UNIVERSITÉ JOSEPH KI-ZERBO
- - - - - - - - - - - - - - - - -
Unité de Formation et de Recherche en Sciences Exactes et Appliquées (UFR-SEA)
- - - - - - - - - - - - - - - - -
Département d'Informatique
- - - - - - - - - - - - - - - - -
LAboratoire de Mathématiques et d'Informatique (LAMI)

MÉMOIRE DE MASTER

Domaine : **Sciences et Technologies**
Mention : **Informatique**
Option : **Ingénierie Logicielle et Système d'Information Informatisé : ILSII**

****

DÉTECTION ET CLASSIFICATION DES INTRUSIONS EN UTILISANT DES TECHNIQUES DE MACHINE LEARNING

****

présenté par :
MANEGAWINDIN LEONARD SAWADOGO
Sous l'encadrement de
Dr Didier BASSOLE
et sous la supervision du
Pr Oumarou SIE

Soutenu le 19 Janvier 2021 devant le jury composé de :

Président :	**Oumarou SIE**	Professeur titulaire, Université Joseph KI-ZERBO
Encadrant :	**Didier BASSOLE**	Maître Assistant, Université Joseph KI-ZERBO
Rapporteur :	**Justin KOURAOGO**	Maître Assistant, Université Joseph KI-ZERBO
Examinateur :	**Boureima ZERBO**	Maître Assistant, Université Thomas SANKARA

Année académique : 2018-2019

Dédicaces

Je dédie ce mémoire :

- *A mon père ;*
- *A ma mère ;*
- *A François Frédéric SAWADOGO ;*
- *A W. Gatien SAWADOGO ;*

qui n'ont cessé et qui ne cessent de me soutenir.

Remerciements

Je remercie Dr Didier BASSOLE de l'Université Joseph KI-ZERBO d'avoir initier ce projet de recherche, d'avoir eu confiance en moi et de m'avoir permis d'y participer. Je le remercie également pour son encadrement, sa disponibilité et ses précieux conseils durant toute la période de recherche.

Mes remerciements vont particulièrement aussi à Pr Oumarou SIE coordonnateur du Master en Ingénierie Logicielle et Système d'Information Informatisé et par ailleurs Directeur adjoint du LAboratoire de Mathématiques et d'Informatique (LAMI).

Merci à M. Gouayon KOALA, doctorant au département informatique, pour les conseils et le suivi, et à M. W. Gatien SAWADOGO, enseignant des lycées et collèges, pour les lectures et les corrections.

Mes remerciements vont également aux membres du jury de m'avoir fait l'honneur d'accepter d'examiner ce travail et à tous les enseignants du Département Informatique de l'UFR/SEA pour la qualité de leur formation et leurs précieux conseils.

Enfin, je remercie mes parents, frères et sœurs, mes ami(e)s et tous les autres membres de ma famille pour leur soutien multiforme, sans quoi ce travail n'aurait pu voir le jour.

Résumé

De nous jours, il existe de nombreux systèmes de détection d'intrusion(en anglais Intrusion Detection System : IDS), mais ceux-ci sont toujours confrontés à deux (02) problématiques majeures à savoir le taux de fausses alertes et la capacité de détection de nouvelles attaques (Attaques *"zero-day"*). La question de savoir comment détecter les différentes attaques du réseau, en particulier celles qui n'ont jamais été vues auparavant, est une question clé que les chercheurs tentent de résoudre.

Dans ce mémoire, nous avons proposé un modèle d'IDS utilisant une technique de deep learning qui est un réseau de neurones convolutif combiné à un arbre de décision dont l'ensemble est appelé Tree-CNN. La structure du modèle permet un apprentissage incrémental. Ce qui le rend donc capable d'apprendre à classifier de nouveaux types d'attaques au fur et à mesure que de nouvelles données arrivent. L'entrainement du modèle a été réalisé en utilisant CSE-CIC-IDS-2018, un ensemble de données contenant les attaques de réseau les plus courantes et les plus récentes. Nous avons, par la suite, évalué les performances du modèle en le comparant avec d'autres résultats de l'état de l'art sur l'apprentissage profond. Le modèle a été implémenté avec Tensorflow. Nous avons obtenu, pour la détection, un score de 99,94% et 97.54% pour la classification.

Mots clés : *Détection d'Intrusion, Deep Learning, CNN, Machine Learning, Apprentissage Incrémental, Tensorflow*

Abstract

Nowadays, there are many intrusion detection systems (IDS), but they are still confronted to two (02) major problems, namely the rate of false alarms and the capacity to detect new attacks (*"zero-day"* attacks). The question of how to detect the different attacks on the network, especially those that have never been seen before, is a key issue that researchers are trying to resolve.

To address these issues, we propose in this master thesis an intrusions detection technique using a deep learning algorithm that can classify differents types of attacks based on user behaviour and not on attack signatures. The deep learning we used is a supervised learning model called Convolutional Neural Networks (CNN) coupled with a tree structure whose set is called Tree-CNN. This structure allows for incremental learning. This makes the model capable of learning how to classify new types of attacks as new information comes in. The model was implemented with Tensorflow and trained with the CSE-CIC-IDS2018 dataset. For intrusions detection, we achieved a accuracy of 99,94% and 97.54% for the classification.

Keyswords : *Intrusion Detection, Deep Learning, CNN, Machine Learning, Incremental Learning, Tensorflow*

Sigles et Abréviations

AM	**A**ttention **M**ecanism
ANN	**A**rtificial **N**eural **N**etwork
API	**A**pplication de **P**rogramming **I**nterface
ARP	**T**ransport **C**ontrol **P**rotocol
CNN	**C**onvolutional **N**eural **N**etwork
DCNN	**D**eep **C**onvolutional **N**eural **N**etwork
GRU	**G**ated **R**ecurrent **U**nit
HIDS	**H**ost **I**ntrusion **D**etection **S**ystem
ICMP	**I**nternet **C**ontrol **M**essage **P**rotocol
IDS	**I**ntrusion **D**etection **S**ystem
IP	**T**ransport **C**ontrol **P**rotocol
LSTM	**L**ong **S**hort-**T**erm **M**emory
MAC	**T**ransport **C**ontrol **P**rotocol
ML	**M**achine **L**earning
NIDS	**N**etwork **I**ntrusion **D**etection **S**ystem
RNN	**R**ecurrent **N**eural **N**etwork
TCP	**T**ransport **C**ontrol **P**rotocol
UDP	**U**ser **D**atagram **P**rotocol
URL	**U**niform **R**esource **L**ocator
KDD	**K**nowledge **D**iscovery **D**ata Mining
CSE	**C**ommunications **S**ecurity **E**stablishment
CIC	**C**anadian **I**nstitute for **C**ybersecurity
SVM	**C**anadian **I**nstitute for **C**ybersecurity
KNN	**C**anadian **I**nstitute for **C**ybersecurity
RBM	**C**anadian **I**nstitute for **C**ybersecurity
DOS	**C**anadian **I**nstitute for **C**ybersecurity
DDOS	**C**anadian **I**nstitute for **C**ybersecurity
IDC	**I**nternationa **D**ata **C**orporation

Table des matières

Liste des tableaux

Table des figures

Introduction générale

De nos jours, les systèmes d'informations et les réseaux informatiques sont au cœur de la société et des entreprises actuelles dû aux multiples avantages incontestables qu'ils offrent. Ils sont dans tous les domaines, même ceux sensibles comme la santé, l'éducation, l'administration, la sécurité, etc. Notre dépendance aux services fournis et/ou aux données qu'ils contiennent est tellement grande et intense à tel point que la compromission de leur sécurité peut entraîner des conséquences très graves. La probabilité des scénarios les plus pessimistes augmente avec l'évolution très rapide du nombre des utilisateurs, du nombre et des capacités des appareils (vitesse de calcul, capacité de stockage, etc.) et des capacités des réseaux. À titre d'exemple, en 2010, le monde ne comptait que deux zettaoctets de données numériques. En 2015, ce chiffre avait été multiplié par six (06). Selon IDC [1], le volume mondial de données sera multiplié encore par 3,7 entre 2020 et 2025, puis par 3,5 tous les cinq ans jusqu'en 2035, pour atteindre la taille vertigineuse de 2 142 zettaoctets [92]. En 2020, les appareils connectés seraient au nombre de 30,73 milliards dans le monde[90]. Un autre exemple, entre 2000 et 2019, l'Afrique a connu une augmentation remarquable de l'utilisation de l'Internet de près de 10 000% [87].

En Informatique, les intrusions consistent en général à des tentatives pour accéder, sans autorisation, aux données d'un système informatique ou d'un réseau en contournant ou en désamorçant les dispositifs de sécurité mis en place. Autrement, une intrusion se définit comme toute tentative pouvant nuire à la confidentialité, l'intégralité ou la disponibilité d'un système informatique ou d'un réseau ainsi que toute tentative visant à contourner les dispositifs de sécurité mis en place.

Pour contrer les intrusions, un système de réseau doit utiliser un ou plusieurs outils de sécurité tels qu'un pare-feu, un logiciel antivirus ou un système de détection d'intrusions pour protéger les données et les services importants. S'appuyer sur un système de pare-feu seul ne suffit pas à empêcher un réseau d'entreprise de subir tous les types d'attaque de réseau. En effet, un pare-feu ne peut pas défendre le réseau contre les tentatives d'intrusion sur les ports ouverts nécessaires aux services du réseau. C'est pourquoi un système de détection d'intrusion en anglais Intrusion Detection System (IDS)[11] qui est fait pour répondre aux caractéristiques dynamiques des attaques, est généralement installé en complément du pare-feu.

Un système de détection d'intrusion, est une technique de cybersécurité importante,

1. `https://www.idc.com/`

qui surveille l'état des logiciels et/ou du matériel et son rôle est de repérer les actions d'un attaquant tentant de tirer parti des vulnérabilités du système pour nuire aux objectifs de sécurité. Il envoie un signal à l'utilisateur de l'ordinateur ou à l'administrateur du réseau lorsqu'il suspecte une intrusion. Ce signal permet au destinataire d'inspecter le système afin de détecter et de contrer l'attaque éventuelle.

Les systèmes de détection d'intrusion, pour identifier les attaques, peuvent s'appuyer sur les informations relatives aux transitions ayant lieu dans le système (l'exécution de certains programmes, de certaines séquences d'instructions, l'arrivée de certains paquets réseau, ...) ou sur l'étude de l'état de certaines propriétés du système (intégrité des programmes ou données, les privilèges des utilisateurs, les transferts de droits, ...). Ils sont nombreux et diversifiés en terme de fonctionnalités et mode de fonctionnement[81][80][83][84][82][44][35][54][59][49][42][19].

Avant l'invention des IDS, la détection d'intrusion se faisait à la main, car toutes les traces devaient être imprimées afin que les administrateurs puissent y déceler des anomalies. C'est une activité très chronophage et pas très efficace, car utilisée après les attaques afin de déterminer les dommages et de retrouver comment les assaillants s'y sont pris pour entrer dans le système. Puis des chercheurs ont développé les premiers programmes d'analyse de traces, mais cela reste inefficace, car ces programmes sont lents et fonctionnent la nuit lorsque la charge sur le système est faible, donc les attaques sont le plus souvent détectées après coup. Ce n'est qu'au début des années 90, qu'apparaissent les premiers programmes d'analyse en temps réel, qui permettent d'analyser les traces dès qu'elles sont produites. Entre 2006 et 2010, le nombre d'attaques est passé d'environ 5000 à plus de 35000[25]. En 2018 on a eu trente millions (30.000.000) d'attaques dans le monde[78]. Une attaque de hackers se produit toutes les 39 secondes [79]. L'Afrique, malgré son taux d'utilisation du numérique encore faible, selon un rapport publié en septembre 2020, Kaspersky (éditeur de logiciels de cybersécurité) y a recensé 28 millions de cyberattaques entre janvier et août 2020. D'où le besoin d'avoir des IDS plus performants. Les tentatives d'intrusions en passant outre les mesures de sécurité sont de plus en plus nombreuses, de plus en diversifiées et surtout de plus en plus sophistiquées. Ajouter à cela, l'augmentation des capacités des équipements et des réseaux rend possible des attaques plus subtiles ou plus puissantes. Il est alors difficile pour un IDS qui est basé sur des règles établies même par des experts du domaine, de détecter efficacement ces attaques. A l'heure du changement perpétuel de l'environnement informatique et des innombrables nouvelles attaques dévoilées chaque jour, comment avoir des IDS capables d'y faire face ? L'idée d'exploiter la puissance des techniques de machine learning est alors apparue pour contribuer à y répondre. En effet le machine learning nous aide à modéliser la normalité. Un modèle mis au point permettra de décrire une situation donnée (le fonctionnement d'un réseau, les événements sur une machine, etc.). On pourra alors qualifier une situation donnée et d'en déterminer la distance par rapport à la normalité. Si l'écart est trop important, cela peut donner lieu à une alerte. Il faut aussi qualifier cette « anormalité » et permettre à l'administrateur d'appliquer la solution adéquate. L'option de la classification des attaques permet de qualifier ces « anormalités » .

Les techniques de machine learning ont été activement utilisées pour la sécurité des systèmes d'information ces dernières années. De nombreuses études sur les IDS basées sur le machine learning ont utilisé l'ensemble de données KDD CUP 98 (Knowledge Discovery Data Mining) ou ses versions mises à jour. Dans ce travail, nous développons un modèle d'IDS en utilisant CSE-CIC-IDS-2018(Communications Security Establishment-Canadian Institute for Cybersecurity-Intrusion Detection System-2018), un ensemble de données contenant les attaques de réseau les plus courantes et les plus récentes. Nous utilisons des techniques d'apprentissage profond en développant un modèle de réseau neuronal convolutif et hiérarchique appelé Tree-CNN[75]. Nous évaluons ensuite ses performances en le comparant avec d'autres résultats de l'état de l'art sur l'apprentissage profond.
L'objectif global de ce travail est de contribuer à améliorer les performances des IDS notamment le taux de fausses alertes et la détection des attaques *zero day* en exploitant la puissance du machine learning. Les objectifs spécifiques de ce travail sont :

- proposer une démarche et une architecture d'un système de détection et de classification des intrusions intégrant des techniques de Machine Learning avec un processus de détection puis de classification ;
- implémenter un système de détection et de classification d'intrusions et analyser ses performances à travers une étude comparative avec d'autres systèmes existants rencontrés dans la littérature.

La suite de ce mémoire est organisée en quatre (04) chapitres comme suit :

- dans le chapitre 1 nous nous attarderons sur quelques concepts sur les attaques et les systèmes de détection d'intrusion ;
- le chapitre 2 sera consacré à la présentation de l'état de l'art sur l'implication des techniques de machine learning dans le domaine des systèmes de détection d'intrusions ;
- au niveau du chapitre 3 nous nous étalons sur la démarche et l'architecture de notre modèle ;
- dans le quatrième chapitre nous expliquerons l'implémentation et présenterons les tests et les résultats de notre modèle.

Chapitre 1

État de l'art

Sommaire

De nos jours la recherche autour des IDS intéresse beaucoup les chercheurs en particulier celle qui implique les méthodes de machine learning. Comme l'atteste le nombre important d'articles dans le domaine. Ces auteurs traitent de diverses méthodes de machine learning(supervisées, non supervisées, etc.). Les problématiques de données étant très importantes pour le machine learning, plusieurs autres chercheurs traitent de celles-ci. Ces auteurs discutent sur la quantité, la qualité, la disponibilité et la compatibilité avec les différentes méthodes de machine learing. Nous présentons dans ce chapitre, un panorama des différentes propositions de système de détection d'intrusion utilisant le machine learning, afin de bien situer notre travail et notre contribution dans ce domaine.

1.1 Moyen de détection des attaques : les IDS

Un bien ou une propriété de valeur est très généralement protégé contre le vol et la destruction, et caché à l'abri. Les banques par exemple, sont équipées de systèmes de sécurité(alarme, système anti-incendie, etc.) qui peuvent décourager les voleurs ou rendre leur but très difficile à atteindre. Cette même préoccupation d'assurer la disponibilité et l'intégrité, en gros la sécurité des biens, est aussi pressente dans le cadre des systèmes, réseaux et données informatiques. Des alarmes sont nécessaires pour prévenir les administrateurs et les membres de l'équipe de sécurité qu'une effraction s'est produite afin qu'ils puissent répondre à temps au danger. L'idée derrière les systèmes de détection d'intrusions est d'avoir un système capable de jouer le rôle d'une telle alarme. Le premier système de détection d'intrusion a été proposé en 1980 par Anderson et al. dans leur article *Computer Security Threat Monitoring and Surveillance* [2]

1.1.1 Définition de IDS

Un système de détection d'intrusion (IDS, de l'anglais Intrusion Detection System) est utilisé pour détecter les attaques contre un réseau ou un système informatique à un stade précoce. Les systèmes de détection d'intrusion sont des périphériques ou des processus qui surveillent et analysent toutes les activités du réseau ou du système, pour détecter toute entrée non autorisée ou toute activité malveillante. Puis ils avertissent l'utilisateur ou l'administrateur. Un IDS est une application de sécurité informatique qui vise à détecter un large éventail de violations de la sécurité, allant des tentatives d'effraction par des personnes extérieures aux pénétrations dans le système en passant par des abus par des initiés [3]. Fondamentalement, une distinction est faite entre les systèmes de détection, selon leur mode de fonctionnement et selon leur position sur le réseau. Il existe d'autres critères de distinction des IDS, mais qui sont moins mise en emphase. Les principales catégories des IDS sont décrites ci-dessous.

1.1.2 Selon le mode de fonctionnement

Les IDS disposent de trois (03) principales approches différentes selon leur manière de fonctionnement, pour la détection des intrusions :

- **basée sur la signature** : cette approche consiste à rechercher dans l'activité de l'élément surveillé les empreintes d'attaques connues, à l'instar des antivirus. C'est l'approche la plus basique et la plus ancienne. Une signature est habituellement définie comme une séquence d'événements et de conditions relatant une tentative d'intrusion. La reconnaissance est alors basée sur le concept de « pattern matching ». C'est-à-dire en analysant les chaînes de caractères présentes dans le paquet, à la recherche de correspondance au sein d'une base de connaissance. Il est aisé de comprendre que ce type d'IDS est purement réactif ; il ne peut détecter que les attaques dont il possède la signature. De ce fait, il nécessite des mises à jour quotidiennes. De plus, ce système de détection est aussi bon que l'est la base de signature. Si les signatures sont erronées ou incorrectement conçues, l'ensemble du système est inefficace. C'est pourquoi ces systèmes sont souvent contournés par les pirates qui utilisent des techniques dites « d'évasion » qui consistent à maquiller les attaques. Ces techniques de maquillage tendent à faire varier les signatures des attaques qui ainsi ne sont plus reconnues par l'IDS. Ce modèle est par contre très aisé à implémenter et à optimiser. Il permet la séparation du moteur logiciel de la base de signature qui peut ainsi être mise à jour indépendamment. Il permet également une classification relativement facile de la criticité des attaques signalées ;
- **basée sur le comportement** : les IDS comportementaux ont pour principale fonction la détection d'anomalie, d'un comportement déviant ou d'un cas inhabituel. Leur déploiement nécessite une phase d'apprentissage pendant laquelle l'outil va apprendre le comportement "normal". De par leur conception, ils pourront ainsi détecter toute activité "anormale" mais ne pourront pas spécifier l'attaque ni son niveau de criticité. Ce type d'IDS pourra détecter de nouvelles attaques non encore connu ou des variantes des attaques existantes en signalant tout comportement inhabituel. En effet le point fort des IDS comportementaux est leur capacité de détection des nouveaux type d'attaque. Mais l'utilisation de tels IDS peut s'avérer délicate dans le sens où les alarmes remontées contiendront une quantité importante de fausses alertes qui sont dues au fait que tout cas inhabituel n'est une attaque. Ces IDS signaleront par exemple tout changement dans le comportement d'un utilisateur qu'il soit hostile ou non.
- **les hybrides** : cette approche essaie de combiner les deux décrites ci-dessus.

1.1.3 Selon la position

De par la position de l'IDS, on peut classer les IDS en plusieurs catégories qui sont :

- **IDS basé sur un hôte(HIDS : hote IDS)** : le premier IDS basé sur un hôte fut utilisé déjà dans les années 80 pour protéger les structures informatiques centralisées. Ils analysent le fonctionnement ou l'état des machines sur lesquelles ils sont installés, les logs ainsi que toute information se trouvant sur la machine afin de détecter les attaques. Ils sont très dépendants du système sur lequel ils sont installés. Il faut donc des outils spécifiques en fonction des systèmes déployés. Ces IDS peuvent

s'appuyer sur des fonctionnalités d'audit propres au système d'exploitation hôte ou non pour vérifier l'intégrité du système et générer des alertes. Mais ces IDS ne sont pas à mesure de détecter les attaques affectant les couches réseau de la machine, par exemple les attaques DoS comme SYN FLOOD ou autre ;

- **système de détection d'intrusion basé sur le réseau(NIDS :Network IDS)** : ces IDS analysent le trafic réseau, ils comportent généralement une sonde qui « écoute » sur le segment de réseau à surveiller et un moteur qui réalise l'analyse du trafic afin de détecter les signatures d'attaques ou les divergences face au modèle de référence. Les NIDS à base de signatures sont confrontés actuellement à deux problèmes majeurs qui sont : le développement de l'utilisation du cryptage et le développement des réseaux commutés. En effet, il est d'une part plus difficile « d'écouter » sur les réseaux commutés et le cryptage rend l'analyse du contenu des paquets presque impossible. La performance de ces IDS est très importante, car ils sont amenés à analyser des volumes de plus en plus importants pouvant transiter sur les réseaux, et cela avec la contrainte de temps de réponse qui doit être instantanée ;
- **système de détection d'intrusion hybride** : dans ce type d'IDS, les sources d'information viennent à la fois du réseau et des machines sur le réseau. De ce fait, la complexité du système augmente, mais les avantages aussi dus à la combinaison des points forts des NIDS et des HIDS. De plus, ils permettent une meilleure détection d'attaques distribuées.

1.1.4 En ligne ou Hors ligne

Une autre critère de différenciation peut se faire sur la caractéristique *en ligne* ou *hors ligne* de l'IDS. Le premier détecte une attaque au moment où elle se produit alors que le deuxième s'exécute périodiquement et ne constate que le résultat de l'attaque. La deuxième méthode est préférable pour avoir une dépense plus faible du point de vue du temps de calcul que la première, due à l'aspect temps-réel de l'approche *en ligne*. Cependant, l'IDS *hors ligne* , contrairement celui *en ligne*, ne permet pas de détecter une attaque à temps pour donner une possibilité à ce qu'il soit évitée. Plus le temps mis avant la détection de l'attaque est grand, plus les dégâts et dommages sont susceptibles d'être grands aussi. Généralement, les attaques réseaux nécessitent une activité préalable sur le réseau par l'attaquant comme le scanne des ports, l'envoi de certains types de requête, etc. Si cette activité est détectée avant l'attaque, elle pourra alors être évitée.

1.1.5 Active ou Passive

Les IDS sont aussi différents selon le type de réponse apportée en cas de détection d'une attaque. Certains IDS envoient une alerte à un humain et lui laisse le soins de trouver et d'apporter une solution pour contrer l'attaque. On dit que ces IDS sont *passives*. Mais cela implique que l'humain doit avoir une certaine compétence dans le domaine et surveiller continuellement l'IDS pour ne pas « rater » les alertes. L'autre catégorie d'IDS qui est dite *active*, en cas de détection d'attaque, effectue des actions pour contrer l'attaque. Ainsi la

réaction à l'attaque est rapide. Néanmoins, ceci peut être dangereux. En effet, supposons qu'un attaquant usurpe l'identité d'un système « A » ou arrive à prendre son contrôle et lui demande d'attaquer un autre système « B ». Si l'IDS de « B » détecte et répond à l'attaque en attaquant « A », et l'IDS de « A » aussi à son tour détecte et répond à l'attaque de « B » en attaquant « B », ceci peut avoir des conséquences dramatiques. Par exemple tous les deux systèmes pourraient être indisponibles pour répondre aux autres demandes de service. Ce qui correspond à un déni de service. De plus, dû à la possible existence de fausses alertes, il existe un grand risque d'avoir des réponses inappropriées.

1.1.6 Autres types d'IDS

A part les principaux types d'IDS cités ci-haut on distingue plusieurs autres types dont :

- **WIDS** de l'anglais « Wireless Intrusion Detection Systems », c'est un système de détection d'intrusion utilisant les réseaux sans fil. Il sont destinés à prévenir toute intrusion provenant d'un appareil utilisant un réseau sans fil.
- **VMBIDS** : de l'anglais « Virtual Machine Based Intrusion Detection Systems », il utilise une machine virtuelle pour renforcer la sécurité du réseau contre toute menace potentielle. Le principal avantage de ce système est la possibilité d'analyser les états de la machine virtuelle. Le système est aussi plus complexe, compliquant ainsi la tâche aux pirates.

1.2 Le machine learning appliqué à la sécurité

1.2.1 Les modèles

Les techniques de machines learning sont souvent classifiées en plusieurs groupes comme le montre la figure 4.1. Deux groupes principaux nous intéressent à savoir l'apprentissage supervisé et l'apprentissage non-supervisé. Plus de détails sur la classification des modèles sont abordés par *Ayon Dey* dans son article[39]. On a aussi une autre manière de les distinguer à savoir l'apprentissage profond et l'apprentissage peu profond. Nous présentons quelques modèles liés à ces différents groupes dans l'annexe du document.

1.2.1.1 Arbre de décision

Il fait partie du groupe d'apprentissage peu et supervisé. Un arbre de décision est une structure arborescente constituée de nœuds non terminaux(une racine et des nœuds internes) et de nœuds terminaux(feuilles). Le nœud racine est le premier nœud avec des

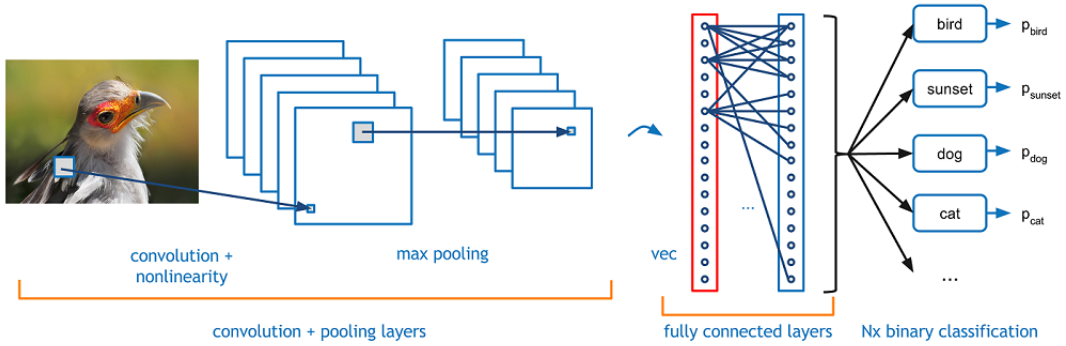

FIGURE 1.1 – Les couches de CNN[77]

conditions de test pour diviser chaque enregistrement de données d'entrée vers chaque nœud interne en fonction des caractéristiques de l'enregistrement de données[5].

Tout d'abord, l'arbre de décision est formé avec des données étiquetées avant de pouvoir classer des données qui ne sont pas utilisées pour l'entraînement. Le processus d'entraînement comprend la sélection des attributs, la génération d'arbres et l'élagage des arbres. Lors de l'apprentissage d'un modèle d'arbre de décision, l'algorithme sélectionne un à un les attributs les plus appropriées et génère des nœuds enfants à partir du nœud racine. L'algorithme d'arbre de décision peut automatiquement exclure les attributs non pertinents et redondants. Après la formation, l'arbre peut prévoir ou classer de nouvelles données en partant d'un nœud racine et en traversant les nœuds internes en fonction des attributs des conditions de test jusqu'à ce qu'il arrive à un nœud feuille (terminal) constitué d'une classe de réponse[7].

Le modèle ressemble à un arbre, ce qui rend son interprétation beaucoup facile. Aussi, l'arbre de décision a une grande précision de la classification et simple à mettre en œuvre. Certains algorithmes avancés, tels que la forêt aléatoire (Random Forest) et le renforcement du gradient extrême (XGBoost), sont constitués de plusieurs arbres de décision[34].

1.2.1.2 CNN

Ils font partie du groupe d'apprentissage profond et supervisé.Les CNN[45] sont conçus pour imiter le système visuel humain. Par conséquent, ils ont fait de grandes avancées dans le domaine de la vision par ordinateur. Un CNN est empilé avec des couches de convolution et d'autres couches alternées. Les couches de convolution sont utilisées pour extraire les *features*, et les couches de *pooling* sont utilisées pour améliorer la généralisation des *features*.

Les quatre principales opérations de CNN sont la couche de convolution, la fonction d'activation telle que ReLu ou Softmax, la couche de *pooling* et la couche entièrement connectée (*fully connected*) comme le montre la figure fig.1.1.

- **La couche de convolution** : CNN tire son nom des opérations de convolution. La tâche principale de la convolution est d'extraire des *features* de la donnée d'entrée. La donnée de sortie issue de l'opération de convolution est appelée « *Activation Map* », « *Convolved Feature* » ou « *Featured Map* ». Les valeurs du filtre ou du noyau sont

mises à jour automatiquement pendant le processus de formation de CNN pour mieux apprendre les *features* d'une donnée. Des informations plus détaillées sur la couche de convolution sont présentées par Jianxin Wu(2017) [52].

- **Fonction d'activation ReLu** : après chaque opération de convolution, avant de générer la « *Activation Map* », une fonction non linéaire supplémentaire telle que *ReLu* est utilisée. *ReLu* signifie en anglais *Rectified Linear Unit* et est une opération non-linéaire. *ReLu* introduit le comportement non linéaire dans CNN. D'autres fonctions d'activation non linéaires telles que *tanh* et *sigmoïde* peuvent également être utilisées à la place de *ReLu*. Des informations plus détaillées sur la fonction d'activation *ReLu* et d'autres fonctions d'activation peuvent être trouvées dans Agarap(2018) [55].
- **Couche de *pooling*** : L'opération de *pooling* réduit la dimension de chaque « *Activation Map* » mais conserve les informations les plus importantes. L'opération de *pooling* peut être de différents types tels que *Max*, *Moyenne*, *Somme*, etc. La mise en commun Max s'est avérée plus efficace dans de nombreuses applications. Des informations plus détaillées sur la couche de mise en commun sont présentées par Haibing Wu(2015) [38].
- **Couche entièrement connectée ou *fully connected*** : Il s'agit d'un perceptron multicouche traditionnel qui utilise une fonction d'activation *Softmax* dans la couche de sortie. Le terme « entièrement connecté » implique que chaque neurone de la couche précédente est connecté à chaque neurone de la couche suivante. La sortie des couches de convolution et de *pooling* représente les *features* de haut niveau de la donnée d'entrée. La couche entièrement connectée utilise ces *features* pour classer la donnée d'entrée en différentes classes sur la base de l'ensemble de données d'entrainement. Plus de détails sur les couches *fully connected* sont présentées par Karn(2016) [41].

Les CNN travaillent sur des données sur plusieurs dimensions ou unidimensionnelles (1D), de sorte que les données d'entrée doivent être traduites en matrices pour la détection des attaques.

Le pré-traitement requis dans CNN est beaucoup moins important que pour les autres algorithmes de classification. Alors que dans les méthodes primitives, les *features* sont sélectionnés par un expert, avec une formation suffisante, les CNN ont la capacité de sélectionner les *features* les plus représentatifs. Nous reviendrons sur CNN dans le chapitre 2.

1.2.2 Les datasets

La tâche de l'apprentissage machine consiste à extraire des informations précieuses des données. Par conséquent, la performance de l'apprentissage machine dépend de la qualité des données d'entrée. La compréhension des données est la base de la méthodologie du machine learning. Pour les IDS, les données adoptées doivent être faciles à acquérir et refléter les comportements des hôtes ou des réseaux. Les types de données sources les

plus courantes pour les IDS sont les paquets, les flux, les sessions et les journaux. La constitution d'un ensemble de données est complexe et prend du temps. Une fois qu'un ensemble de données de référence est construit, il peut être réutilisé à plusieurs reprises par de nombreux chercheurs. Outre la commodité, l'utilisation d'ensembles de données de référence présente deux autres avantages :

1. Les ensembles de données de référence font autorité et rendent les résultats expérimentaux plus convaincants.
2. De nombreuses études publiées ont été réalisées à l'aide d'ensembles de données de référence communs, ce qui permet de comparer les résultats des nouvelles études avec ceux des études précédentes.

Les ensembles de données les plus utilisés dans la recherche sont cités dans l'annexe du document.

1.2.2.1 CSE-CIC-IDS2018[91]

L'utilisation des techniques de machine learning dans le cadre des IDS est très en vogue dans le domaine de la recherche scientifique. Mais ces techniques nécessitent de grandes quantités de données qui reflètent au mieux le monde réel des réseaux informatiques et systèmes d'information. C'est en soi un défi important, car la disponibilité des ensembles de données est rare. La raison est que d'une part, beaucoup de ces ensembles de données sont internes et ne peuvent être partagés pour des raisons de confidentialité. D'autre part, les autres sont fortement anonymisés et ne reflètent pas souvent les tendances actuelles, ou bien il leur manquent certaines caractéristiques statistiques, de sorte qu'un ensemble de données parfait n'existe pas encore.

Pour pallier ces lacunes, l'Institut canadien de cybersécurité en anglais, Canadian Institute for Cybersecurity (CIC) en collaboration avec le Centre de sécurité des télécommunications en anglais, Communications Security Establishment (CSE) ont mis en place une approche pour générer des ensembles de données permettant d'analyser, de tester et d'évaluer les systèmes de détection des intrusions, en mettant l'accent sur les détecteurs d'intrusions sur le réseau. L'objectif principal de leur projet est de développer une approche systématique pour générer des ensembles de données de référence, divers et complets pour la détection des intrusions. L'institut se base sur la création de profils d'utilisateurs qui contiennent des représentations abstraites des événements et des comportements vus sur le réseau. Les profils seront combinés pour générer un ensemble diversifié de données. Chaque profil a un ensemble unique de caractéristiques, qui couvre une partie du domaine d'évaluation. Ils ont réussi à capturer une importante quantité de données en 2018 qu'ils ont nommé *CSE CIC-IDS2018*.

L'ensemble de données que nous avons utilisé est le CSE CIC-IDS2018 [91]. C'est l'un des ensembles de données publiques les plus récents, les plus réalistes et les plus complets en terme de types d'attaques qu'ils contiennent. Les attaques que le CSE CIC-IDS2018 contient sont d'actualité et reflètent au mieux les menaces actuelles auxquelles font face les réseaux et les systèmes d'informations de nos jours. L'ensemble de données CSE CIC-

Label	Nombre
Benign	13484708
DDOS attack-HOIC	686012
DDoS attacks-LOIC-HTTP	576191
DoS attacks-Hulk	461912
Bot	286191
FTP-BruteForce	193360
SSH-Bruteforce	187589
Infilteration	161934
DoS attacks-SlowHTTPTest	139890
DoS attacks-GoldenEye	41508
DoS attacks-Slowloris	10990
DDOS attack-LOIC-UDP	1730
Brute Force -Web	611
Brute Force -XSS	230
SQL Injection	87

TABLE 1.1 – Répartition des données de CSE CIC-IDS2018

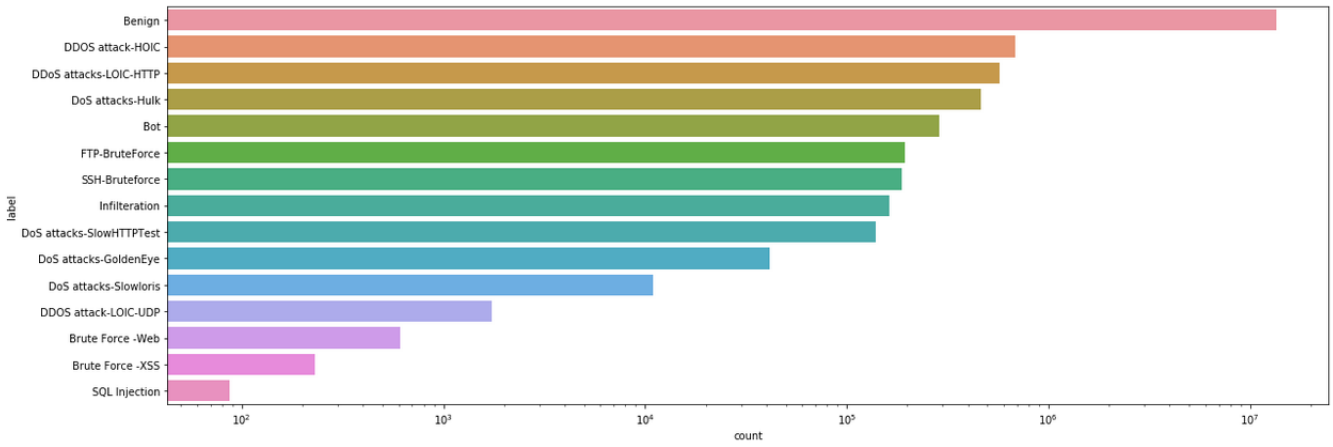

FIGURE 1.2 – Types d'attaque contenus dans CSE CIC-IDS2018

IDS2018 contient 86 "features" ou caractéristiques qui sont dans le tableau Table 4.2.

Les données que contient CSE CIC-IDS2018 sont étiquetées avec quatorze (14) étiquettes(selon le type d'attaque et l'outil utilisé pour effectuer l'attaque) qui sont présentées dans le tableau Table 1.2. Le nombre de ligne de l'ensemble de données est plus de seize millions (16.000.000) et est réparti comme le montre le tableau Table 1.1 selon le type d'attaque.

Les figures Figure 1.2 et Figure 1.2, montrent clairement qu'il y a un déséquilibre des données. Le label "benign" représente à lui seul plus du 80% Table 1.2 des données. Effectivement, dans l'usage de tous les jours, les cas bénins dépassent largement les cas d'attaque. Mais pour que le modèle n'accorde plus de poids au label "benign" par rapport aux autres lors de l'entraînement, il faut équilibrer les données.

Un autre constat est qu'il existe dans l'ensemble de données, des valeurs *nulles* ou *vides*, des valeurs *infinies* et l'entête des données se répète plusieurs fois. Tout cela implique une

Label	Nombre	pourcentage
Bénigne (0)	13484708	0,8307
Attaque (1)	2748235	0,1693

TABLE 1.2 – Répartition des données de CSE CIC-IDS2018 en « Bénigne » et « Attaque »

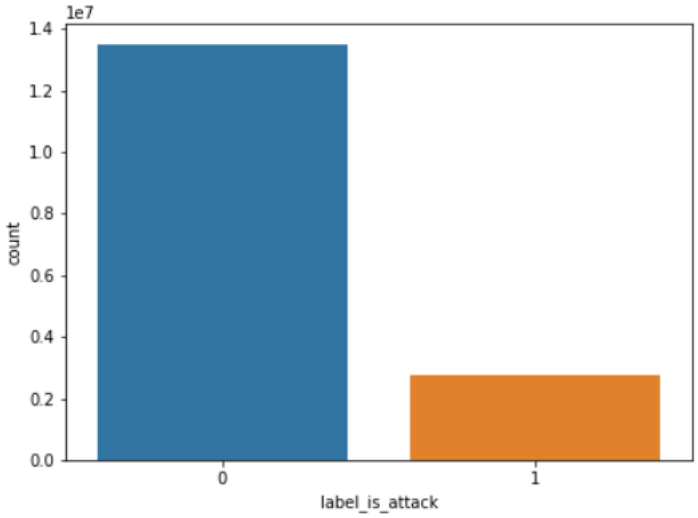

FIGURE 1.3 – Répartition des données de CSE CIC-IDS2018 en « Bénigne » et « Attaque »

nécessité de pré-traitement des données.

1.3 Les travaux dans le domaine

Plusieurs travaux ont été effectués dans le domaine des IDS depuis plusieurs décennies. Nous avons choisi de présenter quelques un qui ont eu lieu entre 2015 et 2020 représentant donc l'état de l'art actuel. Nous avons ajouté quelques autres qui sont des références du domaine. Cette option a été prise pour montrer la tendance de recherche actuelle dans ce domaine, les acquis et les défis de nos jours.

Alex Shenfield et al. [62] présentent une approche pour détecter les trafics malicieux sur les réseaux à l'aide de réseaux neuronaux artificiels basés sur l'inspection approfondie des paquets du réseau. Mille (1000) octets de données de l'ensemble de données KDD CUP 1999 ont été extraits et utilisés comme entrée pour l'ANN. La structure optimale de l'ANN a été trouvée grâce à un processus de recherche par grille(Grid Search [1]), la meilleure structure (en termes de précision de classification) pour l'ANN s'est avérée être un perceptron multicouche (MLP) avec deux couches cachées de 30 neurones chacune. La stratégie d'apprentissage par rétro-propagation résiliente (utilisant un taux d'apprentissage par défaut de 0,01 et une formation pour un maximum de 1000 époques) a été utilisée pour former le réseau de neurones. L'architecture de réseau neuronal artificiel proposée obtient une précision moyenne de 98% et un taux moyen de faux positifs inférieur à 2% lors d'une 10-fold cross-validation. Cela montre que la technique de classification proposée

1. `https://www.sciencedirect.com/topics/engineering/grid-search`

est assez robuste, précise et exacte, mais le taux de fausses alarmes reste quand même élevé.

Li et al. [66] ont proposé une approche de deep leraning pour la détection des intrusions au niveau du réseau. Dans ce travail, les auteurs ont utilisé deux réseaux de neurones récurrents (RNN) à apprentissage profond avec un nombre variable de couches cachées : Long Short Term Memory (LSTM) et Gated Recurrent Unit (GRU). Ils ont évalué également le système d'apprentissage large "Bureau of Labor Statistics" (BLS) et ses extensions. Les modèles sont formés et testés en utilisant les ensembles de données du Border Gateway Protocol (BGP) qui contiennent des enregistrements de routage collectés auprès des Réseaux IP Européens (RIPE) et de BCNET [2] ainsi que l'ensemble de données NLS-KDD contenant les enregistrements de connexion au réseau. Ils ont soutenu la technique BLS comme une alternative possible aux méthodes d'apprentissage profond sophistiquées, car elle offre des taux de performance similaires(F1-measure compris entre 90% et 95%) avec moins de temps de formation. Ces scores restent néanmoins moins en dessous des meilleurs du domaine.

N. Chockwanich et al.[68] ont proposé une technique de détection des intrusions en utilisant un modèle d'apprentissage profond qui peut classer différents types d'attaques sans règles générées par l'homme ou cartographie des signatures. Ils ont utilisé des modèles d'apprentissage profond supervisé dont RNN et CNN pour classer cinq types d'attaques populaires en utilisant Keras avec TensorFlow. Ils ont précisé que leur technique ne requiert que les informations d'en-tête du paquet et ne nécessite aucune charge utile pour l'utilisateur. Pour évaluer les performances, les auteurs ont utilisé l'ensemble de données MAWI qui sont des fichiers pcap et ils ont comparé leurs résultats avec ceux de Snort. Les résultats montrent que Snort n'a pas pu détecter l'attaque de scan réseau via ICMP et UDP. Ils ont prouvé que RNN et CNN peuvent être utilisés pour classer les attaques de type Port scan, Network scan via ICMP, Network scan via UDP, Network scan via TCP, et DoS attack avec une grande précision. RNN est le modèle qui offre la plus grande précision (0.9976 pour RNN et 0.9956 pour CNN) dans leur cas mais il consomme plus de puissance de calcule et plus de temps d'entraînement que le modèle CNN. Ces résultats ont été obtenu en mode *hors ligne*, mais dans le monde réel, la détection se fait en mode *en ligne* c'est à dire en temps réel. Cependant cela n'est pas testé et ne permet donc pas d'apprécier les performances du modèle dans la détection des attaques en temps réel.

Liang Zhou et al.[67] ont proposé un système qui utilise un modèle de réseau neuronal profond pour la classification des cyberattaques. Les auteurs ont proposé une nouvelle approche qui utilise le modèle de réseau neuronal profond (DNN).Ils sélectionnent les paramètres globaux optimaux pour obtenir une haute performance de généralisation. Le résultat de l'évaluation démontre que la méthode proposée peut efficacement identifier les cyber-attaques dans le réseau avec une précision pouvant atteindre 96%. Le score est assez bon mais il est un peu en dessous des meilleurs du domaine.

N. Shone [63] et ses collaborateurs ont proposé un modèle qu'ils ont appelé NDAE(*Non-Symmetric Deep Auto-Encoder*). C'est un auto-encodeur comportant de multiples couches

2. `https://www.bc.net/about-us`

cachées non symétriques. Fondamentalement, cela implique le passage du paradigme de l'encodeur-décodeur (symétrique) à l'utilisation de la seule phase de l'encodeur (non symétrique). Le raisonnement sous-jacent est qu'avec une structure d'apprentissage correcte, il est possible de réduire les coûts de calcul et de temps, avec un impact minimal sur la précision et l'efficacité. Le résultat des tests du modèle atteint jusqu'à 0,995999 +/- 0,000556, ce qui est un résultat très prometteur. Ils ont mis en œuvre le modèle en utilisant TensorFlow et l'ont évalué sur les ensembles de données de référence KDD Cup'99 et NSL-KDD. Contrairement à la plupart des travaux, les auteurs ont évalué les performances de leur modèle sur la base des deux ensembles de données de référence, révélant un niveau constant de précision de la classification. Bien que leur modèle ait obtenu les résultats prometteurs mentionnés ci-dessus, les auteurs reconnaissent qu'il n'est pas parfait et qu'il y a encore des possibilités d'amélioration. La première piste d'amélioration consiste à évaluer et à étendre la capacité du modèle à traiter les attaques de type "zero-day" et la deuxième piste consiste à tester le modèle dans les conditions réelles.

Dans leur article [57] Leila Mohammadpour et al. ont obtenu des résultats expérimentaux, un taux de détection de 99,79%, en utilisant CNN et l'ensemble de données NSL-KDD. Cela montre que les CNN peuvent être appliqués comme méthode d'apprentissage pour les systèmes de détection d'intrusion (IDS). Ils ont utilisé deux couches de convolution, deux couches de "pooling" et trois couches entièrement connectées. Cependant Leila Mohammadpour et al. n'ont pas donné d'informations sur les autres métriques de performance comme le taux de fausses alarmes.

WEI WANG et al. ont proposé dans leur article [51], un nouveau modèle appelé système de détection d'intrusion hiérarchique spatio-temporel (HAST-IDS), qui apprend d'abord les *features* spatiaux de bas niveau du trafic réseau en utilisant des réseaux neuronaux convolutifs (CNN) et ensuite les *features* temporels de haut niveau en utilisant des réseaux à mémoire long et court terme (LSTM). L'ensemble du processus d'apprentissage des *features* est effectué automatiquement par les réseaux neuronaux profonds. Aucune technique d'ingénierie des *features* n'est nécessaire. Les ensembles de données DARPA1998 et ISCX2012 sont utilisés pour évaluer les performances du système proposé. Ils obtiennent pour ISCX2012, un score de 99,86% et un taux de fausses alertes de 2%. Le score est assez bon cependant les performances du modèle pour la détection dans des ensembles de données déséquilibrés doivent être améliorées. Les données expérimentales montrent que les performances du HAST-IDS ne sont pas assez bonnes pour les classes de trafic ayant moins d'échantillons(classes minoritaires).

Peng Lin et al. [71] utilisent la mémoire à long et à court terme (LSTM) pour construire un modèle de réseau neuronal profond et ajoute un mécanisme d'attention (AM) pour améliorer les performances du modèle. Ils utilisent l'algorithme SMOTE et une fonction de perte améliorée pour traiter le problème du déséquilibre de classes dans l'ensemble de données CSE-CIC-IDS2018. Les résultats expérimentaux montrent que la précision de classification de leur modèle atteint 96,2 %, et ils affirment que ce chiffre est supérieur aux chiffres des autres algorithmes d'apprentissage machine. Néanmoins le score est en dessous de celui d'autres modèles utilisant pourtant le même ensemble de données comme

le montre les travaux ci-dessous.

Dans cet article [69] V. Kanimozhi et al. ont comparé la performance de plusieurs algorithmes sur l'ensemble de données CSE-CIC-IDS2018. ils ont obtenu les résultats suivants :

classifier models	accuracy	précision	recall	F1	AUC
artificial neural network	0.9997	0.9996	1.0	0.9998	1.0
random forest classifier	0.9983	0.9992	0.9988	0.9992	0.9
kn neighbor classifier	0.9973	0.998	0.9988	0.9984	0.998
svm classifier	0.998	0.9	0.9988	0.9994	0.999
ada boost classifier	0.9996	0.9996	0.9988	0.9992	0.9988
naïve bayes classifier	0.992	0.9929	0.9976	0.9953	0.981

Ils ont montré que les réseaux de neurones artificiels sont bien plus performants que les autres modèles.

V. Kanimozhi et al. dans leur article [70] proposent un système qui consiste à détecter une classification d'attaque de botnet qui constitue une menace sérieuse pour les secteurs financiers et les services bancaires. Le système proposé est créé en appliquant un modèle de réseaux de neurones artificiels à un ensemble réaliste de données de cyberdéfense (CSE-CIC-IDS2018), le tout dernier ensemble de données de détection d'intrusion créé en 2018 par l'Institut canadien de cybersécurité (CIC) sur AWS (Amazon Web Services). Le système proposé de réseaux de neurones artificiels offre une performance exceptionnelle : le score de précision est de 99,97% et le taux moyen de faux positifs n'est que de 0,001. Le système proposé pour la détection des attaques de botnets, est puissant, précis et exact. Le nouveau système proposé peut être mis en œuvre pour l'analyse du trafic réseau en temps réel. Cependant le modèle est appliqué seulement aux intrusions de type "botnet" et n'a donc pas la capacité de détecter les autres types d'attaques.

1.4 Discussion

Dans l'état de l'art, le cas des études comparatives entre les techniques de machine learning utilisées dans le domaine des IDS a des limites. Entre autres limites soulevées, les modèles sont utilisés avec des paramètres très variés et aussi les données utilisées sont variées. La variété des données n'est pas due à la disponibilité abondante des données mais à l'utilisation de différents morceaux(parties) des ensembles de données existants. De ce fait il est difficile d'affirmer qu'un tel modèle est le meilleur de tous. Néanmoins la remarque est faite de la dominance de plus en plus des modèles d'apprentissage profond(Deep Learning) dans les travaux récents sur les IDS. Nous avons relevé un certain nombre de points que nous en discutons dans les lignes suivantes.

1.4.1 Sur la complexité

Au regard des travaux récents, les IDS basés sur l'apprentissage profond ont gagné une énorme popularité grâce à leur capacité d'apprentissage de fonctionnalités approfondies

permettant de produire d'excellents résultats dans la détection des attaques. Les modèles basés sur les algorithmes d'apprentissage profond sont assez complexes et nécessitent des ressources importantes en termes de puissance de calcul, de capacité de stockage et de temps. Ces structures complexes posent des défis supplémentaires pour la mise en œuvre des IDS dans des environnements en temps réel. Pour pallier ce problème des plateformes ou des services GPU basés sur le cloud peuvent être explorés pour la formation des modèles(Dans notre cas nous avons utilisé *Google Colab*). Une autre solution à ce problème est d'essayer de réduire la complexité des algorithmes et prendre le risque de sélectionner nous-même les *features* importants de façon à avoir une précision de détection presque identique à celle obtenue avec l'ensemble des *features*. Dans tous les cas, il faut un bon compromis entre la complexité et la performance du modèle pour le cas d'un environnement "temps réel".
En règle générale, les algorithmes $O(n)$ et $O(n \times logn)$ sont considérés comme linéaires et sont utilisables pour des approches dans les environnements dits *temps réel* . $O(n^2)$ est considéré comme une complexité temporelle acceptable pour la plupart des pratiques. $O(n^3)$ et plus sont considérés comme des algorithmes beaucoup plus lents et sont utilisés pour des approches *hors ligne*.

1.4.2 Sur la performance

Une bonne capacité de généralisation est nécessaire pour que le modèle formé ne s'écarte pas radicalement du modèle de départ lorsque de nouvelles données sont vues. La plupart des méthodes de machine learning de pointe ont une très bonne capacité de généralisation. Cependant, les réseaux informatiques deviennent de plus en plus complexes au fur et à mesure que de nouvelles technologies de l'information et de la communication (NTIC) de pointe sont intégrées au réseau, offrant ainsi aux attaquants de plus en plus de possibilités. L'un des facteurs les plus importants liés à la performance des IDS est le type et la qualité de l'ensemble de données utilisé pour l'entraînement. L'autre facteur lié aux performances des IDS est le type d'algorithme de machine learning utilisé et la conception globale du système. Les performances des IDS peuvent être améliorées en utilisant des ensembles de données actualisés et équilibrés.

1.4.3 Sur l'efficacité

Les travaux récents montrent des limites au niveau de la détection des attaques de type "zero-day" et au niveau du taux de fausses alertes. Les IDS devraient inclure un mécanisme permettant de continuer à entraîner régulièrement le modèle avec de nouvelles données pour que le modèle apprenne de nouveaux types d'attaques ou pour qu'il puisse s'ajuster. Cela permettra d'améliorer le modèle dans la détection des attaques "zero-day" et de réduire le taux des fausses alertes.

Dans ce chapitre, après avoir regardé très attentivement les concepts liés aux IDS, les modèles de machine learing puis les ensembles de données pour les IDS, nous avons fait le tour sur les travaux récents dans le domaine. Par la suite, nous avons discuté sur les notions de complexité, de performance et d'efficacité des techniques de machine learning profond et peu profond. Dans le chapitre suivant nous présentons notre approche et notre démarche.

Chapitre 2

Proposition de démarche et architecture

Sommaire

Ce chapitre explique l'idée conceptrice de notre système. Cette description concerne l'hypothèse, l'architecture de notre système et les données utilisées. L'hypothèse repose sur le fait d'avoir un arbre de décision avec des nœuds capables d'effectuer une classification en utilisant un réseau de neurones à convolution (CNN).

2.1 Hypothèse

Notre hypothèse est d'avoir d'une part une capacité d'apprentissage incrémental qui permettra d'apprendre de nouvelles intrusions. Et d'autre part de pouvoir détecter puis de classer(afin de donner plus de précision sur l'intrusion) les intrusions à travers un arbre de décision, ce qui répond à la logique de détection et de classification des intrusions. Cette hypothèse est inspirée de la démarche de Deboleena ROY et al. dans leur article *Tree-CNN : A hierarchical Deep Convolutional Neural Network for incremental learning*[75].

Au cours de la dernière décennie, les réseaux neurones à convolution profonds (Deep CNN) ont montré des performances remarquables dans la plupart des tâches de vision par ordinateur. Ces performances sont exploitées vers la tâche de détection d'intrusions dans la cybersécurité [50] [76] [58] . Mais ces travaux utilisent traditionnellement un ensemble de données fixe, et le modèle, une fois formé, est déployé tel quel. L'ajout de nouvelles informations à un tel modèle représente un défi en raison des problèmes d'entraînement complexe, tels que l'*oubli catastrophique* et la sensibilité au réglage des hyper-paramètres. Cependant, dans ce monde moderne, les données sont en constante évolution et nos modèles d'apprentissage profond doivent évoluer aussi pour s'adapter. Alors Deboleena ROY et al. ont proposé une structure de réseau hiérarchique adaptative composée de Deep CNN qui peut croître et apprendre au fur et à mesure que de nouvelles données sont disponibles. Le réseau se développe de manière arborescente pour accueillir de nouvelles classes de données, tout en préservant la capacité à distinguer les classes précédemment formées. Le réseau organise les données progressivement en super-classes axées sur les *features* et améliore les modèles hiérarchiques existants des Deep CNN en y ajoutant la capacité d'auto-développement.

2.1.1 Apprentissage incrémentale

Pour que les IDS soient efficaces dans la détection des attaques « zero-day », ils doivent être entraînés régulièrement avec les nouvelles données obtenues suite à la surveillance du trafic du réseau. Plus le modèle de l'IDS sera formé, plus il détectera efficacement les intrusions.
Traditionnellement, l'apprentissage pour les modèles d'intelligence artificielle se passe en une seule fois puis ces modèles sont utilisés, quelle que soit la durée, sans apprendre des nouvelles informations qui arrivent au jour le jour. Cette manière d'apprendre est appelé apprentissage "One-Shoot" qui veut dire, un apprentissage en un seul coup.
Or, dans la vie réelle d'un humain, on apprend chaque jour à chaque événement qui arrive,

à chaque information. La vie d'un humain est faite de continuelle et de quotidienne apprentissage. Il apprend à travers les flux d'informations qu'il capte par ses cinq (05) sens (l'ouïe, l'odorat, le touché, le goût, la vue). Ainsi l'être humain développe ses connaissances, son intelligence. Cela constitue sa plus grande force de sa capacité à s'adapter aux situations nouvelles et aux changements. Cette intelligence lui permet aussi d'avoir une bonne capacité de généralisation qui est très recherchée dans le domaine du machine learning.
L'apprentissage incrémental ou l'apprentissage continu tente d'imiter cette manière d'apprendre de l'être humain. C'est-à-dire pouvoir faire apprendre à l'ordinateur de façon continue ou incrémentale à travers de nouvelles informations. L'intelligence artificielle qui est définie comme une imitation (tentative d'imitation) du comportement de l'homme, tente cette approche d'appendre continuellement en introduisant différentes techniques dont l'apprentissage continue et l'apprentissage incrémental [12] [10]. Ceci est devenu un domaine de recherche très en vogue [13] [53] [48] [12]. Mais l'apprentissage incrémental, dans l'état de l'art est confronté à un problème. La modification d'une partie de l'espace des paramètres affecte immédiatement le modèle dans son ensemble[33]. Un autre problème lié à l'entraînement progressif d'un modèle de Deep CNN est la question de l'*oubli catastrophique*[28]. Lorsqu'un Deep CNN formé est recyclé exclusivement sur de nouvelles données, il en résulte la destruction des *features* existants appris à partir de données antérieures. Cela oblige à utiliser des données antérieures lors de la formation avec les nouvelles données. Notre démarche pour notre IDS est dans la logique de pouvoir répondre aux problèmes posés afin de pouvoir exploiter efficacement les avantages de l'apprentissage incrémental. En effet en nous basant sur un algorithme proposé par Deboleena ROY et al. nous pouvons exploiter les avantages l'apprentissage incrémental. Dans leur travail, pour éviter le problème de l'*oubli catastrophique*, et pour garder les fonctionnalités apprises dans la tâche précédente, Deboleena Roy et al. proposent un réseau composé de CNN qui se développe hiérarchiquement au fur et à mesure que de nouvelles classes sont introduites. Le réseau ajoute les nouvelles classes en tant que nouvelles feuilles à la structure hiérarchique. La ramification est basée sur la similarité des *features* entre les nouvelles et les anciennes classes. Un tel modèle nous permet d'exploiter les couches de convolution apprises précédemment pour les utiliser dans le nouveau réseau plus grand.

2.1.2 CNN

Les réseaux de neurones à convolution (CNN)[45] sont très utilisés dans le domaine de vision par ordinateur. Ils offrent de très bons résultats dans ce domaine. Leur performance est démontrée aussi dans plusieurs autres domaines comme la détection des intrusions pour les systèmes informatiques. De nombreux articles dont [57] [51] [71] [69] [50] [76] [58] ont utilisé le deep learing et ont obtenu des résultats très bons. Notamment [57] [50] [76] [58] ont utilisé les CNN et ont prouvé clairement son efficacité dans le domaine des IDS. Dans ces articles il ressort que les CNN et ses variantes d'architecture ont obtenu des résultats significatifs par rapport aux classificateurs classiques de machine learning. Ceci est principalement dû au fait que les CNN sont capables d'extraire des *features* de haut niveau

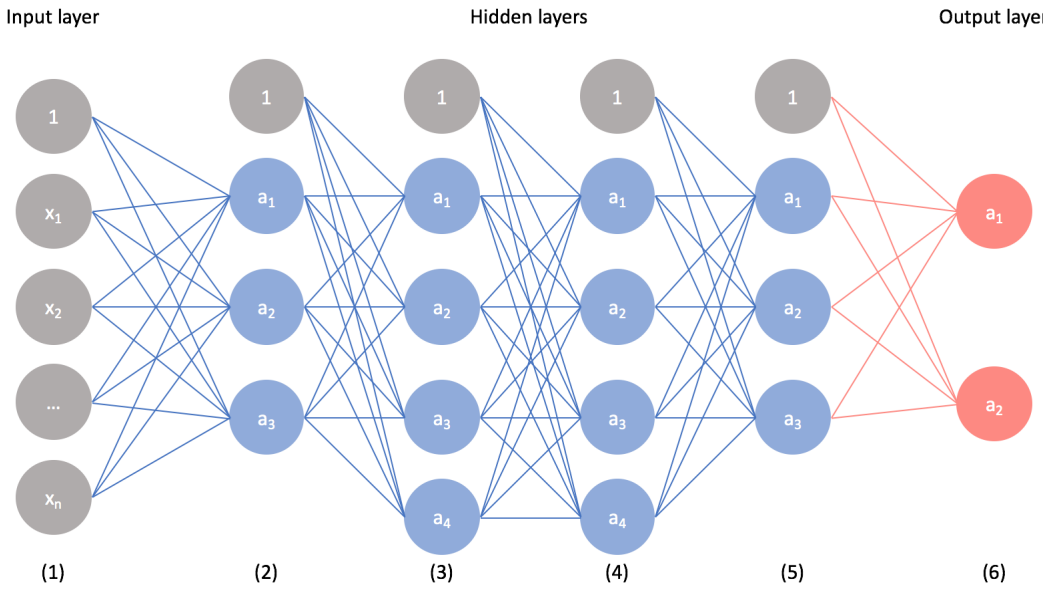

FIGURE 2.1 – Structure d'un CNN

qui représentent au mieux la forme abstraite des *features* de bas niveau des connexions de trafic réseau. Nous avons choisi d'utiliser les CNN au vu des nombreux avantages qu'ils offrent particulièrement leur capacité à sélectionner eux même les *features* les plus significatifs. En effet les CNN réalisent eux-mêmes le pénible processus d'extraction des *features*. En plus de cela, l'architecture du réseau de neurone à convolution lui donne la capacité de hiérarchiser les *features* sélectionnés du plus simple au plus sophistiqués.

CNN est le premier algorithme d'apprentissage véritablement réussi pour la formation de structures de réseaux multi-couches, c'est-à-dire la structure illustrée à la figure fig.2.1. Il réduit le nombre de paramètres qui doivent être appris pour améliorer les performances d'apprentissage de l'algorithme de Rétro-propagation du gradient (Backpropagation : BP) par le biais de relations spatiales.

CNN a besoin de peu d'effort de pré-traitement et est indépendant de la conception des *features* et des connaissances préalables(des experts du domaine). Pour tout ceci il est très promoteur pour la détection des intrusions.

2.1.3 Architecture

Inspiré des classificateurs hiérarchiques, le modèle que nous proposons, Tree-CNN est composé de plusieurs nœuds liés entre eux de manière à avoir une structure arborescente. Le premier nœud de l'arborescence est le nœud « racine » où se déroule la première classification. Tous les autres nœuds de l'arborescence, à l'exception des nœuds « feuilles », ont un Deep Convolutinal Neuronal Network(Deep CNN) qui est formé pour classer l'entrée pour leurs nœuds « enfants ».

Le processus de classification commence au niveau du nœud « racine » et la donnée est ensuite transmise au nœud « enfant » suivant, en fonction du résultat de la classification du nœud « racine ». Ce nœud procède ensuite à une classification de la donnée à son

niveau puis le transmet à son tour, en fonction du résultat de la classification, à l'un de ses nœuds enfants. Ce processus est répété jusqu'à ce qu'on atteigne un nœud « feuille ». C'est la fin des étapes de classifications.

Les nœuds ont chacun un nœud parent et au moins un nœud enfant sauf le nœud « racine » et les nœuds « feuilles ». Chacun des nœuds « feuilles » est lié exclusivement à une classe et une classe, exclusivement à un et un seul nœud. De telle sorte que deux nœuds « feuilles » ne soient associés à une même classe.

La figure Figure 3.3 montre une architecture à deux niveaux de notre modèle. La méthode d'entraînement d'un réseau de neurones artificiels avec une telle architecture est décrite par l'algorithme Figure 3.1. L'une des avantages de la structure en arbre est qu'elle permet une réduction considérable du temps de décision(prédiction).

2.2 Traitement des données

Les problèmes soulevés dans 1.2.2.1 concernant les données sont traités et les solutions apportées au cas par cas de la façon suivante :

- **entête répété** : les entêtes répétés sont recherchés et supprimés ;
- **valeur infinie** : les valeurs infinies qui sont dans un format(*"infinity"*) non reconnu par *Pandas*(la bibliothèque que nous avons utilisée pour le traitement des données), sont d'abord transformées en format reconnu par Pandas. Par la suite, les valeurs infinies sont recherchées et remplacées par le maximum de la colonne concernée ;
- **valeurs vides ou nulles** : les valeurs vides ou les valeurs nulles sont traitées de la même façon. Elles sont recherchées et remplacées par la moyenne de la colonne concernée ;
- **ensemble de données déséquilibré** : ce problème est atténué en réduisant le nombre de lignes avec le label "benign" puis en utilisant la technique *"SMOTE"*[16] (Synthetic Minority Oversampling Technique : SMOTE) pour essayer d'équilibrer les données. Comme son nom l'indique, SMOTE est une méthode de sur-échantillonnage. Elle fonctionne en créant des échantillons synthétiques à partir de la classe ou des classes minoritaires au lieu de créer de simples copies.

D'autres traitements des données ont été effectués pour faciliter et améliorer l'apprentissage. Ce sont :

- **traitement des *features* catégoriels** : les *features* comme « protocol » sont des données catégorielles et doivent être transformées pour ne pas être considérées par le modèle comme des simples nombres. Nous avons utilisé le principe de « one hot encoding ».
- **traitement des *features* numériques** : les *features* contenant des valeurs numériques sont transformé avec la formule 2.1. Cette transformation permet de faciliter les calculs et améliorer les performances du modèle. L'approche de normalisation Min-Max est utilisée pour normaliser les valeurs numériques de sorte que les nou-

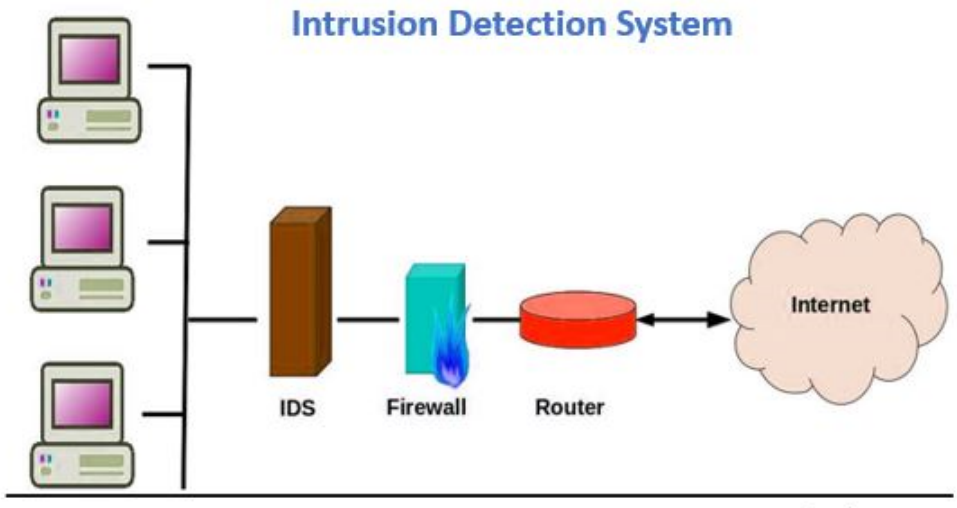

FIGURE 2.2 – Position du NIDS[88]

velles valeurs soient dans la plage [0, 1]. La formule mathématique de la normalisation Min-Max est l'équation Eq.2.1

$$x_{nouveau} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.1}$$

Où :
x est une valeur contenu dans la colonne du *feature* ;
$x_{nouveau}$ est la nouvelle valeur ;
x_{min} est la valeur minimale de la colonne du *feature* ;
x_{max} est la valeur maximale de la colonne du *feature*.

2.3 L'IDS

L'IDS que nous avons implémenté est de type Network IDS (NIDS), c'est-à-dire destiné à être sur le réseau et sa position est comme celle sur la figure Figure 2.2. Il est basé sur le comportement. L'état de l'art sur les NIDS est très riche en contribution mais ici nous nous concentrons sur le principe de détecter puis de classifier puis celui de l'apprentissage incrémental.

Le principe consiste à détecter d'abord s'il s'agit d'une attaque ou pas. Ensuite d'alerter l'administrateur et enfin de classifier l'intrusion afin de donner plus de détails à l'administrateur afin qu'il puisse intervenir adéquatement.
Cela se réalise par une structure sous la forme d'arbre comme le montre la figure Figure 3.3. Il y a trois niveaux :

- le premier niveau est le nœud racine ;
- le deuxième niveau est constitué de deux nœuds : un nœud « feuille » lié à la classe *Benign* et un nœud lié à la classe *Attack* ;

- le troisième niveau contient des nœuds « feuilles », tous issus du nœud « Attack ».

Tous les nœuds, sauf les nœuds feuilles, contiennent un Deep CNN qui est entraîné à classifier les classes de ses nœuds enfants, chacun à son niveau. Le premier nœud qui est le nœud racine sert à prendre la décision (à prédire) s'il s'agit d'une attaque ou d'un événement bénin. Et s'il s'agit d'une attaque, une alerte est émise. Par la suite, pour donner plus de détails sur l'attaque afin de permettre à l'administrateur d'intervenir efficacement, le nœud lié à la classe "Attaque" est amené à effectuer une classification. Le nœud lié à la classe *"Attack"* classifie les différents types d'attaques. Les quatorze (14) types d'attaques regroupés en sept (07), sont classifiés par ce nœud. À l'issue de la classification, une alerte est émise avec le nom du type d'attaque prédite.

Après cette exposition de notre hypothèse et l'explication de notre démarche, nous avons justifié nos choix notamment le modèle et l'ensemble de données. Dans le chapitre suivant nous passons l'implémentation proprement dit et puis les tests.

Chapitre 3

Implémentation, tests et résultats

Sommaire

Après les explications sur la théorie et les concepts clé qui sont nécessaires à la compréhension, nous passons à la phase pratique en mettant en œuvre la technique proposée et décrite ci-haut dans le chapitre 2. Nous décrivons dans ce chapitre l'implémentation de notre modèle, les outils utilisés puis les résultats obtenus sont comparés avec ceux d'autres modèles de l'état de l'art.

3.1 Implémentation

Dans cette partie, nous présentons les outils que nous avons utilisés, l'implémentation de l'algorithme utilisé et l'architecture de notre modèle.

3.1.1 Outils utilisés

Nous avons réalisé le travail avec un certain nombre d'outils principalement *open source*. Ces outils sont très utilisés, acceptés et adoptés dans le monde de la recherche scientifique. L'implémentation du modèle est faite en python avec l'éditeur Jupyter Notebooks qui est un environnement de développement python performant. Jupyter Notebook est une application flexible web et open-source : il permet de configurer et d'organiser l'interface utilisateur de manière à prendre en charge un large éventail de flux de travail dans le domaine des sciences des données, le traitement des données, la visualisation des données, les modèles statistiques, le machine learning, etc. Notre environnement de travail est Google Colab, un environnement cloud dans lequel nous avons accès à des ressources partagées. Ces ressources sont : des GPU et une mémoire RAM de 25Gb, utilisé à travers Jupyter Notebook.
Nous avons utilisé pour le traitement des données et l'implémentation du modèle, un ensemble de bibliothèques et outils comme Tensorflow, Pandas, Keras, Scikit-learn, etc. **Tensorflow** [1] est une plate-forme open-source de Google dédié au machine learning. Elle propose un écosystème complet et flexible d'outils, de bibliothèques et de ressources communautaires permettant aux chercheurs d'avancer dans le domaine du machine learning, et aux développeurs de créer et de déployer facilement des applications qui exploitent cette technologie. Son code source a été ouvert le 9 novembre 2015 mais la version 1.0.0 est sortie le 11 février 2017.

Pandas [2] est une bibliothèque que nous avons utilisée pour la manipulation et le traitement des données. C'est un outil d'analyse et de manipulation de donnée rapide, puissant, flexible et facile à utiliser. Elle est écrite avec le langage de programmation Python. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles. Pandas est open source.

Keras est une API(Application Programming Interface) cohérente et simple, elle minimise le nombre d'actions de l'utilisateur requises pour les cas d'utilisations courantes et

1. `https://www.tensorflow.org/`
2. `https://pandas.pydata.org/`

il fournit des messages d'erreur clairs et exploitables. C'est un API d'apprentissage pour les réseaux de neurones profonds et est écrit en Python, fonctionnant sur la plate-forme d'apprentissage machine TensorFlow. Elle a été développée dans le but de permettre une expérimentation rapide. Il se concentre sur son ergonomie, sa modularité et ses capacités d'extension. Elle est née dans le cadre du projet ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System) [3]. Elle a été initialement écrite par François Chollet dans son livre *Deep learning with Python*.

Scikit-learn [4] est une bibliothèque libre Python destinée à l'apprentissage automatique. Elle a des outils simples et efficaces pour l'analyse prédictive des données. Construit sur NumPy, SciPy, et matplotlib, elle est accessible et réutilisable dans divers contextes. Elle est *open source*, et est développée par de nombreux contributeurs notamment dans le monde académique.

NumPy [5] est une bibliothèque écrite en langage de programmation Python, destinée à manipuler des matrices ou des tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux. C'est un projet *open-source* visant à permettre le calcul numérique avec Python. Il a été créé en 2005, sur la base des premiers travaux des bibliothèques Numerical et Numarray.

3.1.2 Algorithme d'apprentissage

L'algorithme que nous utilisons est basé sur celui de Roy et al.[75]. C'est un modèle utilisant le CNN et ayant une structure en arbre. Les lignes suivantes décrivent le processus d'apprentissage (incrémental) du modèle et la procédure pour faire évoluer l'arbre Tree-CNN.

Nous initialisons notre modèle avec un nœud racine que nous entraînons à la classification de deux classes : les *attaques* et les *bénignes*.
Pour entraîner le modèle à reconnaître un certain nombre de nouvelles classes d'attaques, nous fournissons les données sur ces attaques au nœud racine.
Nous obtenons une matrice tridimensionnelle à la sortie de la couche de sortie :
$O^{K\times M\times I}$, où
K est le nombre(dans notre cas 02) de nœuds enfants du nœud racine,
M est le nombre(dans notre cas 07) de nouvelles classes d'attaques et
I est le nombre d'échantillons de données par classe.
$O\,(k, m, i)$ indique la sortie du $k^{ième}$ nœud pour la $i^{ième}$ donnée appartenant à la $m^{ième}$ classe, où
$k \in [1, K]$,
$m \in [1, M]$, et
$i \in [1, I]$.

3. https://keras.io/#why-this-name-keras
4. https://scikit-learn.org/
5. https://numpy.org/

$O_{avg}(k,m)$ $(Eq.\,(3.1))$ est la moyenne des sorties(du $k^{ième}$ nœud et la $m^{ième}$ classe) sur I données.
$O_{avg}^{K\times M}$ est la matrice de ces moyennes sur les I données.
La *Softmax* $(Eq.\,(3.2))$ est calculée sur chaque moyenne O_{avg} $(Eq.\,(3.1))$ pour obtenir une matrice $L^{K\times M}$.
Une liste ordonnée S est générée à partir de la matrice $L^{K\times M}$, ayant les propriétés suivantes :

- La liste S a M objets. Chaque objet est lié de façon unique à l'une des nouvelles classes M.
- Chaque objet $S[i]$ possède les attributs suivants :
  1. $S[i].label$ = label de la nouvelle classe.
  2. $S[i].value = [v_1, v_2, v_3]$, les 3 plus grandes valeurs *Softmax* des moyennes (O_{avg}) de cette nouvelle classe, classées par ordre décroissant $v_1 \geq v_2 \geq v_3$.
  3. $S[i].noeuds = [n_1, n_2, n_3]$, les nœuds correspondants respectivement aux valeurs *Softmax* v_1, v_2, v_3.
- S est ordonnée par valeur décroissante de $S[i].value[1]$

$$O_{avg}\,(k,m) = \sum_{i=1}^{I} \frac{O\,(k,m,i)}{I} \tag{3.1}$$

$$L(k,m) = \frac{e^{O_{avg}(k,m)}}{\sum_{k=1}^{K} e^{O_{avg}(k,m)}} \tag{3.2}$$

Cet ordonnancement est fait pour s'assurer que les nouvelles classes ayant des valeurs de *Softmax* élevées soient d'abord ajoutées à l'arbre Tree-CNN. La Softmax est utilisée à la place du nombre de données qui sont classées comme appartenant chacun des nœuds « enfants » car elle reflète mieux la réponse de la couche de sortie aux données à une échelle exponentielle et nous aide à mieux identifier à quel degré une donnée est similaire à l'une des étiquettes déjà existantes. Après avoir construit S, nous examinons son premier élément, S[1] , et nous prenons l'une des trois chemins :

i. ajoutez la nouvelle classe à un nœud enfant existant : Si v_1 est supérieur à la valeur suivante (v_2) d'un seuil, α (dans notre cas $\alpha = 0,1$), cette classe indique une forte ressemblance/association avec un nœud enfant particulier. Par exemple si les trois plus grandes valeurs *Softmax* sont $v_1 = 0,85$, $v_2 = 0,1$, et $v_3 = 0,05$. La nouvelle classe est ajoutée au nœud enfant n_1 correspondant ;

ii. fusionner deux nœuds enfants ajouter la nouvelle classe à ce nœud : S'il existe plus d'un nœud enfant pour lequel la nouvelle classe a une forte valeur *Softmax*, nous pouvons les combiner pour former un nouveau nœud enfant. Cela se produit lorsque $v_1 - v_2 < \alpha$, et $v_2 - v_3 > \beta$ (un autre seuil défini par l'utilisateur, ici nous l'avons défini $\alpha = 0,1$). Par exemple, si les trois plus grandes valeurs *Softmax* sont $v_1 = 0,48$, $v_2 = 0,45$, et $v_3 = 0,05$. Ensuite, si n_2 est un nœud de feuille, nous fusionnons n_2 dans n_1 et ajoutons la nouvelle classe à n_1 ;

```
1: I = Input Image, node = Root Node of the Tree
2: procedure ClassPredict(I, node)
3:     count = # of children of node
4:     if count = 0 then
5:         label = class label of the node
6:         return label
7:     else
8:         nextNode = EvaluateNode(I, node)
9:         ▶ returns the address of the child node of highest
   output neuron
10:        return ClassPredict(I, nextNode)
11:    end if
12: end procedure
```

FIGURE 3.1 – Algorithme d'inférence [75]

iii. ajouter la nouvelle classe comme un nouveau nœud enfant : Si la nouvelle classe n'a pas une valeur *Softmax* supérieure aux autres valeurs par une bonne marge ($v_1 - v_2 < \alpha, v_2 - v_3 < \beta$: par exemple si les trois plus grandes valeurs *Softmax* sont $v_1 = 0,35$, $v_2 = 0,33$, et $v_3 = 0,31$), ou si tous les nœuds enfants sont pleins(si on a atteint le nombre maximum de nœuds enfant), le réseau s'étend horizontalement en ajoutant la nouvelle classe comme nouveau nœud enfant. Ce nœud sera un nœud feuille.

Même si dans notre cas précis, le problème ne se pose pas, il est important de le souligner. Comme le nœud racine continue d'ajouter de nouvelles branches et sous-branches, les nœuds ayant plus d'enfants ont tendance à devenir plus lourds. Les nouvelles classes entrantes ont tendance à avoir une vraisemblance Softmax plus élevée pour ces nœuds ayant un plus grand nombre d'enfants.
Pour éviter que l'arbre Tree-CNN ne devienne déséquilibré, on peut fixer le nombre maximum d'enfants qu'un nœud de branche peut avoir.

La procédure décrite ci-dessus est répétée de manière itérative jusqu'à ce que toutes les nouvelles classes se voient attribuer un emplacement sous le nœud racine. Le pseudo-code est décrit dans l'algorithme Figure 3.2. Nous illustrons également un exemple d'apprentissage incrémental dans Tree-CNN avec la Figure Figure 3.3. Le réseau commence avec un seul nœud DCNN qui peut classer *2 classes*, *"benigne"* , *"attaque"*. Nous augmentons la capacité du réseau en ajoutant *7 nouvelles classes.C2 et C3, C4, C5, C6, C7, C8* sont alors ajoutés au nœud « attaque », convertissant ce nœud en branche comme le montre la figure Figure 3.3, conformément à la condition (i).

3.1.3 Modèle

Notre modèle comme le montre la figure Figure 3.4, a l'architecture suivante : il est composé de dix (10) couches de convolutions intercalées par deux (02) couches de Max-pooling, une (01) couche Flatten, deux (02) couches dense(entièrement connectée). Toutes les couches de convolution et *dense*, sauf la dernière, ont une activation *ReLU*. La dernière

```
L = Likelihood Matrix
maxChildren = max. number of children per branch node
RootNode = Root Node of the Tree-CNN
procedure GrowTree(L, Node)
    S = GenenerateS(L, Node, maxChildren)
    while S is not Empty do
        ▶ Get attributes of the first object
        [label, value, node] = GetAttributes(S[1])
        if value[1] − value[2] > α then
            ▶ The new class has a strong preference for n1
            ▶ Adds label to node[1]
            RootNode = AddClasstoNode(RootNode, label,
node[1])
        else
            if value[2] − value[3] > β then
                ▶ The new class has similar strong preference n1
and n2
                Merge = CheckforMerge(Node, node[1], node[2])
                ▶ Merge is True only if node[2] is a leaf node, and,
                ▶ the # of children of node[1] less than
maxChildren − 1
                if Merge then
                    ▶ Merge node[2] into node[1]
                    RootNode = MergeNode(RootNode, node[1],
node[2])
                    RootNode = AddClasstoNode(RootNode, label,
node[1])
                else
                    ▶ Add new class to the smaller output node
                    sNode = Node with lesser children
(node[1], node[2])
                    RootNode = AddClasstoNode(RootNode, label,
sNode)
                end if
            else
                ▶ Add new class as a new Leaf node to Root Node
                RootNode = AddNewNode(RootNode, label)
            end if
        end if
        ▶ Remove the columns of the added class from L
        ▶ Remove the rows of "full" nodes from L
        ▶ Regenerate S
        S = GenenerateS(L, Node, maxChildren)
    end while
end procedure
```

FIGURE 3.2 – Algorithme de Tree-CNN [75]

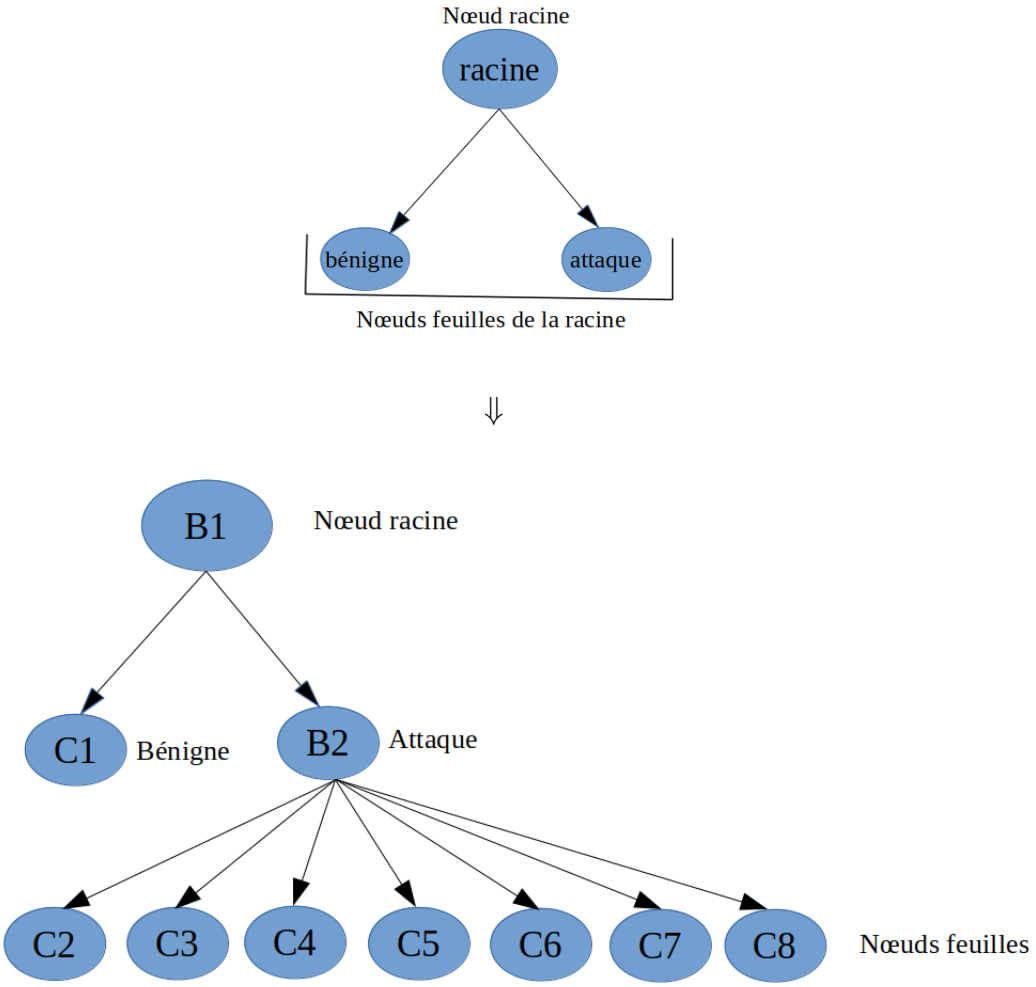

FIGURE 3.3 – Illustration des étapes de l'évolution de l'arbre

Layer (type)	Output Shape	Param #
conv1d_10 (Conv1D)	(None, 81, 16)	32
conv1d_11 (Conv1D)	(None, 81, 16)	272
conv1d_12 (Conv1D)	(None, 81, 32)	544
conv1d_13 (Conv1D)	(None, 81, 32)	2080
max_pooling1d_2 (MaxPooling1	(None, 40, 32)	0
conv1d_14 (Conv1D)	(None, 40, 64)	4160
conv1d_15 (Conv1D)	(None, 40, 64)	8256
max_pooling1d_3 (MaxPooling1	(None, 20, 64)	0
conv1d_16 (Conv1D)	(None, 20, 128)	24704
conv1d_17 (Conv1D)	(None, 20, 128)	49280
conv1d_18 (Conv1D)	(None, 20, 128)	49280
conv1d_19 (Conv1D)	(None, 20, 128)	65664
dense_3 (Dense)	(None, 20, 64)	8256
flatten_1 (Flatten)	(None, 1280)	0
dense_4 (Dense)	(None, 64)	81984
dropout_1 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 7)	455

Total params: 294,967
Trainable params: 294,967
Non-trainable params: 0

FIGURE 3.4 – Notre modèle CNN

couche a un activation *Sofmax*

L'entraînement a été effectué sur 2000 époques avec une condition d'arrêt pour ne pas avoir un sur-entraînement. La condition est qu'au bout de 100 époques si le score ne s'améliore pas, on arrête l'entraînement du modèle.

Nous initialisons le modèle avec deux classes. La classe attaque et la classe bénigne. Cela revient à pouvoir, à partir du nœud, décider s'il s'agit d'une attaque ou pas. À la suite, sept (07) autres classes qui sont des types d'attaque, sont ajoutées au modèle. Ces sept (07) classes sont ajoutées comme nœuds « feuilles » de la classe attaques. Ceci dans le but de pouvoir prédire le type de l'attaque détectée.

L'algorithme 3.1 montre comment se déroule l'inférence de notre modèle.

3.2 Métriques de performance

Les techniques de machine learning utilisées pour les IDS sont diverses et différentes. Choisir l'une parmi elles se fait sur la base de certaines caractéristiques et métriques de performances.

Les caractéristiques suivantes ne sont pas liées directement à la performance de la technique mais constituent, en générale, des critères importants pour le choix :

- **durée d'exécution** : le temps d'exécution comprend à la fois le temps d'entraînement et le temps de test. En raison de la grande complexité des modèles d'apprentissage profond, leur temps d'entraînement et de test sont beaucoup plus longs que ceux des modèles peu profonds ;

- **capacité d'apprentissage** : les structures des modèles d'apprentissage profond sont complexes et contiennent un très grand nombre de paramètres (généralement des millions ou plus). Par conséquent, les modèles d'apprentissage profond ont une plus grande capacité d'adaptation que les modèles peu profonds. Cependant, les modèles d'apprentissage profond présentent également un risque plus élevé de sur-apprentissage (*overfitting*) et nécessitent un volume de données beaucoup plus important pour la formation ;
- **interprétabilité** : les méthodes de machine learning en générale et les modèles d'apprentissage profond en particulier sont des *"boîtes noires"*. Les résultats sont presque ininterprétables, ce qui est un point critique. Cependant, certains algorithmes traditionnels d'apprentissage profond, tels que les arbres de décision, ont des résultats facilement interprétables ;
- **nombre de paramètres** : il existe deux types de paramètres : les paramètres d'apprentissage et les hyper-paramètres. Les paramètres d'apprentissage sont calculés pendant la phase d'entraînement, et les hyper-paramètres sont réglés manuellement avant le début de l'entraînement. Les paramètres et hyper-paramètres des modèles profonds sont beaucoup plus nombreux que ceux des modèles peu profonds, par conséquent, la formation et l'optimisation des modèles profonds prennent plus de temps ;
- **représentation des *features*** : l'entrée des modèles de machine learning traditionnels est un vecteur des *features*, et l'ingénierie des *features* est une étape essentielle. En revanche, les modèles d'apprentissage profond sont capables d'apprendre des représentations de *features* à partir de données brutes et ne dépendent pas de l'ingénierie des *features*. Ce qui leur confère un avantage exceptionnel par rapport aux méthodes de machine learning traditionnelles.

Pour ce qui est de l'évaluation(ou la comparaison) des différentes techniques de machine learning, de nombreux paramètres sont utilisés. Les modèles optimaux sont sélectionnés à l'aide de ces mesures. Pour mesurer de manière exhaustive la qualité de la détection, plusieurs mesures sont souvent utilisées simultanément dans l'état de l'art sur les IDS. L'évaluation est faite avec les quatre (04) éléments de base qui sont :

- **les vrais positifs (VP)** : les intrusions détectées ;
- **les faux négatifs (FN)** : les intrusions non-détectées ;
- **les faux positifs (FP)** : les événements bénignes détectés comme intrusions ;
- **les vrais négatifs (VN)** : les événements bénignes détectés effectivement comme bénignes.

On peut les représenter sous la forme d'un tableau croisé tab.3.1 pour mieux les appréhender.

Les métriques proprement dites sont :

- **le score (S)3.3** : il est défini comme le rapport entre les échantillons correctement prédits(VP+VN) et le nombre total d'échantillons(VP+FN+FP+VN). Le score est

	Prédit Positif	Prédit Négatif
Effectivement Positif	VP	FN
Effectivement Négatif	FP	VN

TABLE 3.1 – Éléments de base pour mesurer la performance d'un IDS

une mesure appropriée lorsque l'ensemble de données est équilibré. Dans les environnements de réseau réels, les échantillons normaux sont toutefois beaucoup plus abondants que les échantillons anormaux ; le score peut donc ne plus être une mesure appropriée. Il peut fausser l'évaluation lorsque le nombre de vrais négatifs est considérablement supérieur au nombre de vrais positifs. C'est pourquoi l'on utilise d'autres métriques en complément. Plus la valeur est grande plus le modèle est meilleur. Il est calculé selon la formule 3.3 suivante :

$$\textbf{Score} : S = \frac{VP + VN}{VP + FN + FP + VN} \tag{3.3}$$

- **la précision (P)3.4** : elle est définie comme le rapport des échantillons correctement prédits positifs(VP) sur les échantillons prédits positifs(VP+FP). Elle représente la confiance dans la détection des attaques. Plus la valeur est petite et plus il y a de faux positifs. Elle est calculée selon la formule 3.4 suivante :

$$\textbf{Précision} : P = \frac{VP}{VP + FP} \tag{3.4}$$

- **le rappel (R)3.5** : il est défini comme le rapport des échantillons correctement prédits positifs(VP) sur le total des échantillons réellement positifs(VP+FN) et est également appelé taux de détection. Le taux de détection reflète la capacité du modèle à reconnaître les attaques, ce qui est une mesure importante pour les IDS. Il est calculé selon la formule 3.5 suivante :

$$\textbf{Rappel} : R = \frac{VP}{VP + FN} \tag{3.5}$$

- **la F-mesure (F)3.6** : elle est définie comme la moyenne harmonique de la précision et du rappel. Plus la valeur est élevée, plus le modèle est performant. Elle est calculée selon la formule 3.6 suivante :

$$\textbf{F-mesure} : F = 2 \times \left[\frac{P \times R}{P + R}\right] = \frac{2 \times VP^2}{VP^2 + VP \times (FN + FP)} \tag{3.6}$$

- **le taux de faux négatifs (TFN)3.7** : il est défini comme le rapport entre les échantillons faux négatifs et le total des échantillons positifs. Dans la détection des attaques, le TFN est également appelé taux d'alarmes manquées, et il est calculé selon la formule 3.7 suivante :

$$TFN = \frac{FN}{VP + FN} \tag{3.7}$$

- **le taux de faux positifs (TFP)3.8** : il est défini comme le rapport entre les échantillons faux positifs et le total des échantillons négatifs. Dans la détection des attaques, le TFP est également appelé taux de fausses alarmes, et il est calculé selon la formule 3.8 suivante :

$$TFP = \frac{FP}{VN + FP} \tag{3.8}$$

3.3 Tests et résultats

Les résultats que nous obtenons sont mesurés suivant les métriques que nous avons décrit précédemment(le score, le taux de faux positif, le taux de faux négatifs, etc.). Cela est utile pour connaître les performances de notre modèle et aussi pouvoir le comparer à d'autres travaux.

3.3.1 Score et autres mesures

Nous avons mesuré les performances de notre modèle en utilisant les mesures suivantes : le *score*, la *précision*, la *F1-measure* et le *rappel*. Le *taux de faux positifs(TFP)* et le *taux de faux négatifs(TFN)* sont aussi mesurés pour voir à quel degré notre modèle se trompe.

Les tests pour la détection(classification binaire) donnent les résultats suivants :

- **le score** : nous obtenons un score de **99,94%** pour tous les types d'attaque ;
- **la précision** : nous obtenons une précision global de 0,99 ;
- **la F1-measure** : à ce niveau nous obtenons 0,99 comme score global ;
- **le rappel** : nous obtenons un rappel de 0,99.
- **le taux de faux positifs** : avec cette mesure, nous avons obtenu un score global de 0.0001.
- **le taux de faux négatifs** : avec cette mesure, nous avons obtenu un score global de 0.0001.

Pour la classification multi-classe, nous obtenons un score de *97,54%*. Le tableau suivant précise les détails sur les scores en pourcentage.

Classe	**Précision**	**F1-measure**	**Rappel**
Bénigne	100	99,64	99,28
DDOS	100	100	100
DoS	96,50	92,71	89,21
Bot	100	99,82	99,64
BruteForce	90,00	93,43	97,12
Infilteration	97,19	98,40	99,64
Attaque web et injection	99,63	98,73	97,84

Travaux	Technique	Ensemble de données	Score	Précision	F1-measure
T. Le et al.(2017)[46]	LSTM :binaire	KDD Cup99	97.54	0.97	-
Zeng et al.(2019)[72]	1D-CNN :binaire	ISCX2012	99.85	-	-
Zeng et al.(2019)[72]	LSTM :binaire	ISCX2012	99.41	-	-
Kanimozhi et al.(2019)[70]	ANN :binaire	CSE-CIC-IDS 2018	99,97	1,0	1,0
Notre méthode	**Tree-CNN :binaire**	**CSE-CIC-IDS 2018**	**99,94**	**0.99**	**0.99**

TABLE 3.2 – Tableau des performances de quelques modèles sur la détection

Travaux	Technique	Ensemble de données	Score	Précision	F1-measure
Q. Niyaz et al.(2015)[40]	RNN :5-classes	NSL-KDD	79.10	-	-
Sasanka Potluri et al.(2018)[60]	CNN :5-classes	NSL-KDD	91.14	-	-
Ding Yalei et al.(2018)[56]	CNN :5-classes	NSL-KDD	80,13	-	-
S. Potluri et al.(2017)[47]	DBN+SVM :5-classes	NSL-KDD	92,06	-	-
Notre méthode	**Tree-CNN :7-classes**	**CSE-CIC-IDS 2018**	**97,53**		

TABLE 3.3 – Tableau des performances de quelques modèles sur la classification

3.4 Comparaison avec l'état de l'art

Il est difficile de comparer des travaux n'utilisant pas les mêmes méthodes ni les mêmes données. Alors nous avons choisi de faire la comparaison avec des travaux qui utilisent des techniques de machine learning voisines à la notre.

Nous avons à travers un tableau Table 3.2 résumé les résultats obtenus dans d'autres travaux notamment ceux utilisant l'apprentissage profond. Dans les travaux cités, certains font de la classification d'intrusions (multi-classes) Table 3.3 et d'autres font de la détection d'intrusions Table 3.2 (classification binaire).

Pour ce qui est de la détection nous avons obtenu un score de 99,94%, une précision de 0,99. Ces mesures sont légèrement inférieures à celles de Kanimozhi et al.(2019)[70](les meilleurs) cité dans le tableau tab.3.2. Mais cela peut s'expliquer par le fait que notre modèle prend en compte plus de paramètres puisque en plus de la détection, il fait la classification, contrairement à celui de Kanimozhi et al. En plus le taux de fausses alertes de l'ordre de 0.0001 de notre modèle nous réconforte et nous rend confiant sur sa performance dans les cas réels. C'est une mesure si chère aux IDS.

Aussi à travers le tableau Table 3.3, nous voyons très clairement que notre modèle

est meilleur pour la classification multi-classe. En effet, le score dépasse largement ceux des travaux de classification multi-classe cités dans le tableau Table 3.3. Néanmoins, le score n'est pas une mesure très fiable, mais les détails sur les autres métriques comme la précision, dans les autres travaux n'ont pas été mentionnées. La précision de notre modèle sur chaque classe montre qu'il se trompe peu sur la classification.

De tout ce qui précède, nous pouvons dire que notre méthode est un bon moyen de détection et de classification d'intrusions.

Dans ce chapitre nous avons expliqué l'implémentation de notre modèle d'IDS qui est une technique de *deep learning* appelé Tree-CNN. Nous avons aussi expliqué notre algorithme d'apprentissage incrémental. Dans la partie « tests et comparaison » nous avons montré que notre modèle est meilleur sur le plan de la classification multi-classe par rapport à d'autres travaux utilisant des techniques similaires ou proches.

Conclusion Générale et Perspectives

Dans ce mémoire, il était question de la détection et la classification des intrusions en utilisant des techniques de machine learning. Mais bien avant, nous avons abordé quelques concepts du domaine comme la sécurité informatique, les attaques informatiques et les systèmes de détection d'intrusion.

Par la suite, nous avons fait le tour sur les techniques de machine learning, les ensembles de données utilisés et sur les travaux récents dans le domaine des IDS. Des travaux récents, il ressort que le *deep learning* est de plus en plus utilisé dû aux nombreuses qualités qu'il possède. Notamment, CNN a des qualités énormes pour la détection des intrusions comme l'extraction des *features* importants. Pour ce qui concerne les ensembles de données, CSE-CIC-IDS2018 est visiblement l'un des meilleurs pour l'entraînement et le test des modèles car il contient des attaques d'actualité et reflète au mieux les menaces actuelles auxquelles font face les réseaux et les systèmes d'information de nos jours.

C'est ainsi que nous avons proposé un modèle d'IDS en utilisant l'ensemble de données CSE-CIC-IDS-2018 et une technique d'apprentissage profond qui est un réseau neuronal convolutif et ayant une architecture hiérarchique appelé Tree-CNN. La structure du modèle permet un apprentissage incrémental. Ce qui le rend donc capable, d'apprendre à classifier de nouvelles types d'attaques au fur et à mesure que de nouvelles données arrivent. Nous avons, par la suite, évalué les performances du modèle en le comparant avec d'autres résultats de l'état de l'art sur l'apprentissage profond. De cette évaluation, nous avons trouvé notre modèle plus performant sur la classification multi-classe par rapport aux autres. Tree-CNN permet donc la classification et la détection des intrusions avec de bonnes performances. Néanmoins, des améliorations sont encore possibles.

Les recherches que nous avons menées dans le cadre de ce mémoire peuvent être approfondies afin d'améliorer notre approche et nos méthodes utilisées. Les perspectives dans ce sens sont :

- **améliorer le modèle** : il est encore possible d'améliorer la performance, pour la classification multi-classe du modèle en réglant encore plus les hyper-paramètres. En plus, on peut entraîner le modèle avec d'autres ensembles des données comme CSE-CIC-IDS2017 afin de réduire ses taux de fausses alertes et le rendre encore plus robuste et fiable ;
- **de IDS à IPS** : une autre piste d'amélioration de la solution proposée serait de combiner notre modèle avec des outils de détection d'intrusions basés sur des règles

tel que Snort, qui pourraient accélérer la détection des cas triviaux ou récurrents. En plus, il serait intéressant de continuer à développer notre solution pour qu'elle puisse enclencher des actions en fonction des intrusions détectées, c'est-à-dire qu'elle puisse agir contre une attaque en attendant l'intervention de l'administrateur. La solution finale ne serait plus un système de détection d'intrusion(IDS), mais un système de prévention d'intrusion(IPS).

Bibliographie

[1] A. Abou el kalam. "Modèles et politiques de sécurité pour les domaines de la santé et des affaires sociales". In : *Thèse de Doctorat, Laboratoire d'Analyse et d'Architecture des Systèmes du Centre National de la Recherche Scientifique* (Décembre 2003).

[2] James P Anderson. "Computer security threat monitoring and surveillance". In : *Technical Report, James P. Anderson Company* (1980).

[3] D.E. Denning. "An intrusion-detection model". In : *IEEE Trans. Softw. Eng* (1987).

[4] K. Hornik, M. Stinchcombe et H. White. "Multilayer feedforward networks are universal approximators". In : *Neural Netw.* (1989).

[5] S Rasoul Safavian et David Landgrebe. "A survey of decision tree classifier methodology". In : *IEEE transactions on systems, man, and cybernetics* 21.3 (1991), p. 660-674.

[6] Bishop. "Neural networks for pattern recognition". In : *C. M.* (1995).

[7] J. Ross Quinlan. "Learning decision tree classifiers". In : *ACM Computing Surveys (CSUR)* 28.1 (1996), p. 71-72.

[9] Mike Schuster et Kuldip K Paliwal. "Bidirectional recurrent neural networks". In : *IEEE transactions on Signal Processing* 45.11 (1997), p. 2673-2681.

[10] Mark B Ring. "CHILD : A first step towards continual learning". In : *Learning to learn*. Springer, 1998, p. 261-292.

[11] H. Debar, M. Dacier et A. Wespi. "Towards a taxonomy of intrusion-detection systems". In : *Computer Networks* (1999).

[12] Christophe Giraud-Carrier. "A note on the utility of incremental learning". In : *Ai Communications* 13.4 (2000), p. 215-223.

[13] Robi Polikar et al. "Learn++ : An incremental learning algorithm for supervised neural networks". In : *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)* 31.4 (2001), p. 497-508.

[14] Pascal Soucy et Guy W Mineau. "A simple KNN algorithm for text categorization". In : *Proceedings 2001 IEEE International Conference on Data Mining*. IEEE. 2001, p. 647-648.

[15] Alexander J. Smola Bernhard Schölkopf. “Learning With Kernels : Support Vector Machines, Regularization, Optimization and Beyond”. In : *MIT Press* (2002).

[16] Nitesh V Chawla et al. “SMOTE : synthetic minority over-sampling technique”. In : *Journal of artificial intelligence research* 16 (2002), p. 321-357.

[17] Tapas Kanungo et al. “An efficient k-means clustering algorithm : Analysis and implementation”. In : *IEEE transactions on pattern analysis and machine intelligence* 24.7 (2002), p. 881-892.

[18] David G Kleinbaum et al. *Logistic regression*. Springer, 2002.

[19] S. Mukkamala, A. Sunga et A. Abraham. “Intrusion detection using an ensemble of intelligent paradigms”. In : *J. Netw. Comput. Appl.* (2004).

[20] Alex Graves et Jürgen Schmidhuber. “Framewise phoneme classification with bidirectional LSTM and other neural network architectures”. In : *Neural networks* 18.5-6 (2005), p. 602-610.

[21] William S Noble. “What is a support vector machine ?” In : *Nature biotechnology* 24.12 (2006), p. 1565-1567.

[22] Jean-François Pillou. *Tout sur les systèmes d'information Grandes, moyennes et petites entreprises*. Collection Commentçamarche.net, 2006.

[23] Ruslan Salakhutdinov, Andriy Mnih et Geoffrey Hinton. “Restricted Boltzmann machines for collaborative filtering”. In : *Proceedings of the 24th international conference on Machine learning*. 2007, p. 791-798.

[24] Mahbod Tavallaee et al. “A detailed analysis of the KDD CUP 99 data set”. In : *2009 IEEE symposium on computational intelligence for security and defense applications*. IEEE. 2009, p. 1-6.

[25] Asmaa Shaker Ashoor et Sharad Gore. “Importance of Intrusion Detection System (IDS)”. In : *International Journal of Scientific and Engineering Research* (2011).

[26] Martin Sundermeyer, Ralf Schlüter et Hermann Ney. “LSTM neural networks for language modeling”. In : *Thirteenth annual conference of the international speech communication association*. 2012.

[27] Laurent Bloch et al. *Sécurité informatique : Principes et méthodes à l'usage des DSI, RSSI et administrateurs*. Editions Eyrolles, 2013.

[28] Ian J Goodfellow et al. “An empirical investigation of catastrophic forgetting in gradient-based neural networks”. In : *arXiv preprint arXiv :1312.6211* (2013).

[29] Xugang Lu et al. “Speech enhancement based on deep denoising autoencoder.” In : *Interspeech*. T. 2013. 2013, p. 436-440.

[30] Paul Wilson et A.T. Chandler. “Social Engineering The Art of Human Hacking”. In : *Brilliance Audio* (2014).

[32] Ian Goodfellow et al. “Generative adversarial nets”. In : *Advances in neural information processing systems* 27 (2014), p. 2672-2680.

[33] Tianjun Xiao et al. “Error-driven incremental learning in deep convolutional neural network for large-scale image classification”. In : *Proceedings of the 22nd ACM international conference on Multimedia*. 2014, p. 177-186.

[35] Kuang F. et al. “A novel SVM by combining kernel principal component analysis and improved chaotic particle swarm optimization for intrusion detection.” In : *Soft Comput.* (2015).

[36] Nour Moustafa et Jill Slay. “UNSW-NB15 : a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)”. In : *2015 military communications and information systems conference (MilCIS)*. IEEE. 2015, p. 1-6.

[37] Christopher Olah. “Understanding lstm networks”. In : (2015).

[38] Haibing Wu et Xiaodong Gu. “Max-pooling dropout for regularization of convolutional neural networks”. In : *International Conference on Neural Information Processing*. Springer. 2015, p. 46-54.

[39] Ayon Dey. “Machine Learning Algorithms : A Review”. In : 2016.

[40] Ahmad Javaid et al. “A deep learning approach for network intrusion detection system”. In : *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. 2016, p. 21-26.

[41] Ujjwal Karn. “An intuitive explanation of convolutional neural networks”. In : *The data science blog* (2016).

[42] Min XU Xingshan LI Na LI. “Intelligent intrusion detection technology NPRIM improved algorithm based on non-parameter clustering”. In : *Advances in Engineering Research 7th International Conference on Mechatronics, Control and Materials (ICMCM 2016)* (2016).

[43] Jaime Zabalza et al. “Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging”. In : *Neurocomputing* 185 (2016), p. 1-10.

[44] Mayank Agarwal et al. “Internal Detection System for PS-Poll DOS attack in 802.11 networks using real-time discrete event system”. In : *IEEE* (2017).

[45] Saad Albawi, Tareq Abed Mohammed et Saad Al-Zawi. “Understanding of a convolutional neural network”. In : *2017 International Conference on Engineering and Technology (ICET)*. IEEE. 2017, p. 1-6.

[46] T. Le, J. Kim et H. Kim. “An Effective Intrusion Detection Classifier Using Long Short-Term Memory with Gradient Descent Optimization”. In : *2017 International Conference on Platform Technology and Service (PlatCon)*. 2017, p. 1-6. doi : 10.1109/PlatCon.2017.7883684.

[47] S. Potluri, N. F. Henry et C. Diedrich. “Evaluation of hybrid deep learning techniques for ensuring security in networked control systems”. In : *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2017, p. 1-8. doi : 10.1109/ETFA.2017.8247662.

[48] Hanul Shin et al. “Continual learning with deep generative replay”. In : *Advances in neural information processing systems*. 2017, p. 2990-2999.

[49] Shaohua Teng et al. “SVM-DT-based adaptive and collaborative intrusion detection”. In : *IEEE/CAA Journal of Automatica Sinica* 5.1 (2017), p. 108-118.

[50] R. Vinayakumar, K. P. Soman et P. Poornachandran. “Applying convolutional neural network for network intrusion detection”. In : *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 2017, p. 1222-1228. doi : 10.1109/ICACCI.2017.8126009.

[51] Wei Wang et al. “HAST-IDS : Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection”. In : *IEEE Access* 6 (2017), p. 1792-1806.

[52] Jianxin Wu. “Introduction to convolutional neural networks”. In : *National Key Lab for Novel Software Technology. Nanjing University. China* 5 (2017), p. 23.

[53] Friedemann Zenke, Ben Poole et Surya Ganguli. “Continual learning through synaptic intelligence”. In : *Proceedings of machine learning research* 70 (2017), p. 3987.

[54] Zheng L. Zhao G. Zhang C. “Intrusion detection using deep belief network and probabilistic neural network”. In : *In Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC),Guangzhou, China* (2017).

[55] Abien Fred Agarap. “Deep learning using rectified linear units (relu)”. In : *arXiv preprint arXiv :1803.08375* (2018).

[56] Yalei Ding et Yuqing Zhai. “Intrusion detection system for NSL-KDD dataset using convolutional neural networks”. In : *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence*. 2018, p. 81-85.

[57] Chee Sun Liew Leila Mohammadpour Teck Chaw Ling et Chun Yong Chong. “A Convolutional Neural Network for Network Intrusion Detection System”. In : *Asia-Pacific Advanced Network (APAN)* (2018).

[58] Sheraz Naseer et Yasir Saleem. “Enhanced Network Intrusion Detection using Deep Convolutional Neural Networks.” In : *TIIS* 12.10 (2018), p. 5159-5178.

[59] Sasanka Potluri, Shamim Ahmed et Christian Diedrich. “Convolutional neural networks for multi-class intrusion detection system”. In : *International Conference on Mining Intelligence and Knowledge Exploration*. Springer. 2018, p. 225-238.

[60] Sasanka Potluri, Shamim Ahmed et Christian Diedrich. “Convolutional neural networks for multi-class intrusion detection system”. In : *International Conference on Mining Intelligence and Knowledge Exploration*. Springer. 2018, p. 225-238.

[61] Iman Sharafaldin, Arash Habibi Lashkari et Ali A Ghorbani. “Toward generating a new intrusion detection dataset and intrusion traffic characterization.” In : *ICISSP*. 2018, p. 108-116.

[62] Alex Shenfield, David Day et Aladdin Ayesh. “Intelligent intrusion detection systems using artificial neural networks”. In : *ICT Express* 4.2 (2018). SI on Artificial Intelligence and Machine Learning, p. 95-99. issn : 2405-9595. doi : https://doi.org/10.1016/j.icte.2018.04.003. url : http://www.sciencedirect.com/science/article/pii/S2405959518300493.

[63] N. Shone et al. “A Deep Learning Approach to Network Intrusion Detection”. In : *IEEE Transactions on Emerging Topics in Computational Intelligence* 2.1 (2018), p. 41-50.

[64] Eric Wilkinson. “Deep Learning : Sparse Autoencoders”. In : • (2018).

[65] Nan Zhang et al. “An overview on restricted Boltzmann machines”. In : *Neurocomputing* 275 (2018), p. 1186-1199.

[66] Guangyu Xu Zhida Li Ana Laura Gonzalez Rios et Ljiljana Trajković. “Machine Learning Techniques for Classifying Network Anomalies and Intrusions”. In : *ICT Express* (2018).

[67] Liang Zhou et al. *Cyber-Attack Classification in Smart Grid via Deep Neural Network*. CSAE ’18. Hohhot, China : Association for Computing Machinery, 2018. isbn : 9781450365123. doi : 10.1145/3207677.3278054. url : https://doi.org/10.1145/3207677.3278054.

[68] N. Chockwanich et V. Visoottiviseth. “Intrusion Detection by Deep Learning with TensorFlow”. In : *2019 21st International Conference on Advanced Communication Technology (ICACT)*. 2019, p. 654-659. doi : 10.23919/ICACT.2019.8701969.

[69] V. Kanimozhi et Dr. T. Prem Jacob. “CALIBRATION OF VARIOUS OPTIMIZED MACHINE LEARNING CLASSIFIERS IN NETWORK INTRUSION DETECTION SYSTEM ON THE REALISTIC CYBER DATASET CSE-CIC-IDS2018 USING CLOUD COMPUTING”. In : *International Journal of Engineering Applied Sciences and Technology* (2019).

[70] V. Kanimozhi et T. P. Jacob. “Artificial Intelligence based Network Intrusion Detection with Hyper-Parameter Optimization Tuning on the Realistic Cyber Dataset CSE-CIC-IDS2018 using Cloud Computing”. In : *2019 International Conference on Communication and Signal Processing (ICCSP)*. 2019, p. 0033-0036. doi : 10.1109/ICCSP.2019.8698029.

[71] Kejiang Ye Peng Lin et Cheng-Zhong Xu. “Dynamic Network Anomaly Detection System by Using Deep Learning Techniques”. In : *Springer Nature Switzerland AG 2019* (2019).

[72] Y. Zeng et al. "*Deep − Full − Range* : A Deep Learning Based Network Encrypted Traffic Classification and Intrusion Detection Framework". In : *IEEE Access* 7 (2019), p. 45182-45190. doi : 10.1109/ACCESS.2019.2908225.

[73] Di Zhao et al. "Bidirectional RNN-based Few-shot Training for Detecting Multi-stage Attack". In : *arXiv preprint arXiv :1905.03454* (2019).

[74] Azam Rashid, Muhammad Jawaid Siddique et Shahid Munir Ahmed. "Machine and Deep Learning Based Comparative Analysis Using Hybrid Approaches for Intrusion Detection System". In : *2020 3rd International Conference on Advancements in Computational Sciences (ICACS)*. IEEE. 2020, p. 1-9.

[75] Deboleena Roy, Priyadarshini Panda et Kaushik Roy. "Tree-CNN : A Hierarchical Deep Convolutional Neural Network for Incremental Learning". In : *Neural Networks* 121 (2020), p. 148-160.

[76] S. Sriram et al. "DCNN-IDS : Deep Convolutional Neural Network Based Intrusion Detection System". In : *Computational Intelligence, Cyber Security and Computational Models. Models and Techniques for Intelligent Systems and Automation*. Sous la dir. de Suresh Balusamy et al. Singapore : Springer Singapore, 2020, p. 85-92.

Webographie

[77] *https ://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/*. consulté le 20/10/2020. URL : https://adeshpande3.github.io/A-Beginner%5C%27s-Guide-To-Understanding-Convolutional-Neural-Networks/.

[78] *https ://purplesc.us/resources/cyber-security-statistics/*. consulté le 10/11/2020. URL : https://purplesc.us/resources/cyber-security-statistics/.

[79] *https ://www.securitymagazine.com/articles/87787-hackers-attack-every-39-seconds*. consulté le 23/12/2020. URL : https://www.securitymagazine.com/articles/87787-hackers-attack-every-39-seconds.

[80] *http ://www.bro-ids.org*. consulté le 06/07/2020. URL : http://www.bro-ids.org.

[81] *http ://www.ossec.net*. consulté le 06/07/2020. URL : http://www.ossec.net.

[82] *http ://www.prelude-ids.com*. consulté le 06/07/2020. URL : http://www.prelude-ids.com.

[83] *http ://www.snort.org*. consulté le 06/07/2020. URL : http://www.snort.org.

[84] *http ://www2.imm.dtu.dk/IDSnet*. consulté le 06/07/2020. URL : http://www2.imm.dtu.dk/IDSnet.

[85] *https ://blog.netwrix.fr/2018/07/04/les-10-types-de-cyberattaques-les-plus-courants*. consulté le 13/09/2020. URL : https://blog.netwrix.fr/2018/07/04/les-10-types-de-cyberattaques-les-plus-courants.

[86] *https ://fr.norton.com/internetsecurity-malware-what-is-a-trojan.html*. consulté le 06/07/2020. URL : https://fr.norton.com/internetsecurity-malware-what-is-a-trojan.html.

[87] *https ://fr.wizcase.com/blog/statistiques-incroyables-sur-internet-et-les-reseaux-sociaux/*. consulté le 23/08/2020. URL : https://fr.wizcase.com/blog/statistiques-incroyables-sur-internet-et-les-reseaux-sociaux/.

[88] *https ://www.comodo.com/ids-in-security.php*. consulté le 06/07/2020. URL : https://www.comodo.com/ids-in-security.php.

[89] *https ://www.editions-eni.fr/open/mediabook.aspxidR=644f97384201b4f93cd4a99753ebebc0.* consulté le 06/07/2020. URL : https://www.editions-eni.fr/open/mediabook.aspx?idR=644f97384201b4f93cd4a99753ebebc0.

[90] *https ://www.alioze.com/chiffres-web.* consulté le 23/09/2020. URL : https://www.alioze.com/chiffres-web.

[91] *http ://www.unb.ca/cic/datasets/ids-2018.html.* consulté le 09/08/2020. URL : http://www.unb.ca/cic/datasets/ids-2018.html.

[92] Marjolaine TASSET. *Le volume de données mondial sera multiplié par 45 entre 2020 et 2035.* 17/05/2019. URL : https://www.journaldunet.com/solutions/dsi/1424245-le-volume-de-donnees-mondial-sera-multiplie-par-45-entre-2020-et-2035-selon-statista/.

Chapitre 4

Annexe

Sommaire

4.1 Sécurité informatique et types d'attaques

L'objectif de notre travail est de contribuer à trouver une solution efficace pour détecter les intrusions. Mais bien avant il faut connaitre les concepts clés sur les attaques et surtout savoir ce qu'est la sécurité informatique.

4.1.1 La sécurité informatique

4.1.1.1 Définition

La sécurité informatique, de façon générale, consiste à s'assurer que les ressources informatiques (matérielles et/ou logicielles) sont utilisées par les personnes prévues dans le cadre prévu et uniquement comme prévu[22][27]. Elle vise à protéger l'intégrité, la confidentialité et la disponibilité des systèmes, des réseaux et des données informatiques

contre les attaques, les dommages ou les accès non autorisés. Le concept de sécurité des systèmes d'information intègre un ensemble de méthodes, techniques et outils dont les IDS.

La norme traitant des systèmes de management de la sécurité de l'information (SMSI) est l'ISO/CEI 27001, publiée en octobre 2005 et révisée en 2013. La norme précise que l'objectif de la sécurité informatique est de protéger les fonctions et les informations de toute perte, vol ou altération, et les systèmes informatiques de toute intrusion et sinistre informatique. Elle insiste sur ces points ; disponibilité, intégrité et confidentialité.

4.1.1.2 Les propriétés de la sécurité

La sécurité informatique repose sur un certain nombre de piliers et de principes. Les principaux sont les suivants :

- **confidentialité** : c'est un principe qui spécifie que seules les personnes autorisées ont accès ou peuvent atteindre la ressource de façon intelligible. Ou encore, elle consiste s'assurer que seuls les concernés comprendrons l'information, la ressource ou le service. Pour garantir la confidentialité, on a recours à des méthodes comme le contrôle d'accès (droits et permissions), le chiffrement, etc ;
- **intégrité** : elle consiste à garantir que la ressource n'aie pas été modifiée, changée ou altérée de manière volontaire ou involontaire. Généralement, les méthodes comme le calcul du checksum ou le hachage sont utilisées pour garantir l'intégrité ;
- **disponibilité** : elle consiste à garantir l'accès à un service ou à des ressources à chaque fois qu'il est demandé ;
- **authentification** : c'est un principe qui vérifie l'authenticité de l'identité prétendue et ainsi autorise l'accès ou pas à la ressource ou le service. En générale, il est demandé à l'entité (un humain ou un autre système) la preuve de son identité. Cette preuve peut être pour l'entité, ce qu'elle sait, ce qu'elle possède, ce qu'elle est, ou ce qu'elle sait faire. Très souvent, il s'agit d'une combinaison de plusieurs de ces dernières ;
- **non-répudiation** : elle consiste à s'assurer que l'auteur d'une action (active ou passive) ne puisse pas nier qu'il en est l'auteur. Imputabilité (responsabilité) et traçabilité sont des termes associés à ce principe de non-répudiation. L'une des solutions est la signature digitale qui est un mécanisme cryptographique qui permet d'assurer la non-répudiation de l'origine. Le mécanisme repose sur un système cryptographique asymétrique où la signature est calculée en utilisant la clé privée de l'émetteur et elle est vérifiée en utilisant la clé publique (de l'émetteur).

4.1.1.3 Menaces et vulnérabilités

Les vulnérabilités, ce sont les failles ou les faiblesses de sécurité dans un système permettant à un attaquant de porter atteinte à l'intégrité, à la confidentialité et/ou à la disponibilité de ce système. Elles peuvent être organisationnelles (ex : pas de politique de sécurité), humaines (ex : pas de formation du personnel), logicielles ou matérielles

(ex : utilisation de produits peu fiables ou non testés). Ne pouvant pas être sécurisé à 100%, tout système vu dans son ensemble présente des vulnérabilités, qui peuvent être exploitables ou non. Les menaces sont les « adversaires » identifiés capables d'attaquer un système d'information ou un réseau par exploitation d'une vulnérabilité.

4.1.2 Types d'attaques

Les moyens, méthodes et outils utilisés pour mener une attaque contre un système sont très nombreux et diversifiés, rendant ainsi difficile la classification. Ainsi, on peut regrouper les attaques selon l'objectif de sécurité menacé. On distingue :

- les attaques de modification qui menacent l'intégrité ;
- les attaques d'accès qui menacent la confidentialité ;
- les attaques de répudiation qui menacent la non-répudiation ;
- les attaques par déni de service et déni de service distribué (DoS et DDoS) qui menacent la disponibilité.

4.1.2.1 Attaques de modification

Ces sont les attaques qui tentent de modifier, de changer ou d'altérer la ressource cible. Elles constituent une menace à l'intégrité de la ressource. On retrouve au premier plan des attaques de modification, les codes malveillants ou malwares qui sont des programmes informatiques malicieux conçus et écrits pour qu'ils se reproduisent. Parmi eux on a les virus, les vers, les chevaux de Troie, les bombes logiques, etc qui diffèrent par leur manière de se propager. Leur effet peut aller de l'affichage d'un simple message à la suppression de tous les fichiers en passant par le cryptage des données.

Nous citons ci-dessous quelques malwares qui sont utilisés dans les attaques de modification :

- **le virus classique** : il s'attache à un fichier exécutable. Puis il se copie systématiquement sur tout autre exécutable qui va être lancer. Certains virus détruisent de façon, souvent irrécupérable les données ou les programmes. Par exemple la modification d'un secteur du disque dur avec des paramètres volontairement erronés obligeant à faire un « reformatage bas niveau », ce qui n'est pas toujours possible ;
- **le vers** : il est très semblable au virus classique et a à peu près les mêmes conséquences. Mais contrairement à lui, il n'a pas besoin de se parasiter à un programme pour fonctionner. Les vers se propagent généralement à travers les pièces jointes aux e-mails. A l'ouverture de la pièce jointe, le programme du ver s'exécute et effectue les actions programmées par l'attaquant ;
- **le cheval de Troie** : Très souvent malveillant, il se cache dans un programme utile comme les vers et les virus. La principale différence entre un virus et un cheval de Troie est que ce dernier ne se réplique pas de lui-même. En plus de lancer des attaques contre un système, un cheval de Troie peut établir une porte dérobée qui peut être exploitée par des attaquants pour voler ou altérer des données [85] ;

- **la bombe logique** : elle est une partie de programme qui reste dormante dans le système hôte jusqu'à ce qu'un instant un événement survienne, ou que certaines conditions soient réunies, pour déclencher des effets dévastateurs(destruction de façon, souvent irrécupérable des données et/ou programmes) [1] ;
- **le rançongiciels (ransomware)** : il s'agit d'un type de logiciel malveillant qui bloque l'accès aux données de la victime et menace de les publier ou de les supprimer à moins qu'une rançon ne soit versée. Un rançongiciel avancé utilise une technique appelée extorsion crypto-virale, qui chiffre les fichiers de la victime de manière à les rendre presque impossible à récupérer sans la clé de déchiffrement[85].

4.1.2.2 Attaques d'accès

Cette catégorie d'attaques vise à accéder à des informations sans les autorisations nécessaires. Elle constitue une menace pour la confidentialité.

Voici quelques techniques et malwares utilisés pour effectuer une attaque d'accès :

- **le sniffing** : généralement c'est un logiciel qui renifle les paquets réseaux, appelé Sniffer et que les attaquants utilisent pour dérober tous les paquets qui passent sur le réseau. Cela peut permettre aux attaquants d'obtenir des mots de passent si ils sont transmis en clair. De nombreux protocoles réseau ne chiffrent pas les données, il est donc possible de voir en clair les mots de passe Telnet, POP, FTP... [89]. Il est aussi possible de connaître les pages que visitent les utilisateurs connectés, les courriels en réception ou en envoi, etc ;
- **le cheval de Troie** : il sert généralement à prendre le contrôle de l'ordinateur d'un utilisateur, afin de voler des données et/ou d'injecter davantage de malwares sur l'ordinateur de la victime [86] ;
- **l'ingénierie sociale** : en réalité ce n'est pas vraiment une attaque mais plutôt des procédés et techniques utilisées pour recueillir des informations sur un système ou avoir les identifiants d'une personne. Elle consiste à manipuler psychologiquement une personne pour soutirer frauduleusement des informations. Les pratiques du piratage psychologique exploitent les faiblesses psychologiques, sociales et plus largement, organisationnelles des individus ou des organisations pour soutirer un bien, un service, un accès, une information confidentielle, etc. En utilisant ses connaissances, son charisme, son sens de l'imposture ou son culot, l'attaquant cherche à abuser de la confiance, de l'ignorance, de la crédulité de sa cible pour obtenir ce qu'il souhaite [30] ;
- **le craquage de mots de passe** : Il consiste à essayer à maintes reprises des mots de passe jusqu'à trouver le bon. Voici quelques méthodes pour le faire :

    - **attaque par dictionnaire** : il existe des dictionnaires contenant des mots de passe par défaut de toutes les applications et des mots de passe les plus souvent utilisés, des prénoms, des noms de famille dans divers pays du monde, de personnalités célèbres, de tous les mots d'une langue, etc. Le mot de passe

testé est pris dans cette liste prédéfinie et aussi ses variantes (à l'envers, avec un chiffre à la fin, etc.). Si le mot de passe est crypté de manière irréversible (type MD5 ou SHA1), l'attaque va consister à tenter tous les hashcodes de tous les mots du dictionnaire en utilisant tous les algorithmes de cryptage connus ;

- **attaque par force brute** :l'attaquant tente toutes les combinaisons de chiffres, de lettres et de caractères spéciaux. Généralement c'est un générateur automatique qui génère les mots de passe et ils sont tentés tous un par un.

- **le SQLinjection** : les injections SQL ont comme porte d'entrée, les paramètres d'entrée non vérifiés. Le but est d'injecter du code SQL dans une requête de base de données à travers les formulaires d'une page web, les URL, ou tout type d'entrées. Ainsi, il est possible de récupérer des informations se trouvant dans la base données (exemple : les utilisateurs et leur identifiant) ou encore de détruire les données ;
- **le XSS (Cross-site scripting)** : le principe est d'arriver à injecter un script (JavaScript le plus souvent) dans un site web à travers une des entrées (formulaire, URL, etc). Grâce aux vulnérabilités Cross-Site Scripting, l'attaquant peut récupérer les données échangées entre l'utilisateur et le serveur du site web concerné. Le code injecté dans la page web peut permettre d'afficher un formulaire afin de tromper l'utilisateur et lui faire saisir par exemple ses identifiants.

4.1.2.3 Attaques de répudiation

Ce sont les attaques qui consistent en générale à usurper l'identité d'une personne. C'est-à-dire, à se faire passer pour quelqu'un qu'on est pas. Cela menace le principe de la non-répudiation car le propriétaire réel de l'identité n'est pas le responsable effectif des actions éventuelles de l'attaquant. Plusieurs techniques sont utilisées pour effectuer ce type d'attaque. En voici certaines :

- **le IP spoofing** : cette technique consiste à usurper d'identité en falsifiant l'adresse IP(Internet Protocol). Elle consiste à utiliser une fausse adresse IP pour émettre les paquets IP. L'objectif peut être de camoufler sa propre identité lors d'une attaque, ou d'usurper l'identité d'un équipement pour effectuer des actions auxquelles ce dernier a accès. Il existe plusieurs variantes comme spoofer des adresses e-mail, des serveurs DNS ou NFS, etc ;
- **ARP Spoofing (ou ARP Redirect)** : pour l'attaquant, cette technique consiste à s'attribuer l'adresse IP de l'ordinateur cible en faisant correspondre son adresse IP avec l'adresse MAC de l'ordinateur cible dans les tables ARP des équipements du réseau. En fait, il faut simplement envoyer des paquets ARP_reply diffusés régulièrement contenant l'adresse IP de la cible et l'adresse MAC de l'attaquant. Cela aura pour effet, la modification des tables dynamiques de tous les équipements (y compris les routeurs) sur le réseau. Ce qui entraîne l'envoi des trames Ethernet à l'attaquant en pensant communiquer avec la cible. L'ordinateur de l'attaquant stocke et renvoie le trafic vers la machine cible en forgeant les trames Ethernet contenant la vraie adresse MAC. En conséquence, malgré le fait que tout le trafic de la machine (cible)

ait été intercepté par l'attaquant, la cible ne remarque rien. C'est une technique efficace mais elle est facilement détectable par l'administrateur en consultant les logs ;

- **DNS Spoofing** : le protocole DNS (Domain Name System) assure la correspondance entre un nom de domaine et un adresse IP et vis-versa. Cette méthode d'attaque consiste à envoyer à la victime des fausses réponses à ses requêtes DNS. Généralement, l'objectif est de rediriger les victimes vers un site pirate sans qu'elles ne s'en rendent compte. Il existe deux principales techniques pour le faire :
  - **DNS ID Spoofing** : cela consiste à renvoyer une fausse réponse à la victime avant le serveur DNS tout en prédisant l'ID (numéro d'identification) de la demande. La difficulté de prédiction de l'ID rend cette attaque un peu compliqué ;
  - **DNS Cache Poisoning** : le serveur DNS dispose d'un cache qui peut conserver la correspondance entre un nom d'ordinateur et son adresse IP pendant un certain temps. En fait, le serveur DNS n'a que la correspondance pour les machines du domaine pour lequel il a autorité. Pour les autres, il contacte le serveur DNS qui en a l'autorité. Pour éviter de demander constamment aux autres serveurs DNS, les réponses seront conservées dans un cache. Cette attaque consiste à injecter des informations incorrectes dans ce cache. Cette injection se fait en envoyant des réponses erronées au serveur DNS (demandeur) par l'intermédiaire d'un autre serveur DNS (répondant) contrôlé par l'attaquant.

- **le vol de session TCP (TCP Session Hijacking)** : c'est une technique qui consiste à intercepter une session TCP(Transport Control Protocol) entre deux machines dans le but la rediriger. Dû au fait que l'authentification de l'utilisateur s'effectue uniquement lors de l'établissement de la session, un attaquant qui réussit à intercepter cette session, entre en possession de la connexion pendant toute la durée de la session. Avec TCP la synchronisation est effectuée en numérotant séquentiellement les paquets. Si au cours de l'échange, l'attaquant envoie un paquet de données mal formaté au client, avec l'adresse IP du serveur en plaçant le mauvais numéro de séquence sur le serveur, la session ne sera lus synchronisée. Le client peut alors penser avoir perdu la connexion et arrêtera ses échanges. Ainsi, si l'attaquant envoie le bon numéro de séquence au serveur, il reprendra la connexion et pourra donc injecter des commandes via cette session.

4.1.2.4 Attaques par déni de service et déni de service distribué

Ces attaques consistent à inonder un système de demandes à tel point qu'il ne puisse plus y répondre. Empêchant ainsi les utilisateurs d'avoir accès à la ressource ou au service. Cela menace le principe de la disponibilité. Les techniques utilisées pour effectuer ce type d'attaque sont nombreuses. Nous présentons ici quelques unes :

- **attaque Ping Flood (ICMP Flood)** : de nombreux machines avec des adresses publiques ou privées, renvoient une réponse aux paquets ICMP. Cela rend facile leur inondation avec ces requêtes afin de les rendre indisponibles. En fait, que les cibles répondent ou pas à l'ICMP, le principal objectif est de saturer leurs bandes passantes d'accès réseau, leurs processeurs, leurs mémoires, etc. jusqu'à ce que leurs limites soient atteintes. C'est une technique très répandue pour la catégorie DoS car des amateurs et des curieux s'amusent à envoyer des *pings* à des hôtes en ajoutant des options ou en utilisant des outils permettant d'avoir une cadence élevée ;
- **le flooding** : l'objectif de cette attaque DoS, est de saturer une cible exactement comme le réalise Ping Flood et UDP Flood. Le principe est basé sur l'envoi massif à une machine de nombreux paquets IP de grosse taille. La machine cible ne pourra donc pas traiter tous les paquets et finira par se déconnecter du réseau ;
- **le TCP-SYN flooding** : le TCP-SYN flooding est basé sur une faille du protocole TCP et est en réalité un déclinaison du flooding. Quand une demande de connexion est envoyée au serveur (SYN), il répond en renvoyant un paquet *SYN-ACK* et attend une réponse *ACK* pour établir la connexion. Sans réponse *ACK* le serveur va attendre jusqu'au timeout(expiration) des « demi-connexion ». Ainsi, si on envoie un énorme nombre de demandes de connexion au serveur sans après envoyer les réponses *ACK* et cela, plus vite que le timeout des "demi-connexion", alors le serveur sera saturé et finira par se déconnecter. C'est le principe de cette technique ;
- **UDP Flooding** : le principe de cette attaque est la même à celle du Ping Flood. C'est de saturer les ressources de la cible en terme de débit, processeur, mémoire à l'aide de datagramme UDP et cela que la cible réponde ou pas au flux abondant émis, ne change pas le résultat. Le flux UDP étant prioritaire sur celui de TCP, alors en envoyant des multitudes de paquets UDP occupant toute la bande passante, les connexions TCP seront indisponibles ;
- **smurfing** : cette technique utilise le protocole ICMP. Quand un *ping* (message ICMP ECHO) est envoyé à une adresse de *broadcast* (par exemple 10.255.255.255), celui-ci est démultiplié et envoyé à toutes les machines du réseau. Le principe de l'attaque est de falsifier les paquets envoyés en insérant comme adresse IP source, celle de la cible. Toutes les machines vont naturellement répondre et la machine cible va alors recevoir un très grand nombre de réponses, et ainsi toute sa bande passante sera saturée. Cette attaque peut être amplifiée en le couplant avec d'autres attaques comme Ping Flood, UDP Flood, etc.
- **déni de service distribué (DDoS)** : de nos jours, la grande majorité des attaques DoS s'effectuent à partir de plusieurs sources. C'est à dire que l'attaque est lancée à partir d'un nombre élevé de machines infectées par un malware et contrôlées par l'attaquant, d'où le terme « distribué ». C'est généralement le terme "botnet" qui est utilisé pour qualifier le groupe de machines infectées. En utilisant ces botnets, un plus grand trafic de données est ainsi généré plutôt qu'avec une simple attaque DoS qui n'est lancée qu'à partir d'une seule machine. Les attaques DDoS ont des

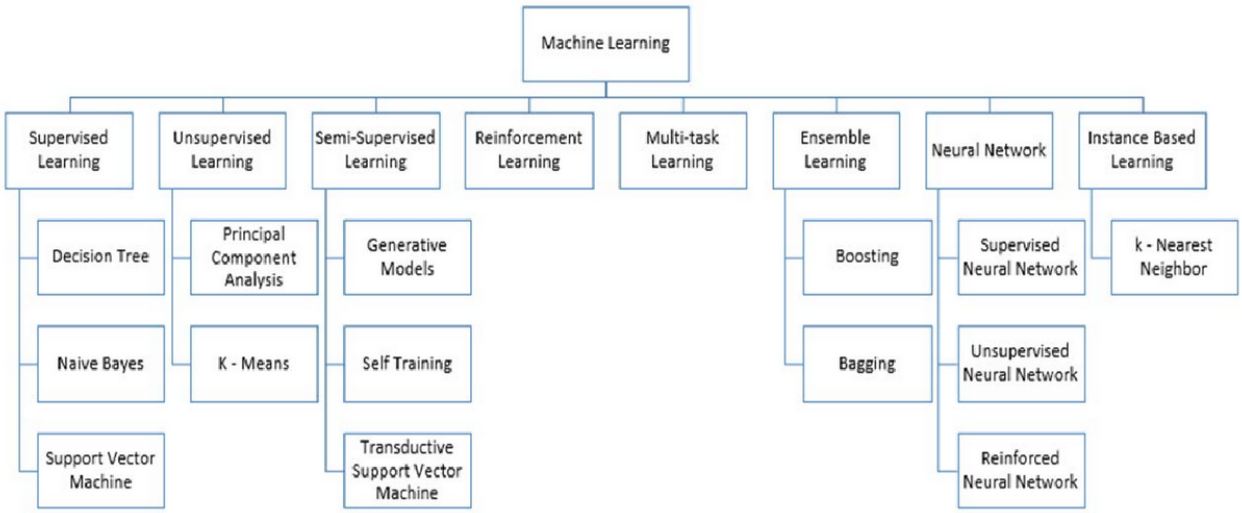

FIGURE 4.1 – Types de modèles d'apprentissage [39]

effets plus importants sur les machines touchées et l'espoir de localiser la source de l'attaque est assez faible.

4.2 Les modèles de machine learning appliqués à la sécurité

Les techniques de machines learning sont souvent classifiées en plusieurs groupes comme le montre la figure fig.4.1. Deux groupes principaux nous intéressent à savoir l'apprentissage supervisé et l'apprentissage non-supervisé. Plus de détails sur la classification des modèles sont abordés par *Ayon Dey* dans son article[39]. On a aussi une autre manière de les distinguer à savoir l'apprentissage profond et l'apprentissage peu profond. Nous présentons quelques modèles liés à ces différents groupes dans les lignes suivantes.

4.2.1 Apprentissage peu profond

L'apprentissage peu profond utilise des modèles de machine learning qui n'ont pas plusieurs couches(couches cachées). Nous les répartissons en deux groupes c'est-à-dire, supervisé et non-supervisé.

4.2.1.1 Apprentissage supervisé

4.2.1.1.1 SVM

C'est un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de discrimination(classification) [15] et de régression(en prédisant la valeur numérique d'une variable).

Le principe des SVM[21] est d'identifier l'hyperplan séparateur avec la marge maximale dans l'espace à n-dimensions des *features*, de telle sorte que, par exemple pour deux classes, la distance entre l'hyperplan et les points de données les plus proches de chaque classe

soit maximisée. Les SVM sont capables de fournir des bons résultats même dans le cas où l'ensemble de données d'entraînement est de taille réduite parce que l'hyperplan séparateur est identifié seulement grâce à un nombre réduit de vecteurs de soutien. Néanmoins ils sont sensibles au bruit. L'approche des SVM est de minimiser les erreurs de classification plutôt que d'optimiser ce dernier. Les SVM sont bien connues pour leur capacité de généralisation et sont particulièrement utiles lorsque le nombre de *features* est largement supérieur au nombre de points de données.

4.2.1.1.2 KNN

Selon [6], les k voisins les plus proches en anglais K-Nearest Neighbor (KNN) est l'une des techniques non paramétriques les plus simples et les plus traditionnelles pour classer les échantillons[14]. Il peut être utilisé aussi bien pour la régression que pour la classification. Dans les deux cas, il s'agit de mettre l'entrée dans le groupe auquel appartiennent les k voisins les plus proches dans l'espace de représentation des données d'entraînement. Le résultat est fonction des fins de l'utilisation :
En classification, le résultat est une classe d'appartenance. Un objet d'entrée est classifié selon le résultat majoritaire des statistiques de classes d'appartenance de ses k plus proches voisins, (k est un nombre entier positif généralement petit). Si k = 1, alors l'objet est affecté à la classe d'appartenance de son proche voisin.
En régression, le résultat est la valeur pour cet objet. Cette valeur est la moyenne des valeurs des k plus proches voisins.
La méthode k-NN est basée sur l'apprentissage préalable, ou l'apprentissage faible, où la fonction est évaluée localement, le calcul définitif étant effectué à l'issue de la classification[34].

Le paramètre k influence grandement la performance des modèles KNN. Plus le paramètre k est petit, plus le modèle est complexe et plus le risque de sur-entraînement est élevé. Inversement, plus k est grand, plus le modèle est simple et plus la capacité d'adaptation est faible.

4.2.1.1.3 Naïve Bayes

Le principe de Naïve Bayes[8] est basé sur la probabilité conditionnelle appliquant le théorème de Bayes et sur l'hypothèse naïve que les attributs sont indépendantes. Pour chaque échantillon, Naïve Bayes calcule les probabilités conditionnelles avec les différentes classes et l'échantillon est classé dans la classe qui a la probabilité maximale. Plus simplement, un classificateur bayésien naïf suppose que l'existence d'une caractéristique pour une classe, est indépendante de l'existence d'autres caractéristiques. Exemple une personne peut être considérée comme un homme si elle a une barbe, des cheveux courts et une proéminence laryngée (pomme d'Adam). Même si ces caractéristiques sont en réalité liées, un classificateur bayésien naïf déterminera que la personne est un homme en considérant indépendamment ces caractéristiques de barbue ou imberbe, de la taille des cheveux et la taille de la proéminence laryngée[34].

Naïve Bayes produit de bons résultats dans la classification lorsqu'il n'existe pas ou peu de relations entre les attributs. Malheureusement, cette condition est difficile à satisfaire dans la réalité. Naïve Bayes ne nécessite qu'un seul balayage des données d'entraînement, ce qui facilite grandement la tâche de classification.

4.2.1.1.4 Logistic regression

Le nom *Logistic regression* provient de la fonction logistique qui est l'élément au cœur de la méthode[18]. La *Logistic regression* modélise les probabilités de problèmes de classification avec deux résultats possibles (binaire)[18]. Elle fait partie des méthodes de référence pour les problèmes de classification binaire. Il s'agit d'une extension du modèle de régression linéaire pour les problèmes de classification.

La *Logistic regression* suppose l'indépendance entre les variables, ce qui n'est pas toujours le cas dans les ensembles de données. Toutefois, comme c'est souvent le cas, l'applicabilité de la méthode (et son efficacité, par exemple l'erreur de classification) l'emporte souvent sur les hypothèses statistiques[34].

Un modèle de *Logistic regression* est facile à construire, et l'entraînement est efficace. Cependant, il ne peut pas bien traiter les données non linéaires, ce qui limite son application.

4.2.1.1.5 Arbre de décision

Un arbre de décision est une structure arborescente constituée de nœuds non terminaux(une racine et des nœuds internes) et de nœuds terminaux(feuilles). Le nœud racine est le premier nœud avec des conditions de test pour diviser chaque enregistrement de données d'entrée vers chaque nœud interne en fonction des caractéristiques de l'enregistrement de données[5].

Tout d'abord, l'arbre de décision est formé avec des données étiquetées avant de pouvoir classer des données qui ne sont pas utilisées pour l'entraînement. Le processus d'entraînement comprend la sélection des attributs, la génération d'arbres et l'élagage des arbres. Lors de l'apprentissage d'un modèle d'arbre de décision, l'algorithme sélectionne un à un les attributs les plus appropriées et génère des nœuds enfants à partir du nœud racine.
L'algorithme d'arbre de décision peut automatiquement exclure les attributs non pertinents et redondants. Après la formation, l'arbre peut prévoir ou classer de nouvelles données en partant d'un nœud racine et en traversant les nœuds internes en fonction des attributs des conditions de test jusqu'à ce qu'il arrive à un nœud feuille (terminal) constitué d'une classe de réponse[7].

Le modèle ressemble à un arbre, ce qui rend son interprétation beaucoup facile. Aussi, l'arbre de décision a une grande précision de la classification et simple à mettre en œuvre. Certains algorithmes avancés, tels que la forêt aléatoire (Random Forest) et le renforcement du gradient extrême (XGBoost), sont constitués de plusieurs arbres de décision[34].

4.2.1.2 Apprentissage non-supervisé

4.2.1.2.1 K-means

K-means[17] est un algorithme typique de clustering, où K est le nombre de *clusters* et *means* est la moyenne des attributs. L'algorithme K-means utilise la distance comme critère de mesure de la similarité afin de regrouper les données similaires en groupes (clusters). Plus la distance entre deux objets est courte, plus il est probable qu'ils soient placés dans le même *cluster*. K-means s'adapte bien aux données linéaires, mais ses résultats sur des données non convexes ne sont pas reluisants. En outre, l'algorithme est sensible à la condition d'initialisation et au paramètre K. Par conséquent, de nombreuses expériences répétées doivent être effectuées pour fixer la valeur correcte du paramètre.
Les moyennes des attributs des données d'un *cluster*, définissent la position de leur centroïde dans l'espace des attributs : ceci est à l'origine du nom de cet algorithme (K-moyennes ou K-means en anglais)[34]. Après avoir initialisé les centroïdes en prenant des données au hasard dans le jeu de données, K-means alterne plusieurs fois ces deux étapes pour optimiser les centroïdes et leurs clusters :

1. regrouper chaque objet autour du centroïde le plus proche ;
2. replacer chaque centroïde selon la moyenne des attributs de son groupe.

Après quelques itérations, l'algorithme trouve un découpage stable de l'ensemble de données : on dit que l'algorithme a convergé.

4.2.2 Apprentissage profond (Deep learning)

4.2.2.1 Apprentissage supervisé

4.2.2.1.1 Réseau de Neurones Artificiels

Un réseau de neurones artificiels en anglais *Artificial Neural Network(ANN)*, est constitué d'une couche de neurones d'entrée, d'une couche de neurones de sortie et entre les deux, d'une ou de plusieurs couches cachées. Chaque couche reçoit les résultats de la couche précédente en entrée, calcule et fournie les résultats en sortie pour la prochaine couche. La première couche est activée par les données d'entrée. L'idée est inspirée du fonctionnement du cerveau humain [4]. Lorsque le réseau de neurones artificiels est utilisé dans le cadre d'une classification, la dernière couche génère en sortie la classification finale. Un réseau de neurones artificiels a une bonne capacité d'adaptation particulièrement pour les fonctions non linéaires grâce à sa couche ou ses couches cachées. Mais en raison de sa structure complexe, le temps d'entraînement est long ,il est généralement utilisé dans le domaine de traitement d'images. Ce domaine emploie de grandes ressources et n'est pas exigent sur le temps d'entraînement.

Les réseaux de neurones profonds en anglais *Deep Neural Network(DNN)* sont une forme raffinée de réseau de neurones artificiels (ANN) ayant plus d'une couche cachée entre les couches d'entrée et de sortie[74]. L'adjectif *profond* vient de l'utilisation de plusieurs couches dans le réseau.
La popularité des DNN augmente avec le développement des réseaux de neurones récur-

rents (RNN), des réseaux de neurones à convolution (CNN), etc.

4.2.2.1.2 CNN

Les CNN[45] sont conçus pour imiter le système visuel humain. Par conséquent, ils ont fait de grandes avancées dans le domaine de la vision par ordinateur. Un CNN est empilé avec des couches de convolution et d'autres couches alternées. Les couches de convolution sont utilisées pour extraire les *features*, et les couches de *pooling* sont utilisées pour améliorer la généralisation des *features*.

Les quatre principales opérations de CNN sont la couche de convolution, la fonction d'activation telle que ReLu ou Softmax, la couche de *pooling* et la couche entièrement connectée (*fully connected*) comme le montre la figure fig.1.1.

- **La couche de convolution** : CNN tire son nom des opérations de convolution. La tâche principale de la convolution est d'extraire des *features* de la donnée d'entrée. La donnée de sortie issue de l'opération de convolution est appelée « *Activation Map* », « *Convolved Feature* » ou « *Featured Map* ». Les valeurs du filtre ou du noyau sont mises à jour automatiquement pendant le processus de formation de CNN pour mieux apprendre les *features* d'une donnée. Des informations plus détaillées sur la couche de convolution sont présentées par Jianxin Wu(2017) [52].
- **Fonction d'activation ReLu** : après chaque opération de convolution, avant de générer la « *Activation Map* », une fonction non linéaire supplémentaire telle que *ReLu* est utilisée. *ReLu* signifie en anglais *Rectified Linear Unit* et est une opération non-linéaire. *ReLu* introduit le comportement non linéaire dans CNN. D'autres fonctions d'activation non linéaires telles que *tanh* et *sigmoïde* peuvent également être utilisées à la place de *ReLu*. Des informations plus détaillées sur la fonction d'activation *ReLu* et d'autres fonctions d'activation peuvent être trouvées dans Agarap(2018) [55].
- **Couche de *pooling*** : L'opération de *pooling* réduit la dimension de chaque « *Activation Map* » mais conserve les informations les plus importantes. L'opération de *pooling* peut être de différents types tels que *Max*, *Moyenne*, *Somme*, etc. La mise en commun Max s'est avérée plus efficace dans de nombreuses applications. Des informations plus détaillées sur la couche de mise en commun sont présentées par Haibing Wu(2015) [38].
- **Couche entièrement connectée ou *fully connected*** : Il s'agit d'un perceptron multicouche traditionnel qui utilise une fonction d'activation *Softmax* dans la couche de sortie. Le terme « entièrement connecté » implique que chaque neurone de la couche précédente est connecté à chaque neurone de la couche suivante. La sortie des couches de convolution et de *pooling* représente les *features* de haut niveau de la donnée d'entrée. La couche entièrement connectée utilise ces *features* pour classer la donnée d'entrée en différentes classes sur la base de l'ensemble de données d'entrainement. Plus de détails sur les couches *fully connected* sont présentées par Karn(2016) [41].

Les CNN travaillent sur des données sur plusieurs dimensions ou unidimensionnelles (1D), de sorte que les données d'entrée doivent être traduites en matrices pour la détection des attaques.

Le pré-traitement requis dans CNN est beaucoup moins important que pour les autres algorithmes de classification. Alors que dans les méthodes primitives, les *features* sont sélectionnés par un expert, avec une formation suffisante, les CNN ont la capacité de sélectionner les *features* les plus représentatifs. Nous reviendrons sur CNN dans le chapitre 2.

4.2.2.1.3 RNN

Les réseaux de neurones récurrents en anglais *Recurrent Neural Nework(RNN)* sont des réseaux conçus pour des données séquentielles et sont largement utilisés dans le traitement du langage naturel. Les *features* des données séquentielles sont contextuelles. L'analyse de données isolées de la séquence n'a aucun sens. Pour obtenir des informations contextuelles, chaque unité d'un RNN reçoit non seulement l'état actuel mais aussi les états précédents. Cette caractéristique fait que les RNN souffrent souvent de la disparition ou de l'explosion des gradients. En réalité, les RNN standards ne traitent que des séquences de longueur limitée. Pour résoudre le problème de la dépendance à long terme, de nombreuses variantes ont été proposées, telles que LSTM, GRU, et Bi-RNN.

- **Bi-RNN** : [73] le principe de Bi-RNN est de relier deux couches cachées de directions opposées à la même sortie. Grâce à cette forme d'apprentissage génératif profond, la couche de sortie peut obtenir des informations à partir des états passés (en arrière) et futurs (en avant) simultanément. Il a été introduit par Schuster et al(1997)[9]. Pour une discussion détaillée ses différentes architectures, voir également l'article de Graves et al. (2005)[20].
- **LSTM** : la mémoire à long et court terme en anglais *Long Short-Term Memory(LSTM)*[26] est un système d'apprentissage profond qui évite le problème d'évanouissement du gradient. LSTM peut apprendre des tâches qui nécessitent des souvenirs d'événements qui se sont produits plus tôt. LSTM fonctionne même avec de longs délais entre des événements importants et peut traiter des signaux qui mélangent des composantes à basse et haute fréquence. LSTM peut apprendre à reconnaître les langages sensibles au contexte, contrairement aux modèles précédents basés sur des modèles de Markov cachés et des concepts similaires. LSTM est bien adaptée à la classification, au traitement et à la prédiction de séries temporelles avec des décalages temporels de durée inconnue.
- **GRU** : les Gated Recurrent Units (GRU) sont des réseaux neuronaux récurrents introduit en 2014 par Cho et al.(2014)[31]. Pour résoudre le problème d'évanouissement du gradient d'un RNN standard, les GRU utilisent ce qu'on appelle une porte de mise à jour et une porte de réinitialisation. En gros, ce sont deux vecteurs qui décident des informations à transmettre à la sortie. Leur particularité est qu'ils peuvent être formés pour conserver des informations datant de longtemps, sans les effacer ou supprimer des informations non pertinentes pour la prédiction. Plus de

details peuvent etre trouver dans cet article de Christopher Olah[37]. Leur performance sur la modélisation de la musique polyphonique et du signal de parole s'est avérée similaire à celle de LSTM.

4.2.2.2 Apprentissage non-supervisé

4.2.2.2.1 GAN

les réseaux adverses génératifs (en anglais generative adversarial networks ou GAN) ont été introduit en 2014 par Goodfellow et al.[32]. Un modèle de GAN comprend deux sous-réseaux, un générateur et un discriminateur. Le générateur vise à générer des données synthétiques similaires aux données réelles, et le discriminateur vise à distinguer les données synthétiques des données réelles. Ainsi, le générateur et le discriminateur s'améliorent mutuellement.

La tâche en aval des GAN est une tâche de discrimination entre les échantillons réels et générés. On pourrait aussi dire une tâche de « non-discrimination » car on veut que la discrimination échoue autant que possible. Ainsi, dans une architecture GAN, nous avons un discriminateur, qui prend des échantillons de données réelles et générées et qui essaie de les classer le mieux possible, et un générateur qui est formé pour tromper le discriminateur autant que possible.

Les GAN sont actuellement un sujet de recherche en vogue utilisé pour augmenter les données dans la détection des attaques, ce qui atténue en partie le problème de la pénurie d'ensembles de données des IDS. En même temps, les GAN font partie des approches d'apprentissage contradictoires qui peuvent augmenter la précision de détection des modèles en ajoutant des échantillons contradictoires à l'ensemble d'entrainement.

4.2.2.2.2 RBM

La Machine de Boltzmann Restreinte(en anglais Restricted Boltzmann Machine : RBM)[23] est un type de réseau de neurones artificiels pour l'apprentissage non supervisé. Un RBM est un réseau de neurones randomisé dans lequel les unités obéissent à la distribution de Boltzmann. Un RBM est composé d'une couche visible et d'une couche cachée. Les unités d'une même couche ne sont pas connectées. Cependant, les unités de différentes couches sont entièrement connectées. Les RBM ne font pas la distinction entre les directions avant et arrière. Ainsi, les poids dans les deux directions sont les mêmes. Ce sont des modèles d'apprentissage formés par l'algorithme de divergence contrastive, et ils sont généralement appliqués pour l'extraction des *features* ou le débruitage. L'article de Nan Zhang et al.(2018) [65] donne plus de détails.

4.2.2.2.3 Autoencoder

Un autocodeur contient deux composants symétriques, un codeur et un décodeur. Le codeur extrait des caractéristiques des données brutes, et le décodeur reconstruit les données à partir des caractéristiques extraites. Il prend des données d'entrée telles que des images ou des vecteurs de très haute dimension et les fait passer à travers le réseau neuronal en essayant de les comprimer en une représentation plus petite avec deux com-

posantes principales. Le premier est l'encodeur, qui est simplement un groupe de couches entièrement connectées ou de couches convolutionnelles. Elles vont prendre l'entrée et la comprimer en une représentation plus petite qui a moins de dimensions que l'entrée, ce qu'on appelle un goulot d'étranglement. À partir de ce goulot d'étranglement, il tente de reconstruire l'entrée en utilisant des couches complètement connectées ou convolutives. Au cours de la formation, la divergence entre l'entrée du codeur et la sortie du décodeur est progressivement réduite. Lorsque le décodeur parvient à reconstruire les données à partir des caractéristiques extraites, cela signifie que les *features* extraits par l'encodeur représentent l'essence des données. Il est important de noter que tout ce processus ne nécessite aucune intervention. Il existe de nombreuses variantes d'autocodeurs célèbres.

- **Stacked Autoencoder** : c'est un réseau de neurones constitué de plusieurs couches de Sparse Autoencoder où la sortie de chaque couche cachée est reliée à l'entrée de la couche cachée suivante [43]. Les couches cachées sont formées par un algorithme non supervisé, puis affinées par une méthode supervisée. Stacked Autoencoder se compose principalement de trois étapes :
  - former l'autocodeur en utilisant les données d'entrée et acquérir les données apprises ;
  - les données apprises de la couche précédente sont utilisées comme entrée pour la couche suivante et cela continue jusqu'à ce que la formation soit terminée ;
  - une fois que toutes les couches cachées sont formées, l'algorithme de rétro-propagation est utilisé pour minimiser la fonction de coût et les poids sont mis à jour avec l'ensemble de données pour obtenir un réglage fin.

  Les récents progrès de Stacked Autoencoder permettent de traiter des données brutes et avoir beaucoup plus d'informations détaillées et prometteuses sur les *features*. Ce qui est ensuite utilisé pour former un classificateur avec un contexte spécifique et trouver une meilleure précision que la formation avec des données brutes.
- **Denoising Autoencoder** : un Denoising Autoencoder[29] essaie d'apprendre une représentation qui est robuste au bruit [1]. Il est une version stochastique des autoencodeurs standards qui réduit le risque d'apprendre la fonction d'identité (où la sortie est simplement égale à l'entrée). En général, plus il y a de couches cachées dans un autocodeur, plus la compression peut être affinée. Toutefois, si un autoencodeur comporte plus de couches cachées que d'entrées, il y a un risque que l'algorithme n'apprenne la fonction d'identité pendant la formation, et devienne alors inutile. Les autocodeurs de débruitage tentent de contourner ce risque en introduisant du bruit, c'est-à-dire en corrompant aléatoirement l'entrée de sorte que l'autocodeur doit ensuite "débruiter" ou reconstruire l'entrée originale. La partie encodeur de l'autoencodeur transforme l'entrée en un espace différent qui tente de préserver les *features*, mais supprime le bruit. Pendant l'entrainement, vous définissez une fonction de perte, similaire à l'erreur quadratique moyenne. À chaque itération de la formation, le réseau calculera une perte entre l'image contenant du bruit produit

1. `https://towardsdatascience.com/denoising-autoencoders-explained-dbb82467fc2`

par le décodeur et l'image d'origine, et essaiera également de minimiser la différence entre l'image reconstruite et l'image originale sans bruit ;

- **Sparse Autoencoder** : un Autoencoder prend l'image ou le vecteur d'entrée et apprend le dictionnaire de codes qui change l'entrée brute d'une représentation à l'autre. Dans Sparse Autoencoder, l'application de la règle de la rareté dirige un réseau à couche unique vers l'apprentissage du dictionnaire de code, ce qui minimise l'erreur de reproduction de l'entrée tout en limitant le nombre de mots de code pour la reconstruction [64]. Le Sparse Autoencoder est constitué d'une seule couche cachée, qui est reliée au vecteur d'entrée par une matrice de poids formant l'étape d'encodage. La couche cachée produit ensuite un vecteur de reconstruction, en utilisant une matrice de poids lié pour former le décodeur [64].

4.3 Les ensembles de données

La tâche de l'apprentissage machine consiste à extraire des informations précieuses des données. Par conséquent, la performance de l'apprentissage machine dépend de la qualité des données d'entrée. La compréhension des données est la base de la méthodologie du machine learning. Pour les IDS, les données adoptées doivent être faciles à acquérir et refléter les comportements des hôtes ou des réseaux. Les types de données sources les plus courantes pour les IDS sont les paquets, les flux, les sessions et les journaux. La constitution d'un ensemble de données est complexe et prend du temps. Une fois qu'un ensemble de données de référence est construit, il peut être réutilisé à plusieurs reprises par de nombreux chercheurs. Outre la commodité, l'utilisation d'ensembles de données de référence présente deux autres avantages :

1. Les ensembles de données de référence font autorité et rendent les résultats expérimentaux plus convaincants.
2. De nombreuses études publiées ont été réalisées à l'aide d'ensembles de données de référence communs, ce qui permet de comparer les résultats des nouvelles études avec ceux des études précédentes.

Les ensembles de données les plus utilisés dans la recherche sont cités dans les lignes suivantes.

4.3.1 Defense Advanced Research Projects Agency (DARPA) 1998 et 1999

Il fait partie des ensembles de données les plus utilisés et les plus cités dans les recherches. C'est un ensemble de données créé par Cyber Systems and Technology Group of the Massachusetts Institute of Technology Lincoln Laboratory (MIT/LL). Les données ont été récoltées en créant un réseau expérimental puis en procédant à des simulations d'attaques sur celui-ci. Ces données ont ensuite été traitées et organisées en plusieurs colonnes et sauvegardées pour les expériences futures.

- Pour DARPA 1998, les simulations d'attaque ont durée neuf (09) semaines et les données recueillies étaient celles concernant le réseau TCP/IP et celles concernant le système d'exploitation (Solaris) dont essentiellement les logs du module de sécurité et le dump du super-utilisateur et l'utilisateur. DARPA 1998 contient quatre types d'attaques : Déni de service (DoS), d'utilisateur à racine (U2R), de distant à local (R2L), et Probe ou Scan. Les données de sept (07) semaines sont utilisées pour l'entraînement et celles des deux (02) autres semaines pour le test des algorithmes de Machine learning.

- Pour ce qui concerne DARPA 99 les données ont été collectées pendant cinq (05) semaines (les données de trois (03) pour les entraînements et les données de deux (02) pour les tests) dans les mêmes conditions que DARPA 98 mais d'autres types d'attaques supplémentaires ont été simulés. La DARPA 1999 contient un nouveau type d'attaque où l'attaquant tente de récupérer des fichiers spéciaux qui doivent rester sur l'ordinateur de la victime.

4.3.2 KDD99

Créée par le KDD Cup challenge, KDD 99 découle des données TCP/IP de DARPA 98. C'est l'un des ensembles de données les plus utilisés pour les expériences. KDD 99 a des caractéristiques supplémentaires au DARPA 98. Ces caractéristiques ont été capturées par pcap et ont été obtenues en analysant les données selon l'estampillage et leur séquence. KDD 99 possède jusqu'à quarante un (41) attributs. L'ensemble de données KDD 1999 est similaire aux données NetFlow, mais possède des caractéristiques plus dérivées et plus détaillées car les attaques ont été simulées. L'ensemble des données du KDD 1999 (avec environ 4 millions d'enregistrements de trafic normal et d'attaque) a été analysé de manière exhaustive par Tavallaee et al.(2009)[24]. Quelques problèmes inhérents ont été relevés, tels que la synthèse du réseau et des données d'attaque (après échantillonnage du trafic réel) en raison de préoccupations liées à la protection de la vie privée, un nombre inconnu de paquets abandonnés en raison du débordement du trafic, et des définitions d'attaque vagues. Tavallaee et al.(2009)[24] ont également effectué des évaluations statistiques et leurs propres expériences de classification. Ils ont signalé un nombre énorme d'enregistrements redondants (78% dans les données de formation et 75% dans les données de test) causant un biais. En outre, dans les expériences de classification menées par le groupe, ils ont souligné qu'en sélectionnant au hasard des sous-ensembles de données d'entraînement et de test, il est possible d'obtenir des précisions souvent très élevées et irréalistes. Ils ont proposé un nouvel ensemble de données, la NSL-KDD, qui consiste en des enregistrements sélectionnés de l'ensemble complet des données KDD et qui ne présente pas les lacunes mentionnées ci-haut. On peut souligner aussi que les données KDD sont très anciennes pour représenter l'environnement réseau actuel.

4.3.3 NSL-KDD

Pour pallier les insuffisances de l'ensemble de données KDD99, la NSL-KDD[24] a été proposée. Les enregistrements de la NSL-KDD ont été soigneusement sélectionnés sur la base de la KDD99. Les enregistrements des différentes classes sont équilibrés dans la NSL-KDD, ce qui évite le problème du biais de classification. La NSL-KDD a également supprimé les enregistrements en double et les enregistrements redondants. Elle ne contient donc qu'un nombre modéré d'enregistrements. Par conséquent, les expériences peuvent être mises en œuvre sur l'ensemble des données, et les résultats des différents documents sont cohérents et comparables. La NSL-KDD atténue dans une certaine mesure les problèmes de biais et de redondance des données. Toutefois, la NSL-KDD ne contient pas de nouvelles données. Ainsi, les échantillons des classes minoritaires font toujours défaut et ses échantillons sont toujours périmés.

4.3.4 UNSW-NB15

L'ensemble de données UNSW-NB15[36] a été compilé par l'Université du Sud du Pays de Galles, où les chercheurs ont configuré trois serveurs virtuels pour capturer le trafic réseau et ont extrait des caractéristiques en 49 dimensions à l'aide de l'outil appelé Bro. L'ensemble de données comprend plus de types d'attaques que l'ensemble de données KDD99, et ses caractéristiques sont plus nombreuses. Les catégories de données comprennent des données normales et neuf types d'attaques. Les caractéristiques comprennent des caractéristiques de flux, des caractéristiques de base, des caractéristiques de contenu, des caractéristiques temporelles, des caractéristiques supplémentaires et des caractéristiques étiquetées. L'UNSW-NB15 est représentatif des nouveaux ensembles de données des IDS, et a été utilisé dans certaines études récentes.

4.3.5 CIC-IDS2017[61]

Cet ensemble de données a été enregistré dans un petit environnement de réseau avec un tracé normal simulé. Sept catégories(Brute Force, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS) d'attaques modernes sont lancées à partir d'un réseau distinct. La capture des paquets bruts (PCAP) et des flux réseaux étiquetés avec 80 fonctionnalités(en fichier CSV) sont tous deux mise à disposition.

4.3.6 CSE-CIC-IDS2018[91]

L'ensemble de données CSE CIC-IDS2018 contient 86 "features" ou caractéristiques qui sont dans le tableau Table 4.2. Se référer à 1.2.2.1 pour plus de détails.

Nom	Description	Nom	Description
Flow ID	Flow ID	Src IP	Source IP address
Dst IP	Destination IP address	Src Port	Source port
Dst Port	Destination port	Protocol	Protocol used
Label	Attack	Timestamp	Timestamp
Flow Duration	Flow Duration	Tot Fwd Pkts	Total packets in the forward direction
Tot Bwd Pkts	Total packets in the backward direction	Totlen Fwd Pkts	Total size of packet in forward direction
Fwd Pkt Len Min	Minimum size of packet in forward direction	Fwd Pkt Len Mean	Average size of packet in forward direction
Fwd Pkt Len Std	Standard deviation size of packet in forward direction	Bwd Pkt Len Max	Maximum size of packet in forward direction
Bwd Pkt Len Min	Minimum size of packet in backward direction	Bwd Pkt Len Mean	Mean size of packet in backward direction
Bwd Pkt Len Std	Standard deviation size of packet in backward direction	Flow Byts/s	flow byte rate that is number of packets transferred per second
Flow Pkts/s	flow packets rate that is number of packets transferred per second	Flow IAT Mean	Average time between two flows
Flow IAT Std	Standard deviation time two flows	Flow IAT Max	Maximum time between two flows
Flow IAT Min	Minimum time between two flows	Flow IAT Tot	Total time between two packets sent in the forward direction
Fwd IAT Mean	Mean time between two packets sent in the forward direction	Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
Fwd IAT Max	Maximum time between two packets sent in the forward direction	Fwd IAT Min	Minimum time between two packets sent in the forward direction
Bwd IAT Tot	Total time between two packets sent in the backward direction	Bwd IAT Mean	Mean time between two packets sent in the backward direction
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction	Bwd IAT Max	Standard deviation time between two packets sent in the backward direction
Bwd IAT Min	Minimum time between two packets sent in the backward direction	Fwd PSH Flags	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
Bwd PSH Flags	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)	Fwd URG Flags	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
Bwd URG Flags	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)	Fwd Header Len	Total bytes used for headers in the forward direction
Bwd Header Len	Total bytes used for headers in the forward direction	Fwd Pkts/s	Number of forward packets per second
Bwd Pkts/s	Number of backward packets per second	Pkt Len Min	Minimum length of a flow
Pkt Len Max	Maximum length of a flow	Pkt Len Mean	Mean length of a flow
Pkt Len Std	Standard deviation length of a flow	Pkt Len Var	Minimum inter-arrival time of packet
FIN Flag Cnt	Number of packets with FIN	SYN Flag Cnt	Number of packets with SYN
RST Flag Cnt	Number of packets with RST	PSH Flag Cnt	Number of packets with PUSH
ACK Flag Cnt	Number of packets with ACK	URG Flag Cnt	Number of packets with URG
CWE Flag Count	Number of packets with CWE	ECE Flag Cnt	Number of packets with ECE
Down/Up Ratio	Download and upload ratio	Pkt Size Avg	Average size of packet
Fwd Seg Size Avg	Average size observed in the forward direction	Bwd Seg Size Avg	Average size observed in the backward direction
Fwd Byts/b Avg	Average number of bytes bulk rate in the forward direction	Fwd Pyts/b Avg	Average number of packets bulk rate in the forward direction
Fwd Blk Rate Avg	Average number of bulk rate in the forward direction	Bwd Byts/b Avg	Average number of bytes bulk rate in the backward direction
Bwd Pyts/b Avg	Average number of packets bulk rate in the backward direction	Bwd Blk Rate Avg	Average number of bulk rate in the backward direction
Subflow Fwd Pkts	The average number of packets in a sub flow in the forward direction	Subflow Fwd Bkts	The average number of bytes in a sub flow in the forward direction
Subflow Bwd Pkts	The average number of packets in a sub flow in the backward direction	Subflow Bwd Bkts	The average number of bytes in a sub flow in the backward direction
Init Fwd Win Byts	Number of bytes sent in initial window in the forward direction	Init Bwd Win Byts	Number of bytes sent in initial window in the backward direction
Fwd Act Data Pkts	Number of packets with at least 1 byte of TCP data payload in the forward direction	Fwd Seg Size Min	Minimum segment size observed in the forward direction
Active Mean	Mean time a flow was active before becoming idle	Active Std	Standard deviation time a flow was active before becoming idle
Active Max	Maximum time a flow was active before becoming idle	Active Min	Minimum time a flow was active before becoming idle
Idle Mean	Mean time a flow was idle before becoming active	Idle Std	Standard deviation time a flow was idle before becoming active
Idle Max	Maximum time a flow was idle before becoming active	Idle Min	Minimum time a flow was idle before becoming active

FIGURE 4.2 – Les *features* de CSE CIC-IDS2018