des mesures correctives appropriées après avoir examiné les alertes. Cela peut inclure des actions telles que le renforcement des politiques de sécurité, la modification des configurations de réseau ou des correctifs de sécurité.

2.2.6.2 Réponse active

La réponse active vise à stopper une attaque dès sa détection. Deux techniques sont disponibles à cet effet : la reconfiguration du pare-feu et l'interruption d'une connexion TCP.

La reconfiguration du pare-feu consiste à bloquer le trafic malveillant au niveau du pare-feu en fermant le port utilisé ou en interdisant l'adresse de l'attaquant. Cette fonctionnalité dépend du modèle de pare-feu utilisé, car tous les modèles ne permettent pas la reconfiguration via un IDS. De plus, cette reconfiguration est limitée par les capacités du pare-feu. L'IDS peut également interrompre une session établie entre un attaquant et sa cible afin d'empêcher le transfert de données ou la modification du système attaqué. Pour ce faire, l'IDS envoie un paquet TCP reset aux deux extrémités de la connexion (cible et attaquant). Ce paquet TCP reset a le flag RST positionné, signalant ainsi une déconnexion de la part de l'autre extrémité. La cible et l'attaquant pensent alors que l'autre extrémité s'est déconnectée, ce qui interrompt l'attaque [16].

Dans le cas d'une réponse active, il est crucial de s'assurer que le trafic détecté comme malveillant l'est réellement, sous peine de déconnecter des utilisateurs légitimes. En général, les IDS ne

réagissent activement qu'aux alertes confirmées comme étant des attaques. L'analyse des fichiers d'alerte générés est donc essentielle pour examiner toutes les attaques détectées.

Cependant, cette fonctionnalité automatique présente des risques potentiels car elle peut entraîner des dénis de service provoqués par l'IDS. Par exemple, un attaquant déterminé peut tromper l'IDS en utilisant des adresses du réseau local, ce qui pourrait être considéré comme une source d'attaque par l'IDS. Il est donc préférable de proposer une réaction facultative à un opérateur humain, qui prendra la décision finale [16].

2.2.7 Les systèmes de détection d'intrusion dans le cadre applicatif

Il existe plusieurs IDS sur le marché, parmi eux :

➢ **ISS RealSecure :** Internet Security Systems (ISS) propose ISS RealSecure IDS, une plate-forme de détection d'intrusions intégrée. ISS RealSecure IDS utilise une approche standardisée pour comparer les flux de trafic réseau et les journaux d'hôtes aux méthodes d'attaque connues et potentielles. ISS RealSecure IDS s'intègre avec de nombreuses applications de gestion des systèmes et du réseau.

RealSecure combine en un seul agent trois fonctionnalités essentielles :

− Un moteur de détection d'intrusions.

− Un firewall personnel.

− Un module de contrôle d'applications et de communications.

➢ **Enterasys DRAGON :** Publié par Enterasys Networks, ce système de détection d'intrusions est reconnu comme un leader sur le marché en raison de ses performances, de sa capacité à s'adapter à divers environnements et de ses capacités d'analyse. Les solutions Dragon comprennent des sondes NIDS (Network Sensor), des agents HIDS (Host Sensor) et un système de gestion qui assure les fonctions opérationnelles de la suite Dragon. Le Network Sensor est un NIDS disponible sous forme logicielle ou en tant qu'appliance dédiée. À partir de la version 6, Enterasys propose les appliances et les logiciels en trois versions, selon la bande passante à analyser, les versions matérielles étant accompagnées de leur équivalent logiciel. [17] L'Host Sensor est un agent HIDS qui détecte les attaques contre le système sur lequel il est installé en surveillant les journaux système et d'audit ainsi qu'en utilisant des mécanismes d'analyse de signatures. Dragon détecte les intrusions dans l'ensemble de l'infrastructure informatique où elles se produisent, offrant ainsi une visibilité globale sur le

système d'information. Cela permet notamment d'optimiser les ressources humaines nécessaires à l'analyse des journaux provenant des différents pare-feu ou serveurs Web en fédérant tous ces journaux au niveau d'une console.

➢ **SNORT :** SNORT est un IDS largement utilisé grâce à sa disponibilité en open source. Son accès gratuit et sa facilité d'obtention en font un choix populaire. En plus de sa gratuité, son principal avantage réside dans sa vaste base de signatures, alimentée par la communauté des utilisateurs. Cela garantit également des mises à jour rapides de la base dès qu'une nouvelle menace est signalée. Initialement conçu pour le système Linux, SNORT a également été adapté pour Windows. Il existe plusieurs ouvrages commerciaux dédiés à son installation et son utilisation. SNORT est généralement utilisé en tandem avec un autre logiciel open source appelé BASE, qui sert de console de gestion et d'analyse. Cependant, malgré ses nombreux avantages, SNORT est considéré comme moins performant en termes de moteur d'analyse par rapport à des solutions commerciales telles que celles proposées par ISS. De plus, bien que la base de signatures soit étendue, elle nécessite un travail constant de la part de l'administrateur, qui doit les télécharger manuellement, car il n'existe pas de procédure de mise à jour automatique [18].

2.3 Les IDS basés sur l'apprentissage automatique

Les IDS basés sur l'apprentissage automatique utilisent des algorithmes d'apprentissage automatique pour détecter des anomalies et identifier des comportements malveillants dans un réseau ou un système. Contrairement aux IDS traditionnels qui se basent sur des signatures d'attaques connues, les IDS basés sur l'apprentissage automatique peuvent apprendre des données existantes et identifier des attaques nouvelles ou inconnues, ce qui les rend plus adaptatifs et plus efficaces face aux menaces émergentes.

2.3.1 L'apprentissage automatique

L'apprentissage automatique est une branche de l'intelligence artificielle qui se concentre sur le développement de techniques permettant aux ordinateurs d'apprendre à partir de données et d'améliorer leur performance sur une tâche spécifique sans être explicitement programmés. Ces techniques permettent aux systèmes informatiques de reconnaître des modèles complexes et de prendre des décisions autonomes en se basant sur des données passées.

Il existe plusieurs types d'algorithmes de l'apprentissage automatique, notamment :

2.3.1.1 L’apprentissage supervisé

L’apprentissage supervisé est une méthode d'apprentissage automatique où un modèle est formé sur un ensemble de données étiquetées. Ces étiquettes sont essentiellement les "réponses" aux exemples donnés, ce qui permet au modèle d'apprendre à faire des prédictions sur de nouvelles données en se basant sur les associations apprises lors de l'entraînement. C'est un cadre très utilisé dans de nombreux domaines, comme la classification, la régression, et bien d'autres.

2.3.1.1.1 Quelques algorithmes d’apprentissage supervisé

2.3.1.1.1.1 Forêt Aléatoire (Random Forest)

La forêt aléatoire est une technique d'apprentissage supervisé qui combine plusieurs arbres de décision pour améliorer la précision de prédiction et réduire le risque de surapprentissage (overfitting). Cette approche est basée sur l'idée de l'Ensemble Learning, où plusieurs modèles faibles sont combinés pour former un modèle fort. La forêt aléatoire est utilisé pour les tâches de classification et de régression [19].

Un arbre de décision est un modèle qui prédit la valeur d'une variable cible en divisant les données en fonction des caractéristiques les plus informatives. Comme illustré dans la figure 2.6 Chaque arbre dans la forêt prédit une classe ou une valeur pour les données d'entrée, et la prédiction finale du modèle est obtenue en faisant la moyenne des prédictions (pour la régression) ou en prenant la classe majoritaire (pour la classification).

Lors de la construction d'une forêt aléatoire, chaque arbre est entraîné sur un sous-ensemble aléatoire des données d'entraînement. Ce sous-ensemble est généralement créé à l'aide de la méthode du Bootstrap, qui consiste à échantillonner les données avec remplacement. Cela signifie que certains exemples de données peuvent être présents plusieurs fois dans l'échantillon tandis que d'autres peuvent être absents. En outre, à chaque nœud de chaque arbre, seule une sous-partie aléatoire des caractéristiques est considérée pour la division. Cela contribue à la diversité des arbres, améliorant ainsi la capacité de généralisation du modèle [19].

Le processus d'Ensemble Learning contribue à la réduction de la variance, ce qui permet à la forêt aléatoire de généraliser bien mieux que des arbres individuels. Alors que les arbres de décision ont tendance à surapprendre sur les données d'entraînement, les forêts aléatoires réduisent ce risque en

moyennant les erreurs de plusieurs arbres, chacun ayant été formé sur des sous-ensembles différents de données et caractéristiques.

La forêt aléatoire offre plusieurs avantages, notamment sa capacité à traiter des jeux de données avec un grand nombre de caractéristiques sans avoir besoin de présélectionner les variables. De plus, la forêt aléatoire permet d'estimer l'importance des caractéristiques. Cela est fait en calculant la diminution moyenne de l'impureté à chaque division, ou en mesurant la réduction moyenne de la précision du modèle lorsqu'une caractéristique particulière est supprimée. Ces métriques aident à identifier quelles caractéristiques sont les plus informatives pour la prédiction.

Malgré ses nombreux avantages, la forêt Aléatoire présente aussi des inconvénients. Le modèle est plus lent à entraîner et à prédire par rapport à un arbre de décision unique, surtout avec un grand nombre d'arbres. En outre, bien que la Foret Aléatoire fonctionne bien sur de nombreux types de données, il peut ne pas être aussi performant sur des données avec des structures très complexes ou des données avec des relations linéaires subtiles que d'autres modèles peuvent capturer plus facilement.

En termes d'hyperparamètres, certains des plus importants pour la Foret Aléatoire incluent le nombre d'arbres (“n_estimators”), la profondeur maximale des arbres (“max_depth”), le nombre de caractéristiques à considérer à chaque division (“max_features”), et le critère pour mesurer la qualité des divisions (“criterion”, tel que “gini” ou “entropy”). Ajuster ces hyperparamètres de manière appropriée permet d'améliorer les performances du modèle en termes de précision et de généralisation.

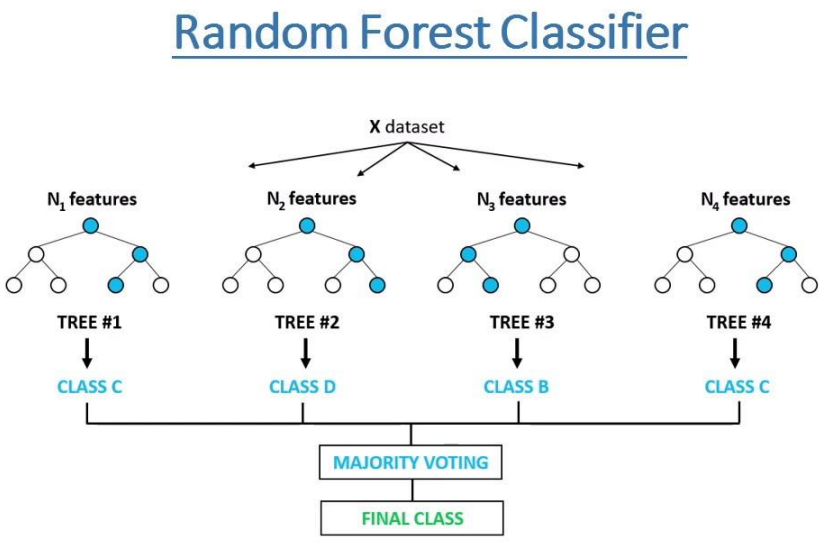

Figure 2.6 : Forêt d'arbres décisionnels[20]

2.3.1.1.1.2 Arbre de Décision

L'arbre de décision est un algorithme de l'apprentissage automatique utilisé pour la classification et la régression. Il fonctionne en divisant les données en sous-ensembles basés sur des critères de décision qui maximisent l'homogénéité de chaque groupe. Cet algorithme est représenté sous forme d'une structure arborescente composée de nœuds où chaque nœud représente une caractéristique ou une condition sur les données. Les feuilles de l'arbre représentent les classes ou les valeurs prédictives.

Chaque division dans un arbre de décision est basée sur une caractéristique qui maximise une certaine mesure de l'homogénéité au sein des groupes créés. Les mesures d'homogénéité les plus couramment utilisées sont l'entropie et l'indice de Gini [21].

L'objectif est de minimiser l'entropie après chaque division, de manière à obtenir des sous-groupes aussi homogènes que possible. Plus l'entropie est faible, plus les sous-groupes sont homogènes.

Comme pour l'entropie, plus l'indice de Gini est faible, plus le nœud est homogène. L'arbre de décision cherche à minimiser l'impureté de Gini à chaque division.

Comme illustré dans la figure 2.7, la construction d'un arbre de décision commence par un nœud racine, représentant l'ensemble complet des données. À chaque étape, l'algorithme choisit la meilleure caractéristique pour diviser les données, en fonction de la mesure d'homogénéité choisie. Cette caractéristique est utilisée pour créer des branches qui subdivisent les données en sous-ensembles de plus en plus homogènes. Ce processus est répété de manière récursive pour chaque sous-ensemble jusqu'à ce que toutes les données soient classées de manière homogène ou que certaines conditions d'arrêt soient atteintes (par exemple, une profondeur maximale de l'arbre, un nombre minimal d'échantillons par nœud, ou une impureté minimale) [21].

Lors de la prédiction, un nouvel échantillon est introduit au nœud racine et parcourt l'arbre en suivant les règles définies par les conditions de chaque nœud. Finalement, il atteint une feuille qui contient la prédiction pour cet échantillon, soit une classe pour la classification, soit une valeur pour la régression.

Pour éviter le surapprentissage, une pratique courante est d'élaguer l'arbre, c'est-à-dire de supprimer certaines branches qui contribuent à la complexité de l'arbre mais n'apportent pas de réelle valeur prédictive. L'élagage se fait généralement après la construction de l'arbre, en utilisant des données de validation pour déterminer quelles branches supprimer.

Les arbres de décision sont appréciés pour leur capacité à gérer des données de différentes natures (catégorielles et numériques) sans nécessiter de normalisation, leur capacité à fournir une interprétation simple et intuitive des décisions, ainsi que pour leur flexibilité. Cependant, ils ont tendance à sur-ajuster les données si leur taille n'est pas contrôlée, et ils peuvent être sensibles à de petites variations dans les données. Pour pallier ces faiblesses, des méthodes d'ensemble comme la forêt aléatoire combinent plusieurs arbres de décision pour améliorer leur robustesse et leur performance globale.

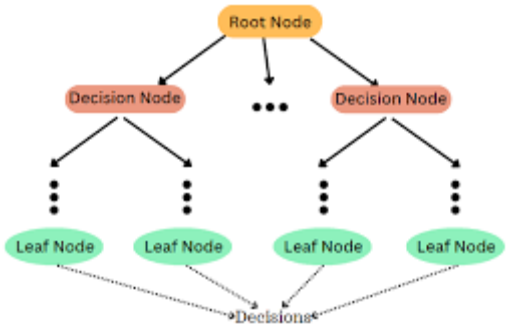

Figure 2.7 : arbre de decision[21]

2.3.1.2 Apprentissage non supervisé

L'apprentissage non supervisé est une autre branche de l'apprentissage automatique où les données ne sont pas étiquetées. Dans ce cas, le modèle cherche à découvrir des structures sous-jacentes ou des schémas intrinsèques dans les données sans être guidé par des étiquettes préexistantes (voir figure 2.8). Cela peut inclure des techniques telles que le clustering pour regrouper des données similaires, la réduction de dimension pour trouver des représentations plus compactes des données, ou encore la détection d'anomalies pour identifier des observations inhabituelles [22]. C'est une approche très utile pour explorer et comprendre des ensembles de données sans avoir besoin d'étiquettes préalables.

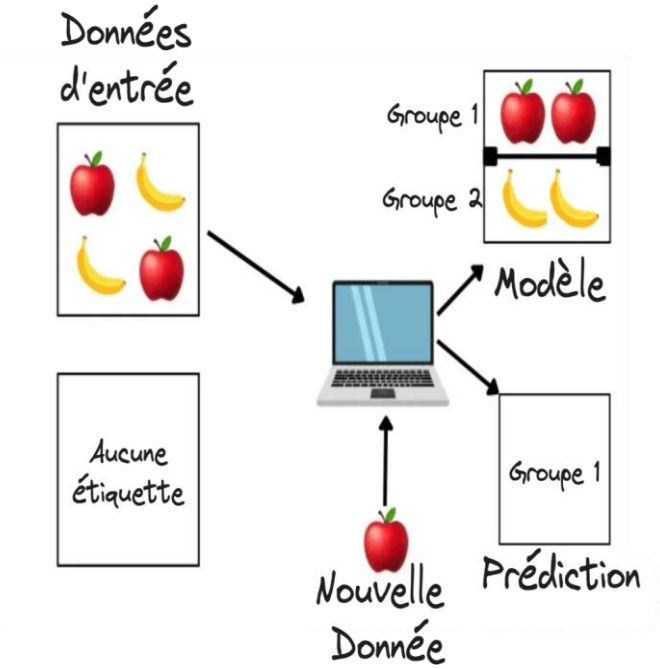

Figure 2.8 : Apprentissage non supervise[23]

2.3.1.3 L`apprentissage par renforcement

L'apprentissage par renforcement est une méthode d'apprentissage automatique où un agent apprend à prendre des décisions en interagissant avec un environnement. Tel qu'illustré dans la figure 2.9, l'agent reçoit des récompenses ou des sanctions en fonction des actions qu'il prend, ce qui lui permet d'apprendre quelles actions sont les plus bénéfiques dans différentes situations. L'objectif principal de l'agent est de maximiser les récompenses cumulées au fil du temps. Cette approche est souvent utilisée dans des domaines tels que les jeux, la robotique, la gestion de ressources, et bien d'autres.

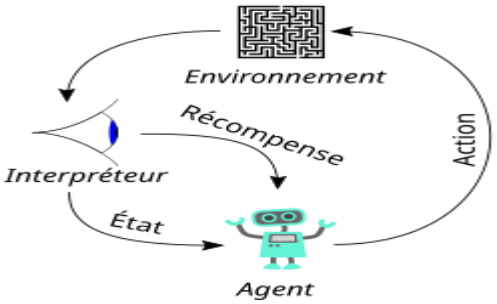

Figure 2.9: Apprentissage par renforcement[24]

2.3.2 L'apprentissage profond

L'apprentissage profond est une technique d'apprentissage automatique basée sur l'intelligence artificielle qui enseigne aux ordinateurs à effectuer des tâches qui sont naturelles pour l'homme, c'est-à-dire apprendre par modèle. L'apprentissage profond est une innovation cruciale derrière les voitures autonomes et les véhicules qui leur permet de comprendre un signal d'arrêt ou de

différencier un piéton d'un lampadaire [25]. C'est la technologie essentielle qui permet le contrôle vocal dans des gadgets tels que les téléphones portables, les ordinateurs portables, la télévision et les écouteurs. Ces temps-ci, l'apprentissage profond est sous les feux de la rampe car il obtient des résultats qui étaient impossibles auparavant. Avec l'aide d'apprentissage profond, un modèle informatique peut apprendre automatiquement des tâches de classification tout comme les humains, directement à partir d'images, de vidéos, de textes ou de la voix. Les modèles d'apprentissage profond sont capables d'atteindre une précision supérieure qui dépasse les résultats au niveau humain. Ces modèles d'apprentissage profond sont formés sur une énorme quantité de données étiquetées et utilisent des architectures neuronales de plusieurs couches. La majorité des techniques d'apprentissage profond utilisent des réseaux neuronaux, et par conséquent, les modèles d'apprentissage profond sont également appelés réseaux neuronaux profonds. Les modèles construits sur des réseaux neuronaux profonds sont formés sur une énorme quantité de données étiquetées, et les couches neuronales sont capables d'apprendre directement des caractéristiques à partir de l'ensemble de données sans avoir besoin d'une extraction manuelle des caractéristiques [25].

Les réseaux neuronaux profonds sont un ensemble de neurones organisés en une séquence de couches interconnectées. Ce qui les distingue, c'est l'architecture du réseau, c'est-à-dire la façon dont les neurones sont disposés et interagissent entre eux.

2.3.2.1 Quelques algorithmes d'apprentissage profond

Parmi les nombreuses implémentations de modèles d'apprentissage profond, on trouve :

2.3.2.1.1 Réseau de Neurones Convolutifs (CNN)

Un réseau neuronal convolutionnel (CNN) est une extension des réseaux de propagation avant traditionnels (feed forward network FFN), inspirée par des principes biologiques [30]. Initialement étudiés pour le traitement d'images, les CNN sont particulièrement efficaces pour identifier des motifs répétitifs, comme des bords ou d'autres éléments récurrents dans une image. Les CNN surpassent tous les autres algorithmes d'apprentissage automatique classiques et ont connu un grand succès dans les tâches de vision par ordinateur (Computer Vision). Ils sont largement utilisés dans le traitement d'images et de vidéos, le traitement du langage naturel (NLP), les systèmes de recommandation, et bien plus encore.

Les réseaux convolutionnel sont particulièrement performants grâce à plusieurs types de couches spécialisées : les couches de convolution, les couches de regroupement (Pooling) et les couches entièrement connectées [26]. La figure 2.10 illustre un modèle de réseau convolutionnel unidimensionnel (1D CNN).

Convolution Layers:

L'objectif de la convolution est d'extraire des caractéristiques de haut niveau. Elle est composée d'un ensemble de filtres (ou noyaux) apprenants, chacun représentant une fonctionnalité spécifique à partir du volume d'entrée. Ces filtres sont constitués d'une couche de poids de connexion et possèdent un petit champ de réception (la taille du noyau). Lors de la phase de passage avant (feedforward), chaque filtre est appliqué sur la largeur et la hauteur du volume d'entrée, calculant le produit scalaire entre les entrées et les valeurs du filtre, ce qui génère une nouvelle carte de caractéristiques qui représente plus fidèlement l'information.

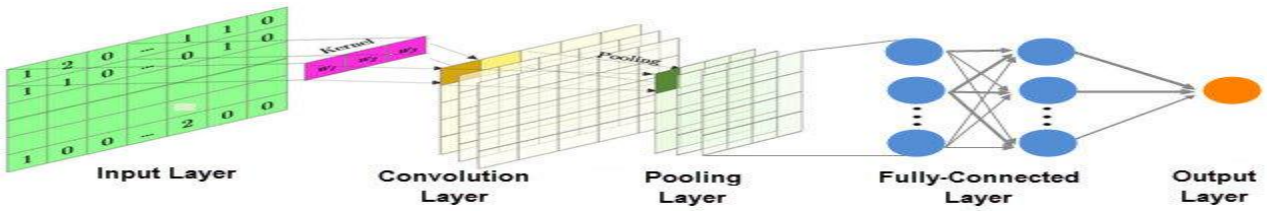

Figure 2.10 :L'architecture d'un modèle de réseau neuronal convolutive[26]

Ainsi, le réseau apprend des filtres qui s'activent lorsqu'ils détectent un type de caractéristique importante et spécifique à une certaine position spatiale dans l'entrée. La figure 2.11 illustre une opération de convolution 1D avec une entrée de dimension 1.

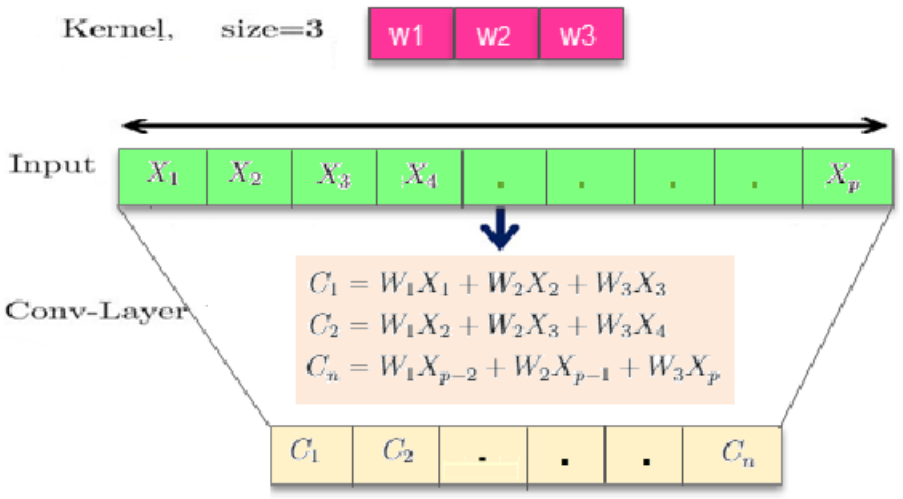

Figure 2.11 : couche Convolutionel[26]

Une couche convolutionnelle utilise le même noyau de convolution pour toutes ses opérations, ce qui réduit considérablement le nombre de paramètres nécessaires. Après chaque couche convolutionnelle, une fonction d'activation non linéaire est appliquée immédiatement. Dans les CNN profonds, la fonction d'activation couramment utilisée est le "Rectified Linear Units" (ReLU).

$$[f(x) = \max(0, x)]$$

Elle renvoie la valeur de x pour toutes les valeurs de $x > 0$ et renvoie 0 pour toutes les valeurs de $x \leq 0$. Les réseaux utilisant cette fonction d'activation s'entraînent plusieurs fois plus rapidement que ceux utilisant des unités "Tanh" [27].

Pooling Layers:

Après l'application de la fonction ReLU, l'opération de mise en commun (Pooling) regroupe les activations des neurones d'une couche en un seul neurone dans la couche suivante. La couche de pooling fonctionne indépendamment sur chaque entité d'entrée, réduisant progressivement la taille des représentations pour diminuer le nombre de paramètres ou de poids, ce qui allège le coût de calcul dans le réseau tout en préservant les informations les plus cruciales. Elle aide également à contrôler le surapprentissage. Il existe deux méthodes de mise en commun :

- **Max-Pooling:** prend la valeur maximale de chaque groupe de neurones de la couche précédente.

- **Average-Pooling:** prend la valeur moyenne de chaque groupe de neurones de la couche précédente.

Le pooling est une forme de sous-échantillonnage non linéaire qui fonctionne de manière similaire à la convolution. Le noyau de pooling est appliqué sur le volume d'entrée, le divisant en un ensemble de régions non chevauchantes, et chaque sous-région produit une seule valeur en sortie : soit la valeur maximale pour le Max-Pooling, soit la valeur moyenne pour l'Average-Pooling. La figure 2.12 illustre l'opération de Max-Pooling avec une entrée 1D et un noyau de taille 2.

La couche de regroupement n'a aucun paramètre à apprendre. En conséquence, ces couches ne sont généralement pas comptabilisées dans le nombre total de couches des réseaux convolutionnels.

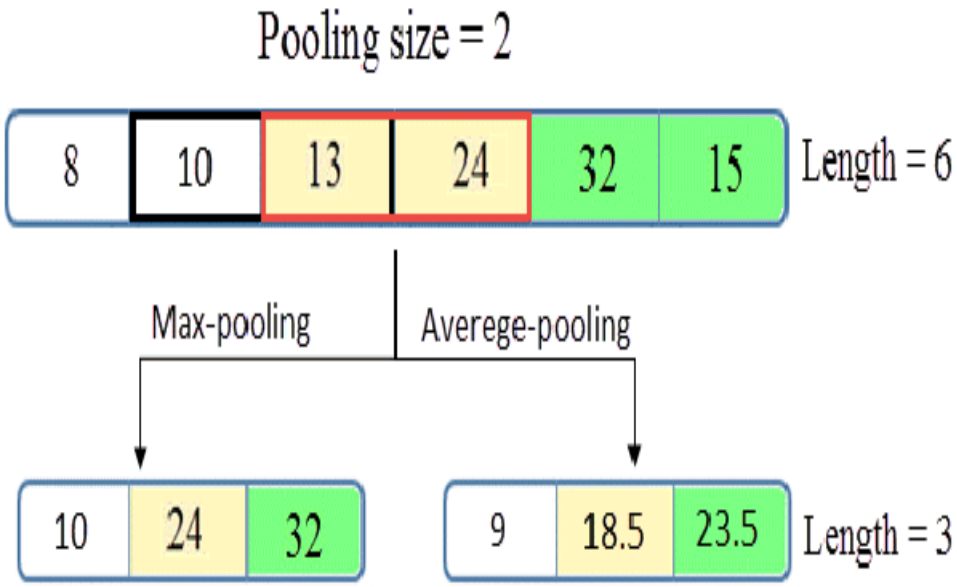

Figure 2.12 : couche de regroupement[27]

Fully Connected Layers:

À la fin d'un réseau CNN, on trouve une ou plusieurs couches entièrement connectées, où chaque nœud de la première couche est relié à chaque nœud de la couche suivante. Ces couches servent à effectuer une classification basée sur les caractéristiques extraites lors des convolutions. La couche finale intègre une fonction d'activation Softmax, qui génère une probabilité entre 0 et 1 pour chaque étiquette de classe que le modèle tente de prédire. Dans certaines architectures CNN récentes, les couches entièrement connectées peuvent être remplacées par plusieurs couches de mise en commun moyenne (average-pooling). Cela permet de réduire considérablement le nombre total de paramètres du réseau, améliorant ainsi la prévention du sur-apprentissage [28].

2.3.2.1.2 Réseaux de Neurones Récurrents (RNN) :

Les réseaux neuronaux s'inspirent du fonctionnement des neurones biologiques du cerveau humain, où les neurones jouent un rôle central dans la réflexion et parfois doivent mémoriser certains événements pour les utiliser ultérieurement avant de prendre une décision. Contrairement aux réseaux neuronaux traditionnels, les réseaux de neurones récurrents (RNN) sont conçus pour imiter ce processus, en se basant sur l'idée qu'un être humain raisonne en s'appuyant sur des connaissances acquises et mémorisées antérieurement [29].

Les RNN sont des réseaux de type propagation avant (feed-forward) avec un état interne (ou mémoire) qui tient compte à la fois des données précédemment vues par le réseau et des nouvelles données actuelles pour ajuster leurs décisions. L'idée clé de ces réseaux repose sur l'utilisation d'un calcul récurrent grâce à des boucles dans l'architecture du réseau. La sortie du réseau est une combinaison de son état interne (mémoire des entrées) et de la dernière entrée reçue, tandis que l'état interne évolue pour intégrer cette nouvelle donnée. Cela permet aux informations de persister en mémoire, comme l'illustre la figure 2.13.

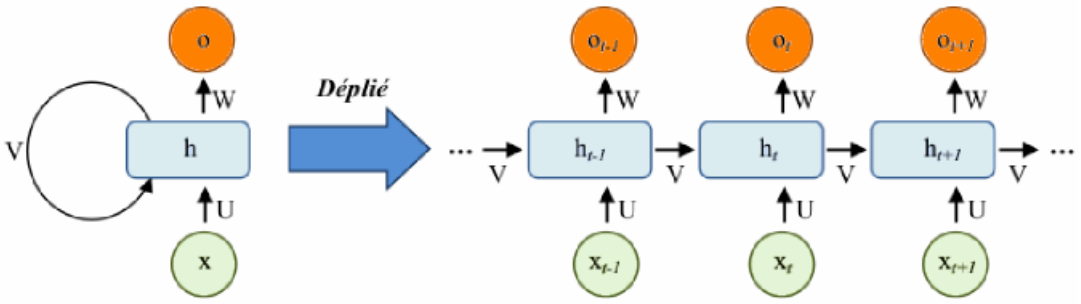

Figure 2.13 : Architecture du modèle RNN[29]

En raison de ces propriétés, les réseaux récurrents sont bien adaptés aux situations où la présence d'une forme n'est pas la seule information discriminante, mais où l'ordre d'apparition joue également un rôle important. Ils sont donc de bons candidats pour les tâches qui traitent des données séquentielles, comme les données textuelles ou les données avec des caractéristiques temporelles. La description mathématique du processus de transfert de mémoire est la suivante :

$$h_t = \delta(Ux_t + Vh_{t-1} + b_h)$$

$$O_t = \delta(Wh_t + b_y)$$

Ou :

- h_t : l'état caché au temps t.

- x_t : C'est l'entrée au même temps t .
- U , V , et W représentent les matrices de pondération, respectivement pour les connexions Input-to-Hidden, Hidden-to-Hidden, et Hidden-to-Output, également connues sous le nom de matrices de transition.
- b_h : C'est la valeur du biais de l'état caché.
- b_y: C'est la valeur du biais de sortie.
- O_t : C'est la valeur de la sortie au temps t .
- δ: C'est une fonction de non-linéarité, appelée fonction d'activation (comme la fonction sigmoïde logistique ou tanh), qui est un outil standard pour transformer des valeurs très grandes ou très petites dans un espace logistique, ainsi que pour rendre les gradients exploitables lors de la rétropropagation.

Un bloc de réseau neuronal examine une entrée x_t et génère une valeur O_t. Une boucle de rétroaction se produit à chaque pas de temps, chaque état caché h_t contenant non seulement des traces de l'état caché précédent, mais aussi de tous les états précédents h_{t-1} aussi longtemps que la mémoire peut persister.

Exemple : prévision de température

Supposons qu'on ait les températures quotidiennes d'une ville. On veut prédire la température du jour t en fonction des jours précédents.

À chaque instant t :

- x_t= température du jour actuel (ex. 18°C).
- h_{t-1} = état caché, représentant ce que le modèle a retenu des jours précédents (ex. tendance : montée ou baisse).

$$h_t = \delta(Ux_t + Vh_{t-1} + b_h)$$

- Le réseau combine la température du jour et la mémoire passée pour mettre à jour sa compréhension du climat.

$$O_t = \delta(Wh_t + b_y)$$

- Le réseau utilise cette mémoire actualisée pour prédire la température du lendemain.

2.3.2.1.3 Unités de mémoire a court terme (LSTM)

Les réseaux neuronaux récurrents (RNN) rencontrent un problème lorsqu'ils traitent des séquences longues, car ils prennent en compte les états sauvegardés précédents lors de la mise à jour des poids. Au fur et à mesure que l'entraînement progresse, les gradients deviennent de plus en plus petits après quelques étapes, ce qui empêche la propagation efficace des erreurs à travers le réseau. Cela signifie qu'il n'y a pas de différence significative dans les résultats, rendant impossible la mise à jour des poids. Ce problème est connu sous le nom de "disparition des gradients" (Vanishing Gradients). Pour surmonter ce problème, les chercheurs allemands Sepp Hochreiter et Juergen Schmidhuber ont proposé dans les années 90 une architecture appelée "Long Short-Term Memory" (LSTM) pour les réseaux neuronaux récurrents, ainsi que des couches supplémentaires appelées "Gated Recurrent Units" (GRU). Ces techniques ont été développées pour améliorer les performances et la précision des RNN [30].

L'idée centrale de la méthode LSTM est l'état de la cellule. Cette méthode permet de supprimer ou d'ajouter des informations à l'état de la cellule. La technique est régulée par des structures appelées "portails" (Gates). Ces portes fonctionnent généralement avec une fonction sigmoïde, où une valeur de 1 signifie que toutes les informations passent, tandis qu'une valeur de 0 signifie que les informations sont bloquées.

Les architectures LSTM et GRU fonctionnent de manière similaire, mais le GRU utilise moins de paramètres d'entraînement, ce qui nécessite moins de mémoire et permet un entraînement plus rapide que les LSTM. Cependant, les LSTM sont plus précis pour les ensembles de données qui utilisent des séquences plus longues.

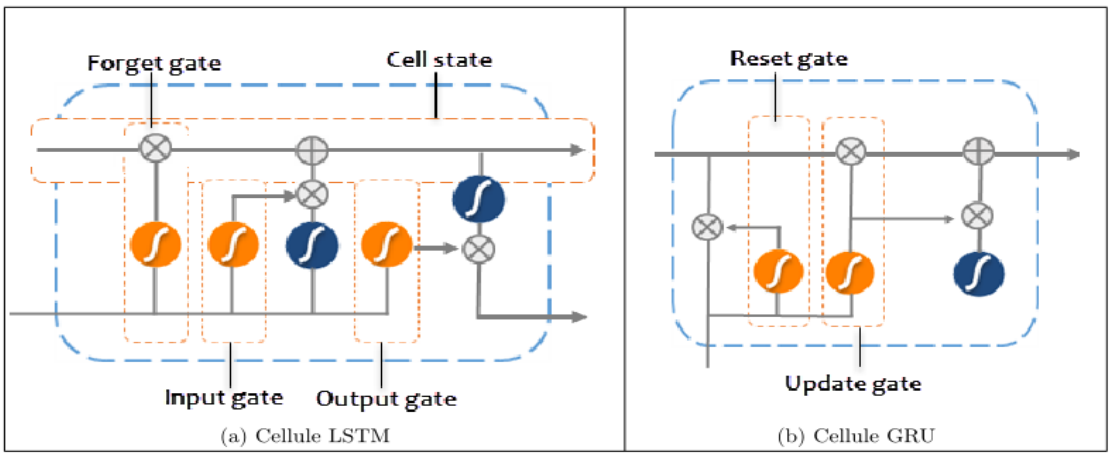

Figure 2.14 : L'architecture d'un modèle LSTM_GRU[30]

Les cellules LSTM sont particulièrement efficaces pour conserver les informations utiles pendant la rétropropagation du gradient. Cette capacité leur permet de corriger les écarts entre les prédictions et les catégories de référence en calculant le gradient de l'erreur pour chaque neurone, en remontant de la dernière couche vers la première. La figure 2.14 illustre les quatre couches interactives (sigmoïde et tanh), les trois portes, et les opérations pointwise qui traitent le vecteur x à l'intérieur d'une cellule LSTM à un temps t.

2.3.2.1.4 Les réseaux siamois

Le réseau siamois est une architecture de réseau de neurones conçue pour apprendre des représentations permettant de mesurer la similarité entre des entrées, plutôt que de simplement effectuer une classification ou une régression. Cette architecture est utilisée dans des tâches telles que la reconnaissance de visage, la vérification de signature, ou des problèmes de recherche de similarités. Un réseau siamois consiste en deux sous-réseaux identiques qui partagent les mêmes poids et biais, prenant chacun une entrée différente, appliquant une série de transformations pour extraire des caractéristiques, puis utilisant une couche qui mesure la similarité entre les représentations extraites [31].

Comme illustré dans la figure 2.15 le réseau siamois est composé d'un sous-réseau partagé qui prend deux entrées et les traite de manière identique, garantissant ainsi une approche symétrique pour mesurer la similarité. Ce sous-réseau est souvent un perceptron multicouche (MLP), un réseau de neurones convolutionnel (CNN), ou une combinaison des deux, selon le type de données à traiter. Après que les deux entrées ont traversé les sous-réseaux, elles passent par une couche qui mesure la similarité entre les représentations. Cela peut être fait en utilisant des mesures telles que la distance Euclidienne, la distance de Manhattan, ou une différence absolue.

Pour mesurer la similarité entre les deux représentations extraites des sous-réseaux, on utilise souvent la distance Euclidienne ou la distance de Manhattan. La distance Euclidienne est définie comme la racine carrée de la somme des carrés des différences entre les deux vecteurs, tandis que la distance de Manhattan est la somme des valeurs absolues des différences. Une fois la distance calculée, elle est transformée en une valeur de similarité, généralement en utilisant une fonction sigmoïde qui donne une sortie entre 0 et 1, où 1 représente une haute similarité et 0 une différence significative [31].

Le réseau siamois est souvent entraîné en utilisant une perte contrastive, qui vise à minimiser la distance entre les paires similaires et à maximiser la distance entre les paires non similaires. La fonction de perte contrastive est conçue pour encourager le réseau à rapprocher les représentations

des paires positives (similaires) tout en éloignant les paires négatives (différentes). Pour entraîner ce réseau, des paires de données sont créées, comprenant des paires positives (même classe) et des paires négatives (classe différente).

Après l'entraînement, le réseau siamois peut être utilisé pour de nombreuses tâches qui impliquent des comparaisons, telles que la vérification d'identité. Pour une nouvelle paire d'échantillons, les deux sous-réseaux génèrent des vecteurs de caractéristiques, et la distance entre ces vecteurs est calculée. Si cette distance est inférieure à un certain seuil, les deux entrées sont jugées similaires.

Les réseaux siamois ont été étendus et combinés avec d'autres techniques pour améliorer leurs performances et leurs applications. Par exemple, ils peuvent être combinés avec des réseaux de convolution (CNN) pour travailler sur des images, ou des réseaux récurrents (RNN) pour traiter des données séquentielles. Une variante courante, appelée Triplet Network, utilise trois échantillons au lieu de deux pour améliorer l'efficacité de l'apprentissage des représentations discriminatives.

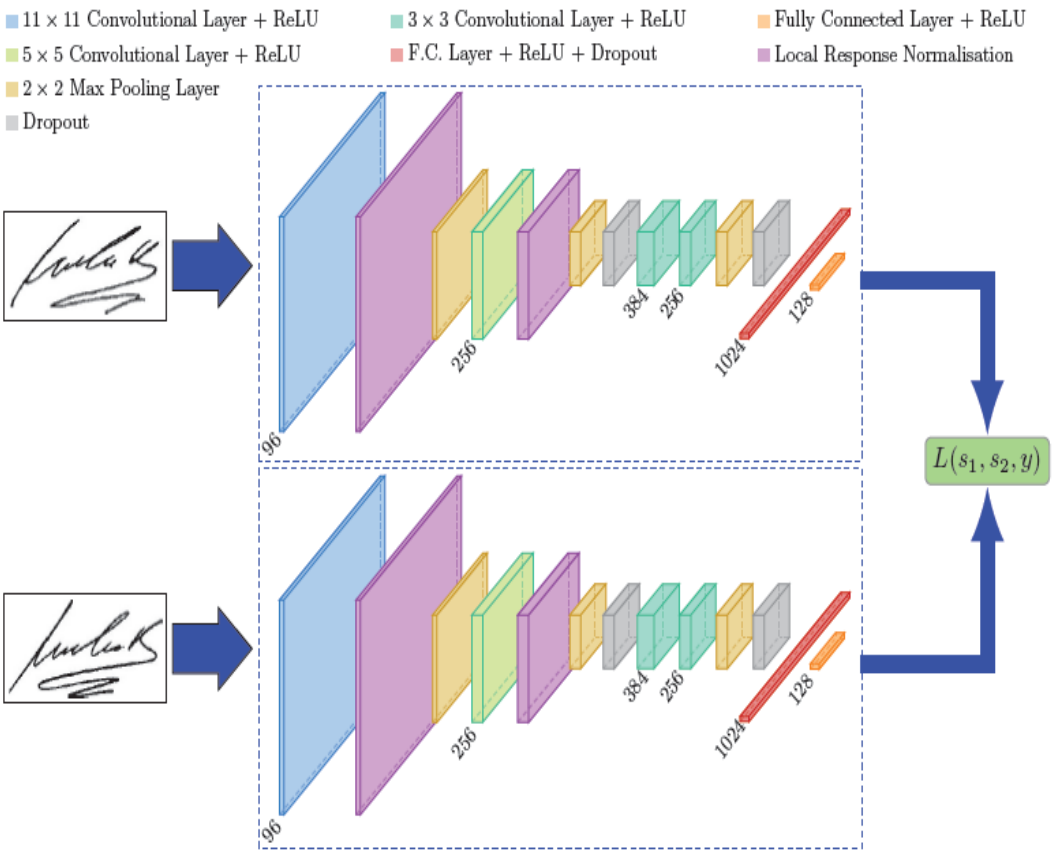

Figure 2.15 : Réseaux siamois[32]

2.4 Stratégies pour Améliorer la Performance et la Robustesse des modèles d'apprentissage machine

L'équilibrage des données et la défense contre les attaques adversaires sont deux approches importantes pour améliorer la performance et la robustesse des modèles d'apprentissage machine. L'équilibrage vise à corriger le biais causé par des classes sous-représentées, tandis que les attaques adversaires permettent de renforcer la résistance des modèles face aux perturbations intentionnelles.

2.4.1 Les attaques contradictoires

Le terme « contradictoire» est couramment employé dans le domaine de la sécurité informatique pour désigner des individus ou des machines qui tentent de pénétrer ou de compromettre un réseau ou un programme informatique. Ces attaques peuvent recourir à diverses méthodes pour perturber le fonctionnement d'un modèle d'apprentissage automatique.

L'équipe de l'apprentissage automatique de l'Université de Berkeley décrit les exemples contradictoires comme étant des « illusions optiques pour les machines. » Il s'agit de données trompeuses conçues pour duper un algorithme en lui faisant croire à quelque chose de faux.

Dans le cadre des algorithmes de classification, un exemple contradictoire représente un ensemble de données synthétiques, méticuleusement élaboré pour provoquer une mauvaise classification. Ces données malicieusement introduites sont souvent constituées de données initialement bien classées, mais auxquelles une perturbation quasiment imperceptible a été ajoutée.

C’est donc très difficile, pour un œil humain, de discerner si un adversaire exemple s’est glissé dans son ensemble de données.[33][34][35].

2.4.1.1 Les méthodes des attaques contradictoires

Les attaques contradictoires sont des techniques utilisées pour tromper les modèles d'apprentissage automatique en modifiant subtilement les données d'entrée. Voici les principales méthodes d'attaques adversaires **:**

2.4.1.1.1 Attaque contradictoire avec FGSM (The Fast Gradient Sign Method) :

La méthode de signe de gradient rapide (FGSM) est une attaque en boîte blanche, ce qui signifie que l'attaque est générée sur la base d'une architecture réseau donnée. Le FGSM est basé sur l'idée que

les réseaux normaux suivent une descente de gradient pour trouver le point de perte le plus bas, et donc si nous suivons le signe du gradient (en allant dans la direction opposée à la descente du gradient), nous pouvons maximiser la perte en ajoutant simplement une petite quantité de perturbation.

- La descente de gradient :

La descente de gradient est un algorithme d'optimisation souvent utilisé pour trouver les poids ou les coefficients des algorithmes d'apprentissage automatique, tels que les réseaux de neurones artificiels et la régression logistique. Cela fonctionne en permettant au modèle de faire des prédictions sur les données d'apprentissage et en utilisant l'erreur sur les prédictions pour mettre à jour le modèle de manière à réduire l'erreur. Le but de l'algorithme est de trouver des paramètres de modèle (par exemple, des coefficients ou des poids) qui minimisent l'erreur du modèle sur le jeu de données d'apprentissage. Pour ce faire, il modifie le modèle en le déplaçant le long d'une pente d'erreur vers une valeur d'erreur minimale. Cela donne à l'algorithme le nom de « descente de gradient ». Il existe trois variantes de cette méthode :

• Batch gradient descent.

• Stochastic gradient descent.

• Mini-batch gradient descent .[36]

Lorsque vous demandez à un humain de décrire comment il détecte un macaw dans une image, il peut rechercher des caractéristiques physiques telles que les ailes, Le bec, plumes, la queue ou la couleur. Elle pourrait également donner un autre contexte, comme le type d'habitat dans lequel elle s'attendrait à voir le macaw.

Pour un réseau de neurones artificiels, tant que l'exécution des valeurs de pixels à travers l'équation fournit la bonne réponse, il est convaincu que ce qu'il voit est bien un macaw. En d'autres termes, en modifiant correctement les valeurs de pixels de l'image, vous pouvez tromper l'IA en lui faisant croire qu'elle ne voit pas de macaw. Les chercheurs en IA ont ajouté une couche de bruit à l'image. Ce bruit est à peine perceptible à l'œil humain. Mais lorsque les nouveaux numéros de pixels passent par le réseau neuronal, ils produisent le résultat attendu de l'image d’une bibliothèque.[37][38]

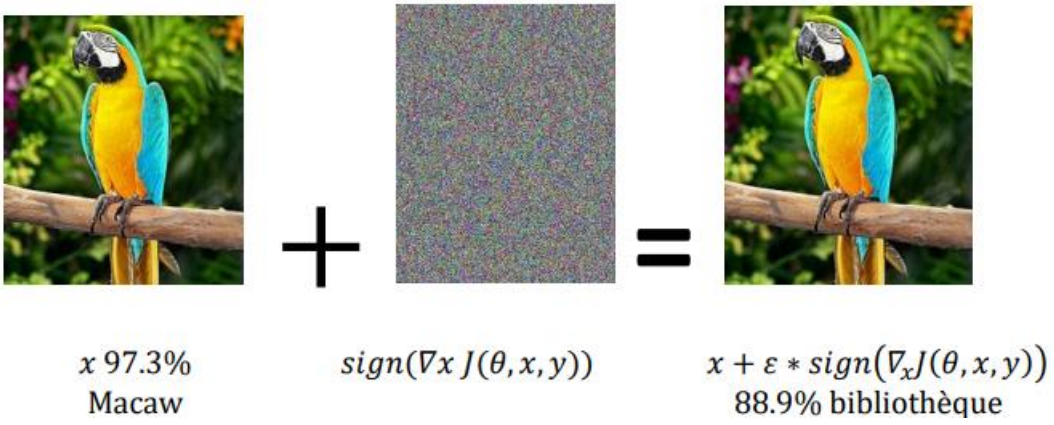

Figure 2.16 : Une démonstration de la génération rapide d'exemple FGSM de confrontation appliquée sur image[40]

FGSM peut donc être décrit comme l'expression mathématique suivante :

$$x' = x + \varepsilon * sign(\nabla x J(\theta, x, y))$$

$\boldsymbol{x'}$: Notre image contradictoire de sortie.

$\boldsymbol{x}$: L'image d'entrée d'origine.

$\boldsymbol{y}$: : l'étiquette de vérité terrain de l'image d'entrée.

$\boldsymbol{\varepsilon}$: Petite valeur nous multiplions les gradients signés par pour nous assurer que les perturbations sont suffisamment petites pour que l'œil humain ne puisse pas les détecter mais suffisamment grandes pour tromper le réseau neuronal.

$\boldsymbol{\theta}$: Notre modèle de réseau neuronal.

$\boldsymbol{J}$: La fonction de perte.

La création d'exemples d'apprentissage machine contradictoire est un processus d'essais et d'erreurs. De nombreux modèles d'apprentissage automatique de classificateurs d'images fournissent une liste de résultats avec leur niveau de confiance (par exemple, Macaw = 90%, perruche= 50%, pigeon= 15%, etc.).[39][40]

La création d'exemples contradictoires implique de faire de petits ajustements aux pixels de l'image et de les réexécuter via l'IA pour voir comment la modification affecte les scores de confiance. Avec suffisamment de réglages, vous pouvez créer une carte de bruit qui réduit la confiance dans une classe et l'augmente dans une autre. Ce processus peut souvent être automatisé.

2.4.1.1.2 L'attaque PGD (Projected Gradient Descent)

L'attaque PGD est une méthode couramment utilisée pour générer des exemples adverses afin de tromper des modèles d'apprentissage profond. Elle est considérée comme l'une des attaques contradictoires les plus puissantes et est particulièrement efficace contre les réseaux de neurones entraînés avec des approches de défense traditionnelles [41].

L'attaque PGD est une version itérative de l'attaque FGSM (Fast Gradient Sign Method). Elle consiste à appliquer plusieurs petites perturbations successives à l'entrée, dans le but de faire classer le modèle de manière incorrecte. Voici le fonctionnement général de l'attaque PGD :

Initialisation de l'Entrée Perturbée :

- L'attaque commence par ajouter une petite perturbation aléatoire à l'entrée originale.

Mise à Jour Itérative:

- À chaque itération, la perturbation est mise à jour en utilisant le gradient de la fonction de perte par rapport à l'entrée.

- Contrairement à FGSM qui applique la perturbation en une seule étape, l'attaque PGD met à jour l'entrée à chaque itération pour s'approcher progressivement de la frontière de décision du modèle.

Projection dans la Zone Admissible :

- Après chaque mise à jour, l'entrée perturbée est projetée dans une région admissible (typiquement une "balle" normée autour de l'entrée originale) afin de maintenir les perturbations dans une certaine limite. Cela garantit que la perturbation reste petite et que les exemples adverses ne s'écartent pas trop de l'entrée originale.

Formulation Mathématique :

La mise à jour de l'entrée perturbée X' à chaque itération t est donnée par la formule suivante :

$$x_{t+1} = Proj_{\beta(x,\varepsilon)}(x_t + \alpha.\sin(\nabla_{xt} L\,(\theta, x_t, y)))$$

- $Proj_{\beta(x,\varepsilon)}$: Projection sur une région admissible définie par une norme.

-L(θ, x_t, y) **:** Fonction de perte du modèle avec les paramètres θ, l'entrée x et l'étiquette y.

- α : Pas de l'attaque, contrôlant la taille de chaque mise à jour.

L'objectif de PGD est de trouver la perturbation minimale qui entraîne une classification erronée par le modèle. PGD est souvent utilisé pour tester la robustesse des modèles, car il simule des attaques réalistes auxquelles les systèmes pourraient être confrontés.

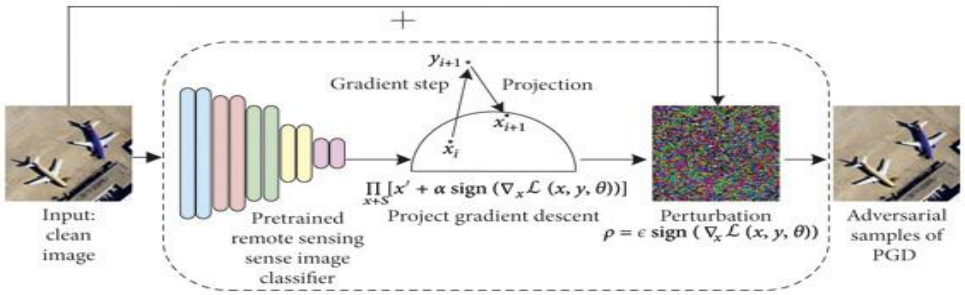

Figure 2.17 : L'attaque contradictoire par Projected Gradient Descent (PGD)[42]

La figure 2.17 illustre le fonctionnement de l'attaque PGD, à partir d'une image propre (une photo aérienne d'un avion), le modèle pré-entraîné produit une prédiction correcte. PGD va ensuite perturber cette image de manière progressive en appliquant plusieurs petites modifications successives, guidées par le gradient de la fonction de perte : à chaque étape, on ajoute une petite quantité $(\alpha.\sin(\nabla_{xt}L(\theta, x_t, y)))$ dans la direction qui maximise l'erreur du modèle. Le résultat final est une image presque identique visuellement, mais qui trompe totalement le classifieur, illustrant la fragilité des modèles d'apprentissage profond face à des attaques ciblées.

2.4.2 L'équilibrage de données

L'équilibrage de données est une technique utilisée en apprentissage automatique pour traiter des ensembles de données déséquilibrés, c'est-à-dire lorsque certaines classes sont significativement moins représentées que d'autres. Ce déséquilibre peut créer un biais dans l'entraînement des modèles, car les classes majoritaires dominent la fonction de perte, rendant le modèle moins performant pour prédire correctement les classes minoritaires[43].

2.4.2.1 Méthodes d'équilibrage des données

Il existe plusieurs méthodes pour équilibrer les données :

2.4.2.1.1 Sous-échantillonnage (Under-sampling)

Le sous-échantillonnage consiste à réduire la fréquence des exemples de la classe majoritaire pour équilibrer l'ensemble de données. Cela peut être fait en supprimant aléatoirement des exemples de la classe majoritaire, ce qui peut toutefois entraîner une perte d'information(voir figure 2.18).

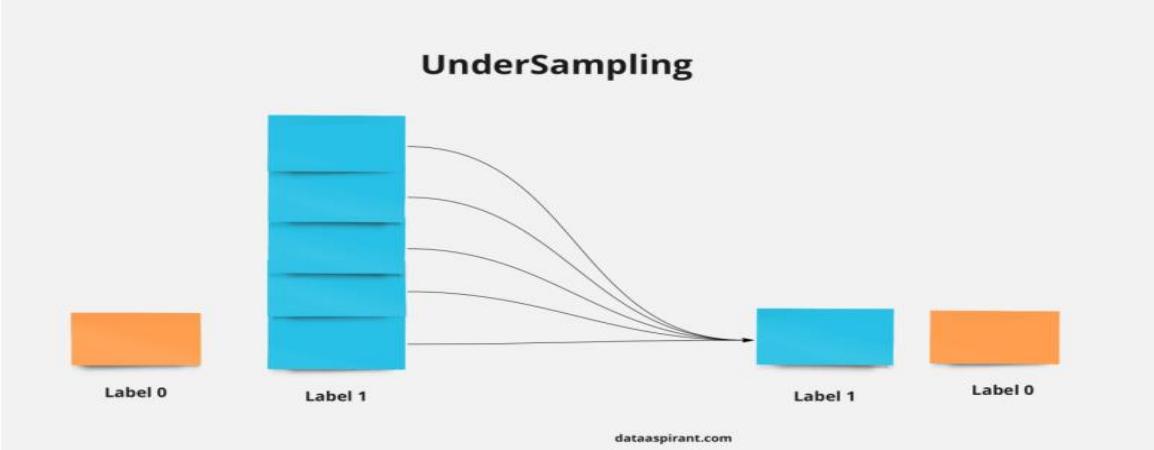

Figure 2.18 : Sous echantillonnage[43]

2.4.2.1.2 Sur-échantillonnage (Over-sampling)

Le sur-échantillonnage consiste à augmenter la fréquence des exemples de la classe minoritaire tel qu'illustré dans la figure 2.19. Une méthode courante est SMOTE (Synthetic Minority Over-sampling Technique), qui crée des échantillons synthétiques en interpolant les points existants de la classe minoritaire. Cette technique permet de créer de nouveaux exemples artificiels pour la classe minoritaire afin de rétablir l'équilibre [43].

SMOTE est un algorithme de suréchantillonnage qui génère des observations synthétiques basées sur les observations des classes minoritaires existantes. En fonction de la quantité de suréchantillonnage nécessaire, SMOTE calcule les k plus proches voisins pour créer des exemples synthétiques.

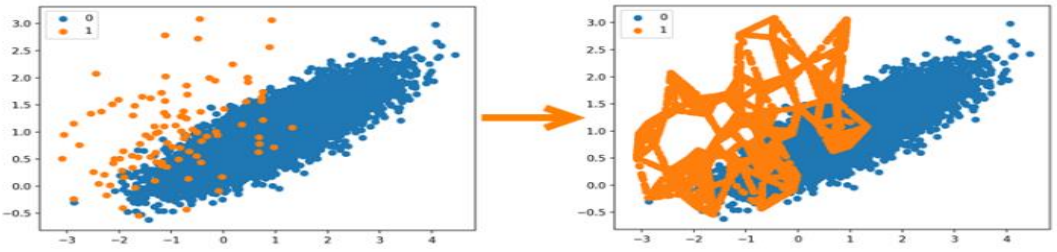

Figure 2.19 : Le sur-échantillonage via SMOTE (Oversampling Technique) [43]

3.1 Métriques d'évaluation de performance :

Les principaux aspects à prendre en compte lors de l'évaluation de la détection et de la précision de classification des attaques sont :

- True Positive (TP): nombre d'intrusions correctement détectées

- True Negative (TN): nombre de non-intrusions correctement détectées

- False Positive (FP) : nombre de non-intrusions mal détectées

- False Negative (FN): nombre d'intrusions mal détectées

Il existe différents types d'erreurs provenant d'un détecteur, affectant plus ou moins son efficacité. Les vrais positifs sont les cas où une alarme se déclenche lorsqu'il y a une violation des politiques de sécurité. Les vrais négatifs se produisent lorsque rien d'anormal ne se passe et qu'aucune alarme ne se déclenche. Les faux positifs se produisent lorsqu'une alarme se déclenche alors qu'il n'y a rien d'anormal. Les faux négatifs sont les cas où aucune alarme ne se déclenche alors qu'une activité anormale se produit. À première vue, on pourrait supposer qu'un faux positif est moins dangereux qu'un faux négatif [46].

Les critères utilisés pour évaluer l'efficacité globale des systèmes de détection d'intrusions, selon Debar et al. [44], sont les suivants :

- **Précision (P) :** Un IDS est considéré précis lorsqu'il détecte les attaques sans générer de fausses alertes. Un manque de précision se produit lorsqu'une action légitime dans l'environnement est incorrectement signalée comme suspecte ou anormale.

$$P = \frac{TP}{TP + FP}$$

-**Accuracy (ACC) :** La capacité d'un système d'intrusion à faire des prédictions ou des décisions correctes sur la base des données sur lesquelles il a été entraîné.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Recall (R):** le pourcentage des attaques identifiées comme des attaques TP parmis tous les attaques dans l'ensemble de données :

$$R = \frac{TP}{TP + FN}$$

- **F1-score (F1)** : la moyenne harmonique pondérée de précision et de rappel (Recall),

il est donné par :

$$F1 = \frac{2 * P * R}{P + R}$$

Généralement, un taux de détection élevé est essentiel pour un système IDS afin de prévenir les attaques avant qu'elles ne causent des violations de sécurité. Pour les systèmes IDS basés sur l'apprentissage automatique, une haute précision de détection avec un faible taux de fausses alertes est essentielle à leur efficacité [45].

3.2 Conclusion

En conclusion, dans ce chapitre, nous avons abordé les différents mécanismes de détection utilisés en cybersécurité, en mettant un accent particulier sur les systèmes de détection d'intrusion basés sur l'apprentissage automatique. Nous avons également discuté des stratégies visant à améliorer la performance et la robustesse de ces modèles, notamment par l'équilibrage des données et l'empoisonnement des données. Ces approches sont essentielles pour garantir une détection efficace et résiliente face aux menaces de plus en plus complexes. Dans le prochain chapitre, nous aborderons les travaux connexes ainsi que les jeux de données existants utilisés dans ce domaine.

Chapitre 3 : Revue des Méthodes et Jeux de Données en Détection des Intrusions

3.1 Introduction

La détection des intrusions est un domaine de recherche en pleine croissance qui est essentiel pour assurer la sécurité des systèmes informatiques face aux menaces croissantes dans le cyberespace. Alors que les cyberattaques se multiplient et que la quantité de données échangées sur les réseaux augmente, le développement de solutions efficaces pour identifier et répondre aux activités malveillantes est devenu une priorité.
Les chercheurs de la cybersécurité travaillent d'arrache-pied pour concevoir des mécanismes avancés permettant de détecter et de prévenir les intrusions en temps réel, en s'appuyant sur des méthodes de plus en plus sophistiquées, notamment l'intelligence artificielle et l'apprentissage automatique.

Les travaux menés dans ce domaine ont conduit à la mise en œuvre de différents systèmes de détection d'intrusion basés sur diverses techniques telles que la détection basée sur les signatures, l'analyse comportementale ou encore des modèles hybrides combinant plusieurs stratégies. L'efficacité de ces systèmes dépend en grande partie des ensembles de données sur lesquels ils sont formés et évalués. En fait, l'obtention d'ensembles de données de haute qualité représentant des menaces réelles est un facteur clé dans le développement et l'amélioration des modèles de détection.

Dans ce chapitre, nous passons en revue les principales recherches menées dans le domaine de la détection d'intrusion, en nous concentrant sur les méthodes utilisées. Nous passerons également en revue les principaux ensembles de données utilisés dans ce contexte, en soulignant leur importance dans la formation et l'évaluation des modèles de détection.

3.2 Travaux connexes

3.2.1 Travaux connexes à l'apprentissage automatique supervisé

Les méthodes de détection d'anomalies de trafic basées sur l'apprentissage automatique ont rencontré un grand succès. Dans [47], les auteurs proposent une nouvelle méthode de sélection de caractéristiques et de classification basée sur la machine à vecteurs de support (SVM). Les résultats expérimentaux sur le jeu de données NSL-KDD Cup 99 de détection d'intrusions ont montré que

l'exactitude de classification de cette méthode avec toutes les caractéristiques d'entraînement a atteint 99%.

Dans l'étude menée par les auteurs de [44], l'accent a été mis sur la classification des intrusions à l'aide du jeu de données NSL-KDD. Le processus débute par un prétraitement des données afin de résoudre les problèmes de données manquantes et de catégoriser les attributs numériques. Par la suite, les données sont subdivisées en quatre groupes distincts, puis partitionnées en ensembles d'entraînement et de test. Un classificateur de forêt aléatoire est ensuite utilisé pour évaluer les performances, notamment à travers la précision et le taux de fausses alertes (FPR). Une méthode de sélection des caractéristiques basée sur la mesure d'incertitude symétrique est également appliquée, ce qui a permis d'améliorer légèrement les performances. Les résultats obtenus ont été comparés à ceux de l'algorithme C4.5, où la forêt aléatoire a démontré de meilleures performances. Après la sélection des caractéristiques, une précision moyenne de 99,67 % et un taux de fausses alertes de 0,5% ont été rapportés.

De manière similaire, l'étude de [37] a analysé les performances de dix algorithmes de classification sur le jeu de données NSL-KDD, en adoptant une approche différente pour la sélection des caractéristiques. Cette méthode repose sur l'utilisation d'évaluateurs d'attributs et d'un processus de filtrage. Les algorithmes testés incluent notamment Naïve Bayes, Bayes-Net, Logistique, Random Tree, Random Forest, J48, Bagging, OneR, PART et ZeroR. Parmi eux, la forêt aléatoire s'est distinguée avec une précision de 99,9 % et un faible taux de fausses alertes de 0,1%. Bagging se classe en deuxième position avec une précision de 99,8 %, suivi de l'algorithme PART, qui présente des résultats similaires.

Dans une autre étude [38], le jeu de données NSL-KDD a été utilisé sans appliquer de sélection de caractéristiques, dans le cadre d'une validation croisée par rétention (hold-out). Quatre algorithmes de classification ont été évalués : forêt aléatoire, SVM, régression logistique et modèle de mélange Gaussien. La forêt aléatoire a démontré les meilleures performances avec une précision de 99 %, suivie par la régression logistique, qui a obtenu une précision de 84 %.

Plus récemment, les auteurs de [39] ont examiné les performances des réseaux de neurones artificiels (ANN) et des SVM sur un échantillon représentant 20 % du jeu de données NSL-KDD. Deux méthodes de sélection de caractéristiques ont été testées : l'une basée sur la corrélation et l'autre sur le test du chi-carré. La première a retenu 17 caractéristiques, tandis que la seconde en a sélectionné 35. Après la sélection, les données ont été introduites dans les classificateurs ANN et SVM. Les meilleures performances ont été obtenues avec la méthode basée sur la corrélation et les ANN, atteignant une précision de 94 %.

Dans une autre approche, les auteurs de [40] ont utilisé le jeu de données KDD'99 pour classifier des attaques dans des catégories telles que normal, Prob, DoS, U2R et R2L. Deux algorithmes de sélection des caractéristiques, CFSSubSet Eval et Best First, ont été appliqués en combinaison avec une méthode non supervisée (k-means) et trois méthodes supervisées (SVM, Naïve Bayes et forêt aléatoire). Les méthodes supervisées, basées sur les huit meilleures caractéristiques, ont surpassé l'approche non supervisée, la forêt aléatoire atteignant une précision de 99 %.

Le jeu de données KDD'99 a également été analysé par les auteurs de [48], qui ont utilisé l'algorithme des colonies de fourmis pour sélectionner un sous-ensemble représentatif de 550 échantillons. Ensuite, une méthode innovante appelée Gradual Feature Removal (GFR) a réduit la dimensionnalité à 19 caractéristiques, utilisées avec SVM pour la classification. La précision avant sélection des caractéristiques était de 98,67 %, et après réduction, elle a légèrement diminué à 98,62 %.

Dans l'étude de [49], l'effet de la réduction de la dimensionnalité sur les performances globales de classification a été examiné. Les auteurs ont appliqué un algorithme de sélection basé sur le gain d'information (IG), combiné à un réseau neuronal à rétropropagation (BBNN). Les résultats montrent que la précision reste constante à 91 %, même après réduction des caractéristiques.

Par ailleurs, le jeu de données CICIDS2017 a également suscité de l'intérêt. Par exemple, les travaux de [30, 33] ont proposé une approche combinant échantillonnage, réduction des caractéristiques et boosting pour la détection d'intrusions. Les données ont été prétraitées à l'aide de SMOTE afin de résoudre le problème de déséquilibre des classes, puis une réduction basée sur l'ensemble de sélection de caractéristiques (EFS) et l'analyse en composantes principales (PCA) a réduit l'espace à 25 caractéristiques. Avec Adaboost et une validation hold-out, une précision de 81,83 % a été rapportée.

Enfin, dans une étude récente [34], les auteurs ont ciblé la classification des attaques en utilisant des techniques de forêt aléatoire et ANN sur le jeu de données CICIDS2017. Ils ont appliqué le package Boruta pour sélectionner les 10 caractéristiques les plus importantes, utilisées ensuite pour entraîner les classificateurs. La précision moyenne signalée est de 96 % avec les ANN et de 96,4 % avec la forêt aléatoire.

3.2.2 Travaux connexes à l'apprentissage profond

Des recherches antérieures ont démontré que les méthodes d'apprentissage profond (DL) surpassent d'autres algorithmes d'apprentissage automatique, tels que les machines à vecteurs de support (SVM) et les réseaux neuronaux artificiels traditionnels (ANN), pour la détection des anomalies. L'utilisation de l'apprentissage profond pour la détection d'intrusions a débuté en 2011, lorsque Salama et al [50], ont proposé une approche hybride combinant les réseaux de croyances profondes (DBN) et les SVM pour classifier les intrusions en deux catégories : normale ou attaque. Le réseau DBN, constitué de plusieurs couches de machines de Boltzmann restreintes (RBM), a été utilisé comme méthode de réduction de dimension pour extraire des caractéristiques d'apprentissage optimales, suivie par une classification binaire (normale/attaque) avec un SVM.

Les performances de cette approche DBN-SVM ont été évaluées sur le dataset NSL-KDD, où le nombre de caractéristiques a été réduit de 41 à 5 grâce au DBN, avant d'être classées par le SVM.

La méthode a atteint une précision supérieure à 90%. Les résultats ont également montré que l'utilisation du DBN pour la réduction des caractéristiques est plus efficace que d'autres méthodes d'analyse de données telles que l'ACP, le Gain Ratio et le Chi-Square.

Kang et al [51], ont proposé un système de détection d'intrusion basé sur un réseau neuronal profond (DNN) pour sécuriser les réseaux de véhicules. Le scénario d'attaque simulé consistait à injecter des paquets de données malveillants dans un bus de réseau de contrôleurs de véhicules. Le système proposé utilise un vecteur de caractéristiques introduit dans les nœuds d'entrée afin de classer les paquets en deux catégories : paquets normaux et paquets d'attaque. Les auteurs ont employé un réseau de croyances profondes (DBN) pour former efficacement les paramètres des poids cachés, servant ainsi à initialiser le réseau DNN. Les couches cachées suivantes sont connectées à ces sorties, calculées à l'aide de la fonction d'activation ReLU. Le système proposé a atteint un taux de fausse alerte (FPR) inférieur à 1 ou 2 %, et un taux de détection (DR) de 99 %.

Sandee et al [52], ont proposé un système NIDS basé sur un DNN, composé d'un auto-encodeur parcimonieux pour l'apprentissage non supervisé des caractéristiques, et d'une régression logistique pour la classification binaire sur l'ensemble de données NSL-KDD (normal/intrusion). Le système prend en compte 115 caractéristiques en entrée. L'auto-encodeur parcimonieux a été utilisé pour former et apprendre de nouvelles caractéristiques, réduisant ainsi leur nombre à 50, puis à 10. Ces caractéristiques réduites ont ensuite été attribuées au classificateur de régression logistique pour la

classification. La performance du système a été évaluée en termes de précision, de rappel et de taux de reconnaissance. Le modèle a atteint une précision globale de 87,2 %.

Tang et al [53] ont utilisé un réseau neuronal profond (DNN) pour implémenter un IDS dans un contrôleur SDN (Software-Defined Network) capable de surveiller tous les flux des commutateurs OpenFlow. Le modèle a été entraîné sur l'ensemble de données NSL-KDD pour une classification binaire (normal/anomalie), comprenant quatre types d'attaques : DoS, R2L, U2R, et probe. Ils ont sélectionné uniquement 6 caractéristiques parmi les 41 disponibles. Le modèle a été optimisé en ajustant l'hyper-paramètre, ce qui a permis d'obtenir des résultats de classification optimaux. Pour cela, ils ont testé différents taux d'apprentissage allant de 0,1 à 0,0001. Les performances se sont améliorées progressivement à mesure que le taux d'apprentissage diminuait de 0,1 à 0,001, avec les meilleurs résultats obtenus à un taux de 0,001. Le modèle a atteint une précision de 75,75 %. Leurs expériences confirment que l'approche DNN présente un fort potentiel pour la détection d'anomalies dans les NIDS.

Javaid et al[54] ont proposé un système de détection d'intrusion réseau (NIDS) basé sur le deep learning, en utilisant l'apprentissage autodidacte (Self-Taught Learning, STL). Cette méthode se déroule en deux étapes pour la classification. La première étape, appelée apprentissage de caractéristiques (non supervisé), consiste à extraire une bonne représentation des caractéristiques à partir d'une vaste collection de données non étiquetées. Dans la seconde étape, cette représentation apprise est appliquée aux données étiquetées et utilisée pour la tâche de classification. L'approche STL a été testée sur le jeu de données de référence NSL-KDD pour la détection d'anomalies réseau, visant une classification multi-classes entre le trafic normal et différents types d'attaques. Les meilleurs résultats obtenus affichent une précision de 85,44 % et un rappel de 95,95 % pour une classification binaire (trafic normal/trafic malveillant) lors de l'application aux données de test. Bien que cette approche STL présente un taux de faux positifs notable, elle surpasse les performances de la simple régression softmax (un algorithme de machine learning pour la classification multi-classes) et d'autres travaux antérieurs.

Il existe quelques études qui ont exploré l'utilisation des réseaux de neurones convolutifs (CNN) dans le domaine de la détection d'intrusion. Le principal avantage des CNN réside dans leur capacité à partager les mêmes noyaux de convolution, ce qui permet de réduire le nombre de paramètres et la charge de calcul lors de l'apprentissage. Une fois que le système est capable d'identifier rapidement le type d'attaque dans les données de trafic, il devient plus facile de détecter et d'identifier ces données de manière efficace .

Vinayakumar et al. [55] ont étudié l'efficacité des réseaux de neurones convolutifs (CNN) pour la détection d'intrusions en modélisant les événements de trafic réseau sous forme de séries temporelles de paquets TCP/IP sur des périodes de temps définies. Les auteurs ont adopté la méthode CNN appliquée au traitement du langage naturel avec une couche Convolution 1D . Sur cette base, ils ont modélisé les événements de trafic réseau comme une série chronologique de données, où, au lieu d'utiliser des images 2D en entrée, le CNN traite une séquence de données en 1D, organisée dans un intervalle de temps. Les auteurs ont proposé différentes architectures CNN, comprenant une couche d'entrée, une ou plusieurs couches CNN dans la couche cachée, suivies de réseaux FFN, RNN, LSTM ou GRU pour déterminer l'architecture optimale. Toutes les expériences ont été menées sur 1000 époques, et les performances de chaque modèle ont été évaluées sur l'ensemble de test du jeu de données KDD Cup 99. Les résultats ont montré que le CNN-LSTM a surpassé les autres structures de réseaux CNN en atteignant une précision de 99 %.

Selon les auteurs, les algorithmes CNN ont dépassé les résultats de la compétition KDD Cup 99 et d'autres travaux publiés sur la détection d'intrusions.

Kim et al. [56] ont proposé un système de détection d'intrusions (IDS) basé sur les réseaux neuronaux récurrents (RNN). Les paquets TCP/IP du trafic réseau ont été représentés sous forme de séquences. Les auteurs ont utilisé la méthode d'optimisation sans hessien (Hessian-Free Optimisation) pour surmonter le problème d'explosion ou de disparition des gradients, ce qui aide l'algorithme de gradient à trouver le minimum global pendant l'entraînement, améliorant ainsi la précision. Le modèle proposé a été évalué à l'aide de l'ensemble de données DARPA, comprenant 41 caractéristiques et 22 types d'attaques différents. Le taux de détection était de 95,37 %, avec un taux de fausses alertes de 2,1 %. La précision de classification pour chaque type d'attaque était satisfaisante, et les auteurs concluent que l'utilisation de RNN avec l'optimisation sans hessien pour la détection des intrusions est une approche prometteuse.

GAO et al. [57] ont appliqué le Deep Belief Network (DBN) au domaine de la détection d'intrusions. L'architecture de ce modèle est une combinaison d'un réseau d'apprentissage non supervisé à plusieurs couches, appelé "machine de Boltzmann restreinte" (Restricted Boltzmann Machine), et d'un réseau d'apprentissage supervisé, appelé réseau de rétro-propagation. Les résultats expérimentaux sur l'ensemble de données KDDCUP99 montrent que les performances du modèle DBN surpassent celles du SVM et du réseau neuronal artificiel (ANN).

Rezvy et al. [58] ont adopté une approche basée sur l'AutoEncoder, utilisé comme un constructeur de caractéristiques. Le modèle a d'abord été pré-entraîné avec l'AutoEncoder pour reconfigurer les caractéristiques, puis un classificateur DNN a été utilisé pour classer le trafic. Les auteurs ont testé le modèle sur le jeu de données NSL-KDD'99 en utilisant cinq classes. L'AutoEncoder a reconstruit les 122 caractéristiques, qui ont ensuite été intégrées dans un DNN à trois couches. Ce DNN sert de classificateur pour les cinq classes. La précision des différentes classes varie de 89,2% à 99,9%, avec une précision globale de 99,3% pour la détection des attaques.

Alom et al. [59] ont utilisé un réseau de croyance profond (Deep Belief Network) pour la détection des intrusions. Le système proposé est capable de détecter les attaques et de classer avec précision les activités réseau en cinq groupes, en se basant sur des sources de données limitées, incomplètes et non linéaires. Par rapport aux systèmes existants, le taux de détection du système atteint 97,5 % après seulement 50 itérations.

Wu et al. [60] ont proposé un modèle de détection d'intrusion utilisant des réseaux neuronaux convolutifs (CNN). Le CNN est employé pour sélectionner automatiquement les caractéristiques du trafic à partir d'un ensemble de données brutes. Les auteurs ont utilisé l'ensemble de données NSL-KDD pour l'évaluation et ont abordé le problème des ensembles de données déséquilibrés en ajustant les coefficients de pondération de la fonction de coût en fonction du nombre d'exemples de chaque classe. Afin de réduire le coût de calcul, ils ont converti les vecteurs de trafic brut en format image. Le modèle proposé améliore la précision pour les classes rares tout en réduisant le taux de fausses alertes (FAR).

Torres et al. [61] ont examiné la viabilité des réseaux neuronaux récurrents (RNN) pour détecter les comportements du trafic réseau en le modélisant comme une séquence d'états évoluant avec le temps. Ils ont proposé et évalué la méthode de détection LSTM sur deux ensembles de données différents : CTU13-42 et CTU13-47, chacun contenant deux classes : connexion au botnet et connexion normale. Un botnet est un groupe d'ordinateurs compromis pouvant être contrôlés à distance pour exécuter des attaques coordonnées ou commettre des actes frauduleux. Les auteurs ont aussi analysé le nombre d'états par connexion pour le modèle comportemental et ont trouvé que le RNN est capable de classer le trafic avec un taux de détection des attaques (ADR) de 97% et un taux de fausses alertes (FAR) très faible de 1.8%. L'impact des techniques d'échantillonnage (sur-échantillonnage/sous-échantillonnage) pour traiter les classes déséquilibrées a également été analysé, montrant que l'ADR ne présente pas de différence significative dans les trois cas évalués. Cependant, les expérimentations ont révélé que les modèles de détection RNN rencontrent des

difficultés pour traiter les comportements du trafic difficiles à différencier, ainsi que certains cas particuliers de trafic déséquilibré.

Le tableau 1 présente une sélection d'études portant sur la détection des intrusions basées sur des modèles DL, les caractéristiques utilisées, le secteur d'intérêt, les ensembles de données et les mesures de performance.

Methode	**Data set**	**Messurede performance**	**Auteurs**
Svm	NSL-KDD CUP 99	Acc=99%	Pervez et al [47]
DBN-SVM	NSL-KDD	Acc=90%	Salama et al [50]
DNN-DBN	Vehicular network communication	DR=99%-99.46% FPR=1-2%	Kang et al [51]
DBN	NSL-KDD	Acc=97.5%	Alom et al[59]
CNN	NSL-KDD	Acc=97.88%-99.46% DR=68.66% FPR=27.9%	Wu et al[60]
STL	KDD 99	ACC=79.10%-88.39% PR=85.44% RC=95.95% F1=90.39%	Javid et al[54]
RNN	DARPA	DR=95.37% FPR=2.1%	Kim et al[56]
RNN-LSTM	CTU-13	DR=97.96% FPR=2.27%	Torres et al [61]
CNN-RNN	KDD 99	ACC=99.99% PR=99.99% RC=99.99% F1=99.99%	Vinayak et al[55]

Deep Auto Encoder	NSL-KDD	ACC=99.99% PR=84.6% RC=92.8% Specificity=80.7%	Sandee et al [52]
DNN	NSL-KDD	ACC=75.75% PR=83% RC=76% F1=75% ROC-AUC=86%	Tang et al [53]
Deep Auto Encoder	NSL-KDD	ACC=99.3%	Rezvy et al [58]
DBN	KDD 99	ACC=74.22%-93.49% DR=75.6%-92.33% FAR=3.15%-0.76%	Gao et al [57]

Tableau 3.1 :Travaux antérieurs connexes pour la détection d'intrusion basé sur le deep learning

3.3 Les jeux de données

3.3.1 Darpa-KDDCUP99

La première tentative de création d'un jeu de données IDS a été réalisée par la DARPA (Defence Advanced Research Project Agency) en 1998, lorsqu'ils ont créé le jeu de données KDD98 (Knowledge Discovery and Data Mining). En 1998, la DARPA a lancé un programme au MIT Lincoln Labs afin de fournir un environnement complet et réaliste pour l'évaluation des IDS (MIT Lincoln Laboratory, 1999). Bien que ce jeu de données ait été une contribution importante à la recherche sur les IDS, son exactitude et sa capacité à refléter les conditions réelles ont été largement critiquées[62]

Ces jeux de données ont été collectés à l'aide de plusieurs ordinateurs connectés à Internet pour modéliser une petite base de l'armée de l'air américaine avec un personnel restreint. Des paquets réseau et des fichiers journaux des hôtes ont été collectés. Lincoln Labs a mis en place un banc d'essai expérimental pour obtenir deux mois de captures de paquets TCP dans un réseau local (LAN), modélisant un réseau local typique de l'armée de l'air américaine. Ils ont simulé ce réseau comme s'il s'agissait d'un véritable environnement militaire, en y intégrant plusieurs intrusions simulées[62].

Les paquets réseau collectés représentaient environ quatre giga-octets de données, contenant environ 4 900 000 enregistrements. Les données de test sur deux semaines comprenaient environ 2 millions d'enregistrements de connexions, chacun comportant 41 caractéristiques et étant classé comme normal ou anormal.

Les données extraites sont une série de sessions TCP débutant et se terminant à des moments bien définis, au cours desquelles des données circulent entre une adresse IP source et une adresse IP cible, contenant une grande variété d'attaques simulées dans un environnement militaire. Le jeu de données DARPA de 1998 a servi de base à la création du jeu de données KDD Cup99, utilisé lors de la troisième compétition internationale d'outils de découverte de connaissances et d'exploration de données (KDD, 1999). Les 41 caractéristiques du jeu de données KDD Cup99 sont présentées dans le tableau

Ces jeux de données sont aujourd'hui obsolètes, car ils ne contiennent pas de traces des attaques récentes de logiciels malveillants. Par exemple, les comportements des attaquants varient en fonction des topologies de réseau, des systèmes d'exploitation, des logiciels et des kits d'outils criminels.

La figure 3.1 illustre les différentes classes d'attaques de ce dataset, en les classifiant en Probe, DoS, R2L et U2R, avec les types d'attaques correspondants à chaque catégorie.

Attack Class	Attack Type
Probe	portsweep, ipsweep, queso, satan, msscan, ntinfoscan, lsdomain, illegal-sniffer
DoS	apache2, smurf, neptune, dosnuke, land, pod, back, teardrop, tcpreset, syslogd, crashiis, arppoison, mailbomb, selfping, processtable, udpstorm, warezclient
R2L	dict, netcat, sendmail, imap, ncftp, xlock, xsnoop, sshtrojan, framespoof, ppmacro, guest, netbus, snmpget, ftpwrite, httptunnel, phf, named
U2R	sechole, xterm, eject, ps, nukepw, secret, perl, yaga, fdformat, ffbconfig, casesen, ntfsdos, ppmacro, loadmodule, sqlattack

Figure 3.1 : Les types d'attaques du jeux de données DARPA-KDDCUP99[63]

3.3.2 CAIDA

Ce jeu de données contient des traces de trafic réseau issues d'attaques par déni de service distribué (DDoS) et a été collecté en 2007. Ce type d'attaque vise à interrompre le trafic normal d'un ordinateur ou d'un réseau cible en le submergeant avec un flux massif de paquets réseau, empêchant ainsi le

trafic légitime d'atteindre sa destination. Un inconvénient du jeu de données CAIDA est qu'il ne contient pas une grande diversité d'attaques. De plus, les données collectées ne comprennent pas des caractéristiques issues de l'ensemble du réseau, ce qui rend difficile la distinction entre les flux de trafic normaux et anormaux[64].

3.3.3 NSL-KDD

NSL-KDD est un jeu de données public, développé à partir du précédent jeu de données KDD Cup99 . Une analyse statistique réalisée sur le jeu de données KDD Cup99 a révélé d'importants problèmes qui influencent fortement la précision de la détection d'intrusions et conduisent à une évaluation trompeuse des IDS .

Le principal problème du jeu de données KDD est la quantité massive de paquets dupliqués. Tavallaee et son équipe ont analysé les ensembles d'entraînement et de test du jeu de données KDD et ont découvert qu'environ 78 % des paquets réseau dans l'ensemble d'entraînement et 75 % dans l'ensemble de test étaient des doublons . Cette grande quantité d'instances dupliquées dans l'ensemble d'entraînement biaise les méthodes d'apprentissage automatique en faveur des instances normales, ce qui les empêche d'apprendre correctement à reconnaître les instances irrégulières qui sont généralement plus nuisibles pour le système informatique. Pour résoudre ces problèmes, Tavallaee et son équipe ont créé en 2009 le jeu de données NSL-KDD à partir du jeu KDD Cup'99, en éliminant les enregistrements dupliqués[65] .

Le jeu de données d'entraînement NSL-KDD contient 125 973 enregistrements, tandis que le jeu de test en contient 22 544. La taille du jeu de données NSL-KDD est suffisante pour permettre son utilisation complète sans avoir besoin de procéder à un échantillonnage aléatoire. Cela a permis d'obtenir des résultats cohérents et comparables dans diverses recherches. Le jeu de données NSL-KDD comprend 22 types d'attaques d'intrusion (voir figure 3.2)dans l'ensemble d'entraînement et 41 attributs (ou caractéristiques). Dans ce jeu de données, 21 attributs concernent la connexion elle-même, tandis que 19 décrivent la nature des connexions au sein du même hôte[65] .

Attacks Class	Attacks Types
DoS	Back, Neptune, Land, Pod, Smurf, Udpstorm, Teardrop, Apache2, Worm, Processtable
Probe	Nmap, Ipsweep, Portsweep, Saint, Mscan, Satan
R2L	Ftp_write, Warezmaster, Httptunnel, Guess_Password, Phf, Warezclient, Snmpguess, Multihop, Xlock, Spy, Xsnoop, Sendmail, Snmpgetattack, Imap, Named
U2R	Rootkit, Loadmodule, Xterm, Sqlattack, Perl, PS, Buffer_overflow

Figure 3.2: Les types d'attaques du jeux de données NSL-KDD[66]

3.3.4 CICIDS2017

Le jeu de données CICIDS2017 comprend à la fois des comportements bénins ainsi que des détails sur de nouvelles attaques de logiciels malveillants, telles que les attaques par force brute FTP, force brute SSH, DoS, Heartbleed, attaques Web, infiltration, botnet et DDoS. Ce jeu de données est étiqueté en fonction de l'horodatage, des adresses IP source et destination, des ports source et destination, des protocoles, et des attaques[67].

Une topologie réseau complète a été configurée pour collecter ces données, incluant un modem, un pare-feu, des commutateurs, des routeurs et des nœuds avec différents systèmes d'exploitation . Ce jeu de données contient 80 caractéristiques de flux réseau extraites du trafic capturé.

Category		Total	Total(-rows with lack info)	Training	Test
BENIGN	BENIGN	2273097	2271320	20000	20000
DOS	DDoS	128027	128025	2700	3300
	DoS slowloris	5796	5796	1350	1650
	DoS Slowhttptest	5499	5499	2171	1169
	DoS Hulk	231073	230124	4500	5500
	DoS GoldenEye	10293	10293	1300	700
	Heartbleed	11	11	5	5
PortScan	PortScan	158930	158804	3808	4192
Bot	Bot	1966	1956	936	624
Brute-Force	FTP-Patator	7938	7935	900	1100
	SSH-Patator	5897	5897	900	1100
Web Attack	Web Attack-Brute Force	1507	1507	910	490
	Web Attack-XSS	652	652	480	160
	Web Attack-SQL Injection	21	21	16	4
Infiltration	Infiltration	36	36	24	6
Total Attack		471454	470365	20000	20000
Total		2830743	2827876	40000	40000

Figure 3.3 : Le jeux de données cicids2017[67]

3.3.5 CICIDS2018

Le jeu de données CICIDS2018, également connu sous le nom de Canadian Institute for Cybersecurity Intrusion Detection System 2018, est un ensemble de données destiné à l'étude et à l'évaluation des systèmes de détection d'intrusion (IDS). Ce dataset a été conçu pour reproduire des scénarios de trafic réseau réalistes, avec des données capturées en utilisant des infrastructures variées et des configurations spécifiques pour refléter fidèlement les comportements normaux et malveillants.

Le CICIDS2018 est constitué de différentes captures réseau représentant des activités normales ainsi que plusieurs types d'attaques malveillantes, permettant aux chercheurs de développer et d'évaluer l'efficacité des algorithmes de détection d'intrusion. L'ensemble de données a été collecté sur plusieurs jours (du 19 au 23 février 2018, puis du 16 au 19 juillet 2018) dans un environnement contrôlé, où des acteurs simulaient différents types d'utilisateurs avec des comportements variés. Cette diversité garantit que l'ensemble de données est suffisamment riche pour représenter de nombreuses situations rencontrées dans des environnements de production réels[67].

.

Category	Attack Type	Flow Count	Training	Test
Brute-force	SSH-Bruteforce	230	184	46
	FTP-BruteForce	611	489	122
Web attack	Brute Force -XSS	187589	7504	1876
	Brute Force -Web	193360	15469	3867
	SQL Injection	87	70	17
DoS attack	DoS attacks-Hulk	466664	18667	4667
	DoS attacks-SlowHTTPTest	139890	55956	13989
	DoS attacks-Slowloris	10990	4396	1099
	DoS attacks-GoldenEye	41508	16603	4151
DDoS attack	DDOS attack-HOIC	686012	27441	6860
	DDOS attack-LOIC-UDP	1730	1384	346
	DDOS attack-LOIC-HTTP	576191	23048	5762
Botnet	Bot	286191	11448	2862
Infilteration	Infilteration	161934	6478	1620
Benign	/	12697719	50791	12698
Total	/	15450706	231127	57782

Figure 3.4 : Le jeux de données cic-ids-2018[67]

Le jeu de données comprend des caractéristiques essentielles de chaque connexion réseau, enregistrées sous forme de flux de données. Ces caractéristiques incluent, entre autres, la durée de la connexion, le nombre de paquets échangés, la taille des paquets, les adresses IP des hôtes source et destination, les numéros de port, les protocoles utilisés, ainsi que des métriques statistiques calculées sur les connexions. Chaque enregistrement est étiqueté pour indiquer si le trafic est normal ou malveillant, et le type spécifique d'attaque.

En ce qui concerne le prétraitement des données, les caractéristiques sont généralement utilisées sous forme de vecteurs de caractéristiques, ce qui facilite l'application de divers algorithmes d'apprentissage automatique. Les caractéristiques les plus importantes qui influencent la classification sont également identifiées, permettant aux modèles d'apprentissage de mieux comprendre la distinction entre trafic normal et trafic malveillant.

Les caractéristiques techniques des données comprennent un total de 80 à 90 caractéristiques extraites de chaque session réseau. Ces caractéristiques incluent, entre autres, des informations temporelles, des statistiques sur les paquets, ainsi que des métriques calculées pour chaque flux, telles que le taux de paquets par seconde ou le nombre d'octets envoyés et reçus. L'étiquetage précis des données facilite l'utilisation de techniques supervisées pour l'entraînement des modèles, et l'abondance de types d'attaques offre une grande diversité pour les tests de robustesse et de généralisation des algorithmes[67].

3.3.6 CICIDS2019

L'ensemble de données CICIDS2019, également connu sous le nom de Canadian Institute for Cybersecurity Intrusion Detection System 2019, est une mise à jour de son prédécesseur, le CICIDS2018. Ce jeu de données a été conçu pour offrir un ensemble de données plus varié, récent, et adapté aux défis actuels rencontrés dans le domaine de la cybersécurité. Le CICIDS2019 a été créé pour simuler un environnement réseau réaliste, intégrant diverses activités normales ainsi que plusieurs types d'attaques malveillantes, permettant ainsi une évaluation efficace des systèmes de détection d'intrusion (IDS).

Les données ont été collectées entre le 10 et le 13 juillet 2019, puis entre le 16 et le 20 juillet 2019, dans un environnement contrôlé où différents utilisateurs simulaient des comportements réels. L'objectif était de fournir des informations sur des scénarios variés, notamment des communications réseau légitimes et diverses attaques, en reproduisant au mieux le trafic observé dans un environnement de production[67].

Le CICIDS2019 inclut des caractéristiques précises pour chaque flux réseau, telles que le nombre de paquets échangés, la durée de la connexion, la taille des paquets, les protocoles utilisés, les adresses IP des hôtes source et destination, ainsi que des statistiques calculées sur chaque connexion. Ces caractéristiques sont utiles pour l'analyse et la modélisation, et facilitent

la tâche des chercheurs qui souhaitent développer des algorithmes de détection d'intrusion robustes.

Le jeu de données est étiqueté, chaque connexion réseau étant annotée pour indiquer si le trafic est normal ou malveillant, ainsi que le type spécifique d'attaque (par exemple, DDoS, brute force, etc.). Ces annotations permettent d'utiliser des approches supervisées pour entraîner des modèles d'apprentissage automatique. En outre, chaque enregistrement contient un ensemble d'environ 80 à 90 caractéristiques spécifiques aux sessions réseau, qui incluent des informations temporelles, des mesures de fréquence de paquets, et d'autres métriques importantes.

Attack Type	Flow Count
Benign	56,863
DDoS_DNS	5,071,011
DDoS_LDAP	2,179,930
DDoS_MSSQL	4,522,492
DDoS_NetBIOS	4,093,279
DDoS_NTP	1,202,642
DDoS_SNMP	5,159,870
DDoS_SSDP	2,610,611
DDoS_SYN	1,582,289
DDoS_TFTP	20,082,580
DDoS_UDP	3,134,645
DDoS_UDP-Lag	366,461
DDoS_WebDDoS	439

Figure 3.5 : Le jeux de données CICIDS2019[67]

En termes de prétraitement, les caractéristiques du dataset sont souvent utilisées sous forme de vecteurs, ce qui permet aux chercheurs d'utiliser une variété d'algorithmes d'apprentissage automatique. Les caractéristiques les plus discriminantes sont sélectionnées pour aider les modèles à identifier les anomalies et les comportements malveillants dans le trafic réseau.

Le CICIDS2019 est un outil précieux pour tester la capacité des modèles de détection d'intrusion à détecter non seulement des attaques connues, mais également des anomalies en général. L'environnement de collecte a été structuré pour reproduire des activités réalistes, ce qui le rend adapté aux conditions rencontrées dans des réseaux de production. L'ensemble de données a été soigneusement conçu pour répondre aux besoins actuels des chercheurs et des praticiens, notamment en matière d'évaluation des systèmes d'intrusion basés sur l'apprentissage profond et l'apprentissage automatique.

3.4 Conclusion

Dans ce chapitre, nous avons passé en revue les travaux connexes relatifs aux systèmes de détection d'intrusion, explorant différentes approches et techniques proposées dans la littérature. Nous avons également présenté les jeux de données existants utilisés dans ce domaine, en mettant l'accent sur leurs caractéristiques, ainsi que leurs avantages et limites dans le contexte de la détection d'intrusion.

Ce chapitre a permis de poser les bases nécessaires pour justifier la suite de notre démarche expérimentale et méthodologique. Dans le prochain chapitre, nous présenterons notre solution, en détaillant les choix techniques et les approches utilisées pour atteindre nos objectifs.

Chapitre 4 : Contribution

4.1 Introduction

Les systèmes de détection d'intrusion basées sur l'apprentissage profond ont fait l'objet de nombreux travaux de recherche. Nous avons discuté dans le chapitre précédant les différentes méthodes d'apprentissage profond qui ont été appliqué avec succès dans la tâche de détection d'intrusion en utilisant des différents ensembles de données dédiés pour la cybersécurité. Les performances des IDS basés sur l'apprentissage profond dépendent fortement de l'ensemble de données utilisées et aucun modèle de référence pour la détection d'intrusion n'a été trouvé. Le processus de détection de fraude de notre méthode repose sur l'utilisation d'algorithmes évolués.

Notre méthode vise à la fois l'amélioration de la performance des modèles de détection d'intrusion, mais également la robustesse des modèles de détection d'intrusion contre les attaques contradictoires. Ce chapitre peut être subdivisé en deux grandes parties. La première partie est axée sur la présentation de la méthodologie ainsi que de l'architecture de notre solution. Dans la seconde partie, nous présenterons les résultats obtenus lors de notre phase d'expérimentation.

4.2 Description de notre approche

4.2.1 Architecture globale de notre système :

La figure 25 présente un pipeline qui se compose de quatre phases distinctes, chacune jouant un rôle important dans la préparation et l'analyse de données pour renforcer la robustesse du modèle.

La première phase, dédiée au nettoyage des données, consiste à préparer le jeu de données en éliminant les anomalies, les valeurs manquantes, et les doublons. Cette étape permet de garantir que les données sont dans un état optimal pour l'entraînement du modèle, minimisant les biais et les erreurs qui pourraient affecter les résultats.

Dans la deuxième phase, un équilibrage des données est appliqué à l'aide de techniques comme SMOTE (Synthetic Minority Over-sampling Technique). Cette étape est particulièrement importante pour corriger le déséquilibre entre les classes, un problème courant qui pourrait influencer négativement les performances de l'algorithme et entraîner des prédictions biaisées. En équilibrant les données, nous améliorons la représentativité des classes minoritaires et préparons un jeu de données plus robuste pour la modélisation.

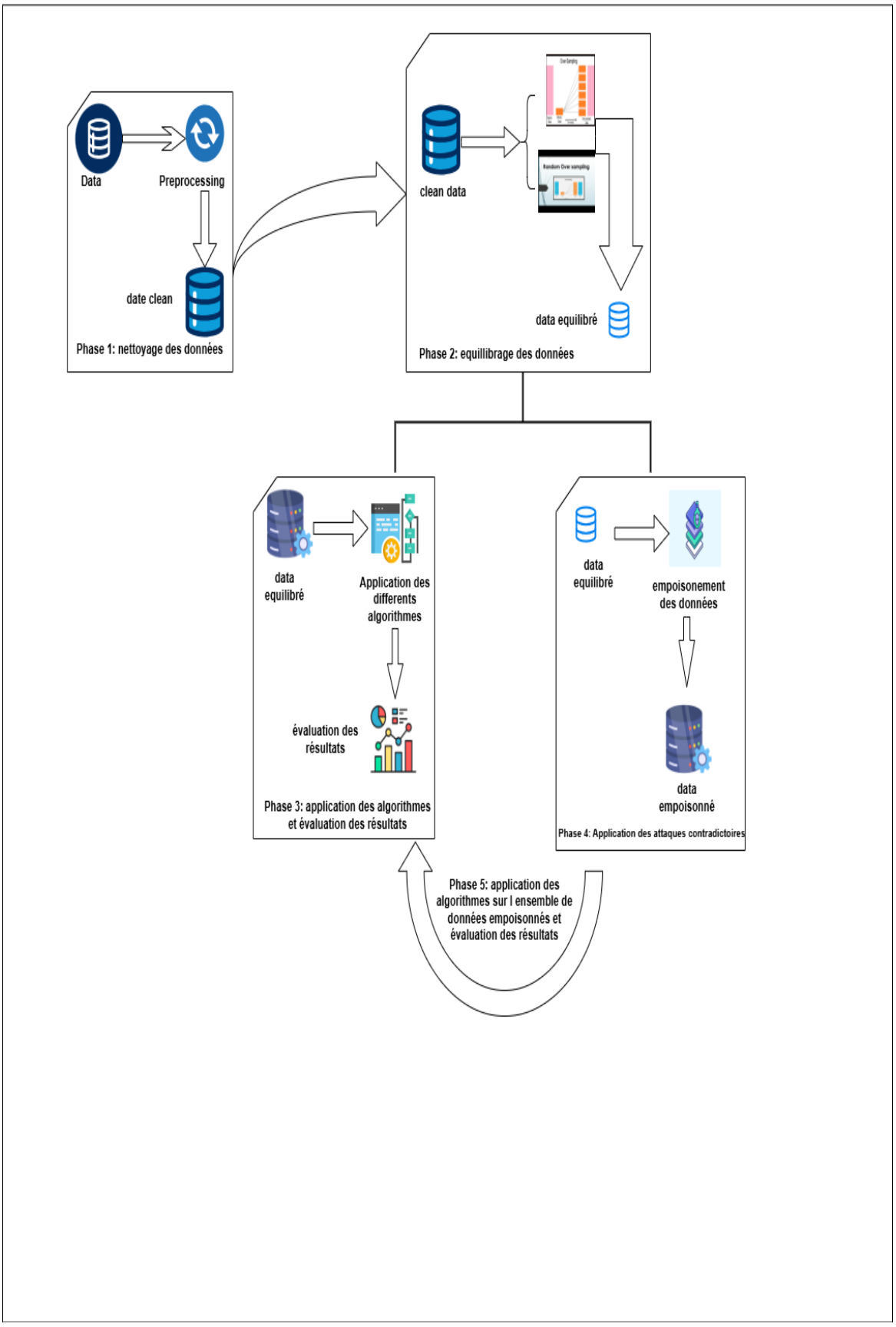

Figure 4.1:Architecture de notre système.

La troisième phase, différents algorithmes sont appliqués sur les données préparées et équilibré. Les performances de chaque modèle sont évaluées selon des critères prédéfinis afin de déterminer leur efficacité.

La quatrième phase introduit un empoisonnement des données par des attaques contradictoires. Cette étape permet de simuler des situations dans lesquelles des exemples perturbés sont ajoutés au jeu de données, testant ainsi la capacité du modèle à résister aux manipulations. L'introduction de ces exemples contradictoires nous aide à évaluer la robustesse et la résilience du modèle face aux attaques, un aspect essentiel pour les applications où la sécurité est primordiale.

Enfin, dans la cinquième phase, différents algorithmes sont appliqués sur les données préparées et empoisonnées. Les performances de chaque modèle sont évaluées selon des critères prédéfinis afin de déterminer leur efficacité. Cette dernière étape permet d'identifier les modèles les plus performants, en tenant compte de leur résilience face aux données adverses et de leur capacité à généraliser sur des jeux de données variés.

Ce pipeline, structuré en cinq phases, constitue une approche rigoureuse pour garantir des modèles performants et robustes, adaptés à des environnements complexes et potentiellement hostiles. Les détails et les justifications de chaque étape seront développés dans les sections suivantes.

4.2.2 Présentation des jeux de données:

Nous avons choisi d'utiliser le jeu de données CICIDS2018 dans le cadre de cette étude, car ce dernier offre plusieurs motivations pertinentes qui justifient son adoption dans ce contexte. Tout d'abord, il est important de noter que le jeu de données CICIDS2018 n'a, à ce jour, jamais été utilisé en conjonction avec un algorithme de type réseau siamois combiné avec CNN et LSTM. Cette originalité représente une réelle opportunité de recherche, car elle permet d'explorer la performance des réseaux siamois dans la détection des intrusions, en particulier pour des données à haute dimensionnalité et comportant des caractéristiques complexes, comme c'est le cas du CICIDS2018.

En outre, le CICIDS2018 présente des caractéristiques qui constituent un défi supplémentaire, rendant sa manipulation et son utilisation intéressantes. En effet, cet ensemble de données est déséquilibré, ce qui signifie que le nombre de données de classes différentes (attaques versus trafic normal) est inégalement distribué. Cela crée un challenge important en matière de détection des intrusions, car un modèle standard d'apprentissage supervisé pourrait être biaisé envers la classe majoritaire, rendant difficile la détection correcte des attaques, en particulier celles qui sont moins

fréquentes. L'utilisation d'un réseau siamois, bien adapté à la comparaison des paires d'échantillons, pourrait offrir une solution efficace à ce problème d'équilibrage.

4.2.3 Le pré-traitement des données :

Avant d'utiliser nos données, il est essentiel de les traiter d'abord pour les rendre plus propices à nos algorithmes d'apprentissage automatique. Le prétraitement est une procédure d'exploration et d'analyse de données qui consiste à convertir des données brutes en informations pouvant être comprises et étudiées par des ordinateurs et des algorithmes d'apprentissage automatique. Cette étape est nécessaire pour corriger toute irrégularité dans l'ensemble de données. Cette étape implique la transformation, le filtrage ou l'amélioration des données avant leur analyse. Le prétraitement des données est crucial pour le processus d'exploration de données, ce processus implique la résolution des données anormales, des valeurs manquantes, des valeurs en double ou d'autres problèmes avec les données brutes. Il peut impliquer des étapes telles que le nettoyage des données, la normalisation des données, l'encodage des données, la transformation des données, l'extraction de caractéristiques de nos données, etc. Cette étape facilite la conversion des données dans un format qui est traité plus facilement et plus efficacement par les algorithmes d'apprentissage automatique. Cela facilite également une interprétation plus précise des résultats de l'apprentissage automatique. Dans notre contexte, les principales étapes de prétraitement que nous avons appliquées sont les suivantes :

- **Filtrage des Données :** Le filtrage des données est une étape essentielle du prétraitement en apprentissage automatique et en analyse de données. Il permet d'éliminer ou de sélectionner certaines observations ou caractéristiques pour améliorer la qualité et la pertinence des données avant leur utilisation dans un modèle.

- **Suppression des Colonnes Non Informatives:** consiste à identifier et éliminer les caractéristiques qui n'apportent pas de valeur ajoutée à l'entraînement du modèle, ce qui permet d'améliorer l'efficacité et la précision des prédictions.

- **Encodage des Caractéristiques Catégorielles :** Les caractéristiques catégorielles sont des variables qui prennent des valeurs sous forme de catégories. Les algorithmes d'apprentissage automatique nécessitent généralement des données numériques, donc il faut transformer ces catégories en nombres grâce à un encodage**.**

- **Normalisation des Données :** Une fois que les données sont de qualité, c'est-à-dire que chaque instance décrit bien sa classe, la prochaine étape consiste à normaliser les données avant l'apprentissage. La normalisation des données d'entrée améliore la construction du modèle en accélérant la convergence et en réduisant les erreurs de généralisation.

4.2.4 Équilibrage des données avec l'algorithme SMOTE:

La grande majorité des techniques d'apprentissage automatique fonctionnent mal sur des ensembles de données déséquilibrés. En effet, ces techniques d'apprentissage ignorent les classes minoritaires, même si ces classes sont généralement les plus importantes.

L'une des méthodes utilisées dans le passé pour traiter les ensembles de données déséquilibrés consiste simplement à copier les exemples dans la classe minoritaire, mais en copiant la classe minoritaire, nous n'ajoutons aucune nouvelle information à l'ensemble de données. Une autre solution pour produire de meilleurs résultats consiste à synthétiser de nouvelles données à partir d'échantillons de données existants.

Cette technique est connue sous le terme anglais SMOTE, qui fait référence au suréchantillonnage de minorités synthétiques. Il s'agit d'une technique statistique qui nous permet d'augmenter le nombre d'échantillons dans une classe donnée pour rendre notre ensemble de données plus équilibré. Les nouvelles données générées par SMOTE ne sont pas des copies des données existantes, elles combinent les caractéristiques du cas cible avec les caractéristiques de ses voisins. Dans notre contexte, nous utiliserons SMOTE pour générer des données synthétiques, que nous comparerons ensuite avec des méthodes de sous-échantillonnage pour déterminer lequel des deux algorithmes est le meilleur et qui sera utilisé dans la suite de notre approche.

4.2.5 Équilibrage des données par l'algorithme de sous-échantillonnage :

Le sous-échantillonnage est une technique utilisée pour traiter des ensembles de données déséquilibrés, où une ou plusieurs classes sont surreprésentées par rapport à d'autres. L'objectif est de réduire le nombre d'exemples dans la classe majoritaire pour obtenir une distribution plus équilibrée entre les classes, ce qui peut améliorer les performances des modèles d'apprentissage automatique en évitant qu'ils ne se biaisent vers la classe dominante.

Dans notre contexte, nous avons opté pour le sous-échantillonnage afin d'équilibrer notre jeu de données déséquilibré. Plus précisément, nous avons utilisé la méthode de Sous-Échantillonnage Aléatoire (Random Under-Sampling). Cette technique consiste à sélectionner de manière aléatoire un sous-ensemble d'exemples issus de la classe majoritaire, de sorte à réduire son effectif pour se rapprocher de celui de la classe minoritaire.

4.2.6 Entraînement et comparaison des performances des différents modèles de classification :

Les algorithmes de classification sont arbre de décision, foret aléatoire, LSTM, CNN, les réseaux siamois sont ceux que nous utiliserons pour l'entraînement et les tests de performance. Notre choix s'est porté sur ces algorithmes du fait de leurs performances dans la résolution des problématiques se rapportant aux détection des intrusions. Les algorithmes classiques, tels que les arbres de décision et les forêts aléatoires, sont souvent utilisés en cybersécurité pour leur simplicité, leur rapidité d'entraînement et leur capacité à bien gérer des données tabulaires, caractéristiques du jeu de données CIC IDS 2018. Les réseaux de neurones convolutifs (CNN) ont été choisi en raison de leur aptitude à extraire des motifs locaux et leur succès croissant dans l'analyse de flux réseau transformés en représentations spatiales. Les réseaux LSTM, quant à eux, permettent de modéliser les dépendances temporelles, ce qui est pertinent pour la détection de comportements malveillants dans les séquences de trafic. Enfin, les architectures siamoises (CNN et CNN+LSTM) ont été explorées pour leur capacité à comparer les similarités entre les échantillons, une approche prometteuse pour détecter les anomalies ou attaques nouvelles. Leur inclusion permet d'évaluer si ces architectures, plus complexes, offrent une meilleure résilience face aux perturbations adverses.

Ces différents algorithmes de classification seront entraînés et testés sur notre jeux de données équilibrés. En entraînant, nos modèles sur les jeux de données équilibrés, nous pourrons améliorer drastiquement la performance des différents modèles, également nous déterminerons parmi les différents algorithmes que nous avons utilisés lequel s'avère le plus performant pour la détection des intrusions.

4.2.7 Amélioration de la robustesse des différents modèles :

Dans notre contexte, nous choisissons d'utiliser la méthode FGSM pour générer des exemples contradictoires et évaluer la robustesse des modèles d'apprentissage automatique appliqués à la détection d'intrusions. FGSM est une méthode idéale en raison de sa pertinence méthodologique face aux contraintes du jeu de données CIC-IDS 2018, caractérisé par sa taille importante et sa complexité. Bien que des techniques plus élaborées telles que PGD puissent être envisagées, leur mise en œuvre nécessite une puissance de calcul plus élevée et une gestion fine des hyperparamètres, ce qui aurait alourdi la phase expérimentale sans garantie d'apport significatif dans une première étape. FGSM, en revanche, constitue une méthode d'attaque contradictoire simple mais efficace, permettant de générer rapidement des exemples perturbés tout en maintenant un contrôle strict sur

l'intensité des modifications apportées aux données d'entrée. Son adoption dans cette étude permet ainsi d'évaluer, de manière rigoureuse et reproductible, la robustesse de modèles variés (CNN, LSTM, architectures siamoises, Random Forest, Decision Tree) face à des attaques de type bruit contrôlé.

Le fonctionnement de FGSM repose sur la perturbation des données d'entrée à l'aide de gradients de modèle. Le paramètre clé Ɛ contrôle la force de cette interférence dans notre cas, on a utilisé Ɛ =0.1 pour garantir que les exemples contradictoires restent réalistes tout en mettant en évidence les vulnérabilités du modèle.

Du côté de la mise en œuvre, FGSM fournit une prise en charge native dans des cadriciels populaires tels que PyTorch et TensorFlow, ainsi que des bibliothèques spécialisées telles que Adversarial Robustness Toolbox (ART), ce qui facilite son intégration dans nos pipelines de traitement.

4.3 Environnement logiciel :

4.3.1 Présentation du langage python :

Python est un langage de programmation orienté objet interprété de haut niveau avec une sémantique dynamique. Ses structures de données de haut niveau intégrées, combinées au typage dynamique et à la liaison dynamique, le rendent très attractif pour le développement rapide d'applications.

Python est un langage facile à apprendre avec une syntaxe intuitive, et il prend également en charge plusieurs modules et packages, encourageant la programmation modulaire et la réutilisation du code.

Python est un langage qui améliore la productivité car il n'y a pas d'étape de compilation et le débogage devient incroyablement rapide lorsque l'interpréteur détecte une erreur et génère une exception.

4.3.2 Présentation de la plateforme Google Colab

Google Colab est un environnement de développement en ligne basé sur Jupyter Notebook qui permet d'exécuter du code Python directement depuis un navigateur. Développé par Google, il offre une alternative puissante et accessible aux utilisateurs souhaitant exécuter des scripts de l'apprentissage automatique et profond et d'analyse de données sans nécessiter d'installation locale de bibliothèques ou d'outils complexes.

L'un des principaux avantages de Google Colab est qu'il fournit un accès gratuit à des ressources matérielles puissantes, notamment des GPU (Graphics Processing Units) et des TPU (Tensor Processing Units), ce qui est particulièrement utile pour entraîner des modèles d'apprentissage

profond nécessitant une grande puissance de calcul. Contrairement aux environnements locaux, où la configuration matérielle peut être une limitation, Google Colab permet d'exécuter des modèles complexes sans se soucier des contraintes de performance. De plus, il offre une intégration fluide avec Google Drive, permettant aux utilisateurs de stocker et de partager leurs projets facilement.

4.3.3 Présentation de la bibliothèque Tensorflow

Tensorflow est une bibliothèque open source gratuite utilisé pour l'intelligence artificielle. Cet outil peut servir à diverses tâches, mais il est principalement destiné à l'entraînement et à l'inférence de réseaux de neurones profonds.

Il a été développé par Google Brain (l'équipe de recherche en IA), a vu sa première version publiée sous la licence Apache 2.0 en 2015.

Tensorflow supporte une variété de langages de programmation, y compris Python, Javascript, C++ et Java, rendant son utilisation possible dans divers domaines et capable de fonctionner sur différents CPU et GPU (CUDA, SYCL), et il est également compatible avec plusieurs systèmes d'exploitation tels que Linux, Mac OS, Windows, Android et iOS. Sa structure adaptable permet de réaliser aisément des calculs sur diverses plateformes (CPU, GPU, TPU), allant des ordinateurs personnels aux grappes de serveurs, sans oublier les appareils portables et périphériques. Tensorflow tire son nom des actions réalisées par ces réseaux neuronaux sur des matrices de données à plusieurs dimensions, connues sous le nom de tenseurs[68].

4.3.4 Présentation de la bibliothèque Keras

Keras est une API de haut niveau conçue pour créer et former des modèles d'apprentissage profond basé sur Python. Il a été développé pour permettre l'expérimentation rapide.

Keras contient plusieurs implémentations communes sur les réseaux de neurones comme les couches de réseaux de neurones, les fonctions d'activation(sigmoid, tangente, relu, leaky-relu.....) , des fonctions d'optimisation ainsi qu'une série d'outils permettant de travailler plus facilement avec des données d'images et de texte afin de simplifier le codage nécessaire à l'écriture du code des réseaux neuronaux profonds. En plus des réseaux de neurones standard , Keras supporte également les réseaux de neurones convolutionnel et récurrent même la combinaison entre les deux[69].

4.3.5 Présentation de la bibliothèque matplotlib

Matplotlib est une bibliothèque Python open source développée à l'origine en 2002 par le neurobiologiste John Hunter. L'objectif est de visualiser les signaux électriques dans le cerveau des

personnes épileptiques. Pour y parvenir, il souhaitait reproduire les capacités de traçage de MATLAB à l'aide de Python[70].

Après la mort de John Hunter en 2012, de nombreux contributeurs de la communauté open source ont continué à améliorer Matplotlib. Il est utilisé pour créer des tableaux et des graphiques de haute qualité. C'est une alternative open source à MATLAB.

Par exemple, des tracés, des histogrammes, des graphiques à barres et tous les types de graphiques peuvent être créés avec seulement quelques lignes de code. Il s'agit d'un outil très complet qui permet de générer des visualisations de données très détaillées.

Cette bibliothèque est particulièrement utile pour les personnes travaillant avec Python Il est utilisé spécifiquement pour les serveurs d'applications Web, les shells et les scripts Python. Grâce à l'API matplotlib, les développeurs peuvent également intégrer des graphiques dans les applications GUI et une combinaison des deux[70].

4.3.6 Présentation de la bibliothèque Pandas

Pandas est une bibliothèque open source conçue spécifiquement pour la manipulation, l'analyse et la structuration de données. Il est largement utilisé dans la science des données, l'apprentissage automatique et l'analyse statistique, fournissant des structures de données flexibles et performantes adaptées au traitement de grands ensembles de données. Son principal avantage réside dans ses principales structures de données : DataFrames et Series, qui permettent une manipulation intuitive et efficace des données.

DataFrame est une structure bidimensionnelle qui fonctionne comme un tableau avec des lignes et des colonnes, semblable à une feuille de calcul Excel ou à un tableau SQL. Il vous permet de stocker et d'analyser des données hétérogènes tout en offrant des capacités étendues de filtrage, d'agrégation et de transformation des données. Une Series, quant à elle, est une structure unidimensionnelle qui correspond aux colonnes d'un DataFrame et peut être utilisée pour stocker des données d'index.

Pandas se distingue par sa capacité à charger, nettoyer et transformer rapidement des ensembles de données provenant de différentes sources telles que des fichiers CSV, Excel, JSON, des bases de données SQL et des API Web. Il fournit des fonctions avancées de tri, de fusion, de regroupement, d'analyse statistique et de gestion des valeurs manquantes, ce qui en fait un outil important pour le traitement des données. De plus, Pandas s'intègre facilement à d'autres bibliothèques telles que NumPy, Matplotlib et Scikit-learn pour faciliter la visualisation et l'analyse des données.

En raison de sa facilité d'utilisation et de sa puissance, Pandas est devenu une référence importante en matière d'analyse de données, permettant aux analystes, aux data scientists et aux ingénieurs d'utiliser efficacement de grandes quantités d'informations. Dans l'ensemble, cette bibliothèque est un outil essentiel pour quiconque souhaite manipuler des données en Python de manière rapide, intuitive et efficace.

4.3.7 Présentation de la bibliothèque Scikit-learn

Scikit-learn est une bibliothèque open source Python conçue spécifiquement pour l'apprentissage automatique et largement utilisée pour créer et évaluer ses modèles. Il est basé sur NumPy, SciPy et Matplotlib et fournit une interface simple et efficace pour la mise en œuvre d'algorithmes d'apprentissage supervisé et non supervisé, le prétraitement des données et l'évaluation des performances des modèles.

L'un des principaux avantages de Scikit-learn est sa large gamme d'algorithmes disponibles. Pour l'apprentissage supervisé, il fournit des méthodes telles que la régression linéaire et logistique, les arbres de décision, les forêts aléatoires (Random Forest), les machines à vecteurs de support (SVM) et les réseaux de neurones artificiels (MLP). Pour l'apprentissage non supervisé, il fournit des algorithmes de Clustering (K-Means, DBSCAN, Aggregate Clustering). Il comprend également des outils de sélection de modèles, de validation croisée, d'optimisation des hyperparamètres et de gestion du pipeline de prétraitement des données.

4.4 Expérimentations et résultats

4.4.1 Prétraitement des données

Dans notre contexte, les principales étapes de prétraitement que nous avons appliquées sont les suivantes :

- **Filtrage des Données :** L'ensemble de données a été d'abord filtré pour supprimer toutes les lignes redondantes représentant les instances de classe. Ensuite, une analyse a été effectuée pour détecter toute valeur 'NAN' (Not A Number) ou 'INF' (Infinite Value). Ces valeurs, considérées comme des données manquantes, peuvent gravement nuire aux performances des modèles d`apprentissage automatique ou profond. Dans notre cas, les données de la colonne "Flow Bytes" présentaient plusieurs valeurs 'NAN'. Étant donné que nous disposons d'un volume de données suffisant, les lignes contenant des valeurs 'NAN' ou 'INF' ont été supprimées.

- **Suppression des Colonnes Non Informatives:** Les statistiques descriptives ont révélé que certaines colonnes étaient vides (valeurs constamment égales à 0) et ne contenaient donc aucune information discriminatoire pour différencier les classes d'attaque. Au contraire, elles pourraient engendrer de mauvais résultats. Ces colonnes, telles que 'Bw PSH Flag', 'Fw URG Flag', 'Bw URG Flag', 'FIN Count', 'PST Count', 'ECE Count', 'Fw Avg Bytes/Bulk', 'Fw Avg Packets/Bulk', 'Fw Avg Bulk Rate', 'Bw Avg Bytes/Bulk', 'Bw Avg Packets/Bulk', et 'Bw Avg Bulk Rate', ont été supprimées. D'autres colonnes catégorielles, comme les adresses IP et les horodatages ('Timestamp'), ont été supprimées, car ces informations ne sont pas directement liées aux propriétés des attaques . Ces caractéristiques, telles que 'Flow ID', 'Source IP', 'Destination IP', 'Timestamp', ont également été supprimées pour ne conserver que les caractéristiques du trafic réseau.

- **Encodage des Caractéristiques Catégorielles :** Certaines caractéristiques catégorielles dans l'ensemble de données nécessitaient un encodage. Par exemple, la colonne 'Flow Packets/s' a été convertie en une colonne numérique.

- **Encodage de la Colonne 'Label':** La colonne 'Label', représentant la classe de chaque instance, a été encodée à l'aide d'une technique populaire appelée "One-Hot-Encoding". Ce codage transforme les catégories en colonnes individuelles, où une valeur de 1 indique que l'instance appartient à cette classe, et une valeur de 0 qu'elle n'y appartient pas.

- **Normalisation des Données :** Dans notre contexte, nous avons utilisé StandardScaler de la bibliothèque Scikit-learn pour la normalisation des données. StandardScaler fonctionne en centrant et en réduisant chaque variable en lui appliquant la transformation suivante :

$$X_{normalisé} = \frac{X - \mu}{\sigma}$$

où μ représente la moyenne de chaque caractéristique et σ son écart-type. Cela a pour effet de transformer les données pour qu'elles suivent une distribution centrée autour de 0 avec un écart-type de 1, garantissant ainsi une échelle homogène entre toutes les variables.

4.4.2 Génération des données synthétiques avec SMOTE

Nous avons appliqué SMOTE (Synthetic Minority Over-sampling Technique) pour équilibrer les classes dans notre ensemble d'entraînement. Avant l'application de SMOTE, la distribution des classes montrait un déséquilibre significatif, avec une sous-représentation de certaines classes. Après avoir utilisé SMOTE, nous avons généré des exemples synthétiques pour les classes minoritaires, ce qui a permis d'obtenir un équilibre parfait entre les classes. Comme nous pouvons le voir sur la figure 4.3 après l'application de SMOTE, notre jeu de données est devenu équilibré .

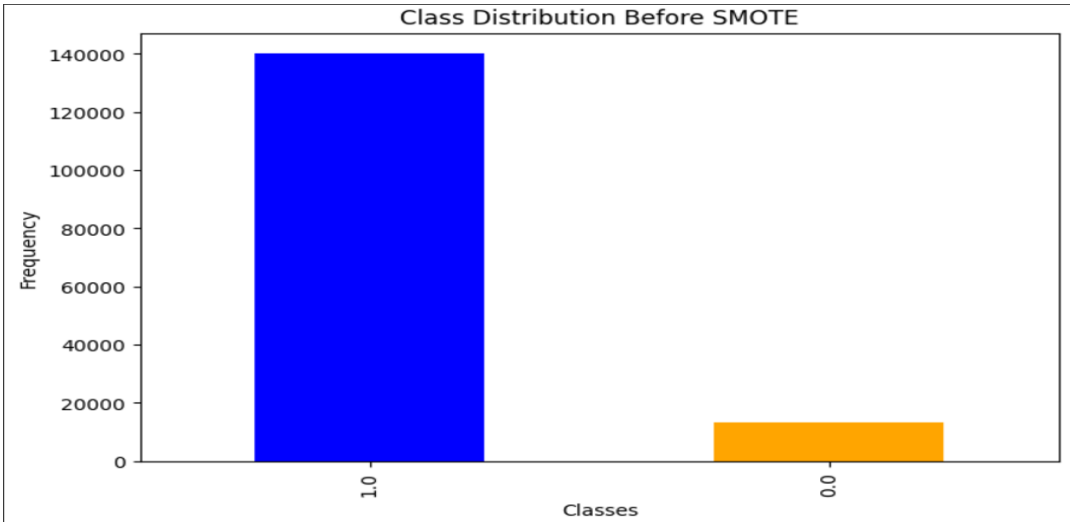

Figure 4.2: distribution des classes avant SMOTE

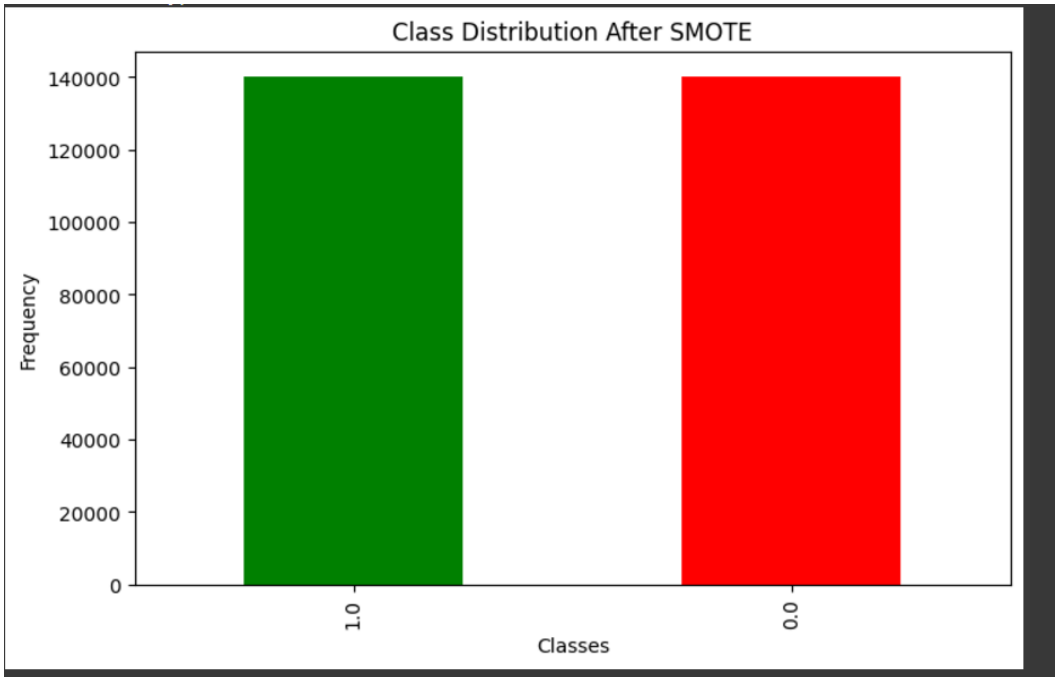

Figure 4.3 : Distribution des classes après SMOTE

4.4.3 Équilibrage des données par l'algorithme de sous-échantillonnage :

Nous avons utilisé la méthode de Sous-Échantillonnage Aléatoire (Random Under-Sampling). Cette technique consiste à sélectionner de manière aléatoire un sous-ensemble d'exemples issus de la classe majoritaire, de sorte à réduire son effectif pour se rapprocher de celui de la classe minoritaire.

Comme nous pouvons le voir sur la figure 4.4 après l'application de l`algorithme Sous-Échantillonnage Aléatoire, notre jeu de données est devenu équilibré .

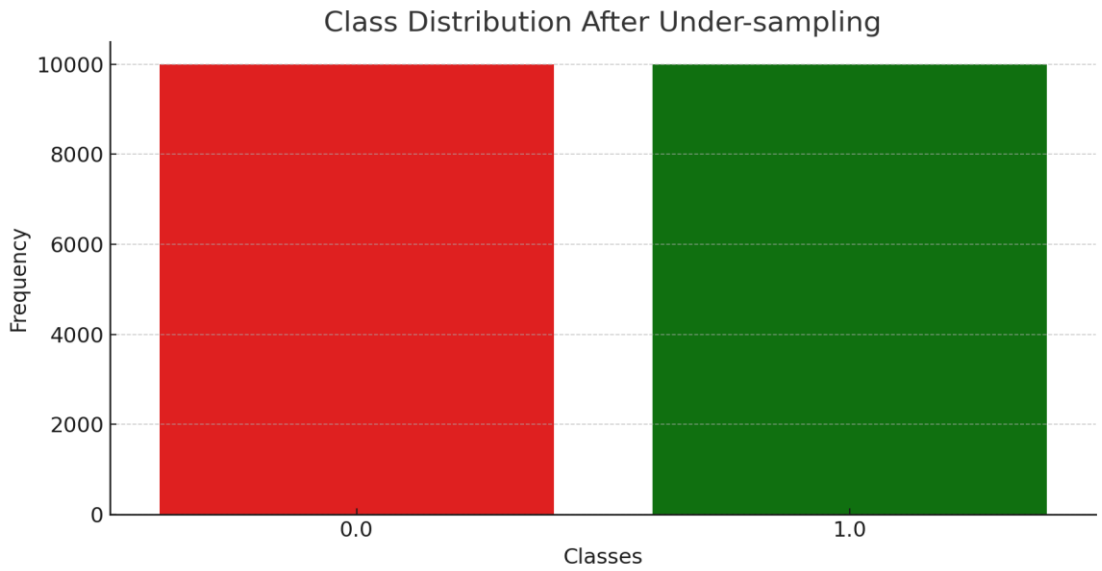

Figure 4.4: Distribution des classes après l'application de l'algorithme Sous-Échantillonnage Aléatoire

4.4.4 Sélection de l'algorithme de génération des données

Le sous-échantillonnage consiste à réduire artificiellement le nombre d'exemples dans la classe majoritaire afin d'équilibrer la distribution des classes, ce qui peut simplifier le problème et accélérer l'entraînement, mais au prix de la perte d'informations potentiellement cruciales et d'une diversité réduite des données. À l'inverse, SMOTE (Synthetic Minority Over-sampling Technique) génère de nouveaux exemples synthétiques pour la classe minoritaire en interpolant entre les observations existantes, permettant ainsi de renforcer sa représentativité sans éliminer les informations de la classe majoritaire. Dans notre contexte, nous privilégions SMOTE, car il enrichit le jeu de données en conservant l'intégralité des données originales tout en offrant une meilleure capacité à détecter et modéliser les comportements rares, ce qui est essentiel pour améliorer la performance et la robustesse des modèles de détection d'intrusions.

4.4.5 Entraînement et test de la performance des différents modèles sur notre ensemble de données équilibré

Dans cette section, nous allons nous intéresser aux performances de nos cinq algorithmes qui vont être testés sur notre jeux de données équilibré. Les algorithmes choisis pour effectuer ces différents tests sont arbre de décision, foret aléatoire, CNN, LSTM et les réseaux siamois. Ces algorithmes ont été choisis en raison de leur bonne performance dans la résolution des problèmes de classification et plus particulièrement la détection d'intrusion. Les différentes expérimentations effectuées vont nous permettre de comprendre comment chaque algorithme performe sur des jeux de données fortement équilibrés. Pour effectuer nos différents tests, nous avons considéré les métriques de performance comme le "Rappel" ,"Précision", "ScoreF1", "Accuracy", ces métriques vont nous permettre de connaître la performance réelle de nos différents algorithmes. Les résultats des expérimentations sur notre jeux de données sont contenus dans le tableaux 4.1 et la figure 4.5 que nous allons analyser par la suite.

L'analyse de la performance des modèles utilisés pour voir si y a des intrusions montre l'arbre décisionnel a la meilleure accuracy (98.2%), après lui Siamese (CNN+LSTM) (97.7%) et CNN (97%), ce qui indique une grande habilité à classer bien les intrusion et le trafic normal. En regardant la précision qui montre le nombre d'alertes correctes parmi celles vues comme intrusions. Random Forest et Siamese (CNN+LSTM) ont 97%, ce qui veut dire qu'ils créent moins de faux positifs. Au contraire LSTM a précision moins bonne (90%), cela montre un surplus de mauvaises alertes. Le rappel, qui mesure la capacité du modèle à détecter toutes les intrusions, est plus haut pour CNN et Siamese (CNN+LSTM) (97.4% et 97%), montrant qu'ils attrapent bien les actes non autorisés. Mais Siamese (CNN) a un rappel plus bas (92%), ce qui veut dire qu'il peut rater certaines attaques. Finalement, le F1-score, est le plus haut pour Siamese (CNN+LSTM) (97%), montrant son bonne capacité à voir les intrusions tout en réduisant les fausse alertes. LSTM a les pire résultats avec un F1 score de 92.10%. Donc, Siamese (CNN+LSTM) parait comme le modèle le meilleur pour trouver les intrusions car il offre un bon compromis entre identification des attaques et limitation des faux positifs, améliorant ainsi la sécurité des systèmes face aux menaces.

Algorithmes	accuracy	précision	rappel	F1 score
Random forest	96.8%	97%	95%	96%
Decision tree	98.2%	94%	96%	95%
CNN	97%	95%	97.4%	96.19%
LSTM	95.68%	90%	94.3%	92.10%
Siamese(CNN)	96.54%	96.25%	92%	94.08%
Siamese(CNN+LSTM)	97.7%	97%	97%	97%

Tableaux 4.1 : Performance des algorithmes sur l'ensemble des données équilibré

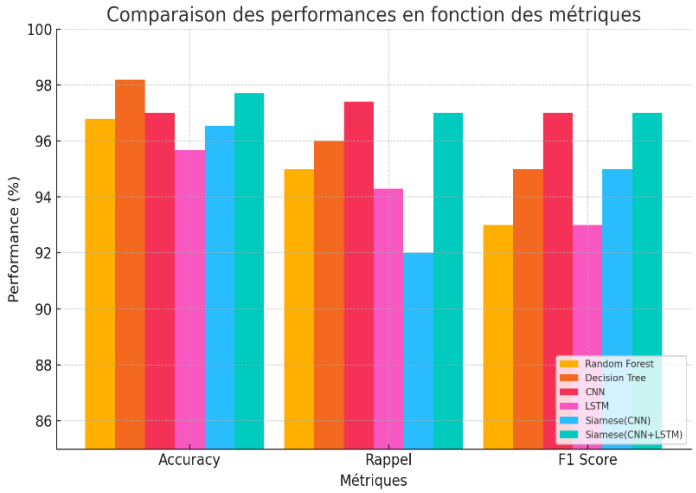

Figure 4.5 : Le graphe de performance des différents algorithmes sur notre jeu de données équilibré

4.4.6 Tests et amélioration de la robustesse des modèles

Dans ce travail, nous avons choisi d'utiliser la méthode FGSM (Fast Gradient Sign Method) pour générer des exemples adversaires, plutôt que la méthode PGD (Projected Gradient Descent)**.** FGSM est également largement reconnu comme une méthode de référence pour évaluer la robustesse des modèles face aux attaques contradictoires, ce qui permet de comparer efficacement les performances des architectures testées.

Dans cette section, nous allons nous intéresser aux performances de nos cinq algorithmes qui vont être testés sur notre jeux de données équilibré et empoisonnées.

Les résultats présentés dans le tableau 4.2 et la figure 4.6 montrent que l'ensemble des algorithmes évalués présente des performances très élevées, avec des scores globaux supérieurs à 96 % pour l'accuracy, la précision, le rappel et le F1 score. Parmi ces modèles, le Siamese combinant CNN et LSTM se distingue particulièrement en affichant une accuracy de 98,9 %, une précision de 98 %, un rappel de 98,43 % et un F1 score de 98,22 %. Cela suggère que cette approche hybride, qui exploite

à la fois les informations spatiales et séquentielles, est particulièrement efficace pour identifier correctement les cas positifs tout en minimisant les erreurs. L'abre décisionnel réalise également d'excellentes performances avec une accuracy de 98,3 %, bien que son rappel légèrement inférieur (96,71 %) indique qu'il pourrait passer à côté de certains cas critiques par rapport au modèle hybride. Les autres algorithmes – Random Forest, CNN, LSTM et le Siamese basé uniquement sur CNN – obtiennent des résultats homogènes autour de 97 % pour l'ensemble des métriques, démontrant leur robustesse générale. En somme, ces résultats illustrent que, même si tous les modèles sont performants, l'approche Siamese (CNN+LSTM) offre une meilleure capacité de détection en équilibrant efficacement précision et rappel, ce qui se traduit par une amélioration globale des performances.

Algorithmes	**accuracy**	**précision**	**rappel**	**F1 score**
Random forest	97.3%	97.23%	97%	97.12%
Decision tree	98.3%	98%	96.71%	97.36%
CNN	97.15%	97.08%	97%	97.04%
LSTM	97.37%	96.12%	97.19%	96.65%
Siamese(CNN)	97.12%	96.78%	97.02%	96.9%
Siamese(CNN+LSTM)	98.9%	98%	98.43%	98.22%

Tableau 4.2 : Performance des algorithmes réentraînée en incluant les attaque contradictoires

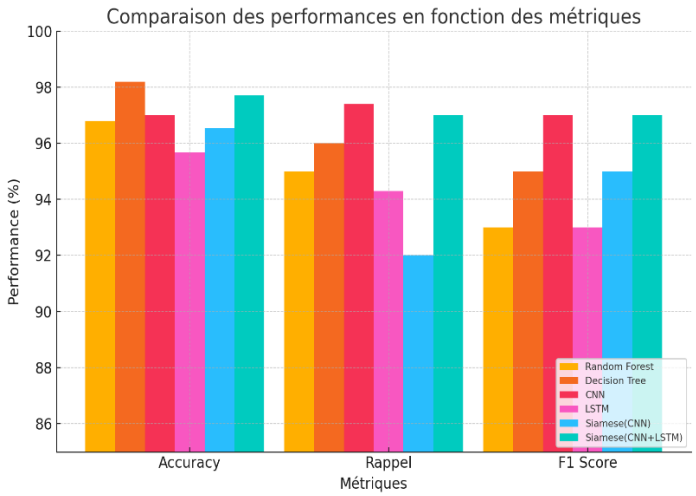

Figure 4.6 : Le graphe de performance des différents algorithmes sur notre jeu de donnée équilibré et empoisonné

4.5 Coclusion :

En conclusion, ce chapitre a présenté l'environnement de travail ainsi que le pipeline détaillé mis en place pour mener à bien notre étude. Nous avons décrit les différentes étapes, allant de la préparation des données et des algorithmes appliqués, jusqu'à l'analyse approfondie des résultats. Les performances des modèles, évaluées dans divers scénarios, ont été discutées en détail, mettant en évidence les forces et les limites de chaque approche. L'intégration des attaques adversaires a permis d'explorer la robustesse des modèles et de confirmer l'efficacité des architectures avancées, notamment les modèles Siamois combinant CNN et LSTM. Ces résultats fournissent une base solide pour les conclusions globales et ouvrent des perspectives pour l'amélioration des systèmes face à des scénarios adversaires complexes.

Chapitre 5 : Conclusion Générale

La détection d'intrusion est une composante essentielle des systèmes de cybersécurité modernes, permettant d'identifier et de prévenir les comportements malveillants dans les réseaux informatiques. Ces systèmes reposent largement sur des techniques d'apprentissage automatique, capables de traiter de grands volumes de données pour détecter des anomalies ou des activités suspectes. Cependant, malgré les progrès significatifs réalisés dans ce domaine, des défis majeurs subsistent, notamment la vulnérabilité face aux attaques contradictoires et la gestion des déséquilibres dans les ensembles de données, qui peuvent compromettre l'efficacité de ces systèmes.

La problématique abordée dans cette thèse s'articule autour de deux axes principaux. D'une part, les modèles d'apprentissage automatique actuels restent vulnérables aux attaques contradictoires, qui exploitent leurs failles en introduisant des perturbations imperceptibles pour tromper leur capacité de classification. D'autre part, les déséquilibres dans les ensembles de données utilisés pour entraîner ces modèles réduisent leur performance, particulièrement pour la détection des intrusions rares mais critiques, entraînant une augmentation des faux positifs ou des faux négatifs. Ces limitations compliquent la capacité des systèmes à détecter correctement les menaces et à protéger les environnements réseau.

Pour répondre à ces défis, nous avons proposé une approche méthodologique intégrée, comprenant un pipeline complet allant de la préparation des données à l'évaluation des modèles. Les données ont été nettoyées et équilibrées à l'aide de la méthode SMOTE (Synthetic Minority Oversampling Technique) pour atténuer les effets du déséquilibre des classes. Divers modèles ont été testés, allant des algorithmes classiques, comme la forêt aléatoire et l'arbre de décision, aux architectures avancées, telles que les réseaux convolutifs (CNN) et les réseaux récurrents à mémoire longue et courte (LSTM). Les modèles classiques ont montré une robustesse naturelle dans des scénarios sans attaques adversaires, mais leurs limites ont été mises en évidence face aux perturbations malveillantes.

Nous avons également intégré des exemples adversaires générés par la méthode FGSM (Fast Gradient Sign Method) dans le processus d'entraînement, renforçant la robustesse des modèles neuronaux face aux attaques contradictoires. Enfin, des modèles Siamois, combinant les forces des CNN et des LSTM, ont été développés pour capturer à la fois les caractéristiques locales et les dépendances temporelles. Ces architectures avancées ont démontré leur efficacité, avec le modèle Siamois basé sur CNN et LSTM atteignant une précision exceptionnelle de **98.9 %** dans des scénarios avec attaques adversaires.

En conclusion, cette thèse démontre que l'intégration de techniques avancées, telles que les modèles Siamois combinant CNN et LSTM, associées à une gestion proactive des déséquilibres des données et à des mécanismes de défense contre les attaques adversaires, permet d'améliorer significativement la performance et la robustesse des systèmes de détection d'intrusions. Ces contributions offrent une avancée substantielle dans la conception de systèmes plus sûrs et fiables, capables de détecter efficacement les menaces, même dans des environnements complexes et adverses. À l'avenir, des recherches pourraient explorer des modèles encore plus performants, comme des mécanismes d'attention renforcés, pour étendre les capacités de ces systèmes dans le domaine de la cybersécurité. D'autres modèles d'attaques contradictoires seront également explorés afin d'approfondir la compréhension des vulnérabilités et de renforcer la robustesse des systèmes d'IA face à ces menace

6. *BIBLIOGRAPHIE*

[1] R.BACE, T. MELL. “NIST SPECIAL PUBLICATION ON INTRUSION DETECTION SYSTEMS”. COMPUTER SCIENCE 2001.

[2] MARTIN ARVIDSON, MARKUS CARLBARK, ‘’ INTRUSION DETECTION SYSTEMS – TECHNOLOGIES, WEAKNESSES AND TRENDS’’, LITH-ISY-EX -3390-2003 STOCKHOLM 2003

[3] Hervé DEBAR et all. « Détection d’intrusion : corrélation d'alertes », March 2004 Techniques et Sciences Informatiques 23(3):359-390 2004

[4] WOOD, MARK AND ERLINGER, MICHAEL. INTRUSION DETECTION MESSAGE EXCHANGE REQUIREMENTS. HTTPS://TOOLS.IETF.ORG/HTML/RFC4766, 2002.

[5] HTTPS://CONNECT.ED-DIAMOND.COM/MISC/MISC-072/LA-DETECTION-D- INTRUSION-UNE-APPROCHE-GLOBALE.

[6] AHMAD JAVAID, QUAMAR NIYAZ, WEIQING SUN, AND MANSOOR ALAM. A DEEP LEARNING APPROACH FOR NETWORK INTRUSION DETECTION SYSTEM. IN PROCEEDINGS OF THE 9TH EAI IN TERNATIONAL CONFERENCE ON BIO-INSPIRED INFORMATION AND COMMUNICATIONS TECHNOLOGIES (FORMERLY BIONETICS), PAGES 21–26, 2016.

[7] Chiba Zouhair, Noreddine Abghour, Khalid Moussaid, Amina El Omri, and Mohamed Rida. A review of intrusion detection systems in cloud computing. In Security and Privacy in Smart Sensor Networks, pages 253–283. IGI Global, 2018.

[8] Wang, Zhen & Zhang, Dan. (2012). HIDS and NIDS Hybrid Intrusion Detection System Model Design. Advanced Engineering Forum. 6-7. 991-994. 10.4028/www.scientific.net/AEF.6-7.991.

[9] https://www.liquidweb.com/blog/host-based-intrusion-detection-system/

[10] Rupa Devi, T., Badugu, S. (2020). A Review on Network Intrusion Detection System Using Machine Learning. In: Satapathy, S.C., Raju, K.S., Shyamala, K., Krishna, D.R., Favorskaya, M.N. (eds) Advances in Decision Sciences, Image Processing, Security and Computer Vision. ICETE 2019. Learning and Analytics in Intelligent Systems, vol 4. Springer, Cham. https://doi.org/10.1007/978-3-030-24318-0_69

[11] Msika, S. (2020). Renforcement de systèmes de détection d'intrusions par des attaques GAN et métaheuristiques [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/4192/

[12] Bhati, Bhoopesh & Chugh, Garvit & Al-Turjman, Fadi & Bhati, Nitesh Singh. (2021). An improved ensemble based intrusion detection technique using XGBoost. Transactions on Emerging Telecommunications Technologies. 32. 10.1002/ett.4076.

[13] ANDERSON, J. (1980). COMPUTER SECURITY THREAT MONITORING AND JAMES P. ANDERSON COMPANY, FORT WASHINGTON, PENNSYLVANIA.

[14] D. E. DENNING, "AN INTRUSION-DETECTION MODEL." IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE- 13(NO. 2):222-232, FEB. 1987.

[15] HTTPS://CONNECT.ED-DIAMOND.COM/MISC/MISC-072/LA-DETECTION-D- INTRUSION-UNE-APPROCHE-GLOBALE.

[16] HTTP://WWW.CSL.SRI.COM/PROGRAMS/INTRUSION/HISTORY.HTML.

[17]HTTPS://WWW.TODDHEBERLEIN.COM/BLOG/2015/5/7/25-YEARS-AGO-A-NETWORK-SECURITY-MONITOR.

[18] HEBERLEIN, L. ET AL. "A NETWORK SECURITY MONITOR." PROCEEDINGS OF THE IEEE COMPUTER SOCIETY SYMPOSIUM, RESEARCH IN SECURITY AND PRIVACY, MAY 1990,PP. 296-303.
[19] Salman, Hasan & Kalakech, Ali & Steiti, Amani. (2024). Random Forest Algorithm Overview. Babylonian Journal of Machine Learning. 2024. 69-79. 10.58496/BJML/2024/007.
[20] https://medium.com/analytics-vidhya/random-forest-classifier-and-its-hyperparameters-8467bec755f6
[21] https://www.learnelectronicswithme.com/2023/01/decision-tree-algorithm-implementation.html
[22] Sublime, Jeremie. (2022). L'apprentissage non-supervisé et ses contradictions. Bulletin 1024. 145-156. 10.48556/SIF.1024.19.145.
[23] https://moncoachdata.com/tutos/apprentissage-automatique-non-supervise/
[24] https://fr.wikipedia.org/wiki/Apprentissage_par_renforcement
[25] Li Deng. A tutorial survey of architectures, algorithms, and applications for deep learning.*APSIPA Transactions on Signal and Information Processing*, 3, 2014.
[26] JIUXIANG GU, ZHENHUA WANG, JASON KUEN, LIANYANG MA, AMIR SHAHROUDY, BING SHUAI, TING LIU, XINGXING WANG, GANG WANG, JIANFEI CAI, ET AL. RECENT ADVANCES IN CONVOLUTIONAL NEURAL NETWORKS. PATTERN RECOGNITION, 77 :354–377, 2018.
[27] ALEX KRIZHEVSKY, ILYA SUTSKEVER, AND GEOFFREY E HINTON. IMAGENET CLASSIFICATION WITH DEEP CONVOLUTIONAL NEURAL NETWORKS. IN ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, PAGES 1097–1105, 2012.
[28] KWANGJO KIM, MUHAMAD ERZA AMINANTO, AND HARRY CHANDRA TANUWIDJAJA. NETWORK INTRUSION DETECTION USING DEEP LEARNING : A FEATURE LEARNING APPROACH. SPRINGER, 2018.
[29] Clément Dalloux, Natalia Grabar, and Vincent Claveau. Détection de la négation : corpus français et apprentissage supervisé. *Revue des Sciences et Technologies de l'Information-Série TSI : Technique et Science Informatiques*, pages 1–21, 2019.
[30] Van Houdt, Greg & Mosquera, Carlos & Nápoles, Gonzalo. (2020). A Review on the Long Short-Term Memory Model. Artificial Intelligence Review. 53. 10.1007/s10462-020-09838-1.
[31] Hindy, Hanan & Tachtatzis, Christos & Atkinson, Robert & Bayne, Ethan & Bellekens, Xavier. (2021). Developing a Siamese Network for Intrusion Detection Systems. 120-126. 10.1145/3437984.3458842.
[32] https://medium.com/towards-data-science/a-friendly-introduction-to-siamese-networks-85ab17522942
[33] "ADVERSARIAL EXAMPLES : VALIDEZ VOS MODÈLES DE DEEP LEARNING | MOOV AI." HTTPS://MOOV.AI/FR/BLOG/VALIDATION-MODELES-ADVERSARIAL-EXAMPLES/
[34] "HOW ADVERSARIAL ATTACKS WORK." HTTPS://BLOG.YCOMBINATOR.COM/HOW-ADVERSARIALATTACKS-WORK/
[35] AHMAD JAVAID, QUAMAR NIYAZ, WEIQING SUN, AND MANSOOR ALAM. A DEEP LEARNING APPROACH FOR NETWORK INTRUSION DETECTION SYSTEM. IN PROCEEDINGS OF THE 9TH EAI IN TERNATIONAL CONFERENCE ON BIO-INSPIRED INFORMATION AND COMMUNICATIONS TECHNOLOGIES (FORMERLY BIONETICS), PAGES 21–26, 2016.
[36] R VINAYAKUMAR, KP SOMAN, AND PRABAHARAN POORNACHANDRAN. APPLYING CONVOLUTIONAL NEURAL NETWORK FOR NETWORK INTRUSION DETECTION. IN 2017 INTERNATIONAL CONFERENCE ON ADVANCES IN COMPUTING, COMMUNICATIONS AND INFORMATICS (ICACCI), PAGES 1222–1228. IEEE, 2017

[37] JIHYUN KIM AND HOWON KIM. APPLYING RECURRENT NEURAL NETWORK TO INTRUSION DETECTION WITH HESSIAN FREE OPTIMIZATION. IN INTERNATIONAL WORKSHOP ON INFORMATION SECURITY APPLICATIONS, PAGES 357–369. SPRINGER, 2015.
[38] NI GAO, LING GAO, QUANLI GAO, AND HAI WANG. AN INTRUSION DETECTION MODEL BASED ON DEEP BELIEF NETWORKS. IN 2014 SECOND INTERNATIONAL CONFERENCE ON ADVANCED CLOUD AND BIG DATA, PAGES 247–252. IEEE, 2014.
[39] SHAHADATE REZVY, MILTOS PETRIDIS, ABOUBAKER LASEBAE, AND TAHMINA ZEBIN. INTRUSION DETECTION AND CLASSIFICATION WITH AUTOENCODED DEEP NEURAL NETWORK. IN INTERNATIONAL CONFERENCE ON SECURITY FOR INFORMATION TECHNOLOGY AND COMMUNICATIONS, PAGES 142–156. SPRINGER, 2018.
[40] https://pyimagesearch.com/2021/03/01/adversarial-attacks-with-fgsm-fast-gradient-sign-method/
[41] Wei, Wei & Dai, Hua & Liang, Weitai. (2020). A novel projected gradient-like method for optimization problems with simple constraints. Computational and Applied Mathematics. 39. 10.1007/s40314-020-01210-x.
[42] https://onlinelibrary.wiley.com/doi/10.1155/2021/6663028
[43] JASON BROWNLEE. IMBALANCED CLASSIFICATION. HTTPS://MACHINELEARNINGMASTERY.COM/ SMOTE-OVERSAMPLING-FOR-IMBALANCED-CLASSIFICATION/..
[44] HERVÉ DEBAR, MARC DACIER, AND ANDREAS WESPI. A REVISED TAXONOMY FOR INTRUSION DETECTION SYSTEMS. IN ANNALES DES TÉLÉCOMMUNICATIONS, VOLUME 55, PAGES 361–378. SPRINGER, 2000.
[45] LI DENG. A TUTORIAL SURVEY OF ARCHITECTURES, ALGORITHMS, AND APPLICATIONS FOR DEEP LEARNING. APSIPA TRANSACTIONS ON SIGNAL AND INFORMATION PROCESSING, 3, 2014.
[46] NASRIN SULTANA, NAVEEN CHILAMKURTI, WEI PENG, AND RABEI ALHADAD. SURVEY ON SDN BASED NETWORK INTRUSION DETECTION SYSTEM USING MACHINE LEARNING APPROACHES. PEER-TO-PEER NETWORKING AND APPLICATIONS, 12(2) :493–501, 2019.
[48] MD ZAHANGIR ALOM, VENKATARAMESH BONTUPALLI, AND TAREK M TAHA. INTRUSION DETECTION USING DEEP BELIEF NETWORKS. IN 2015 NATIONAL AEROSPACE AND ELECTRONICS CONFERENCE (NAECON), PAGES 339–344. IEEE, 2015.
[49] KEHE WU, ZUGE CHEN, AND WEI LI. A NOVEL INTRUSION DETECTION MODEL FOR A MASSIVE NETWORK USING CONVOLUTIONAL NEURAL NETWORKS. IEEE ACCESS, 6 :50850–50859, 2018
[50] MOSTAFA A SALAMA, HEBA F EID, RABIE A RAMADAN, ASHRAF DARWISH, AND ABOUL ELLA HASSANIEN. HYBRID INTELLIGENT INTRUSION DETECTION SCHEME. IN SOFT COMPUTING IN INDUSTRIAL APPLICATIONS, PAGES 293–303. SPRINGER, 2011.
[51] MIN-JOO KANG AND JE-WON KANG. INTRUSION DETECTION SYSTEM USING DEEP NEURAL NETWORK FOR IN-VEHICLE NETWORK SECURITY. PLOS ONE, 11(6), 2016.
[52] SANDEEP GURUNG, MIRNAL KANTI GHOSE, AND AROJ SUBEDI. DEEP LEARNING APPROACH ON NETWORK INTRUSION DETECTION SYSTEM USING NSL-KDD DATASET. INTERNATIONAL JOURNAL OF COMPUTER NETWORK AND INFORMATION SECURITY (IJCNIS), 11(3) :8–14, 2019.
[53] TUAN A TANG, LOTFI MHAMDI, DES MCLERNON, SYED ALI RAZA ZAIDI, AND MOUNIR GHOGHO. DEEP LEARNING APPROACH FOR NETWORK INTRUSION DETECTION IN SOFTWARE DEFINED NETWORKING. IN 2016 INTERNATIONAL CONFERENCE ON WIRELESS NETWORKS AND MOBILE COMMUNICATIONS (WINCOM), PAGES 258–263. IEEE, 2016.

[54] AHMAD JAVAID, QUAMAR NIYAZ, WEIQING SUN, AND MANSOOR ALAM. A DEEP LEARNING APPROACH FOR NETWORK INTRUSION DETECTION SYSTEM. IN PROCEEDINGS OF THE 9TH EAI IN TERNATIONAL CONFERENCE ON BIO-INSPIRED INFORMATION AND COMMUNICATIONS TECHNOLOGIES (FORMERLY BIONETICS), PAGES 21–26, 2016.
[55] R VINAYAKUMAR, KP SOMAN, AND PRABAHARAN POORNACHANDRAN. APPLYING CONVOLUTIONAL NEURAL NETWORK FOR NETWORK INTRUSION DETECTION. IN 2017 INTERNATIONAL CONFERENCE ON ADVANCES IN COMPUTING, COMMUNICATIONS AND INFORMATICS (ICACCI), PAGES 1222–1228. IEEE, 2017
[56] JIHYUN KIM AND HOWON KIM. APPLYING RECURRENT NEURAL NETWORK TO INTRUSION DETECTION WITH HESSIAN FREE OPTIMIZATION. IN INTERNATIONAL WORKSHOP ON INFORMATION SECURITY APPLICATIONS, PAGES 357–369. SPRINGER, 2015.
[57] NI GAO, LING GAO, QUANLI GAO, AND HAI WANG. AN INTRUSION DETECTION MODEL BASED ON DEEP BELIEF NETWORKS. IN 2014 SECOND INTERNATIONAL CONFERENCE ON ADVANCED CLOUD AND BIG DATA, PAGES 247–252. IEEE, 2014.
[58] SHAHADATE REZVY, MILTOS PETRIDIS, ABOUBAKER LASEBAE, AND TAHMINA ZEBIN. INTRUSION DETECTION AND CLASSIFICATION WITH AUTOENCODED DEEP NEURAL NETWORK. IN INTERNATIONAL CONFERENCE ON SECURITY FOR INFORMATION TECHNOLOGY AND COMMUNICATIONS, PAGES 142–156. SPRINGER, 2018.
[59] MD ZAHANGIR ALOM, VENKATARAMESH BONTUPALLI, AND TAREK M TAHA. INTRUSION DETECTION USING DEEP BELIEF NETWORKS. IN 2015 NATIONAL AEROSPACE AND ELECTRONICS CONFERENCE (NAECON), PAGES 339–344. IEEE, 2015.
[60] KEHE WU, ZUGE CHEN, AND WEI LI. A NOVEL INTRUSION DETECTION MODEL FOR A MASSIVE NETWORK USING CONVOLUTIONAL NEURAL NETWORKS.
[61] PABLO TORRES, CARLOS CATANIA, SEBASTIAN GARCIA, AND CARLOS GARCIA GARINO. AN ANALYSIS OF RECURRENT NEURAL NETWORKS FOR BOTNET DETECTION BEHAVIOR. IN 2016 IEEE BIENNIAL CONGRESS OF ARGENTINA (ARGENCON), PAGES 1–6. IEEE, 2016.
[62] G. Creech and J. Hu, "A Semantic Approach to Host-Based Intrusion Detection Systems Using Contiguousand Discontiguous System Call Patterns," in IEEE Transactions on Computers, vol. 63, no. 4, pp. 807-819, April 2014, doi: 10.1109/TC.2013.13.
[63] Thomas, Ciza & Sharma, Vishwas & Balakrishnan, Narayanaswamy. (2008). Usefulness of DARPA dataset for intrusion detection system evaluation. 10.1117/12.777341.
[64] Vaishali Shirsath, May 12, 2023, "CAIDA UCSD DDoS 2007 Attack Dataset", IEEE Dataport, doi: https://dx.doi.org/10.21227/dvp9-s124.
[65] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 2009, pp.
[66] Sakr, Mahmood & Tawfeek, Medhat & El-Sisi, Ashraf. (2019). Network Intrusion Detection System based PSO-SVM for Cloud Computing. International Journal of Computer Network and Information Security. 11. 22-29. 10.5815/ijcnis.2019.03.04.
[67] https://www.unb.ca/cic/datasets/
[68] https://www.tensorflow.org/learn?hl=fr
[69] https://keras.io/api/
[70] https://matplotlib.org/