الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de L'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ BLIDA 1
Faculté de Technologie
Département d'Électronique

MÉMOIRE DE MASTER
En Télécommunication

Spécialité : Réseaux & Télécommunications

THÈME :

L'apprentissage profond appliqué à la détection de bots P2P

Réalisé par
Mr. MLALAZI Mzwakhe
Mr. ZOULIADINE Ibrahim Rabiou

Encadré par
Mr. MEHDI Merouane

Juin 2024

Avant tout, nous offrons notre plus profonde gratitude à Dieu, pour ses bénédictions dans nos vies.

À nos parents bien-aimés, pour leur confiance inébranlable en nos capacités, leurs sacrifices sans fin et leur amour inconditionnel qui ont été le moteur de notre réussite.

À tous nos chers amis, qui ont partagé ce voyage avec nous, en nous offrant leur soutien, en partageant leurs connaissances et en nous poussant à viser l'excellence.

Nous souhaitons également dédier ce travail à toutes les personnes qui ont joué un rôle important dans notre parcours académique. Leur soutien, leur amour et leurs encouragements inébranlables ont été inestimables tout au long des hauts et des bas.

Remerciements

Tout d'abord, nous remercions le bon Dieu qui nous a accordé le don de la vie et sans cette vie rien n'est possible.

Ensuite, nous remercions notre encadreur Monsieur MEHDI Merouane qui nous a accompagné tout au long de ce travail, nous lui sommes très reconnaissants pour sa disponibilité et pour son soutien précieux.

Sans oublier nos très chers parents qui nous ont soutenu tout au long de nos cursus scolaire et universitaire jusqu'à la réalisation de ce projet.

Nos profonde reconnaissance et nos remerciements s'adressent spécialement à nos professeurs de notre spécialité, Réseaux et Télécommunications pour leurs qualités scientifiques qui nous ont permis de mener à bien ce projet de fin d'études. Nous disons un grand merci à notre chef de département d'électronique, Dr. Hocine AIT SAADI pour son expertise, sa détermination et son engagement envers l'excellence de notre département de l'électronique.

Enfin nous tenons à exprimer notre plus profonde gratitude à tous nos collègues, à tous ceux qui ont contribué d'une manière ou d'une autre à la réussite de nos études et ceux qui liront ce modeste travail.

ملخص

يهدف هذا المشروع إلى تحسين أمان شبكات الكمبيوتر من خلال الكشف عن شبكات الروبوت ص٢ص، والتي غالبًا ما تستخدم للهجمات الإلكترونية الضارة مثل ض. باستخدام نماذج التعلم العميق مثل ور، قمنا بتقييم أدائها باستخدام مجموعة بيانات رةو-٣١ بتنت (السيناريو رقم ٢١) جنبًا إلى جنب مع تدفقات حركة المرور من الحقيقة الأساسية لمجموعة بيانات سيشاضي. تشير نتائجنا إلى أن نموذج ر يُظهر كفاءة عالية في اكتشاف شبكات الروبوت ص٢ص، محققًا دقة قدرها ٤٩.٩٩٪ ودقة ٦٥.٩٩٪ مع معدل إيجابي كاذب (سصض) قدره ٥٦٤٠.٠٪.

الكلمات المفتاحية: التعلم العميق (ش)، بوت ص٢ص، شبكات الروبوت، ، ر، ض

Abstract

This research project aims to enhance computer network security by detecting P2P botnets, which are often used for malicious cyber-attacks like DDoS. Utilizing deep learning models such as DNN and CNN, we evaluated their performance using the CTU-13 botnet dataset (scenario n°12) combined with ground-truth traffic flows from the HIKARI dataset. Our findings indicate that the CNN model demonstrates high effectiveness in detecting P2P botnets, achieving an accuracy of 99.94% and a precision of 99.56% with a false positive rate (FPR) of 0.0465%.

Keywords: Deep Learning (DL), P2P bot, Botnets, DNN, CNN, RNN

Résumé

Ce projet vise à améliorer la sécurité des réseaux informatiques en détectant les botnets P2P, souvent utilisés pour des cyberattaques malveillantes comme les DDoS. À l'aide de modèles d'apprentissage profond tels que DNN et CNN, nous avons évalué leurs performances à l'aide de l'ensemble de données du botnet CTU-13 (scénario n°12) combiné aux flux de trafic de ground-truth de l'ensemble de données HIKARI. Nos résultats indiquent que le modèle CNN démontre une grande efficacité dans la détection des botnets P2P, atteignant un accuracy (une exactitude) de 99.94% et une précision de 99.56% avec un taux de faux positifs (FPR) de 0.0465%.

Mots Clée : L'apprentissage profond, Bot P2P, Réseaux de bots, DNN, CNN, RNN

Listes des abréviations

1D : 1 Dimensional
AHP2P : Advanced Hybrid Peer-to-Peer
ANN : Artificial Neural Network
ARPANET : Advanced Research Projects Agency Network
AUC : Area Under Curve
BFU : Best First Union
C&C : Command and Control
CNN : Convolutional Neural Network
CPU : Central Processing Unit
CSV : Comma-Separated Values
CTU-13 : Czech Technical University (dataset avec 13 scénarios)
DDOS : Distributed Denial of Service
DL : Deep Learning
DOS : Denial of Service
DNN : Deep Neural Network
DNS : Domain Name System
ELU : Exponential Linear Unit
FN : False Negative
FPR : False Positive Rate
GIMPS : Great Internet Mersenne Prime Search
GPU : Graphics Processing Unit
GRU : Gated recurrent unit
HD : High Definition
HTTP : Hyper-Text Transfer Protocol
IA : Intelligence Artificielle
ID : Identifiant
IDE : Intergrated Development Environment
IDS : Intrusion Detection System

INF : Infinite Value
IoT : Internet of Things
IP : Internet Protocol
IPS : Intrusion Prevention System
IRC : Internet Relay Chat
LSTM : Long Short-Term Memory
MITM : Man In The Middle
ML : Machine Learning
MLP : Multi-Layer Perceptron
NAN : Not A Number
NCA : Neural Computing and Applications
N-EDPS : Network-based Detection and Prevention Systems of botnet
P2P : peer-to-peer
PCAP : Packet CAPture
PR : Precision
RAM : Random Access Memory
RC : Recall
ReLU : Rectified Linear Unit
RNN : Recurrent Neural Network
ROC : Receiver Operating Characteristic
TCP : Transmission Control Protocol
TN : True Negative
TP : True Possitive
TPR : True Possitive Rate
URL : Uniform Resource Locator

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

Nous vivons à une époque d'information, où tout le monde est connecté et où la civilisation est à son apogée. Chaque entreprise a une présence en ligne, c'est une époque où des volumes inimaginables de données sont transmis en une fraction de seconde. Il existe désormais un besoin pour un système de détection d'intrusion sécurisé et fiable qui garantira la détection de toute forme d'attaque réseau.

Récemment, les techniques d'apprentissage profond ont reçu une attention considérable en raison de leur capacité à gérer des datasets de grande taille. Très peu d'articles ont tenté de détecter les botnets à l'aide de techniques d'apprentissage profond. Dans cette étude, nous développons une approche de détection de robots P2P utilisant des réseaux de neurones profonds incorporés à des techniques de sélection de fonctionnalités pour identifier les connexions informatiques à l'origine de comportements de trafic malveillants.

Nous allons adopter un réseau neuronal profond pour détecter les connexions P2P Bot qui sont les activités malveillantes insensibles. Ça fait partie des principaux objectifs de notre méthode de détecter les P2P Bots sans effectuer une inspection approfondie des paquets dans un court laps de temps, de détecter les P2P Bots sans analyser l'ensemble du trafic réseau et enfin améliorer les systèmes de détection d'intrusion de botnets P2P et obtenir un taux de détection élevé. La réalisation de ce projet en intelligence artificielle se fera à l'aide de l'environnement de développement Anaconda, où le langage Python y est installé ainsi que les librairies open source, mathématiques, graphiques (matplotlib) et spécifiques au deep learning (tensorflow). Parmi, ces bibliothèques open source, Keras exécutée par-dessus des frameworks tels que Theano et TensorFlow, est conçue pour être modulaire, rapide et simple d'utilisation. Créée par l'ingénieur François Chollet de Google, Keras offre une façon simple et intuitive de créer des modèles du Deep Learning.

Dans ce mémoire de master, nous avons quatre chapitres dans lesquels nous allons aborder l'apprentissage profond appliqué à la détection de bots P2P. Dans le premier chapitre, nous allons présenter l'introduction à la sécurité informatique et aux botnets en général et puis la définition des botnets P2P ainsi que les approches traditionnelles de détection des botnets P2P. Enfin nous allons introduire les méthodes de deep learning en sécurité informatique et la détection de botnets P2P utilisant des méthodes de deep learning qui est le coeur de notre recherche.

Dans le deuxième chapitre, nous allons aborder les principes de base du deep learning ensemble avec l'architecture des réseaux de neurones convolutifs (CNN) et des réseaux de neurones récurrents (RNN). Ensuite le pré-traitement des données pour la détection de botnets et les métriques d'évaluation de la performance des modèles de deep learning sera discuté.

Le troisième chapitre est la partie pratique de notre projet de fin d'études. II va expliquer les critères utilisés pour choisir le dataset et aussi la préparation de ces données. Ensuite nous aborderons l'architecture du modèle de détection de botnets P2P, la méthodologie d'entraînement et d'évaluation ainsi que les outils utilisés pour le développement et l'évaluation du modèle. Nous allons décrire l'environnement de travail Anaconda, et aussi discuter sur l'implémentation du modèle de détection de botnets P2P. Enfin nous allons présenter la configuration des expériences et des paramètres et les techniques scientifique d'évaluation de la performance du modèle en question.

Le quatrième chapitre sera le dernier chapitre où nous allons présenter l'analyse des résultats et la discussion des observations. Dans ce chapitre, nous évaluerons la performance du modèle sur différents ensembles de données et nous comparerons notre modèle avec d'autres approches de détection de botnets P2P. À la fin, nous allons proposer une solution future pour implémenter le travail dans un IDS.

Chapitre 1

État de l'art

1.1 Introduction

De nos jours, l'informatique joue un rôle central dans tous les aspects de notre vie, des communications personnelles aux opérations commerciales mondiales en passant par des institutions gouvernementales. Cette révolution technologique rapide s'accompagne toutefois d'un enjeu crucial : la sécurité des données.

Jamais auparavant la protection contre les cyber-menaces n'a été aussi pressante. Avec les progrès constants dans le domaine de la technologie de l'information, les méthodes de sécurisation des échanges de données évoluent constamment, mais aussi, les techniques mises au point par les pirates pour contourner les systèmes de sécurité. Dans ce contexte, renforcer la sécurité des données est devenu une priorité absolue.

Parmi les moyens les plus utilisés par les pirates aujourd'hui pour effectuer des attaques notamment des attaques distribuées de déni de service (DDoS) entraînant une interruption des services en ligne [25], des pertes financières [26] et une altération de la réputation des entreprises touchées, ont retrouve les botnets et les p2p botnets. Ces réseaux d'ordinateurs infectés et contrôlés à distance par des acteurs malveillants sont devenus ces dernières années la plus grande menace qui puisse exister pour la sécurités de système d'information et de communication.

1.2 Introduction à la sécurité informatique et aux botnets

Aujourd'hui les attaquent sont devenues de plus en plus fréquentes à l'ordre de milliard par jour [27] du fait notamment de développement rapide des technologies de l'information et des communications, cela permet un partage rapide des connaissances sur des failles ou des méthodes d'attaques conçu par des pirates chevronnés et partager dans des forums en ligne. Ces outils partagés sont tester par des amateurs curieux pour lancer des attaques sans pour autant avoir de compétences et ou mesurés les conséquences des leurs actions.

Ces méthodes de plus en plus sophistiquées sont à la base de plusieurs types d'attaques chacune ayant ses propres objectifs, méthodes et conséquences. ll existe des attaques très varier de par les moyens et les méthodes utiliser dont les plus connues sont :

Figure 1.1 – Types des cyber-attaques [1]

1.2.1 Attaques par déni de service (DoS) et déni de service distribué (DDoS)

Ces attaques visent à rendre un service indisponible en saturant ses ressources avec un grand volume de trafic. Dans une attaque DoS, un seul point d'origine envoie le trafic, tandis que dans une attaque DDoS, de multiples sources, souvent des botnets , sont utilisées pour amplifier l'attaque.

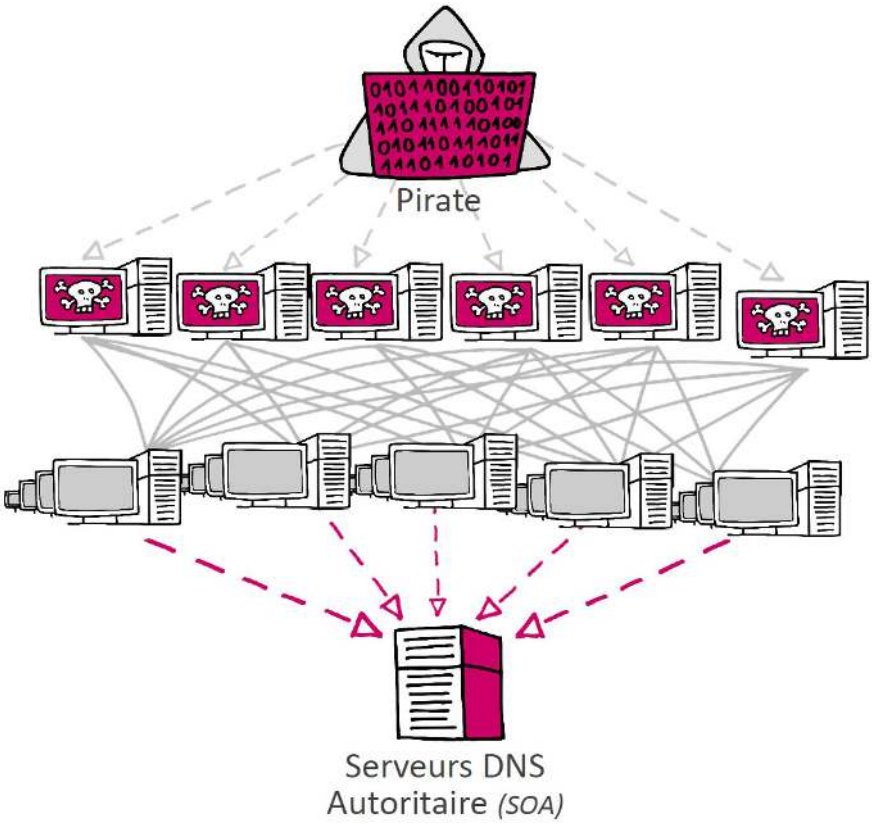

Figure 1.2 – Attaque DDoS [2]

1.2.2 Attaques par hameçonnage (phishing)

Les attaques de phishing utilisent des emails, des messages instantanés ou d'autres moyens de communication pour inciter les utilisateurs à divulguer des informations personnelles telles que des identifiants de connexion, des informations de carte de crédit ou des mots de passe. L'attaque par phishing consiste à tromper les utilisateurs pour obtenir des informations confidentielles. Les attaquants envoient souvent de faux e-mails ou des messages instantanés qui semblent provenir de sources légitimes afin de convaincre les utilisateurs de divulguer leurs informations [28].

Figure 1.3 – Attaque par Phishing [3]

1.2.3 Attaques de l'homme du milieu (MITM)

Lors de ces attaques, un attaquant s'insère entre les communications légitimes entre deux parties et intercepte ou modifie les données échangées. Cela peut permettre à l'attaquant d'espionner les communications ou d'injecter du code malveillant.

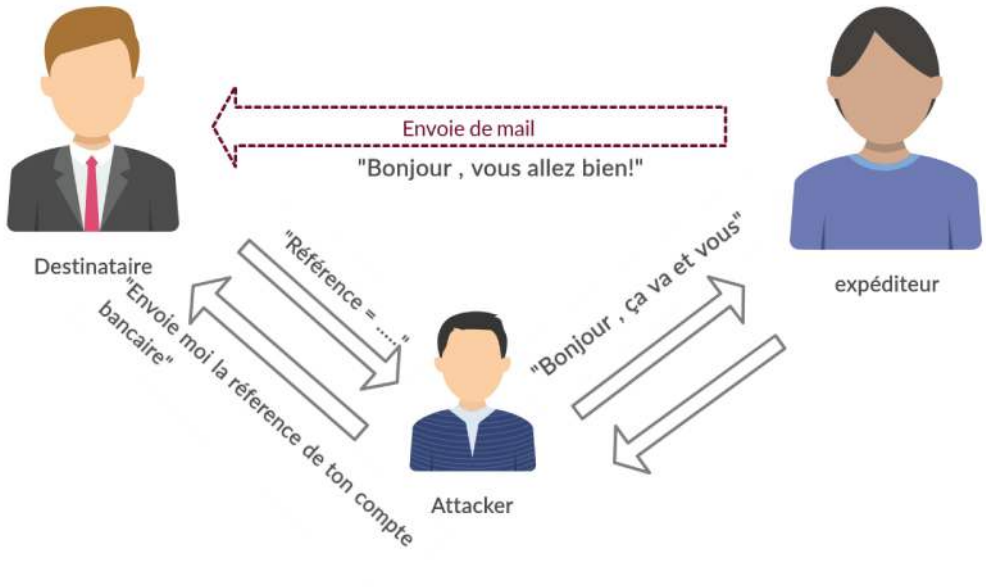

Figure 1.4 – Attaques de l'homme du milieu (MITM) [4]

1.2.4 Les logiciels Malveillants

Les logiciels malveillants, tels que les virus, les vers, les chevaux de Troie et les ransomwares, sont des programmes conçus pour causer des dommages aux systèmes ou voler des données. Ils peuvent être distribués via des emails, des sites web compromis, des périphériques USB infectés, etc [28]. Dans notre cas nous allons nous focaliser sur le botnet P2P.

Les botnets appartiennent à un groupe de robots préprogrammés, contrôlés par une seule entité botmaster [29]. Les botmasters contrôlent un ou plusieurs serveurs C&C qui sont utilisés pour distribuer des commandes à leurs bots (robots), afin d'effectuer à distance plusieurs attaques distribuées et coordonnées. Les bots (robots) doivent signaler au botmaster l'état de l'action une fois ces tâches exécutées et ce reporting est effectué via le serveur C&C [30] [31] [32]. Le botnet fonctionne de manière malveillante en tant que groupe de machines infectées utilisant une structure de canaux C&C [33].

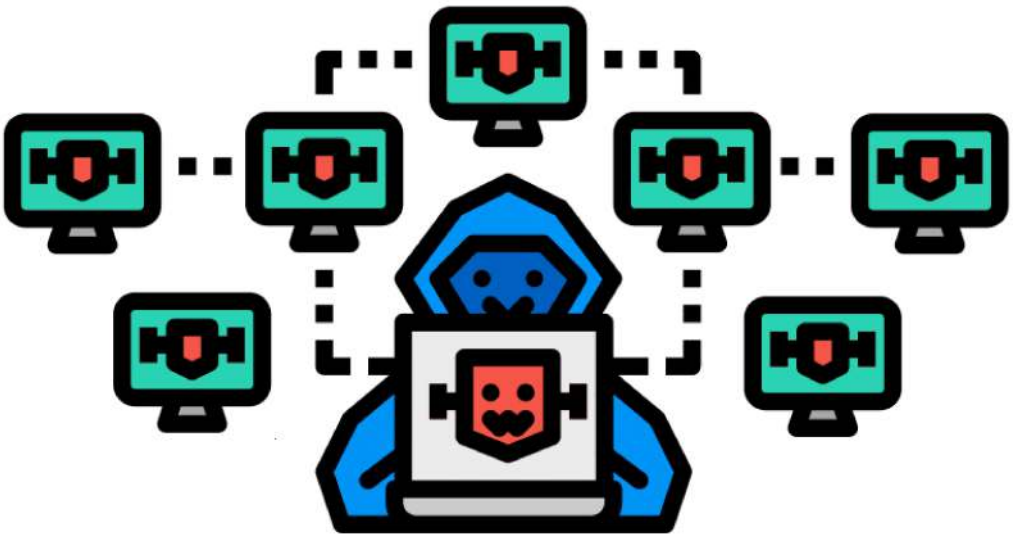

Figure 1.5 – Botnet [5]

Il existe d'autres type de malware dont les plus connues sont :

1. Les Virus : Un virus est un logiciel capable de s'installer sur un ordinateur à l'insu de son utilisateur légitime. Le terme virus est réservé aux logiciels qui se comportent ainsi avec un but malveillant.

Figure 1.6 – Les Virus [6]

2. Les Vers : Un ver (worm) est une variété de virus qui se propage par le réseau. Il peut s'agir d'un bot . En fait aujourd'hui la confusion entre les virus et les vers est presque totale, du fait qu'avant les virus n'étaient pas des vers (ils ne se propageaient pas par le réseau) et les vers n'étaient pas des virus (ils ne se reproduisaient pas), mais de nos jours les virus se comportent comme des vers et vis-versa.
3. Logiciel Espion : Un logiciel espion (spyware), comme son nom l'indique, collecte à l'insu de l'utilisateur légitime des informations au sein du système où il est installé, et les communique à un agent extérieur, par exemple au moyen d'une porte dérobée.
4. Porte Dérobée : Une porte dérobée (backdoor) est un logiciel de communication caché, installé par exemple par un virus ou par un cheval de Troie, qui donne à un agresseur extérieur accès à l'ordinateur victime, par le réseau.
5. Cheval de Troie : Un cheval de Troie (Trojan horse) est un logiciel qui se présente sous un jour honnête, utile ou agréable, et qui une fois installé sur un ordinateur y effectue des actions cachées et pernicieuses.

Figure 1.7 – Le Cheval de Troie [7]

1.2.5 Aperçu général du botnet

Les botnets peuvent être classer en deux catégories, les botnets qui nécessite une architecture centralisée avec des clients et des serveurs de contrôle de commande(Command & control server) et les p2p botnets qui utilisent une architecture décentralisée ce qui les rendent plus robustes et plus difficile à détecter et à contrer.

Le réseau Botnet se compose de bots (robots), d'un canal de commande et de contrôle (C&C) et d'un Botmaster. Une machine peut être infectée par utilisation des plusieurs techniques telles que des portes dérobées, des chevaux de Troie, ou des pièces jointes aux mails(spams). Le Botmaster est l'administrateur du réseau Botnet qui utilise C&C pour exécuter des activités malveillantes, telles que la génération de pages Web de phishing, le vol de données confidentielles d'utilisateurs, l'envoi d'énormes quantité de spams et l'exécution d'une attaque DDoS [34] [30] [31] [35] [36].

1.2.6 Historique du botnet

Les vers informatiques qui se propagent entre les machines en réseau est apparu sur le réseau d'agences de projets de recherche avancée (ARPANET) en 1971. Au début des années 1990, l'attention médiatique accordée à des vers célèbres tels que *Father Christmas Worm* et *Worms Against Nuclear Killers*, le développement d'outils d'accès à distance et l'essor des machines connectées à Internet ont donné naissance aux premiers réseaux de zombies. Le premier bot IRC non malveillant, Eggdrop, publié en 1993, a été développé pour gérer et protéger un canal de discussion.

Peu après la publication, des robots malveillants similaires ont commencé à attaquer les utilisateurs et les serveurs IRC, les attaques par déni de service suivi par la première attaque DDoS à grande échelle survenue en 1999, lorsque le réseau informatique de l'Université du Minnesota a été désactivé pendant deux jours [37]. En 1998, GTbot est devenu un Botnet IRC malveillant largement connu et utilisé pour lancer des attaques DDoS. Agobot de 2002, une collection extrêmement robuste de vers capables du reniflage de paquets, du keylogging, de l'installation de rootkits et des attaques DDoS a été l'un des premiers botnets à grande échelle à nécessiter peu de capacité de programmation. L'Agobot est couramment utilisé comme référence pour la montée des botnets en tant que menace importante [38].

Les années 1990 ont également vu la croissance des réseaux distribués non malveillants projets informatiques dont Great Internet Mersenne Prime Search (GIMPS), considéré comme l'un des plus grands projets de recherche en calcul distribué (distributed computing). En 1996, GIMPS a commencé avec une collaboration de bénévoles, et est distribué gratuitement pour la recherche des nombres premiers de Mersenne qui sont de la forme [39] :

$$M^n = 2^n - 1 \tag{1.1}$$

.

Dans cette équation, "M" représente le nombre de Mersenne et "n" est un nombre entier qui détermine le nombre de Mersenne à tester.

En 1999, SETI@home, influencé par GIMPS, devient un autre projet de recherche de calcul distribué (distributed computing) à grande échelle avec le double objectif de l'utilisation de méthodes de recherche pour rechercher la vie extraterrestre et faire progresser la collaboration de bénévoles [40]. En outre, en 1999, Napster connu comme un pionnier du peer-to-peer partage de fichiers entre pairs dans le but de partager de la musique gratuitement au format MP3 [41].

Les activités malveillantes les plus récentes incluent le Dyn d'octobre 2016 cyber-attaque, une attaque DDoS massive qui a perturbé le service Internet dans une grande partie des États-Unis et de l'Europe. Initialement considérée comme une attaque terroriste, la panne a été attribuée au botnet Mirai. Il convient de noter que Mirai a créé bien plus de dégâts que son auteur ne l'avait prévu lorsqu'il a commencé à infecter les appareils IoT [42].

Dans début 2017, Twitter a découvert 350 000 faux comptes qui faisaient partie d'un botnet. En 2018, Facebook avait découvert plus de 100 000 faux comptes et groupes qui ont été utilisés pour influencer le résultat des élections américaines de 2016 [43]. Les préoccupations actuel et dans un avenir proche incluent, l'incroyable augmentation des appareils IoT, dont la sécurité ne pas une priorité pour les constructeurs. Donc ces appareils sont plus vulnérable et feront agrandir des réseaux de zombies(botnets) pour des futures activités malveillants [42].

1.3 Présentation des botnets P2P : caractéristiques, fonctionnement et impacts

Les botnets p2p sont un ensemble des machines piratées et contrôlées à distance par des pirates(botmasters). Ces botnet peuvent comporter des milliers de machines appelées de zombies qui utilisent le protocole p2p(peer-to-peer) pour communiquer.

Le partage d'information se fait de manière partiellement ou totalement décentraliser, chaque machine est à la fois un client et un serveur 1.9 ce qui permet au botmaster d'envoyer des commandes de contrôle de n'importe quelle machine du réseau botnet, ce qui rendent les botnets P2P extrêmement robustes, contrairement à l'architecture client/serveur 1.8 qui est centralisée et dont le point faible est le serveur de contrôle de commande(C&C).

Le Bot P2P, après avoir infecté la machine, essaiera de trouver d'autres pairs afin de rejoindre le réseau Botnet. Il établira donc un grand nombre de connexions afin de trouver n'importe quel pair encore en vie pour recevoir des mises à jour et donner des notes à l'autre peer dans le même réseau. Les robots P2P après la phase d'infection doivent communiquer avec leurs homologues pour recevoir des mises à jour ou plus d'instructions de la part des homologues de contournement du Botmaster, le nouveau homologue (hôte infecté) doit régulièrement s'appuyer pour mettre à jour l'état et recueillir des informations sur l'état du réseau [44].

Au cours de la dernière décennie les botnets en générale et le P2P botnets en particulier ont constitué la plus grande menace qui puisse exister pour les grandes organismes et les gouvernements. les botnets sont utilisés par des pirates ou loués à des personnes désirant avoir des activités frauduleuses (spamming, proxy, ftp , déni de service distribué

DDoS) [28]. Aujourd'hui la plus grandes craintes pour toutes entreprise est ces attaques DDoS qui ne nécessitent pas une compétence accrue et qui peuvent faire des dégâts considérables et de perte financiers énorme.

Les botnets sont divisés en quatre classes, dont les botnets IRC (Internet Relay Chat), HTTP, P2P et hybrides [31] en termes de leurs environnements de développement. Les botnets IRC et HTTP sont des botnets centralisés tandis que les botnets P2P sont décentralisés. Il existe trois types d'architectures de commande et de contrôle des botnets : centralisée, décentralisée et hybride, en fonction de la manière dont la communication est mise en œuvre [45].

1. **C&C centralisé :** Dans une approche de commande et de contrôle centralisée, les zombies ou robots sont connectés à un serveur C&C central pour recevoir des instructions et des mises à jour. Le serveur C&C peut fournir des services pour enregistrer les robots disponibles et suivre leurs activités. Pour exercer le contrôle sur les bots et leur donner des commandes, un botmaster doit être connecté au serveur C&C. [46] [47]. Le mécanisme C&C centralisé est divisé en types IRC et HTTP en fonction des protocoles de communication qu'ils utilisent pour établir leur connexion.

   • **Basé sur IRC :** IRC ou Internet Relay Chat est un système utilisé par les utilisateurs d'ordinateurs pour communiquer en ligne ou discuter en temps réel [48]. Chaque bot se connecte au serveur et au canal IRC sélectionnés par un botmaster et attend les commandes. Dans cette configuration, le botmaster établit une communication en temps réel avec tous les robots connectés et les contrôle. Les robots IRC suivent l'approche PUSH ce qui signifie que lorsqu'un robot IRC se connecte à un canal sélectionné, il reste en mode connexion [47].

   • **Basé sur HTTP :** La commande et le contrôle HTTP est une nouvelle technique qui permet aux botmasters de contrôler leurs robots en utilisant le protocole HTTP [48]. Dans cette technique, les robots utilisent une URL ou une adresse IP spécifique définie par le botmaster, pour se connecter à un serveur Web spécifique, qui joue le rôle de serveur C&C. Les bots HTTP adoptent l'approche PULL. Ils ne restent pas en mode connexion après avoir établi pour la première fois la connexion au serveur C&C. Les botmasters publient les commandes sur certains serveurs Web et les robots visitent périodiquement ces serveurs Web pour se mettre à jour ou

obtenir de nouvelles commandes. [49] [50].

2. **C&C décentralisé :** L'architecture de commande et de contrôle décentralisée est basée sur le modèle de réseau peer-to-peer (P2P). Dans ce modèle, un ordinateur infecté agit à la fois comme un bot et comme un serveur C&C [51]. En effet dans les botnets P2P, au lieu d'avoir un serveur C&C central, chaque bot agit comme un serveur pour transmettre les commandes à ses bots voisins. Le botmaster envoie des commandes à un ou plusieurs robots, et les robots qui reçoivent les commandes les transmettent à d'autres robots. [45] [50].

3. **C&C hybride :** Afin de tirer parti des avantages de chaque modèle C&C, les différents protocoles et architectures sont utilisés pour former une approche hybride. Par exemple, les botnets HTTP2P [51] communiquent avec le protocole HTTP pour échapper aux pare-feu sur une structure P2P afin d'éliminer les inconvénients traditionnels des serveurs C&C centraux. L'approche hybride ne se limite pas à l'utilisation de certains services ou architectures ; en fait, les botmasters peuvent utiliser n'importe quel protocole applicable pour mettre en œuvre ce modèle. Par exemple, l'AHP2P [52] est un botnet P2P hybride avancé qui utilise la technologie web 2.0 pour cacher ses communications dans des sites Web sociaux.

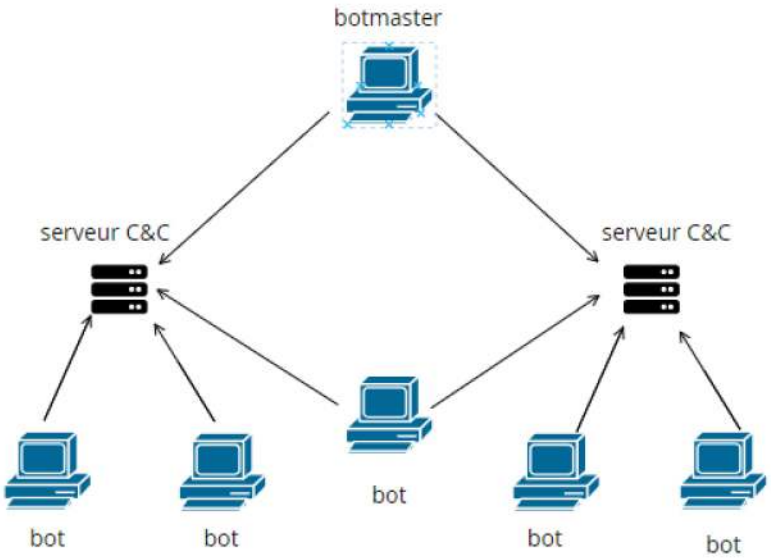

Figure 1.8 – Architecture botnet centralisé

C'est qui fait la résilience des botnets P2P contrairement au botnet normale c'est l'absence d'un serveur de contrôle de commandes (C&C), ça fait que chaque bot est à la

fois un serveur C&C et un bot, donc le botmaster a la capacité d'envoyer des commandes aux autres bots à partir de n'importe quelle machine du réseau.

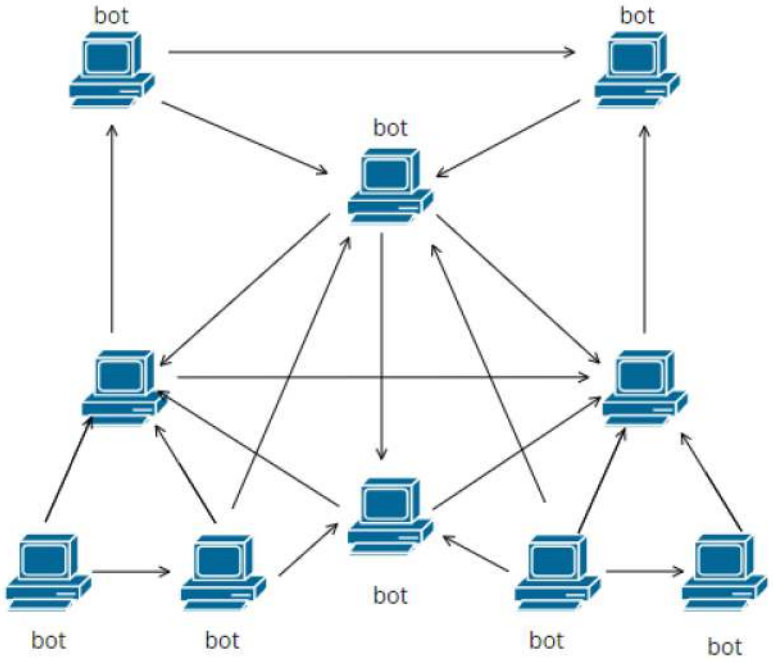

Figure 1.9 – Architecture p2p botnet

Les botnets peuvent être de différentes tailles ou structures mais, en général, ils passent par les mêmes étapes dans leur cycle de vie [53] [54]. La figure 1.10 représente la vue générale du cycle de vie d'un botnet.

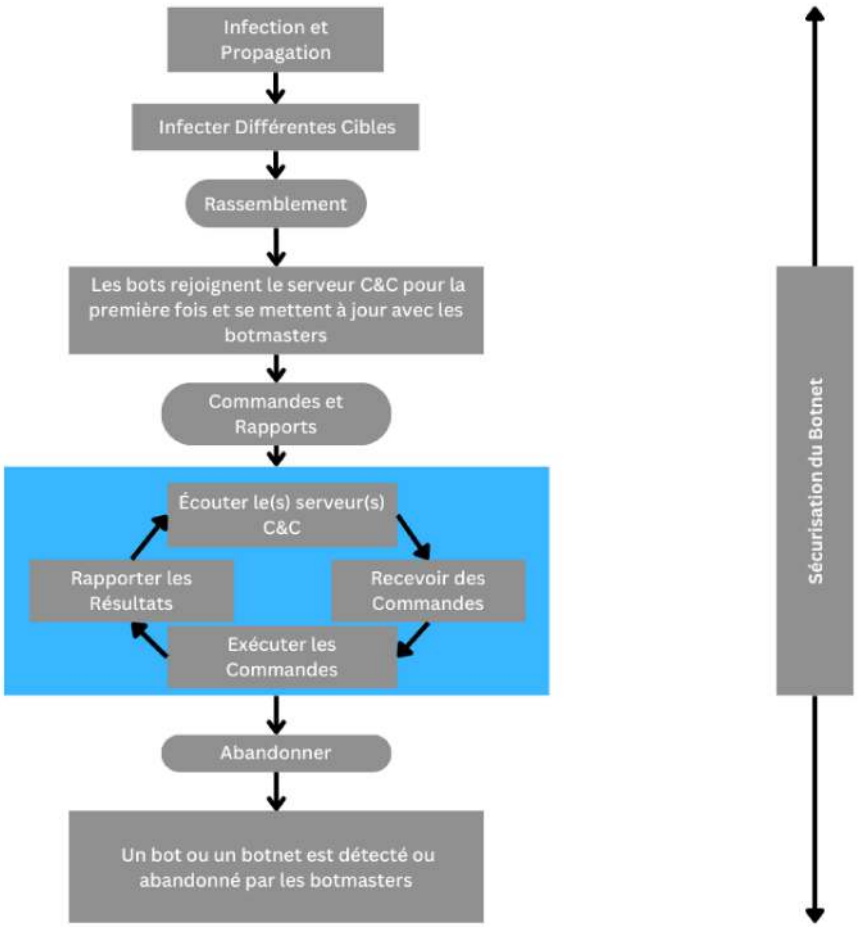

Figure 1.10 – Un schéma de cycle de vie de botnet [8]

1. **Infection et propagation :** Le cycle de vie d'un botnet commence par le processus d'infection au cours duquel les botmasters utilisent différentes méthodes et techniques pour infecter de nouvelles cibles, par exemple des ordinateurs et des appareils mobiles, et les convertir en robots [45]. Les codes infectés joints aux spams ou aux messages instantanés, aux URL malveillantes, aux réseaux de partage de fichiers P2P et même à d'autres botnets peuvent être répertoriés comme exemples de vecteurs pour propager les robots dans les appareils cibles [54]. Les cibles les plus courantes des botmasters sont les ordinateurs moins surveillés dotés d'une connectivité à large bande passante, les serveurs universitaires et les ordinateurs personnels. En général, les botmasters profitent de ceux qui ont peu de connaissances ou de connaissances en matière de sécurité réseau pour obtenir un accès non autorisé à leurs appareils et maintenir leurs robots en vie pendant une longue période sans être détectés [55].
2. **Ralliement :** Il s'agit de la première fois que des robots se connectent au serveur

C&C pour montrer au botmaster qu'il a déjà établi avec succès un zombie [46]. De plus, les robots reçoivent des mises à jour avec des informations essentielles telles que la liste des adresses IP relatives du serveur C&C [56].

3. **Commandes et rapports :** Durant cette étape, les robots écoutent les serveurs C&C ou s'y connectent périodiquement pour recevoir de nouvelles commandes du botmaster. Une nouvelle commande, lorsqu'elle est détectée par les robots, est traitée comme une commande ; ils exécutent la commande et les résultats sont rapportés au serveur C&C ; les robots attendent alors les nouvelles commandes [53] [45].
4. **Abandonner :** Lorsqu'un bot n'est plus utilisable, par exemple si sa bande passante est trop faible ou si le botmaster décide que le bot en question ne convient plus, il peut être abandonné par le botmaster. Dans le cas où un seul bot est désactivé, le botnet est toujours disponible. Un botnet est entièrement détruit lorsque tous ses bots sont détectés ou abandonnés ou lorsque les serveurs C&C sont détectés et bloqués [53] [55].
5. **Sécuriser le botnet :** L'effort constant pour assurer la sécurité de l'ensemble du botnet est l'un des problèmes importants du cycle de vie d'un botnet. Les botnets et les robots sont de nature dynamique et flexible. Ils sont continuellement mis à jour et leurs codes changent de jour en jour. De plus, les botmasters essaient toujours différentes techniques pour protéger leurs robots des solutions de détection existantes [57].

1.4 Approches traditionnelles de détection des botnets P2P

Les botnets pair à pair (P2P) sont difficiles à détecter et à contre en raison de leur nature décentralisée et de l'absence d'un point central de contrôle (C&C). Cependant, plusieurs approches ont été utilisées avec plus ou moins d'efficacité pour tenter de les détecter et de les contrecarrer. Il existe plusieurs méthodes et techniques qui ont été utilisé pour suivre les activités des réseaux de zombies et les détecter, comme l'analyse du trafic réseau, la détection basée sur les signatures, les pots de miel (honeypots), l'analyse du trafic DNS, l'exploration de données (data mining) et l'analyse comportementale (par exemple, active et passive). Voici ces approches en détail :

1. **Analyse du trafic réseau :** L'analyse du trafic réseau est une approche tradition-

nelle et efficace pour détecter les botnets P2P. Elle consiste à identifier les modèles de trafic caractéristiques des botnets P2P en surveillant les ports, les protocoles et le contenu des paquets réseau. L'un des article le plus cité pour l'utilisation de cette méthode est [58] Cet article propose une méthode de détection de botnets P2P basée sur l'analyse du comportement de communication. La méthode utilise des caractéristiques du comportement de communication des bots pour identifier les botnets P2P. La méthode est basée sur l'observation que les bots P2P se comportement différemment des ordinateurs normaux. Les caractéristiques de communication des bots P2P incluent, un grand nombre de connexions entre les bots ,des modèles de communication spécifiques comme des protocoles et un trafic anormal.

2. **La détection basé sur les signatures :** La signature fait référence aux modèles ou caractéristiques connus de menaces d'intrusion dans les systèmes informatiques. En analysant et en comparant ces modèles ou caractéristiques, il est possible pour distinguer les activités malveillantes des activités normales. Un peu d'études ont proposé la méthode de détection des botnets basée sur les signatures [59] [60]. Cependant, la détection par signature est rarement utilisé pour la détection des botnets car cette méthode ne peut pas identifier de nouveaux comportements, modèles ou caractéristiques. Cette méthode est basé sur une simple comparaison des informations collectées avec les caractéristiques prédéfinies des bots existants. Ainsi, cette méthode est bonne pour détecter les robots bien connus, mais moins significatif pour détecter les robots nouveaux et zero-day [61].

Le système de détection de P2P botnet basé sur les signatures repose sur la capacité d'identifier avec succès le motif de signature du trafic traversant le réseau. Le principe de fonctionnement de cette approche basée sur les signatures est similaire à celui des programmes antivirus. Plusieurs études se sont focalisées sur cette méthode comme dans cet article [62] ou les auteurs ont mis en place un système de détection et prévention de botnet et P2P botnet base sur la signature N-EDPS (Network-based Detection and Prevention systems of botnet).

Lorsque la détection de botnet se produit au niveau du réseau, l'approche basée sur les signatures peut être un port ou une adresse IP d'un bot connu. Par exemple, le trafic provenant d'adresses IP suspectées et ouvrant certains ports, comme le port 6667 pour le protocole Internet Relay Chat (IRC) [63].

3. **Data mining (l'exploration de données) :** L'exploration de données est le processus de découverte de modèles, de tendances et d'informations à partir de grands ensembles de données. Cela implique l'utilisation de diverses techniques et algorithmes pour extraire des informations précieuses à partir de données brutes. Les algorithmes d'exploration de données permettent d'automatiser la détection de caractéristiques à partir d'une grande quantité de données, ce que les heuristiques conventionnelles et les méthodes basées sur les signatures ne pourraient pas appliquer.

4. **L'analyse comportementale :** Afin de fournir une approche efficace pour analyser et détecter les botnets, une analyse comportementale est proposée par certaines études. Cette méthode recherche les modèles anormaux dans le trafic réseau et non le contenu des informations transmises [64]. L'un des principal avantage de cette technique est la capacité de détecter les menaces inconnues des botnets [65]. L'analyse comportementale se présente sous deux formes : l'analyse du comportement des attaques (active) et analyse du comportement opérationnel (passive). [45]

1.5 Introduction aux méthodes de deep learning en sécurité informatique

L'intelligence artificielle en générale et le deep learning en particulier sont en train de révolutionner tout les secteurs de la technologie, ces dernières années grâce à la disponibilité des quantités énormes des données et l'augmentation de la puissance de calcul, on peut entraîner des modèles des réseaux des neurones artificiels complexes pour apprendre à partir de données en un temps record.

Le deep learning est utilisé dans le domaine de la sécurité informatique pour principalement la détection des intrusions, analyse des malwares et la protection des données contre des accès non autorisés. Plusieurs algorithmes de deep learning sont utilisés dans la sécurité informatique notamment les réseaux de neurones convolutionnels (CNN), les réseaux de neurones récurrents (RNN), et les perceptions multi-couche (MLPs). Le deep learning offre plusieurs avantages par rapport aux méthodes traditionnelles de sécurités informatiques.

— **Détection d'anomalies et de menaces inconnues :** Les modèles de deep lear-

ning ont la capacité d'apprendre à partir de données et de détecter des modèles complexes, ce qui les rend très efficaces pour identifier des anomalies et des menaces inconnues qui échapperaient aux méthodes de détection traditionnelles basées sur des signatures.

— **Analyse de grands volumes de données :** Avec la quantité massive de données générées par les systèmes et les réseaux informatiques, le deep learning est particulièrement utile pour analyser et tirer des informations pertinentes de ces grandes quantités de données de manière automatisée.

— **Performances accrues** : Les algorithmes de deep learning peuvent souvent atteindre de meilleures performances en termes de précision et de rappel que les méthodes traditionnelles, en particulier pour des tâches complexes comme la détection de malwares, l'analyse de journaux ou la reconnaissance d'images malveillantes.

— **Automatisation et réduction des efforts manuels :** Le deep learning peut automatiser de nombreuses tâches de sécurité qui nécessitaient auparavant une analyse manuelle intensive, comme l'examen de code malveillant ou l'analyse de journaux, réduisant ainsi la charge de travail et les erreurs humaines tout en réduisant le coût de la sécurité.

1.6 Revue de la littérature sur la détection de botnets P2P utilisant des méthodes de deep learning

Les méthodes traditionnelles de détection des botnets P2P, ont été longtemps utilisées dans la sécurité informatiques malgré leurs limites. Aujourd'hui le deep learning, offre une approche différente pour comble et surmonter ces limites. C'est pourquoi plusieurs études ont été réalisés pour la détection de P2P botnet avec des algorithme de deep learning. Parmi ces études Plusieurs ont exploré l'utilisation de réseaux de neurones profonds pour détecter les botnets P2P.

Une approche courante consiste à entraîner des modèles de deep learning (CNN, RNN, MLP) sur des données de trafic réseau contenant des trafics normaux et des trafics des botnets p2p, pour que le modèle apprend à distinguer le trafic légitime du trafic lié aux botnets.

On distingue plusieurs études utilisant le deep learning pour la détection de p2p bot parmi lesquels : "P2P BOT DETECTION USING DEEP LEARNING WITH TRAFFIC REDUCTION SCHEMA" [44] cette étude parût en 2020 est basée sur l'utilisation de deep learning pour la détection de P2P botnet en utilisant ISOT et ISCX dataset avec une précision de 98,2%.

Dans cet article [66] PeerAmbush présente une nouvelle méthode pour identifier les botnets P2P, en utilisant le deep learning précisément la technique d'apprentissage profond Multi-Layer Perceptron (MLP), une rupture par rapport aux approches conventionnelles principalement basées sur des méthodes machine learning. En plus de tirer parti du MLP, PeerAmbush relève le défi de la détection des botnets P2P avec un ensemble réduit de caractéristiques par rapport aux méthodes existantes, grâce à la méthode Best First Union (BFU), ce qui leurs a permis d'avoir une précision de 99,9 %.

1.7 Conclusion

Dans ce chapitre nous avons vue que l'évolution rapide de la technologie de l'information a transformé notre façon de vivre et de travailler, mais elle a également amplifié les risques liés à la sécurité des données. Face à la menace croissante des cyber-attaques, renforcer la sécurité des données est devenu une priorité absolue. Cela nécessite une approche multi-facette, impliquant l'adoption de technologies avancées comme le deep learning et la mise en œuvre de bonnes pratiques de sécurité. Pour travailler à identifier et contrer les menaces potentielles, ainsi nous pouvons mieux protéger nos informations personnelles, nos infrastructures critiques et notre société dans son ensemble contre les cyber-menaces qui ne cessent d'augmenter.

Chapitre 2

Fondements théoriques

2.1 Introduction

L'augmentation de la puissance de calcul et la disponibilité des quantités énormes de données de ce dernières années a permis un développement rapide dans le domaine de l'intelligence artificielle en générale et du deep learning en particulier. Si les algorithmes de deep learning ne datent pas hier, leurs utilisation massive dans des domaines divers et variés est récente. On assiste aujourd'hui à l'utilisation du deep learning dans presque tous les domaines des sciences modères, que se soit en science de santé, en science technique ou en science sociale.

Cet intérêt porté au deep learning est dû à sa capacité à ressouder des problèmes complexes avec une meilleur efficacité que les méthodes utilisées auparavant. Dans ce chapitre nous allons voir les principes de base du deep learning ainsi que les architectures des réseaux des neurones les plus connues et les plus utilisées de nos jours. Ensuite nous allons aborder le pré-traitement des données pour la détection de botnets et enfin les métriques d'évaluation de la performance des modèles de deep learning.

2.2 Botnets P2P

Les botnets peer-to-peer (P2P) représentent une menace sophistiquée et évolutive en matière de cyber-sécurité. Voici un aperçu plus approfondi de leur sophistication et de la manière dont l'apprentissage profond peut être exploité pour leur détection :

2.2.1 Sophistication des Botnets P2P

1. **Décentralisation :**
   — **Structure :** Contrairement aux botnets centralisés traditionnels, les botnets P2P répartissent les fonctionnalités de commande et de contrôle (C&C) parmi les nœuds infectés (bots). Cette décentralisation les rendent plus résilients aux tentatives de démantèlement, car il n'y a pas de point de défaillance unique.
   — **Redondance :** Les bots peuvent communiquer avec plusieurs pairs, assurant ainsi que le réseau reste opérationnel même si certains bots sont supprimés ou perturbés.

2. **Techniques d'évasion :**
   — **Chiffrement :** Les botnets P2P utilisent souvent le chiffrement pour sécuriser leurs communications, rendant difficile l'analyse de leur trafic par les systèmes de détection traditionnels.
   — **Comportement dynamique :** Ces botnets peuvent changer dynamiquement leurs modèles de communication et protocoles, s'adaptant pour échapper aux outils de détection.
   — **Fast Flux :** Cette technique consiste à changer rapidement les adresses IP associées aux noms de domaine du botnet, rendant plus difficiles les efforts de suivi et de blocage.

3. **Propagation avancée :**
   — **Auto-mise à jour :** Les bots peuvent recevoir des mises à jour de leurs pairs, permettant au botnet de s'adapter rapidement et d'améliorer ses capacités.
   — **Découverte des pairs :** Les botnets P2P disposent d'algorithmes sophistiqués pour découvrir de nouveaux pairs, assurant une croissance et une maintenance robustes du réseau.

Les botnets P2P sont de plus en plus sophistiqués, utilisant la décentralisation, le chiffrement, des comportements dynamiques et des techniques de propagation avancées pour échapper à la détection. L'apprentissage profond offre des outils puissants pour détecter ces botnets grâce à l'extraction automatique de caractéristiques, la détection d'anomalies, la classification et le traitement en temps réel. Cependant, une adaptation continue et une formation robuste des modèles sont essentielles pour rester en avance sur les tactiques évolutives des botnets.

2.3 Principes de base du deep learning

Le deep learning, ou l'apprentissage profond, est une sous branche de Machine learning qui a connu un essor remarquable ces dernières années. Il s'agit d'un ensemble des techniques d'apprentissage automatique inspirées du fonctionnement des réseaux de neurones biologiques comme montre dans l'image 2.2. Le deep learning ou apprentissage profond, regroupe actuellement les méthodes les plus efficaces et les plus performantes appliquées dans la communauté de l'apprentissage automatique.

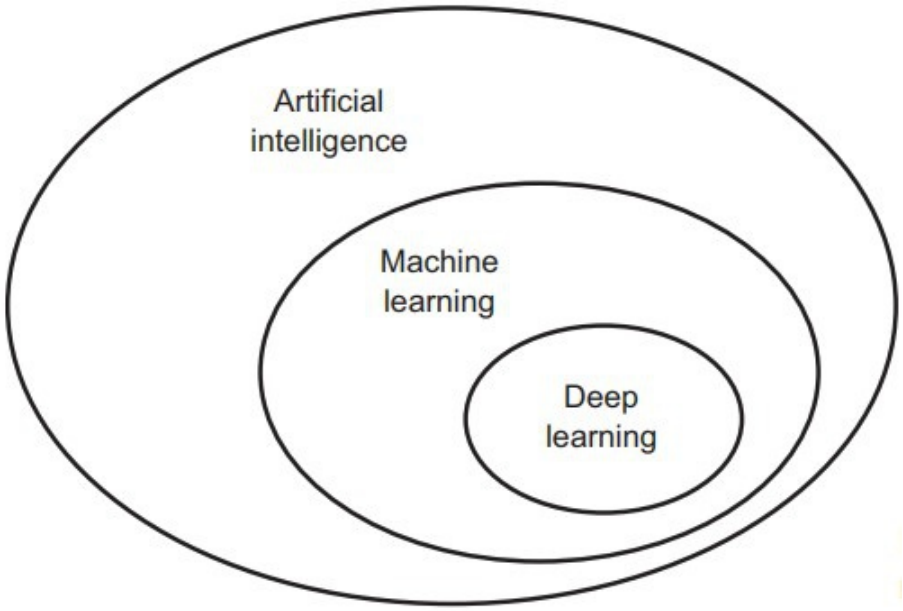

Figure 2.1 – L'intelligence Artificielle, l'apprentissage automatique et l'apprentissage profond [9]

Un neurone biologique, ou cellule nerveuse, est une cellule électriquement excitable qui communique avec d'autres cellules via des connexions spécialisées appelées synapses. C'est le composant principal du tissu nerveux. Les neurones sont généralement classés en trois types en fonction de leur fonction. Les neurones sensoriels répondent à des stimuli tels que le toucher, le son ou la lumière qui affectent les cellules des organes sensoriels et envoient des signaux à la moelle épinière ou au cerveau. Les moto-neurones reçoivent des signaux du cerveau et de la moelle épinière pour tout contrôler, des contractions musculaires à la production glandulaire. Les neurones relais ou inter-neurones connectent les neurones à d'autres neurones dans la même région du cerveau ou de la moelle épinière. Un groupe de neurones connectés s'appelle un circuit neuronal.

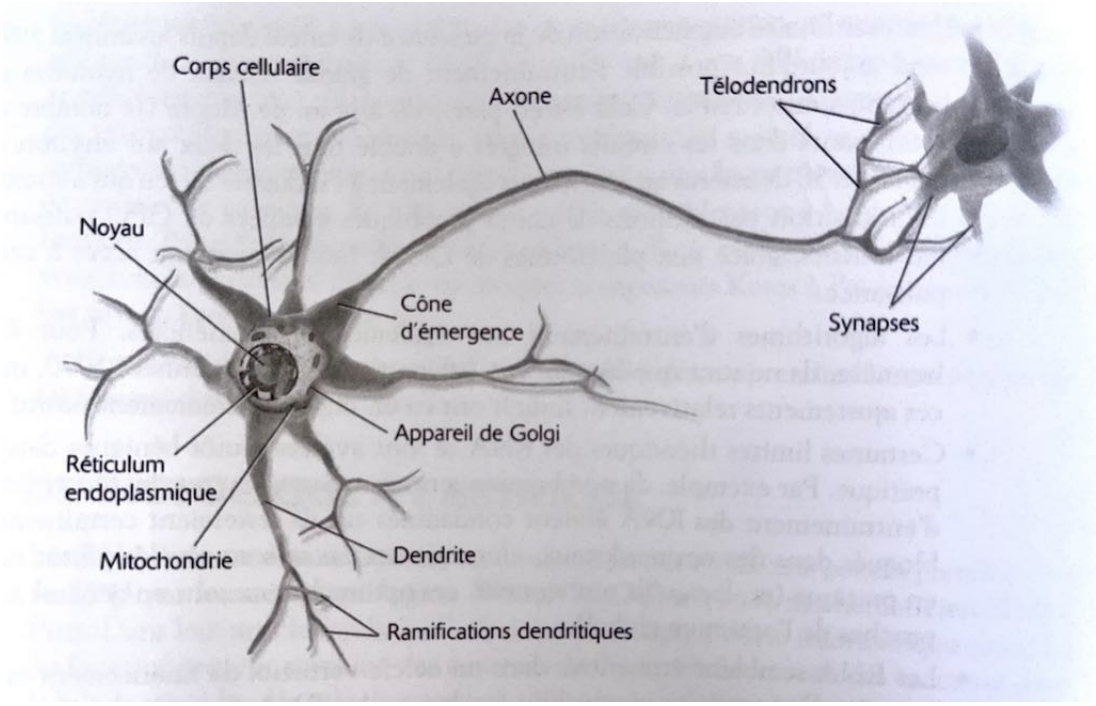

FIGURE 2.2 – Un neurone biologique [10]

Un neurone artificiel (formel) représente le neurone biologique de façon mathématique et informatique. Il s'agit d'une structure en arbre, où les entrées correspondent aux dendrites et l'excitation se fait moyennant l'axone. Le neurone artificiel calcule la somme pondérée des entrées, puis appliques à cette somme une fonction d'activation qui indique la sortie finale du neurone ou bien la classe. Les entrées qui représentent les caractéristiques de données sont assimilées.

Les entrées **x** jouent le rôle des dendrites et apportent un signal qui va être traité et ajusté par les poids qui sont des facteurs significatifs dans le schéma du neurone artificiel. Chaque entrée est multipliée par un poids **w** remplaçant ainsi des synapses de neurone biologique. La somme de ces entrées multipliée par leurs poids respectifs est injecté dans la cellule excitatrice qui est la fonction d'activation **h**. La sortie **y** permet d'afficher la classe ou le neurone gagnant. Cette sortie est calculée à partir des entrées via l'équation :

$$y = h(w_0 + \sum(x_i * w_i)) \tag{2.1}$$

Alors le deep learning est un apprentissage qui consiste à construire des modèles prédictifs à partir de réseaux des neurones profondes tels que les réseaux de neurones convolutif (CNN) et les réseaux de neurones récurrents (RNN) etc. Le deep learning permet

de modéliser des données complexes et d'extraire automatiquement les caractéristiques pertinentes, contrairement aux approches d'apprentissage machine traditionnelles qui nécessitent une extraction manuelle des caractéristiques.

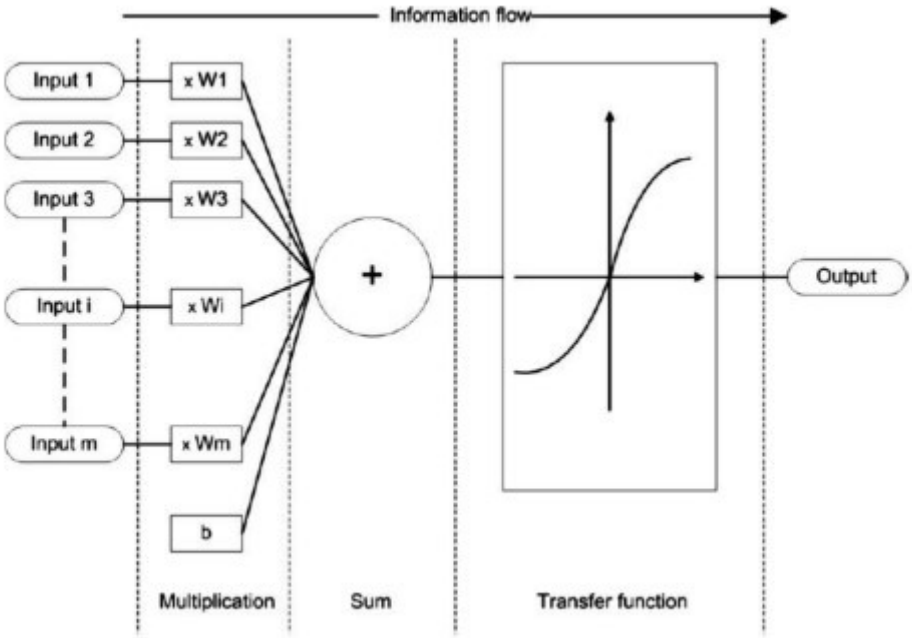

FIGURE 2.3 – Un neurone artificiel [11]

Le parallèle entre un neurone biologique et un neurone informatique est illustré par la figure 2.4. Un modèle constitué d'un seul neurone est appelé perceptron.

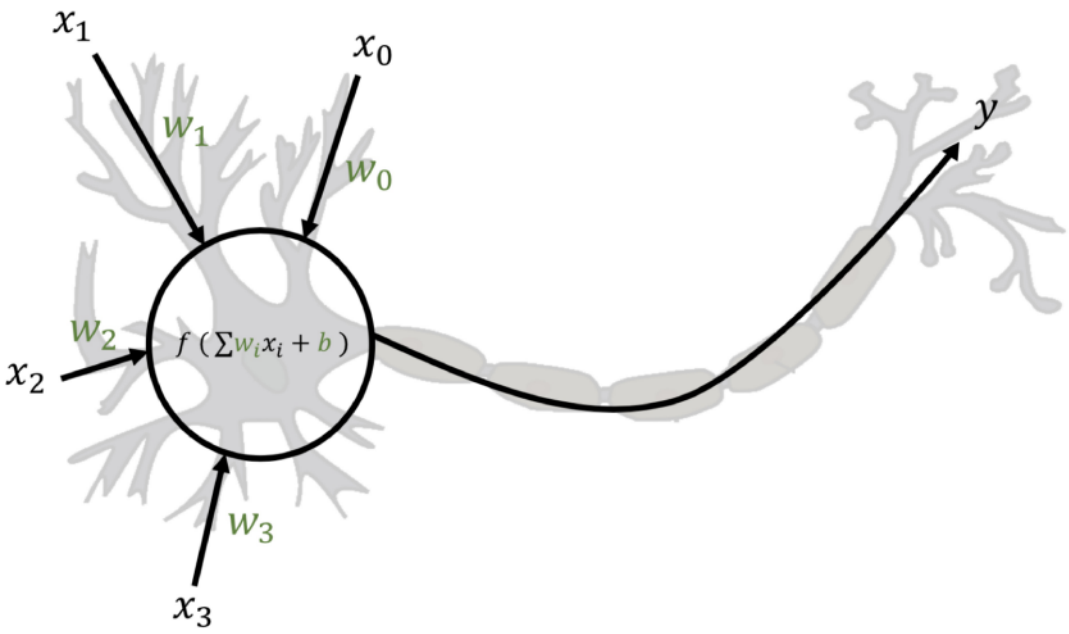

Figure 2.4 – Un schéma d'un neurone informatique superposé à un schéma d'un neurone biologique [12]

2.3.1 Fonction d'activation :

Une fonction d'activation est une équation mathématique qui détermine la sortie de chaque élément (perceptron ou neurone) du réseau neuronal. Il prend l'entrée de chaque neurone et la transforme en sortie, généralement entre 1 et 0 ou entre -1 et 1. Elle peut être définie comme la force ou l'effort supplémentaire appliqué sur l'entrée pour obtenir une sortie exacte. Dans le réseaux de neurones artificiel - ANN (Artificial Neural Network), nous pouvons également appliquer des fonctions d'activation sur l'entrée pour obtenir la sortie exacte [11]. Une fonction d'activation est défini dans l'espace réel permettant d'activer un neurone selon la formule :

$$y = f(x) \tag{2.2}$$

Voici quelques fonctions d'activation :

1. **Fonction d'activation à seuil :** L'exemple de cette fonction est l'activation d'une diode où (y=0, la diode est éteinte) et (y=1, la diode est allumée). Cette fonction est définie par la figure2.5 :

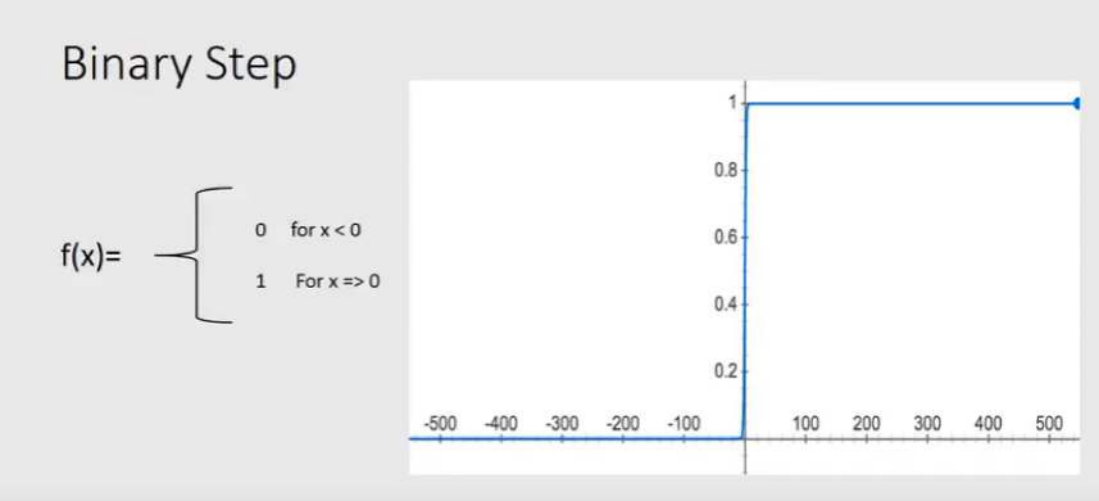

Figure 2.5 – Fonction d'activation à seuil [13]

2. **Fonction ReLu (Rectified Linear Unit) :** Il s'agit d'une fonction permettant une rectification linéaire des valeurs. Bien qu'elle donne l'impression d'une fonction linéaire, ReLU a une fonction dérivée et permet la rétro-propagation tout en la rendant simultanément efficace sur le plan informatique. Le principal problème ici est que la fonction ReLU n'active pas tous les neurones en même temps. Les neurones ne seront désactivés que si le résultat de la transformation linéaire est inférieur à 0.

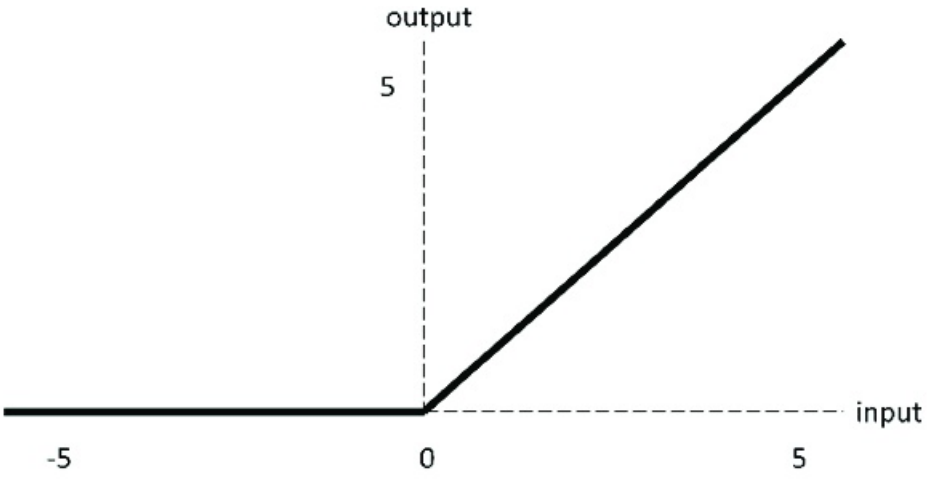

Figure 2.6 – Fonction d'activation ReLU [14]

3. **Fonction sigmoïde ou logistique :** Cette fonction prend n'importe quelle valeur réelle en entrée et génère des valeurs comprises entre 0 et 1. Plus l'entrée **x**

est grande (plus positive), plus la valeur de sortie **f(x)** sera proche de 1, tandis que plus l'entrée est petite (plus négative), plus la valeur de la sortie sera à 0. Elle est utilisée spécialement pour la classification binaire. Cette fonction est donnée par l'équation :

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

x : est l'entrée

f(x) : est la sortie

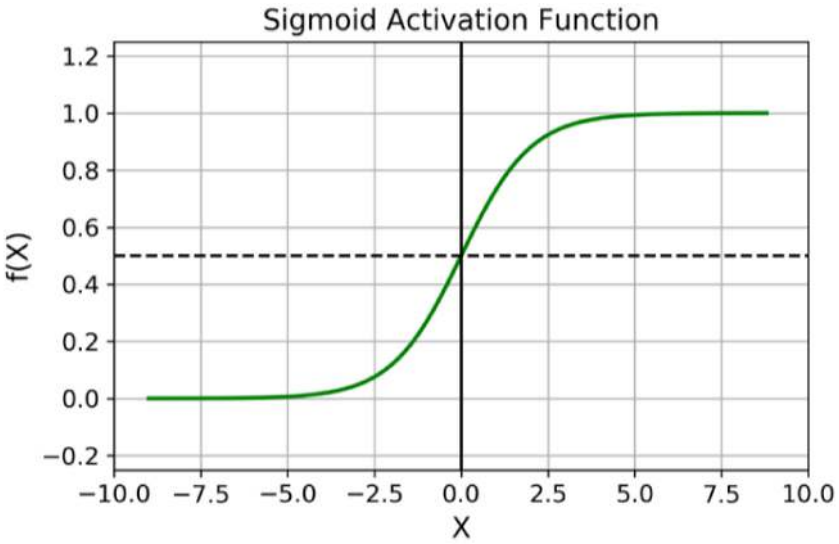

Figure 2.7 – Fonction sigmoïde [15]

4. **Fonction tangente hyperbolique :** La fonction Tanh est très similaire à la fonction d'activation sigmoïde/logistique et a la même forme en S avec une différence de plage de sortie de -1 à 1. Dans Tanh, plus l'entrée **x** est grande (plus positive), plus la valeur de sortie **f(x)** sera proche de 1, tandis que plus l'entrée est petite (plus négative), plus la sortie sera proche de -1. Elle est utilisée pour le cas de multi-classe. Cette fonction est représentée par l'équation :

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.4}$$

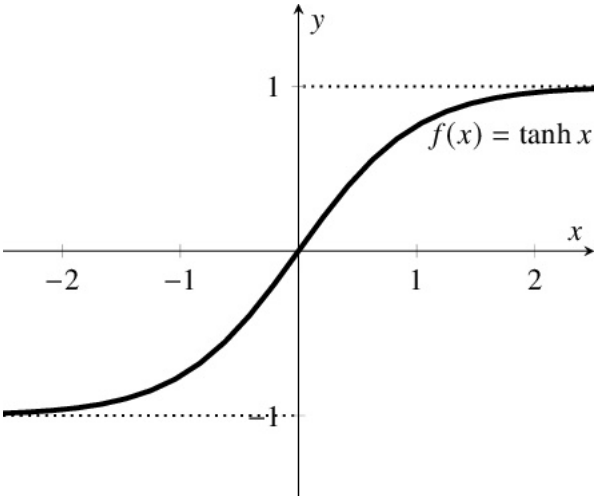

FIGURE 2.8 – Fonction tanh [16]

5. **Fonction Softmax :** La fonction softmax (une fonction exponentielle normalisée) est une fonction d'activation utilisée en dernière couche d'un réseau de neurones, construit pour effectuer une tâche de classification multi-classes. Pour chaque sortie, la fonction softmax donne un résultat entre 0 et 1. Ainsi, cette fonction a la forme d'une probabilité où l'activation de la fonction pour chacun de ses paramètres, est une probabilité parmi l'ensemble des probabilités. La fonction Softmax est donnée par l'équation :

$$f(x_i) = \frac{e^{x_i}}{\sum e^{x_p}} \tag{2.5}$$

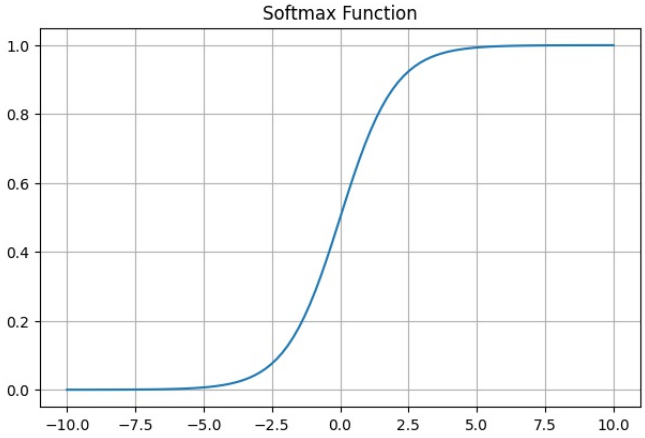

FIGURE 2.9 – Fonction Softmax [17]

Enfin, voici quelques règles pour choisir la fonction d'activation d'une couche de sortie en fonction du type de problème de prédiction que l'on résout :

— **Régression** - Fonction d'activation linéaire
— **Classification binaire** - Fonction d'activation sigmoïde/logistique
— **Classification multi-classe** — Fonction d'activation Tanh et/ou Softmax
— **Classification multi-étiquettes** – Sigmoïde

La fonction d'activation utilisée dans les couches cachées est généralement choisie en fonction du type d'architecture de réseau neuronal.

— **Réseau neuronal convolutif (CNN)** - Fonction d'activation ReLU.
— **Réseau neuronal récurrent (RNN)** - Fonction d'activation Tanh et/ou Sigmoïde.

En résumé, le deep learning permet à la machine de réaliser des corrélations complexes et non-linéaires entre les données. L'entraînement du deep learning est beaucoup plus long à cause d'importante quantité de données à traiter, et des nombreux paramètres et formules mathématiques impliquées. Grâce aux réseaux de neurones, les voitures sont capables de déterminer les objets à éviter, de reconnaître les feux tricolores et les panneaux et de savoir quand accélérer et ralentir.

Les applications du deep learning sont tellement nombreuses et couvrent d'autres domaines variés tels que la vision par ordinateur (reconnaissance d'objets, de visages, etc.), le traitement du langage naturel (traduction automatique, analyse de sentiments, etc...), la reconnaissance de la parole, la bio-informatique, la cyber-sécurité, et bien d'autres encore. Dans ce travail nous allons utiliser ces méthodes du deep learning appliquées à la sécurité informatiques notamment la détection de p2p botnet.

2.4 Architecture des réseaux de neurones convolutifs (CNN) et récurrents (RNN)

2.4.1 Architecture des réseaux de neurones convolutifs (CNN) :

Les réseaux de neurones convolutifs CNN, aussi appelés ConvNets, sont une architecture spécifique du réseau de neurones profonds conçue pour le traitement des données en grille, telles que les images et ces ConvNets sont largement utilisée pour la classification et la reconnaissance des objets ou des formes. Les CNNs ont été largement utilisés avec succès. Un réseau de neurones convolutif est constitué de deux parties fondamentales qui sont l'extraction des caractéristiques et la classification des données selon le schéma présenté par la figure 2.10

Un CNN typique comprend plusieurs couches de convolution, de sous-échantillonnage, de normalisation, de couches entièrement connectées, et une couche de sortie. Chaque couche de convolution est suivie d'une fonction d'activation ReLU et chaque paire de convolution et d'activation est souvent suivie d'une couche de pooling. Ce schéma se répète plusieurs fois, permettant au réseau de hiérarchiser les caractéristiques apprises. Voici les différents couches en détail :

1. **Couche de convolution :** La première couche d'un CNN est généralement une couche de convolution. Cette couche applique un ensemble de filtres convolutifs (ou des noyaux de convolution) de taille 3x3, 5x5 ou 7x7, aux images en entrée pour extraire des caractéristiques de ces images. Chaque filtre détecte des motifs spécifiques dans l'image, tels que des bords, des textures ou d'autres motifs visuels.

   Voici quelques paramètres de la couche convolutifs :

   — *Nombre de filtres (ou kernels) :* C'est le nombre de filtres ou noyaux de convolution appliqués à l'image d'entrée. Chaque filtre extrait des caractéristiques spécifiques.
   — *Taille du filtre :* La taille spatiale du filtre, généralement définie par une matrice, par exemple 3x3, 5x5 ou 7x7. Cette taille influence la taille de la région locale analysée à chaque étape de la convolution.
   — *Stride :* Le pas ou l'écart entre les positions successives du filtre.
   — *Padding (remplissage) :* Le remplissage consiste à ajouter des pixels supplémentaires autour de l'image d'entrée afin de bien garder les dimensions. Cela

est pour éviter que la taille de l'image ne diminue trop, après chaque couche de convolution.

2. **Couche de pooling :** Après la couche de convolution, une couche de pooling est souvent ajoutée pour réduire la dimensionnalité de l'image résultante tout en préservant les caractéristiques les plus importantes. La méthode de pooling la plus courante est le max pooling, où la valeur maximale dans chaque région de la carte de caractéristiques est conservée. La couche de pooling réalise un sous-échantillonnage non-linéaire, ce qui permet de réduire le nombre de paramètres qu'un CNN doit apprendre.

   Voici quelques paramètres de la couche de pooling :
   - *Max Pooling :* Pour chaque région de la carte de caractéristiques, le pooling max sélectionne la valeur maximale.
   - *Average Pooling :* Pour chaque région de la carte de caractéristiques, le pooling moyen calcule la moyenne des valeurs. Ainsi, il réalise une réduction en prenant en compte l'ensemble des informations de la région.
   - *Global Pooling :* Dans cette variante du pooling moyen, une seule valeur est calculée pour chaque carte de caractéristiques en prenant la moyenne de toutes les valeurs.

3. **Couche d'activation :** Après chaque couche de convolution et de pooling, une fonction d'activation est généralement appliquée. La fonction d'activation la plus couramment utilisée est la fonction ReLU (Rectified Linear Unit), qui introduit une non-linéarité dans le modèle en remplaçant les valeurs négatives par zéro. Les caractéristiques activées sont ainsi, transmises à la couche suivante.

4. **Couche flatten :** Elle symbolise l'aplatissement des caractéristiques spatiales, pour les introduire dans une couche entièrement connectée (fully connected). Cette couche génère un vecteur de K dimensions, où K est le nombre de classes que le réseau doit prédire.

5. **Couche entièrement connectée :** Après plusieurs couches de convolution et de pooling, les caractéristiques extraites sont généralement aplaties et connectées à une ou plusieurs couches entièrement connectées, souvent utilisées dans les réseaux de neurones classiques. Ces couches entièrement connectées sont utilisées pour com-

biner les caractéristiques extraites et effectuer la classification finale.

6. **Couche de sortie :** La dernière couche du CNN est une couche de sortie, qui utilise une fonction d'activation appropriée en fonction du problème spécifique. Par exemple, pour la classification binaire, une fonction d'activation de sigmoïde est souvent utilisée, tandis que pour la classification multi-classe, une fonction d'activation de softmax est couramment utilisée.

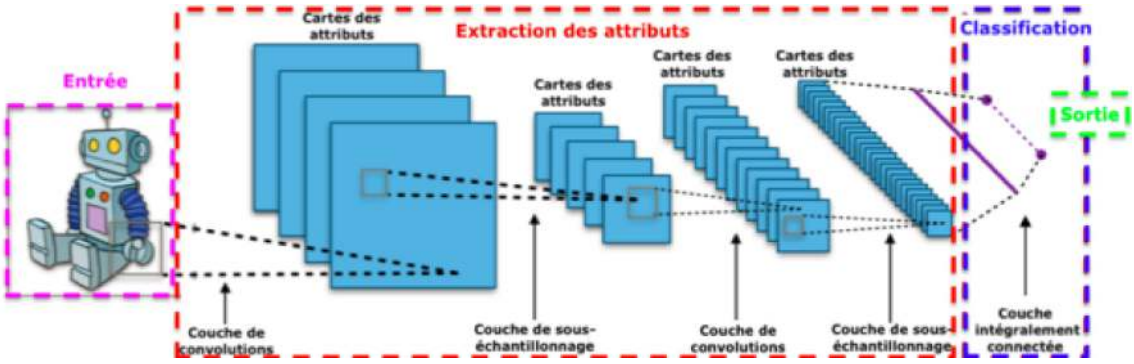

Figure 2.10 – Architecture réseaux de neurones convolutifs (CNN) [18]

2.4.2 Architecture des réseaux de neurones récurrents (RNN) :

Les RNNs sont idéales pour résoudre des problèmes où la séquence est plus importante que les éléments individuels eux-mêmes. On les utilise généralement pour traiter des données séquentielles, qui peuvent être des données chronologiques ou des données textuelles de n'importe quel format. Les séquences sont des objets dont chaque élément possède un ordre, une position, une inscription dans le temps. Par exemple, dans une phrase, chaque mot vient dans un certain ordre et il est prononcé sur un intervalle de temps distinct de celui des autres [67].

En bref, le modèle de réseau neuronal compare la différence entre sa sortie et la sortie désirée et renvoie ces informations au réseau pour ajuster des paramètres tels que les poids à l'aide d'une valeur appelée gradient. Une plus grande valeur de gradient implique de plus grands ajustements aux paramètres, et vice versa. Ce processus se poursuit jusqu'à ce qu'un niveau satisfaisant de précision soit atteint.

Le problème de disparition de gradient fait que les RNNs doivent avoir une mémoire à court terme où les sorties antérieures ont de plus en plus peu ou pas d'effet sur la sortie

actuelle. Le problème du gradient de fuite peut être résolu par différentes variantes des RNNs [67]. **Deux d'entre eux sont :**

— *LSTM (appelés Long Short-Term Memory) :* Ils sont capables de se souvenir des entrées sur une longue période de temps. C'est parce qu'ils contiennent des informations dans une mémoire, un peu comme la mémoire d'un ordinateur. Le LSTM peut lire, écrire et supprimer des informations de sa mémoire. Dans cette mémoire, on décide à chaque fois de stocker ou de supprimer des informations en fonction de l'importance qu'elle accorde à l'information. L'attribution de l'importance se fait par des poids, qui sont également appris par l'algorithme.

— *GRU (Gated Recurrent Units) :* Ce modèle est similaire aux LSTM car il permet aussi de résoudre le problème de mémoire à court terme des modèles RNN. Pour résoudre ce problème, GRU intègre les deux mécanismes de fonctionnement de porte appelés Update gate et Reset gate : les portails de ré-initialisation et de mise à jour sont concrètement des vecteurs qui contrôlent les informations à conserver et à négliger.

Les réseaux neuronaux récurrents sont un outil polyvalent qui peut être utilisé dans une variété de situations. Ils sont employés dans diverses méthodes de modélisation linguistique et de génération de texte. Ils sont aussi employés dans la reconnaissance vocale. Cependant, ils ont un défaut. Ils sont lents à entraîner et ont de la difficulté à apprendre les dépendances à long terme [67].

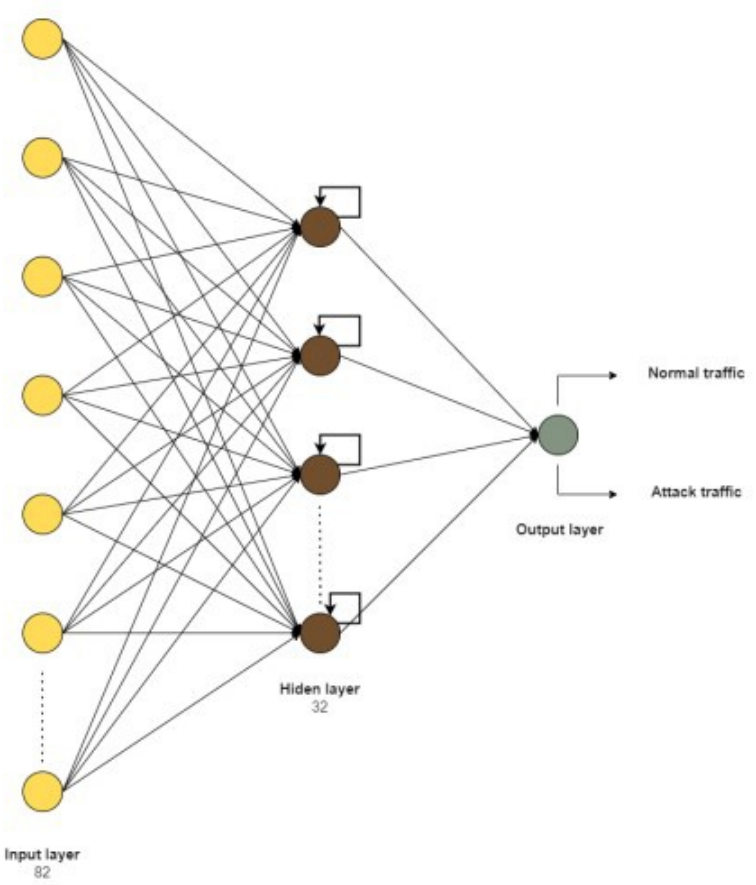

Figure 2.11 – Architecture réseaux de neurones récurrents (RNN) [19]

2.5 Pré-traitement des données pour la détection de botnets

Après avoir collecté de données ou choisi un ensemble de données qu'on utilisera pour l'entraînement et l'évaluation du modèle de deep learning en question, c'est essentiel de passer à l'exploration des données. Cette analyse des données a pour but comprendre la distribution des caractéristiques, identifier les valeurs manquantes et détecter les éventuels données anormales. Le pré-traitement de l'ensemble des données est effectué avant d'appliquer le dataset au réseau de neurones convolutifs pour la détection de botnets P2P. C'est vraiment nécessaire de normaliser ou standardiser les données, de gérer les valeurs manquantes et de convertir aussi les variables catégorielles en variables numériques.

Normalement toutes les valeurs 'NAN' (Not A Number) ou 'INF' (Infinite Value), peuvent être considérées comme des valeurs manquantes. Les algorithmes du deep lear-

ning en général, traitent très mal ces valeurs qui affectent négativement la performance du modèle final. Alors, les lignes avec des valeurs NAN ou INF doivent être supprimés. Dans cette étude, la colonne d'étiquette sera codée pour une classification binaire, le trafic normal étant affecté d'une valeur de 0 et tout le trafic d'attaque (Botnets P2P) étant affecté d'une valeur de 1. La colonne des paquets de flux doit être convertie de chaîne en type de données numériques. Afin d'améliorer les performances du modèle d'apprentissage profond, plusieurs colonnes non pertinentes doivent être éliminées de l'ensemble de données et ces colonnes incluent l'adresse IP source, l'adresse IP de destination, la version, l'ID de flux, le HTTP similaire, unamed0 et timestamp etc.

2.6 Métriques d'évaluation de la performance des modèles de deep learning

Les métriques d'évaluation jouent un rôle énorme et crucial dans l'évaluation des performances et de l'efficacité des modèles d'apprentissage profond. Ces métriques fournissent des mesures quantitatives pour évaluer la capacité du modèle à classer avec précision le trafic réseau et à détecter les attaques potentielles de botnet p2p. Il est important de choisir la métrique d'évaluation appropriée lors de la sélection entre les modèles d'apprentissage profond et l'ajustement des hyper-paramètres afin de faire des prédictions précises qui aideront dans le processus de prise de décision. Chaque métrique mesure quelque chose de différent sur les performances d'un algorithme.

2.6.1 Matrice de confusion

La matrice de confusion permet la visualisation pour un problème de classification des résultats de la manière synthétique. Elle permet de mesurer la qualité d'un problème de classification et cette matrice est indispensable pour déterminer les différentes métriques de classification telles que l'Accuracy, le F1-score ou encore l'AUC et la courbe ROC.

1. **Vrai négatif (True Negative ou TN) :** La prédiction est négative et c'est la réalité par exemple un médecin prédit que la patiente n'est pas malade et c'est vrai dans la réalité.

2. **Vrai positif (True Positive ou TP) :** Ici, la prédiction est positive et c'est la réalité par exemple la patiente est malade et c'est vrai dans la réalité.

3. **Faux positif (False Positive ou FP) :** Dans ce cas, la prédiction est positive mais ce n'est pas la réalité des choses par exemple le médecin prédit que la patiente est malade alors qu'elle n'est pas.

4. **Faux négatif (False Negative ou FN) :** La prédiction est négative mais ce n'est pas la réalité par exemple un médecin prédit que la patiente n'est pas malade et c'est faux dans la réalité.

Matrice de Confusion		Classes Prédites	
		Négatif	**Positif**
Classes réelles	**Négatif**	Vrai Négatif (TN)	Faux Positif (FP)
	Positif	Faux Négatif (FN)	Vrai Positif (TP)

Figure 2.12 – Matrice de confusion [20]

2.6.2 Exactitude (accuracy)

C'est une mesure de la proximité des mesures par rapport à une valeur spécifique, tandis que la précision est la proximité des mesures les unes par rapport aux autres, c'est-à-dire pas nécessairement par rapport à une valeur spécifique. En d'autres termes : si nous disposons d'un ensemble de points de données provenant de mesures répétées de la même quantité, l'ensemble est dit exact (accurate) si leur moyenne est proche de la valeur réelle de la quantité mesurée. En revanche, on appelle l'ensemble précis si les valeurs sont proches les unes des autres. Les deux concepts sont indépendants l'un de l'autre, ce qui signifie que l'ensemble de données peut être exact (accurate), précis, les deux, ou ni l'un ni l'autre. [20].

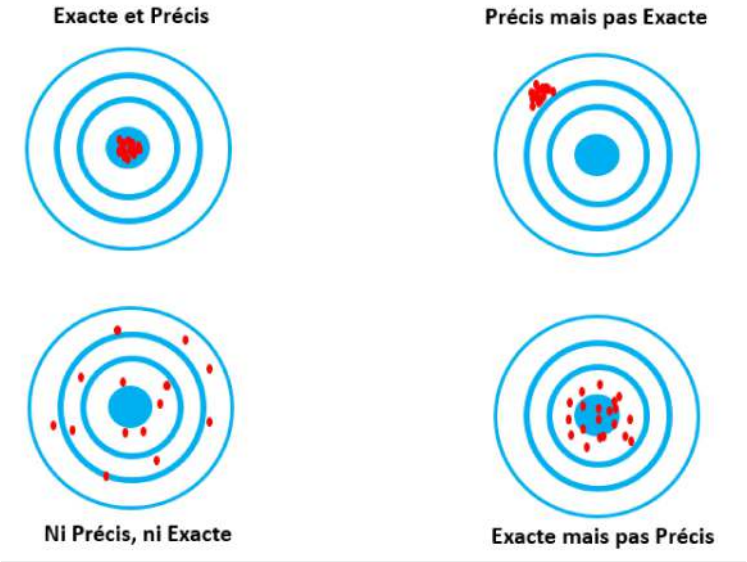

Figure 2.13 – Exactitude et Précision [21]

Accuracy est la proportion de prédictions correctes (vrais positifs et vrais négatifs) parmi le nombre total de cas examinés. Accuracy est donné par l'équation :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.6}$$

2.6.3 Précision

La précision indique combien du nombre total de prédictions spécifiées comme positives sont correctement affectées. Cette métrique est également connue sous le nom de valeur prédictive positive. La Précision est représentée par l'équation :

$$Precision = \frac{TP}{TP + FP} \tag{2.7}$$

2.6.4 Rappel

Le rappel est le nombre total de cas positif réels qui ont été correctement prédits. Cette métrique est également connue sous le nom de sensibilité. Le rappel est donné par l'équation :

$$Rappel = \frac{TP}{TP + FN} \tag{2.8}$$

2.6.5 F1-score

F1-mesure combine la précision et le rappel. Le résultat est la moyenne harmonique des deux valeurs. Elle est représentée par l'équation :

$$F1 - score = 2 * \frac{Precision * Rappel}{Precision + Rappel} \tag{2.9}$$

2.6.6 AUC (Area Under Curve) et courbe ROC (Receiver Operating Characteristic curve)

La courbe ROC (Receiver Operating Characteristic) représente la sensibilité en fonction de 1 – spécificité pour toutes les valeurs seuils possibles du marqueur étudié. La sensibilité est la capacité du test à bien détecter les malades et la spécificité est la capacité du test à bien détecter les non-malades [22].

L'AUC correspond à la probabilité pour qu'un événement positif soit classé comme positif par le test sur l'étendue des valeurs seuil possibles. Pour un modèle idéal, on a AUC=1 , pour un modèle aléatoire, on a AUC=0.5.

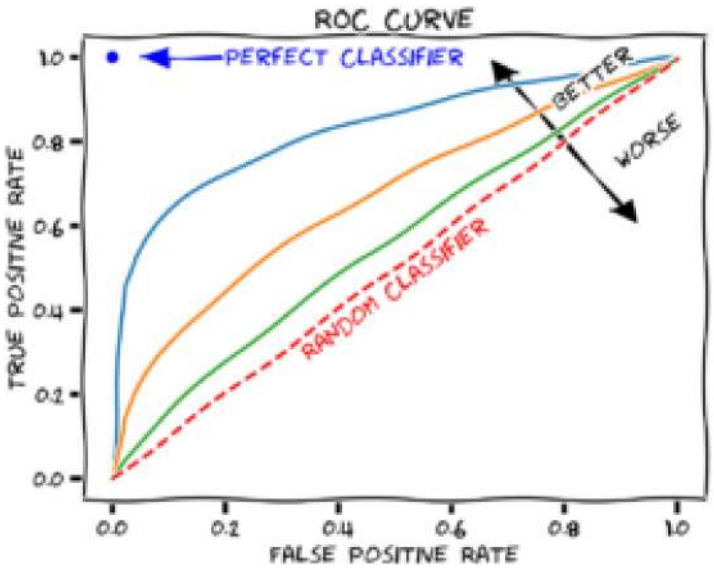

Figure 2.14 – Courbe ROC [22]

2.7 Conclusion :

L'objectif de ce chapitre a été de s'initier aux notions de base de l'apprentissage profond tels que les neurones biologique et artificiel, la fonction d'activation, les réseaux de neurones convolutifs et récurrents et le pré-traitement des données surtout pour la détection de botnets P2P. L'accent a été mis particulièrement sur les métriques d'évaluation de la performance des algorithmes de deep learning pour déterminer leur intervention et importance dans le processus de prise de décision.

Chapitre 3

Méthodologie et Implémentation

3.1 Introduction

Notre recherche se focalise sur le développement d'un système de détection des botnet qui utilisent le protocole P2P pour les réseaux en utilisant des techniques d'apprentissage profond. Les cyberattaques, en particulier les attaques par botnets P2P, représentent des menaces significatives pour la sécurité des réseaux dans l'environnement numérique actuel.

Notre objectif de recherche est d'exploiter la puissance des algorithmes d'apprentissage profond pour détecter efficacement ces attaques, en mettant un accent particulier sur la détection des botnets P2P. Pour répondre à cette problématique, des ensembles de données complets sont indispensables pour entraîner et évaluer les modèles de détection des botnets P2P. Ce chapitre expose notre méthodologie proposée en décrivant l'ensemble de données et en le préparant pour l'entraînement des modèles d'apprentissage profond, en mettant en lumière la détection des botnets P2P.

3.2 Collecte et préparation des données

3.2.1 Construction du dataset

Construire un dataset est très important car le dataset est la norme cruciale pour évaluer l'efficacité d'un modèle d'apprentissage profond proposé. Le problème dans la plupart des travaux existants est que les datasets utilisés étaient incomplets ou n'incluaient pas de véritables fonctionnalités de réseau. Le dataset doit contenir du trafic d'attaque mélangé à du trafic normal afin que le modèle ou l'approche proposé en sache davantage sur le

trafic normal et anormal.

Nous avons choisi le dataset CTU-13 combiné avec ground truth - HIKARI. Pour CTU-13, nous avons choisi le scénario 12 qui possède des P2P Bots, cependant ce dataset ne contient pas de trafic normal (trafic non malveillant qui transitera en parallèle du trafic malveillant). Et le dataset ground truth possède du trafic normal généré avec des machines non infectée par des bots.

Nous avons sélectionné le dataset CTU-13 pour trois raisons :

— Il est fiable et de nombreux chercheurs ont utilisé ce dataset pour évaluer leurs solutions,
— Il a été fourni sous forme de fichiers PCAP et non de fichiers CSV, ce qui permet de mieux comprendre le trafic réseau et le comportement des botnets P2P.
— Il contient le scénario de botnet P2P.

Cependant, ce dataset manquait de flux de réseau normal. Par conséquent, nous avons fusionné l'ensemble de données CTU-13 avec un flux normal réseau récent collecté à partir de l'ensemble de données HIKARI [68]. En conséquence, le nouveau dataset construit contient à la fois le comportement anormal du botnet P2P et le flux normal du réseau pour permettre au modèle formé de reconnaître le trafic normal et malveillant.

3.2.2 Contenu du dataset CTU-13

Ce dataset est considéré comme l'un des datasets le plus fiable de la communauté de l'apprentissage automatique. CTU-13 capture une grande partie du trafic réel des botnets, y compris 13 scénarios différents de différents types de botnets. Ce dataset contient de nombreux protocoles, tels que ICMP, TCP, DNS, etc. L'une de ses fonctionnalités utiles est qu'il a été fourni sous forme de fichiers PCAP, permettant une compréhension réaliste et plus approfondie du trafic réseau [69].

La capture du scénario 12 se dérouler le vendredi 19 août 2011, ainsi une série d'actions a été entreprise pour capturer et analyser l'activité de logiciels malveillants, en se concentrant particulièrement sur un bot qui scanne les ports UDP. Les événements ont été menés dans un environnement virtuel contrôlé avec des configurations de bande passante spécifiques.

Événements détaillés

— 10 :32 :08 CST (Central European Summer Time) - Début de la Capture Globale.

La configuration initiale pour la capture du trafic réseau a commencé. La bande passante était fixée à 100 kbps avec une capacité de rafale de 1000 kbps. 10 :41 :22 CST (Central European Summer Time) - Début de la Capture du Bot

Début de la capture des données spécifiquement pour le bot connu pour scanner les ports UDP.

— *10 :41 :59* CST (Central European Summer Time) - Démarrage de la VM 'saruman'

La machine virtuelle nommée 'saruman' a été lancée.

— *10 :42 :39* CST (Central European Summer Time) - Infection de la VM 'saruman'

La VM 'saruman' a été délibérément infectée par le logiciel malveillant.

— *10 :46 :39* CST (Central European Summer Time) - Démarrage de la VM 'saruman1'

La machine virtuelle nommée 'saruman1' a été lancée.

— *10 :47 :1*8 CEST (Central European Summer Time) - Infection de la VM 'saruman1'

La VM 'saruman1' a été délibérément infectée par le logiciel malveillant.

— *10 :49 :09* CEST (Central European Summer Time) - Démarrage de la VM 'saruman2'

La machine virtuelle nommée 'saruman2' a été lancée.

— *10 :50 :25* CEST (Central European Summer Time) - Infection de la VM 'saruman2'

La VM 'saruman2' a été délibérément infectée par le logiciel malveillant.

— *10 :53 :11* CEST (Central European Summer Time) - Ajustement de la Limite de Bande Passante

La limite de bande passante a été ajustée à 5000 kbps avec une capacité de rafale de 1000 kbps. Il y avait une incertitude quant à l'effet des nouveaux paramètres de bande passante, car aucun changement du volume de flux n'a été observé.

— *11 :44 :17* CEST (Central European Summer Time) - Arrêt des VMs

Toutes les machines virtuelles impliquées dans l'expérience ont été arrêtées.

— *11 :45 :29* CEST (Central European Summer Time) - Fin de la Chronologie

La journalisation de la chronologie a été complétée. la capture a durée 1 heure 43 minutes.

3.2.3 Ground Truth - HIKARI Dataset

Ce dataset a été fusionné avec le CTU-13 pour compléter ce qui manquait dans le premier dataset. Le dataset HIKARI capture complètement le trafic réseau, tel que la communication entre les hôtes, les messages diffusés et les requêtes de recherche de domaine. Nous avons choisi le dataset "Ground - truth" car il fournit un trafic bénin réaliste provenant d'un réseau de production réel, et non un trafic synthétique, comme trouvé dans certains datasets [68].

CTU-13 Datasset p2p botnet + "Ground - truth" dataset traffic normal.

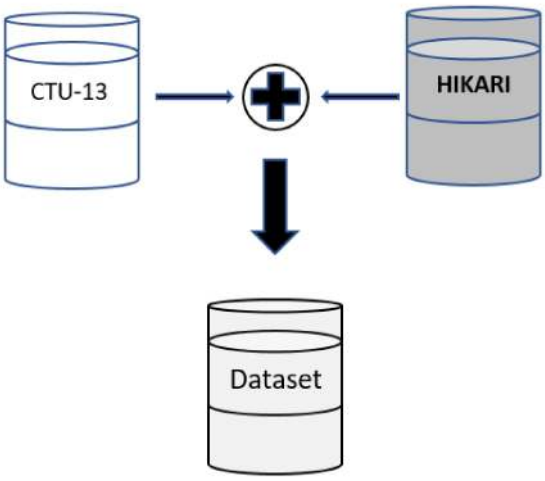

Figure 3.1 – Construction de Dataset

HIKARI ground truth Saturday_2021-04-10_1628.anonymized comprend 533,848 paquets et CTU-13 Scénario n°12 P2P botnet comprend 352,266 paquets. C'est qui fait un total de 886,114 paquets. Nous allons faire l'étude des différents caractéristiques dans la partie pré-traitement.

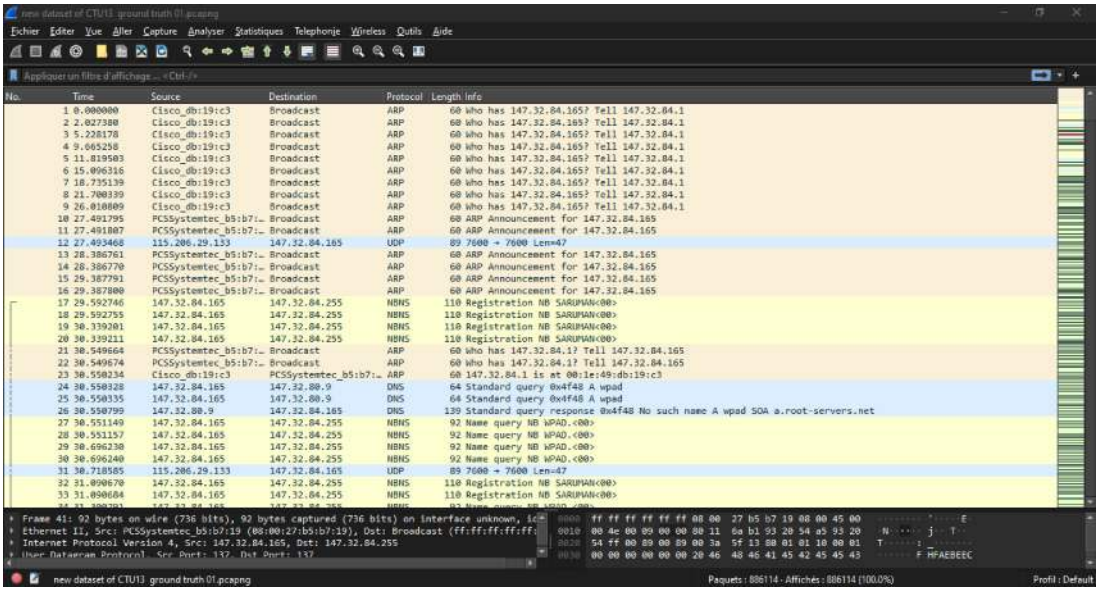

FIGURE 3.2 – Extrait du dataset avec wireshark

WireShark est un analyseur de paquets open-source qui permet la capture de trafic en temps réel sur une interface réseau. WireShark est utile pour observer le trafic réseau et pour une analyse en détail des paquets [70].

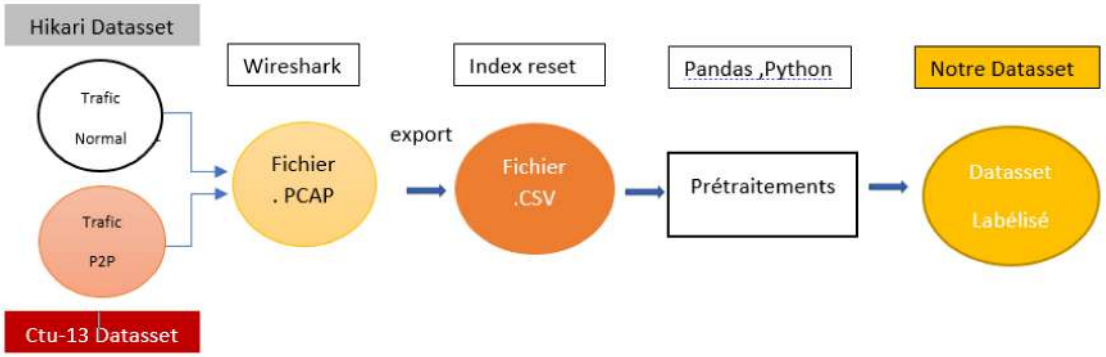

FIGURE 3.3 – Processus de construction de dataset

Pour la construction de notre dataset, nous avons suivi plusieurs étapes clés, comme le montre la figure 3.3. Tout d'abord, nous avons fusionné deux datasets à l'aide de l'application d'analyse de réseau Wireshark. Ensuite, nous avons sélectionné tous les principaux champs qui constituent un paquet, un segment et une trame. Enfin, nous avons exporté ce fichier au format "CSV" afin de faciliter sa manipulation avec la bibliothèque ***"Pandas"*** du langage de programmation ***"Python"***.

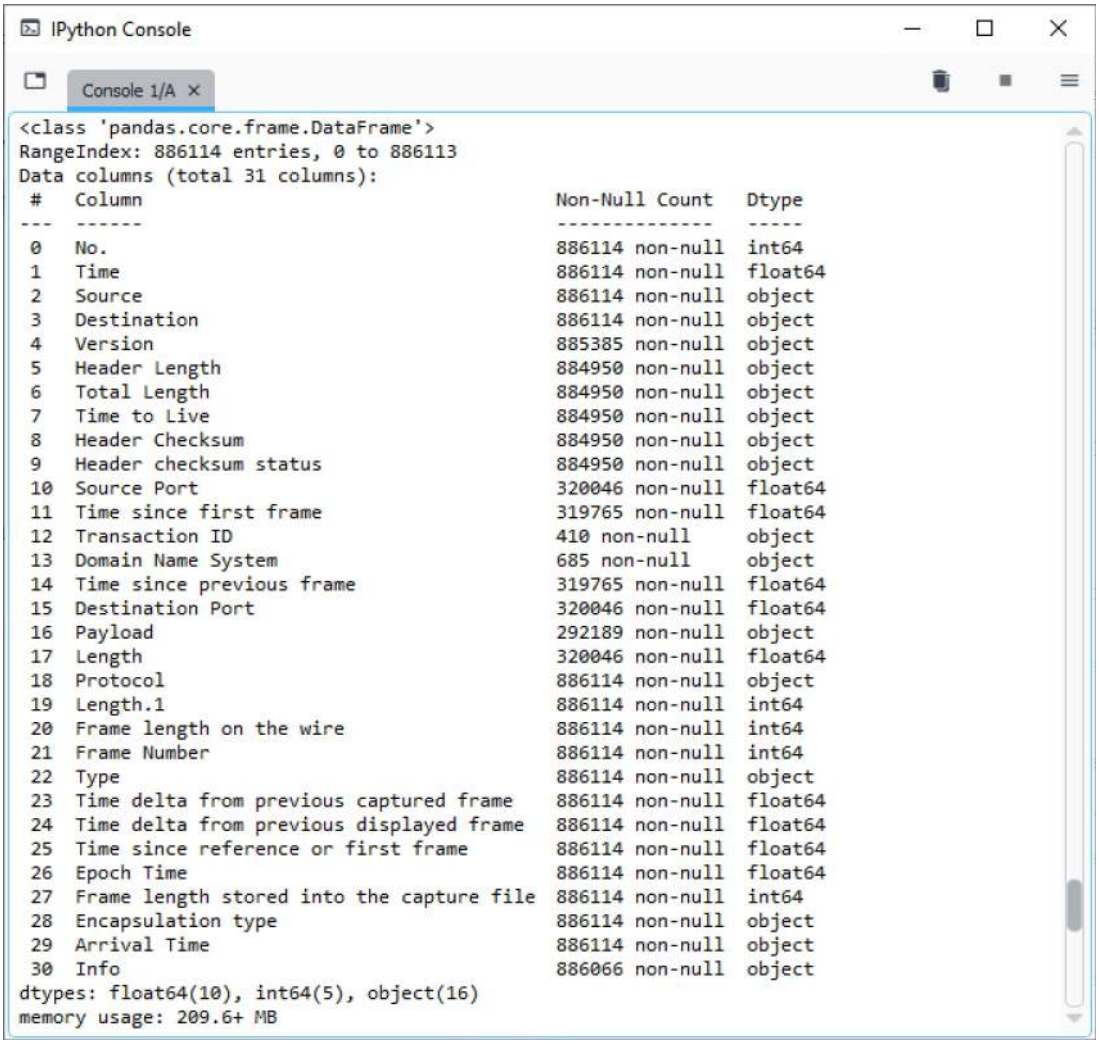

Figure 3.4 – Visualisation des caractéristiques du dataset au niveau de console Python

Pour mieux comprendre notre dataset, nous avons utilisé plusieurs fonctions de la bibliothèque ***"Pandas"***.

```
import pandas as pd
data = pd.read_csv('CTU13 ground truth 01.csv', encoding='utf-8')
print(data.isnull().sum())
print(data.info())
```

isnull() : Cette méthode vérifie pour chaque élément du DataFrame s'il est nul (NaN).

sum() : Ensuite, la méthode sum() additionne ces valeurs par colonne, fournissant le nombre total de valeurs manquantes pour chaque colonne.

info() : Cette méthode fournit des informations sur le DataFrame, notamment le nombre de valeurs non nulles dans chaque colonne et les types de données (int, float, string, etc.).

Tableau 3.1 – Nombres total des colonnes avec des données manquantes

N0	Nom de la colonne	Total	Manquant
1	No.	886114	0
2	Time	886114	0
3	Source	886114	0
4	Destination	886114	0
5	Version	885385	729
6	Header Length	884950	1164
7	Total Length	884950	1164
8	Header Checksum	884950	1164
9	Header checksum status	884950	1164
10	Source Port	320046	566068
11	Time since first frame	319765	566349
12	Transaction ID	410	885704
13	Domain Name System	685	885429
14	Time since previous frame	319765	566349
15	Destination Port	320046	566068
16	Payload	292189	593925
17	Length	320046	566068
18	Protocol	886114	0
19	Length.1	886114	0
20	Frame length on the wire	886114	0
21	Frame Number	886114	0
22	Type	886114	0
23	Time delta from previous captured frame	886114	0
24	Time delta from previous displayed frame	886114	0
25	Time since reference or first frame	886114	0
26	Epoch Time	886114	0
27	Frame length stored into the capture file	886114	0
28	Encapsulation type	886114	0
29	Arrival Time	886114	0
30	info	886114	0

Ce tableau montre les nombres et la somme des champs disponible par paquet, comme on peut le constater parmi les 30 champs ou caractéristiques sont au complet et d'autres pas, donc on a choisi d'éliminer les caractéristiques qui sont absent dans plus 90% des paquets c'est notamment le champ *Transaction ID* qui est présent que parmi 410 paquets et le champ *Domain Name System* qui lui est présent dans 685 paquets sur 886114 paquets.

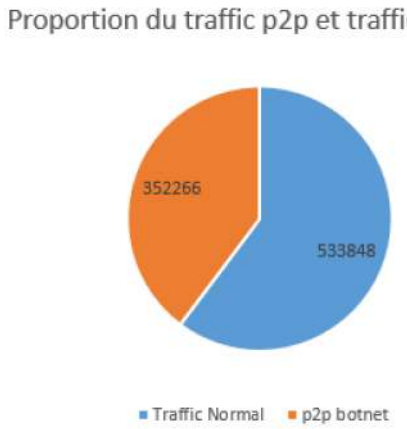

Figure 3.5 – Proportion entre p2p botnet et trafic normal

Notre nouveau dataset est constitué de 60% des paquets du trafic normal et de 40% des paquet du trafic P2P botnet c'est-à-dire pour HIKARI ground truth il y a 533,848 paquets et pour CTU-13 Scénario-12 P2P botnet il y a 352,266 paquets.

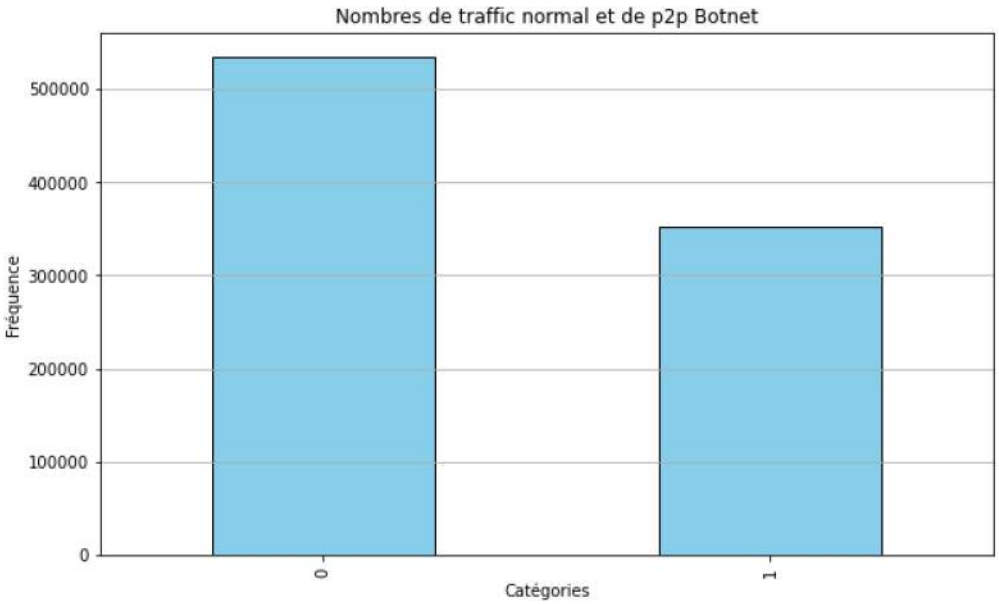

Figure 3.6 – Proportion de label p2p botnet et trafic normal

Après avoir étiqueté notre dataset grâce au langage de programmation python, voilà la proportion de trafic normal qui sont étiquetés en 0 et le trafic P2P botnet étiquetés en 1. On a ainsi un dataset constitué de trafic normal et de trafic P2P botnet.

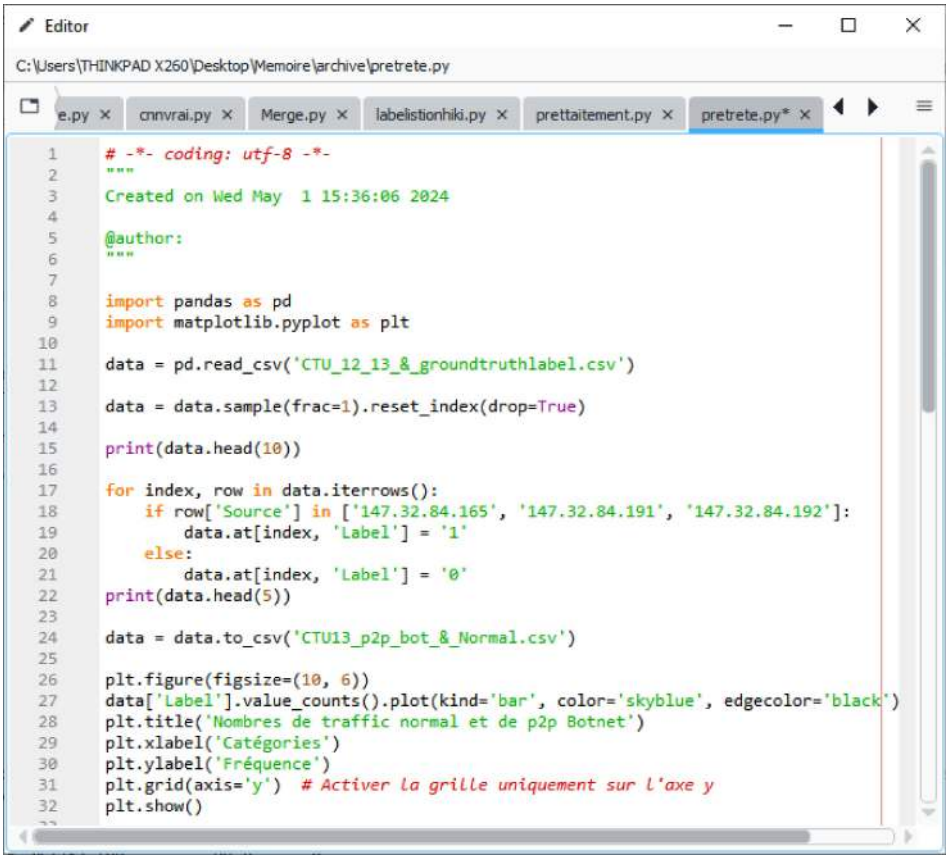

Figure 3.7 – Code python pour l'étiquetage et la figure 3.6

On a étiqueté le trafic botnet P2P en fonction de l'adresse IP source, on a donné l'étiquette "1" aux adresses des machines infectées et du botmaster et l'étiquette "0" aux adresses IP des machines non infectées. Le dataset CTU-13 est constitué de trafic P2P entre les machines infectées (bots) et le botmaster, et ensuite de trafic P2P entre les bots eux-mêmes.

L'adresse IP du botmaster : 147.32.84.165

Les adresses IP des machines infectées : 147.32.84.165, 147.32.84.191, 147.32.84.192

3.3 Architecture du modèle de détection de botnets P2P

Dans ce travail nous allons développé un modèle de détection de P2P botnet en utilisant deux architectures de réseaux de neurones. Notre modèle sera basé sur les réseaux

de neurone convolutifs 1 dimension CNN 1D.

```
37
38  # Définir le modèle CNN
39  model = tf.keras.models.Sequential([
40      tf.keras.layers.Reshape((x.shape[1], 1), input_shape=(x.shape[1],)),
41      tf.keras.layers.Conv1D(filters=32, kernel_size=3, activation='relu'),
42      tf.keras.layers.MaxPooling1D(pool_size=2),
43      tf.keras.layers.Flatten(),
44      tf.keras.layers.Dense(64, activation='relu'),
45      tf.keras.layers.Dropout(0.4),  # Ajout de dropout pour la régularisation
46      tf.keras.layers.Dense(64, activation='relu'),
47      tf.keras.layers.Dense(32, activation='relu'),
48      tf.keras.layers.Dense(1, activation='sigmoid')
49  ])
```

Figure 3.8 – Notre modèle CNN 1 dimension

3.3.1 Architecture de base de CNN 1D

Couche d'entrée : La couche d'entrée est la première couche du réseau qui reçoit les données d'entrée.

Convolution 1D : Une ou plusieurs couches de convolution 1D, chacune suivie d'une fonction d'activation (comme ReLU). Ces couches sont responsables de l'extraction de caractéristiques locales des séquences en balayant des filtres sur les données d'entrée.

Pooling 1D : Des couches de pooling 1D, on a utilisé MaxPooling car plus adapter pour la CNN 1D, qui réduisent la dimensionnalité des caractéristiques extraites tout en préservant les caractéristiques les plus importantes. Le pooling aide également à rendre le modèle plus robuste aux variations mineures dans les données d'entrée.

Couche de Classification : Une ou plusieurs couches dense (entièrement connectées) pour la classification. Ces couches prennent les caractéristiques extraites par les couches précédentes et les utilisent pour effectuer la classification des classes soit c'est un P2P botnet soit c'est un paquet normal.

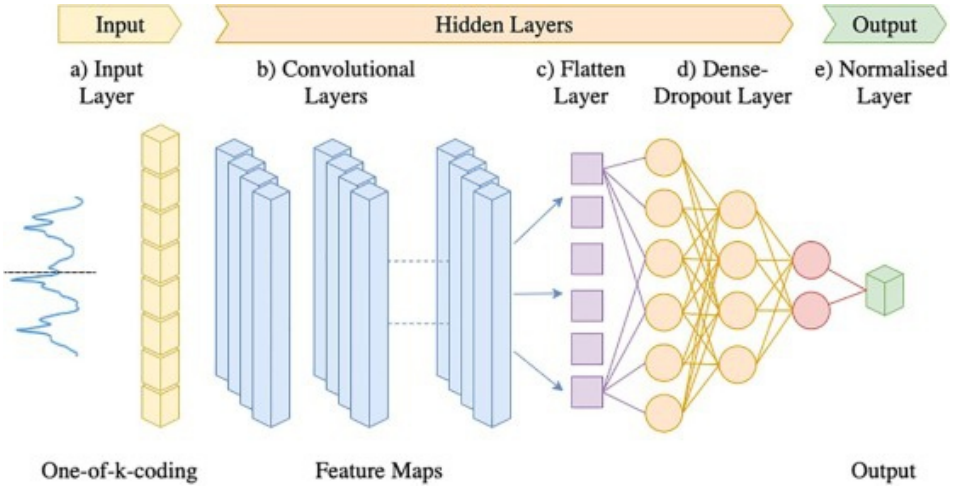

Figure 3.9 – Architecture de CNN 1D (dimension) [23]

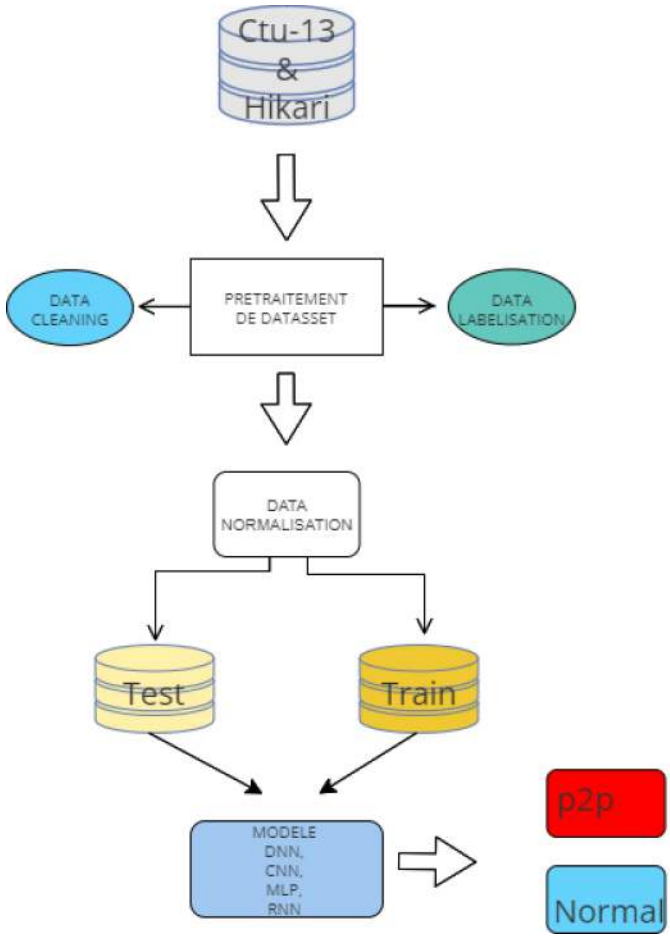

Figure 3.10 – Processus de détection de p2p botnet

3.4 Méthodologie d'entraînement et d'évaluation

Après avoir effectué le pré-traitement de données nous avons choisi d'utiliser deux modèle de deep learning le DNN (Deep Neural Network) et le CNN (Convolutional Neural Network). Pour l'entraînement de ces modèles nous avons procédé de comme suit :

Premièrement nous avons divisé notre ensemble de données en ensembles d'entraînement, de validation et de test, ensuite nous avons entraîné nos modèles sur l'ensemble de données d'entraînement en utilisant un algorithme d'optimisation Adam (Adaptive Moment Estimation) qui fait l'adaptation du taux d'apprentissage pour chaque paramètre et ajuste automatiquement le taux d'apprentissage pour chaque paramètre en fonction des estimations de premier et de deuxième ordre des gradients.

La fonction perte binary_crossentropy qui est une fonction de perte couramment utilisée dans les problèmes de classification binaire comme le notre, lors de l'entraînement de modèles de réseaux de neurones. Elle mesure la divergence entre la distribution de probabilité prédite par le modèle et la distribution de probabilité réelle des étiquettes (label dans notre cas) de classe. Nous avons suivi les performances du modèle sur l'ensemble de validation pour éviter le surajustement (overfitting) en ajustant les hyperparamètres et en utilisant la technique de régularisation dropout.

Une fois l'entraînement terminé, nous avons évalué les performances du modèle sur l'ensemble de test. Ainsi nous avons utilisé les métriques d'évaluation suivantes pour évaluer les performances de nos modèles :

Exactitude (Accuracy) : Qui mesure le nombre de prédictions correctes parmi toutes les prédictions.

Précision : Qui mesure la proportion de vrais positifs parmi tous les exemples prédits comme positifs.

Recall (Rappel) : Qui mesure la proportion de vrais positifs parmi tous les exemples qui sont réellement positifs.

Matrice de Confusion : Une représentation tabulaire des prédictions du modèle par rapport aux valeurs réelles, permettant de visualiser les faux positifs, les faux négatifs.

3.5 Description de l'environnement de développement

L'apprentissage profond est un domaine coûteux en termes de calcul qui nécessite un équipement capable de gérer efficacement des calculs complexes et les projets d'apprentissage profond nécessitent le minimum de logiciels et de matériels fondamentaux nécessaires pour réaliser une expérience. Pour notre projet, nous avons utilisé un ordinateur ASUS contenant un GPU AMD Radeon HD 8500M Discrete/Hybrid Graphics R5 M420 2Go, Intel(R) HD Graphics 620 comme GPU, un processeur Intel(R) Core(TM) i7-7500U CPU à 2,70 GHz - 2,90 GHz 2 Cores et une RAM de 8,00 Go et pour le logiciel, nous avons commencé par mettre en place un environnement de développement Python local à l'aide de la plate-forme Anaconda.

Vu que les projets en IA se concrétisent aujourd'hui en usant de l'environnement de travail "Anaconda", où le langage "Python" y est installé, nous avons utilisé l'environnement de développement Anaconda pour effectuer la détection des botnets P2P en utilisant les modèles de l'apprentissage profond.

Anaconda est une distribution open source du langage de programmation Python, elle a été spécialement conçue pour les activités impliquant des projets d'apprentissage automatique et d'apprentissage profond en science des données. Il offre une sélection complète de bibliothèques, d'outils et de frameworks pré-installés, ce qui simplifie la gestion et la configuration de l'environnement logiciel requis pour les applications d'analyse de données et d'apprentissage automatique. Les bibliothèques populaires telles que "NumPy", "Pandas" et "Scikit-learn" sont toutes incluses dans Anaconda, ainsi que Jupyter Notebook, qui permet une analyse interactive et exploratoire des données [71].

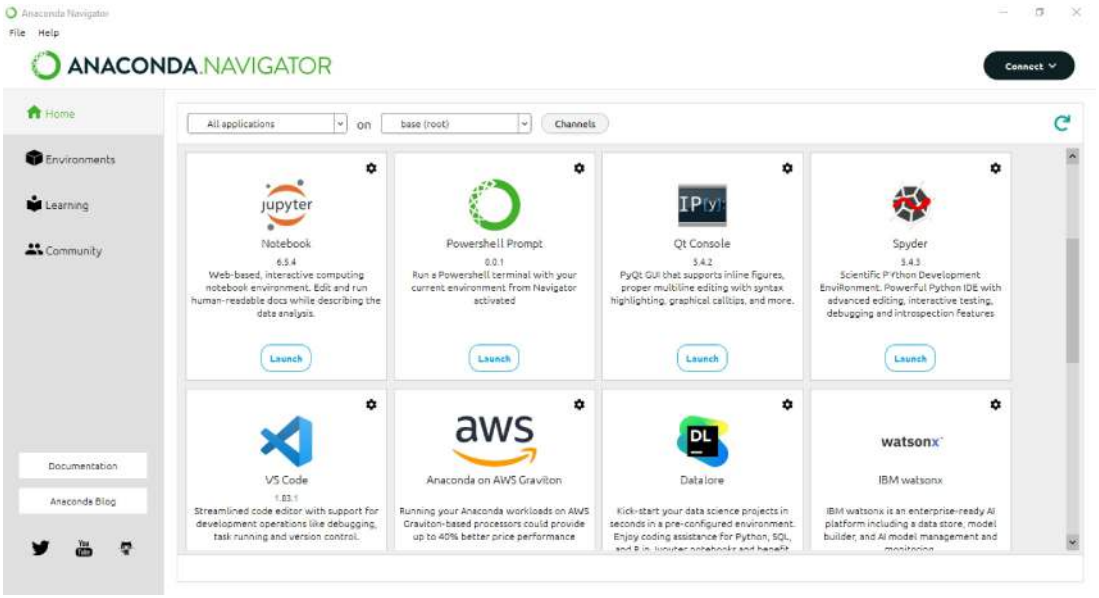

FIGURE 3.11 – Navigateur anaconda

Avant d'utiliser le langage Python (Spyder - Integrated Development Environment), nous avons lancé Anaconda Prompt pour créer un environnement Conda avec Python 3.9 et installé les librairies open source, mathématiques, graphiques (matplotlib) et spécifiques au deep learning. L'environnement "Conda" est principalement utilisé pour la gestion d'environnements Python et "PIP" est utilisé pour installer des packages Python.

3.5.1 NumPy

Il s'agit d'un package Python essentiel pour le calcul scientifique. Il propose des objets tableaux multidimensionnels efficaces ainsi qu'une vaste sélection d'opérations mathématiques pouvant être appliquées à ces tableaux. Dans l'environnement de science des données Python, NumPy sert de bibliothèque principale sur laquelle s'appuient de nombreuses autres bibliothèques et frameworks [72]. NumPy est essentiel pour de nombreuses tâches de calcul scientifique et de manipulation de données.

3.5.2 Pandas

Il s'agit d'un puissant module d'analyse et de manipulation de données Python. Il propose des structures de données et des opérations pour gérer et travailler avec succès avec des données structurées, telles que des données dans un tableau ou une série chronologique. Dans les projets de science des données, Pandas est fréquemment utilisé pour le nettoyage des données, opérations de traitement et d'exploration [73].

3.5.3 Keras

Il s'agit d'une bibliothèque de l'apprentissage profond open source basée sur Python. Les utilisateurs peuvent concevoir des modèles complexes avec moins de code car ils offrent une interface de haut niveau pour la construction et l'entraînement de CNN. Keras est réputé pour son architecture flexible, ses fonctionnalités conviviales et son inter-opérabilité avec d'autres frameworks d'apprentissage profond, notamment TensorFlow [74].

3.5.4 TensorFlow

Il s'agit d'un cadre d'apprentissage profond bien connu qui est une bibliothèque de logiciels open source pour l'apprentissage automatique sur une gamme de tâches et il a été créé par l'équipe Google Brain pour une utilisation interne de Google. Il fournit une large gamme de supports, de cadres et d'outils pour créer et mettre en œuvre des modèles d'apprentissage automatique. TensorFlow est une abstraction de graphiques informatiques flexible qui permet d'exécuter efficacement des réseaux neuronaux sophistiqués sur diverses plates-formes matérielles [75].

Tensorflow est une bibliothèque mathématique symbolique, également utilisée comme système pour créer et entraîner des réseaux de neurones afin de détecter et déchiffrer des modèles et des corrélations, analogues à l'apprentissage et au raisonnement humains. Un tenseur peut être représenté comme un tableau multidimensionnel de nombres.

3.5.5 Scikit-learn

Il s'agit d'une bibliothèque d'apprentissage automatique Python flexible. Il offre une grande variété d'outils et de méthodes pour des tâches telles que la réduction de dimensionnalité, le regroupement, la régression et la classification. Dans les projets d'apprentissage automatique, Scikit-learn est fréquemment utilisé pour la préparation des données, l'évaluation et la sélection de modèles [76].

3.5.6 Matplotlib

Il s'agit d'une bibliothèque de traçage Python qui permet de créer des visualisations statiques, animées et interactives. Il peut produire une variété de graphiques, de diagrammes et de tracés pour examiner et présenter efficacement les données, car il dispose d'une large gamme d'outils de traçage et de possibilités de personnalisation [77].

3.6 Implémentation du modèle de détection de botnets P2P

L'implémentation d'un modèle DNN (Deep Neural Network) et CNN (Convolutional Neural Network) pour la détection des botnets P2P consiste à créer et entraîner des réseaux neuronaux profonds capables de reconnaître les schémas caractéristiques du trafic réseau associé aux botnets P2P.

1. **Modèle DNN (Deep Neural Network) :** Un modèle DNN est un réseau neuronal artificiel (ANN) composé de plusieurs couches cachées entre la couche d'entrée et la couche de sortie. Chaque couche cachée est entièrement connectée à la couche précédente, ce qui permet au réseau d'apprendre des représentations hiérarchiques des données. Pour la détection des botnets P2P, un modèle DNN peut être conçu pour prendre en entrée des caractéristiques extraites du trafic réseau et prédire si ces données correspondent à un botnet P2P ou à un trafic légitime. L'implémentation de ce modèle implique les étapes suivantes :

   — **Collecte et pré-traitement des données :** Nous avons acquis des données de trafic réseau représentatives de botnets P2P et de trafic légitime, et on les a pré-traité pour les rendre compatibles avec l'entrée du modèle.

   — **Conception de l'architecture du modèle :** Nous avons défini l'architecture du réseau des neurones profond, y compris le nombre de couches, le nombre de neurones par couche, les fonctions d'activation, etc.

   — **Entraînement du modèle :** Nous avons utilisé les données d'entraînement pour ajuster les poids du modèle afin qu'il puisse apprendre à reconnaître les schémas caractéristiques des botnets P2P.

   — **Évaluation du modèle :** Nous avons utilisé des ensembles de données de validation et de test pour évaluer les performances du modèle en termes de précision, de rappel, de F-score, etc.

   — **Optimisation et ajustement :** Nous avons optimisé les hyper-paramètres du modèle et ajusté son architecture pour améliorer ses performances.

— **Déploiement et utilisation :** Normalement nous étions censés déployer le modèle entraîné dans un environnement de production pour la détection en temps réel du trafic botnet P2P, mais nous avons testé le modèle avec 20% de dataset qu'on a utilisé pour cette étude.

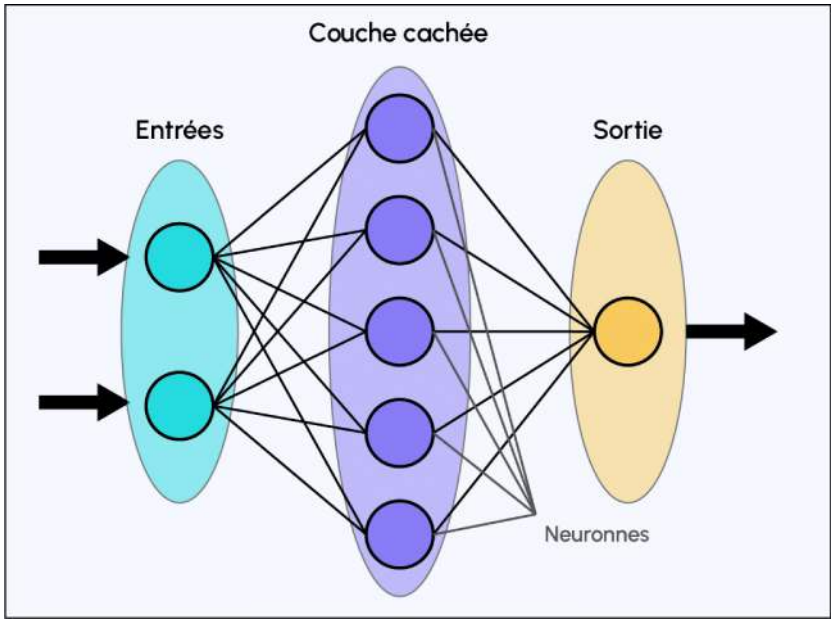

Figure 3.12 – Architecture DNN simple [24]

2. **Modèle CNN (Convolutional Neural Network) :** Un modèle CNN est un type de réseau neuronal qui utilise des opérations de convolution pour extraire automatiquement des caractéristiques des données. Les couches de convolution identifient des motifs locaux dans les données d'entrée en appliquant des filtres (kernels) à travers les données. Les couches de pooling réduisent la dimensionnalité des caractéristiques extraites tout en préservant les informations les plus importantes. Les CNN sont couramment utilisés pour la reconnaissance d'images, mais peuvent également être appliqués à des séquences de données, comme les séquences temporelles de trafic réseau, pour la détection des botnets P2P. L'implémentation de ce modèle implique les mêmes étapes comme le modèle DNN qu'on vient de voir.

Pour notre modèle, il n'existe qu'une seule couche convolutive unidimensionnelle avec 32 filtres et une taille de noyau de 3, activée à l'aide de la fonction ReLU. La couche de pooling qui est implémentée avec MaxPooling 1D, sous-échantillonne la sortie de la couche convolutive, réduisant la dimensionnalité tout en préservant les informations les plus pertinentes. Après la couche de pooling, la couche flatten

et la couche dense, une couche dropout est ajoutée pour éviter le sur-ajustement et rendre le modèle plus flexible.

Après les couches convolutives et de pooling, la sortie est transformée en un vecteur unidimensionnel avant d'être transmise aux couches entièrement connectées. Les couches dense, ou comme on les appelle, les couches entièrement connectées, sont responsables de la classification basée sur les caractéristiques apprises. La couche dense finale est une unité unique dotée d'une fonction d'activation sigmoïde spécialement destinée à la classification binaire. Le modèle est entraîné avec des données étiquetées et optimisé avec Adam qui a un taux d'apprentissage de 0,001. Il utilise également les fonctions d'activation ReLU et Sigmoïde afin de générer des prédictions de classification binaire.

3.7 Configuration des expériences et des paramètres

La configuration des expériences et des paramètres pour l'entraînement et l'évaluation de modèles de détection des botnets P2P est une étape cruciale pour obtenir des résultats fiables et généralisables. L'expérimentation itérative et la compréhension approfondie des données et des modèles sont indispensables pour obtenir des résultats optimaux. Voici quelques aspects qu'on a considéré lors de la configuration de nos expériences et paramètres :

1. **Choix des données d'entraînement, de validation et de test :** Nous avons divisé nos données en ensembles d'entraînement, de validation (80% de dataset pour les deux) et de test (20% de dataset) de manière à avoir une représentation équilibrée du trafic P2P légitime et des botnets P2P.

2. **Pré-traitement des données :** Nous avons déterminé les techniques de prétraitement appropriées pour nos données, telles que la normalisation, la standardisation, l'étiquetage etc. Nous avons assuré aussi que le pré-traitement des données est cohérent entre les ensembles d'entraînement, de validation et de test pour éviter tout biais.

3. **Architecture du modèle :** Nous avons choisi des architectures de modèles appropriées (DNN et CNN) en fonction de la complexité et de la nature des données.

C'est à dire qu'on a expérimenté avec différentes architectures de DNN et de CNN, en ajustant le nombre de couches, la taille des filtres, le nombre de neurones par couche, etc. Nous avons aussi considéré l'utilisation de techniques telles que le dropout et la régularisation pour éviter le sur-apprentissage.

4. **Hyperparamètres d'entraînement :** Nous avons bien choisi les hyperparamètres d'entraînement tels que le taux d'apprentissage, le nombre d'époques, la taille du lot (batch size), etc pour l'optimisation.

5. **Fonction de perte et optimiseur :** Nous avons choisi une fonction de perte appropriée pour nos tâches de classification, telle que la perte d'entropie croisée pour la classification binaire (binary_cross-entropy loss). On a expérimenté également avec l'optimiseur tels que Adam.

6. **Métriques d'évaluation :** Ici, nous avons sélectionné les métriques d'évaluation appropriées pour mesurer les performances de notre modèle, telles que la précision, le rappel, le score F1, l'aire sous la courbe ROC (AUC), etc.

3.8 Méthodes d'évaluation de la performance du modèle

Les méthodes d'évaluation jouent un rôle crucial dans l'évaluation des performances et de l'efficacité des modèles d'apprentissage profond. Ces métriques fournissent des mesures quantitatives pour évaluer la capacité du modèle à classer avec précision le trafic réseau et à détecter les botnets P2P potentiels. Dans cette section, nous explorerons plusieurs méthodes d'évaluation clés que nous avons utilisées pour détecter les botnets P2P.

3.8.1 Matrice de confusion

La matrice de confusion permet la visualisation pour un problème de classification des résultats de la manière synthétique comme dans notre cas de détection des botnets P2P. Elle permet de mesurer la qualité d'un problème de classification et cette matrice est indispensable pour déterminer les différentes métriques de classification telles que l'Accuracy, le F1-score ou encore l'AUC et la courbe ROC.

1. **Vrai positif (True Positive ou TP) :** Les vrais positifs sont des instances correctement classées comme positives (intrusions de botnet P2P) par le modèle d'apprentissage profond utilisé. Ces sont des cas positifs réels prédits comme positifs.

2. **Vrai négatif (True Negative ou TN) :** Les vrais négatifs sont des instances correctement classées comme négatives (trafic réseau légitime) par le modèle CNN. Il s'agit d'instances négatives réelles prédites comme négatives.

3. **Faux positif (False Positive ou FP) :** Les faux positifs sont des instances incorrectement classées comme positives (trafic de botnet P2P malveillant) par le modèle CNN. Il s'agit d'instances négatives réelles (trafic légitime normal) prédites comme positives.

4. **Faux négatif (False Negative ou FN) :** Les faux négatifs (FN) sont des instances classées à tort comme négatives (trafic légitime normal) par le système de détection d'intrusion. Il s'agit d'instances positives réelles (trafic de botnet P2P) prédites comme négatives.

3.8.2 Exactitude (Accuracy)

L'accuracy mesure l'exactitude globale des prédictions du modèle d'apprentissage profond. Il est calculé en divisant le nombre d'instances correctement classées (vrais positifs et vrais négatifs) par le nombre total d'instances.

3.8.3 Précision

La précision mesure la proportion d'instances positives correctement identifiées (vrais positifs) par rapport à toutes les instances prédites comme positives. Il se concentre sur la minimisation des prédictions faussement positives.

3.8.4 Rappel

Le rappel, également connu sous le nom de sensibilité ou taux de vrais positifs (TPR), mesure la proportion d'instances positives réelles qui sont correctement identifiées par

le modèle d'apprentissage profond. Il se concentre sur la minimisation des prédictions faussement négatives.

3.8.5 F1-score

Le score F1 est une mesure combinée qui équilibre la précision et le rappel, fournissant une valeur unique qui représente les performances du modèle. C'est la moyenne harmonieuse de la précision et du rappel, avec une importance égale accordée aux deux métriques.

3.8.6 False Positive Rate (FPR)

Le taux de faux positifs mesure la proportion d'instances prédites comme positives (trafic de botnet P2P illégitime malveillant) de manière incorrecte, parmi toutes les instances qui sont réellement négatives (trafic légitime normal). Il est calculé comme suit :

$$FPR = \frac{FP}{FP + TN} \tag{3.1}$$

3.9 Conclusion

Durant ce chapitre instructif, nous avons présenté notre méthodologie novatrice axée sur l'utilisation des algorithmes d'apprentissage profond pour élaborer un modèle robuste de détection de botnets P2P. Cette approche, spécifiquement élaborée pour contrer les attaques de botnets P2P, exploite pleinement les capacités intrinsèques de ces algorithmes, assurant ainsi une détection rapide, une identification précise et une minimisation efficace de ces activités cybernétiques malveillantes, renforçant ainsi la sécurité du réseau.

Chapitre 4

Résultats et analyses

4.1 Introduction

Dans ce chapitre, nous allons discuter des résultats et analyser nos recherches sur le développement d'un modèle de détection des botnets P2P à l'aide de techniques d'apprentissage profond. Nous aborderons l'effet et l'impact des hyper-paramètres, l'évaluation de la performance du modèle sur l'ensemble de données d'entraînement, de validation et de test. Nous allons proposer une solution future pour implémenter le travail dans un IDS.

4.2 Évaluation de la performance du modèle sur différents ensembles de données

Le but de notre travail était d'étudier les performances des modèles d'apprentissage profond pour reconnaître correctement les attaques de botnets P2P et les séparer du trafic réseau régulier et normal. Dans notre recherche, nous avons travaillé avec le modèle CNN et ce modèle a été choisi en raison de sa capacité démontrée pour gérer des modèles de données complexes et des informations séquentielles.

Nous avons entraîné notre modèle CNN sur l'ensemble de données CTU-13 (scénario n°12) combiné avec des flux de trafic de "Ground-truth" issus de l'ensemble de données HIKARI en utilisant une taille de lot de 128, un taux d'apprentissage de 0,001 et 10 époques, avec 80% pour l'entraînement et la validation, et puis 20% pour le test. Cet entraînement a duré 134 secondes sur 10 époques donc 13 secondes par itération.

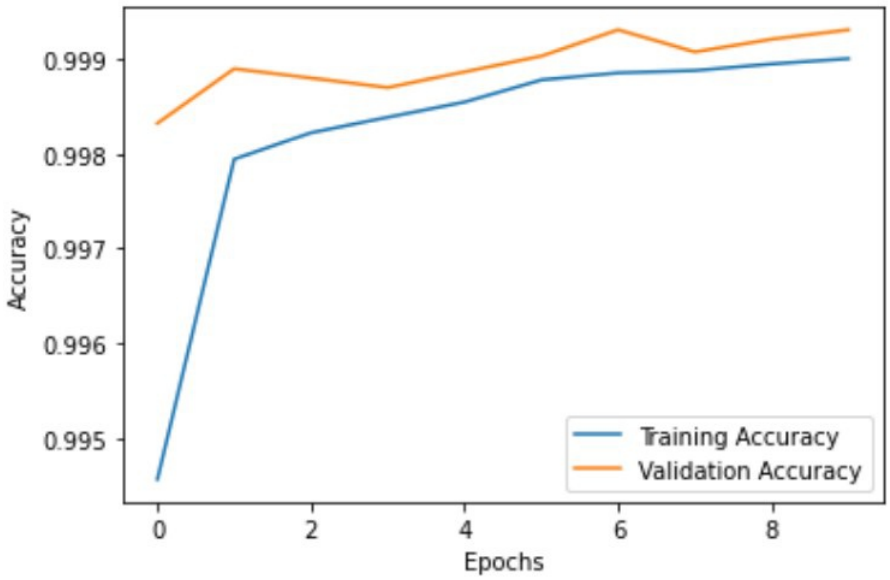

Figure 4.1 – Accuracy CNN

La figure 4.1 montre l'accuracy d'entraînement, la plus élevée est à l'époque 9, elle est de 99,8% et l'accuracy la plus faible est à l'époque 0, elle est de 99,4%, pour l'accuracy de validation, la plus élevée est à l'époque 9 elle est de 99,95% et la plus faible est à l'époque 0 elle est de 99,84%.

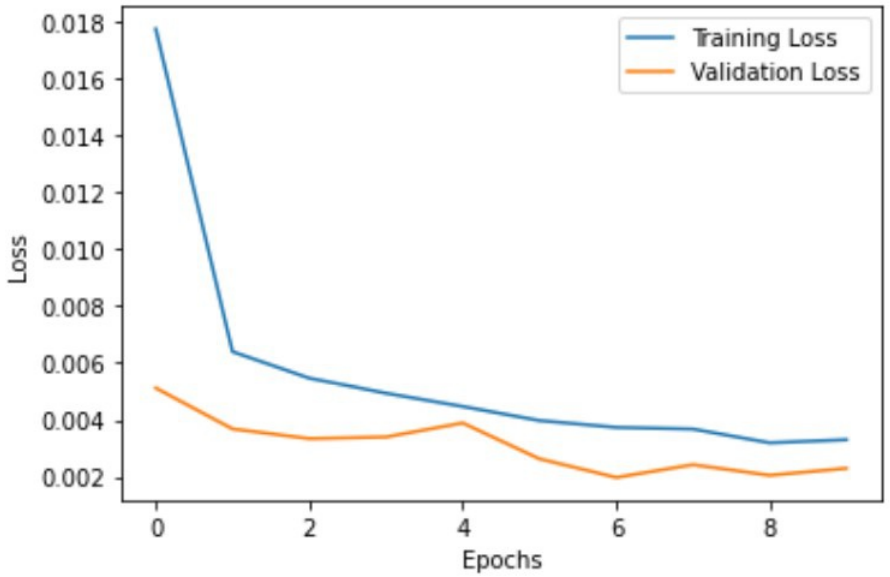

Figure 4.2 – Loss CNN

La figure 4.2 montre la perte d'entraînement et de validation exécutant le modèle CNN sur 10 époques. La perte d'entraînement la plus élevée est à l'époque 0, elle est de 0,018

et la perte d'entraînement la plus faible est à la 9ème époque, elle est de 0,004. La perte de validation la plus élevée est de 0,005 et la perte de validation la plus faible est à la 6ème époque, elle est de 0,002.

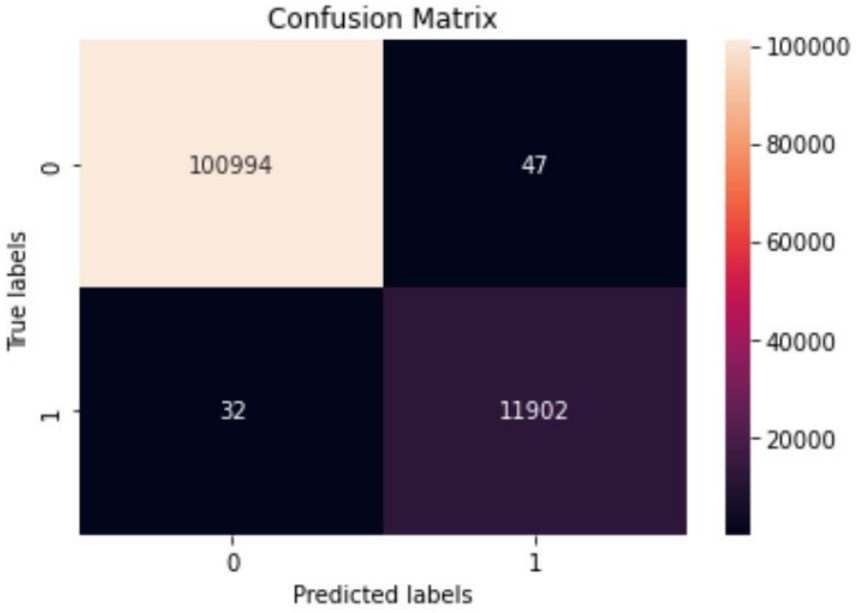

Figure 4.3 – Matrice de confusion CNN

La matrice de confusion fournit une information détaillée des prédictions du modèle CNN par rapport aux étiquettes de vérité réelles ou à la réalité. La matrice de confusion dans ce scénario présente deux classes, c'est-à-dire la classe 0, qui représente le trafic légitime normal, et la classe 1, qui représente les attaques de botnets P2P. La matrice de confusion est composée de quatre parties comme suit :

1. **Vrai positif (True Positive ou TP) :** Les vrais positifs sont des instances correctement classées comme positives (intrusions de botnet P2P) par le modèle d'apprentissage profond CNN. Ces sont des cas positifs réels prédits comme positifs. Le nombre d'attaques de botnets P2P anticipées avec précision est représenté dans l'élément en bas à droite. Dans ce scénario, le modèle CNN a correctement reconnu les tentatives de botnets P2P, produisant un total de 11 902 instances.

2. **Vrai négatif (True Negative ou TN) :** Les vrais négatifs sont des instances correctement classées comme négatives (trafic réseau légitime) par le modèle CNN. Il s'agit d'instances négatives réelles prédites comme négatives. Le nombre d'instances légitimes normaux anticipés avec précision est représenté dans l'élément

supérieur gauche. Le modèle CNN a correctement reconnu 100 994 instances de trafic légitime normal dans ce scénario.

3. **Faux positif (False Positive ou FP) :** Les faux positifs sont des instances incorrectement classées comme positives (trafic de botnet P2P malveillant) par le modèle CNN. Il s'agit d'instances négatives réelles (trafic légitime normal) prédites comme positives. Le nombre d'instances légitimes normales identifiées par erreur comme des attaques de botnets P2P est indiqué dans l'élément supérieur droit. Dans ce cas, la version CNN a identifié à tort 47 instances légitimes normales comme des attaques de botnets P2P.

4. **Faux négatif (False Negative ou FN) :** Les faux négatifs (FN) sont des instances classées à tort comme négatives (trafic légitime normal) par le modèle CNN de détection des botnets P2P. Il s'agit d'instances positives réelles (trafic de botnet P2P) prédites comme négatives. Le nombre d'attaques de botnets P2P authentiques manquées par le modèle CNN est indiqué dans l'élément en bas à gauche. La matrice de confusion affiche 32 faux négatifs.

Tableau 4.1 – Les métriques d'évaluation pour notre modèle CNN

N°	Métrique	Valeur	Pourcentage (%)
1	Accuracy	0.9994	99.94
2	Précision	0.9956	99.56
3	Recall (Rappel)	0.9945	99.45
4	F1-score	0.9950	99.50
5	AUC	1.000	100

Ce tableau 4.1 décrit le rapport de classification du modèle CNN, notamment les valeurs de précision des performances du modèle. Une exactitude (un accuracy) du test de 99.94% indique que les vrais positifs et les vrais négatifs ont été correctement prédits parmi le nombre total d'instances examinées. De même, une précision de 99.56% suggère que presque toutes les prédictions positives ont été correctement classées.

Le rappel détecte les cas positifs et il est de 99.45%, ce qui suggère que le modèle CNN a détecté avec succès 99.45% des cas positifs. Le score F1, c'est-à-dire la moyenne harmonique de la précision et du rappel combinés, est de 99.50%, indiquant une bonne combinaison de précision et de rappel.

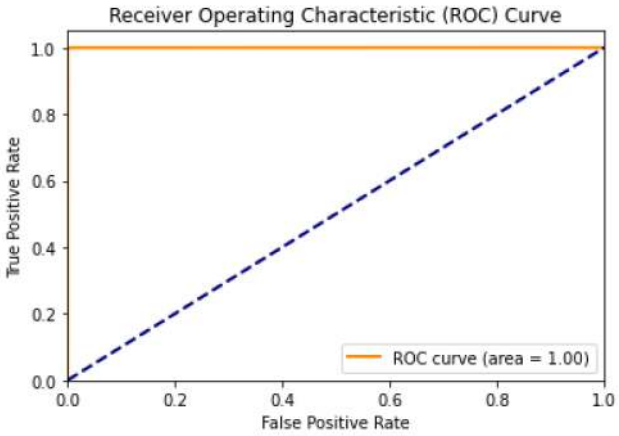

Figure 4.4 – Courbe ROC du modèle CNN

La figure 4.4 montre une courbe ROC qui forme un angle de près de 90 degrés au-dessus de la ligne médiane, indiquant que le modèle de catégorisation a très bien fonctionné. Cette forme désigne un modèle CNN avec un taux de vrais positifs (TPR) élevé et un faible taux de faux positifs (FPR). L'AUC (la zone taux de vrais positifs) est de 100% qui est équivalent à 1 pour un modèle idéal (classificateur parfait).

4.3 Proposition d'une solution future pour implémenter le travail dans un IDS

Après le développement et évaluation d'un modèle de deep learning, une fois que le modèle fonction parfaitement, la prochaine étape est son implémentation dans un système de détection d'intrusion ou un système de prévention d'intrusion. Ainsi notre modèle peut être implémenter dans un IDS/IPS comme Snort ou Suricata, une des meilleur méthode est de développer une extension (plug in) qui sera connecté à un IDS/IPS, grâce à une interface de programmation d'application API.

Grâce à cette API, on peut connecter le modèle au IDS/IPS sans pour autant modifier le code source de IDS/IPS, ainsi le système IDS/IPS sera modifié pour qu'il puisse envoyer des données de trafic réseau à l'API et recevoir les résultats des prédictions. Une API peut être développer à l'aide de framework comme **Flask** ou **FastAPI** en Python pour plus de cohérence, vue que notre modèle est développé avec python. Ensuite cette API doit être déployé sur un serveur ou sur le cloud et en fin intègre d'API à un système.

4.4 Conclusion

Dans ce chapitre, nous avons exploré les résultats de nos recherches sur le développement d'un modèle de détection de botnets P2P en utilisant des techniques d'apprentissage profond. Nous avons discuté de l'impact des hyperparamètres sur les performances du modèle, évalué ses performances sur les ensembles de données d'entraînement, de validation et de test, et comparé notre approche avec d'autres méthodes de détection de botnets P2P. Enfin, nous avons proposé des solutions futures pour intégrer notre modèle dans un système de détection d'intrusion (IDS) ou de prévention d'intrusion (IPS), permettant une surveillance pro-active et réactive des réseaux contre les menaces posées par les botnets P2P.

Conclusion Générale

Dans ce mémoire, nous avons exploré le développement d'un modèle de détection de botnets P2P en utilisant des techniques d'apprentissage profond. À une époque où la connectivité mondiale et les volumes de données atteignent des niveaux sans précédent, il est impératif de disposer de systèmes de détection d'intrusion robustes et fiables pour protéger les réseaux contre des attaques de plus en plus sophistiquées.

Nous avons démontré comment les réseaux de neurones profonds peuvent être utilisés pour identifier des comportements malveillants dans le trafic réseau, en se concentrant spécifiquement sur les botnets P2P. Notre approche permet de détecter efficacement les activités suspectes sans nécessiter une inspection exhaustive des paquets ou une analyse complète du trafic réseau.

Nos résultats indiquent que le modèle CNN démontre une grande efficacité dans la détection des botnets P2P, atteignant une exactitude (un accuracy) de 99.94% et une précision de 99.56% avec un taux de faux positifs (FPR) de 0.0465%.

Ces résultats montrent que les techniques d'apprentissage profond offrent une solution efficace pour la détection des botnets P2P, avec un taux de détection élevé et une capacité à traiter de grandes quantités de données. Cependant, des défis subsistent, notamment la vulnérabilité aux attaques. Nous proposons des pistes d'amélioration pour renforcer la robustesse et l'efficacité de notre modèle.

Il est crucial de reconnaître les limites de cette étude. L'un des défis majeurs était le processus fastidieux de nettoyage d'un vaste ensemble de données de haute qualité. De plus, la formation de modèles d'apprentissage profond nécessite des ressources informatiques considérables, et nos contraintes matérielles limitaient l'échelle et la complexité des modèles que nous avons pu utiliser.

Cette étude souligne l'importance et le potentiel des techniques d'apprentissage profond pour améliorer la sécurité des réseaux contre les menaces posées par les botnets P2P. L'intégration future de notre modèle dans un IDS/IPS pourrait offrir une surveillance pro-active et réactive, renforçant ainsi la résilience des réseaux face aux cyber-menaces.

Bibliographie

[1] Blondel Seumo Ntsiepdjap. Dynamic risk assessment for critical infrastructures under attack. 2022.

[2] https://www.nameshield.com/wp-content/uploads/2020/03/DDoS.jpg,consulté le 10-05-2024.

[3] https://static.vecteezy.com/system/resources/previews/000/173/207/original/phishing-illustration-vector.jpg,consulté le 10-05-2024.

[4] https://sysblog.informatique.univ-paris-diderot.fr/wp-content/uploads/2020/03/database-classic1.png,consulté le 10-05-2024.

[5] https://miro.medium.com/v2/resize:fit:512/1*kI00egJ-dAbQwBd_qPAi_A.png,consulté le 10-05-2024.

[6] https://th.bing.com/th/id/OIP.dv5rC1Syy-4Zb4rFFNAz2QAAAA?rs=1&pid=ImgDetMain,consulté le 10-05-2024.

[7] https://www.nknews.org/pro/wp-content/uploads/sites/4/2023/08/Trojan-Chollima-illustration-935x500.jpg,consulté le 10-05-2024.

[8] Nor Badrul Anuar Meisam Eslahi, Rosli Salleh. Bots and botnets : An overview of characteristics, detection and challenges. pages 349–354, 2012.

[9] François Chollet. *Deep Learning with python*. MANNING Publications, SHELTER ISLAND, 2018.

[10] Aurélien Géron. *Deep Learning avec Keras et Tensorflow : Mise en oeuvre et cas concrets*. O'Reilly Media, 2020.

[11] DEPARTMENT OF COMPUTATIONAL INTELLIGENCE. *Lecture Notes on Neural Networks*. MALLA REDDY COLLEGE OF ENGINEERING TECHNOLOGY, INIDA, 2022.

[12] https://culturesciencesphysique.ens-lyon.fr/images/articles/ia-rousseau/fig2.png, consulté le 30-04-2024.

[13] https://miro.medium.com/v2/resize:fit:1400/1*pmFsxtfiFoo09rXLaKyrOg.png, consulté le 30-04-2024.

[14] https://www.researchgate.net/publication/333411007/figure/fig7/AS:766785846525952@1559827400204/ReLU-activation-function.png, consulté le 30-04-2024.

[15] https://ambrapaliaidata.blob.core.windows.net/ai-storage/articles/Untitled_design_13.png, consulté le 30-04-2024.

[16] https://miro.medium.com/v2/resize:fit:443/1*WeuJzmlt3iNVWsUsvf24Eg.png, consulté le 30-04-2024.

[17] https://www.datocms-assets.com/96965/1703090208-image16.png, consulté le 30-04-2024.

[18] https://commons.wikimedia.org/wiki/File:Typical_cnn_fr.png,consulté le 04-05-2024.

[19] Benslim Borhan Eddine Mahi Sid Ali. Network intrusion detection system using deep learning. 2022-2023.

[20] Bernd Klein. *Machine learning with python tutorial*. Bodenseo, 2021.

[21] https://image.slideserve.com/312588/slide27-l.jpg,consulté le 30-05-2024.

[22] https://www.idbc.fr/tutoriel-comment-lire-une-courbe-roc-et-interpreter-son-auc/, le 02-05-2024.

[23] María T. López Antonio Fernández-Caballero Roberto Sánchez-Reolid, Francisco López de la Rosa. One-dimensional convolutional neural networks for low/high arousal classification from electrodermal activity. *European Alliance of Medical and Biological Engineering and Science (EAMBES).*, 2022.

[24] https://datascientest.com/convolutional-neural-network,consulté le 04-05-2024.

[25] https://www.theverge.com/2020/6/18/21295337/, consulté le 18-04-2024.

[26] https://www.zdnet.com/article/, consulté le 18-04-2024.

[27] https://www.lemondeinformatique.fr/actualites/, consulté le 18-04-2024.

[28] Jean-François PILLOU & Jean Philippe BAY. *Tout sur la Sécurité informatique*. Dunod, Paris, 2013.

[29] L. Zhang R. Alasem M. Alauthman, N. Aslam and M. A. Hossain. A p2p botnet detection scheme based on decision tree and adaptive multilayer neural networks. *Neural Computing and Applications*, pages 1–14, 2016.

[30] http://nrl.northumbria.ac.uk/id/eprint/29617, consulté le 19-04-2024.

[31] N. Aslam O. Dorgham A. Almomani, M. Alauthman and M. Al-Refai. *Computer and Cyber Security : Principles, Algorithm, Applications and Perspectives*. CRC Press, USA, 2018.

[32] F. Meziane A. Alnawasrah, A. Almomani and M. Alauthman. Fast flux botnet detection framework using adaptive dynamic evolving spiking neural network algorithm. 2018.

[33] F. Jahanian E. Cooke and D. McPherson. The zombie roundup : Understanding, detecting, and disrupting botnets. 39(44), 2005.

[34] M. Anbar M. Alauthman R. Abdullah K. Alieyan, A. Almomani and B. Gupta. Dns rule-based schema to botnet detection. *Enterprise Information Systems*, pages 1–20, 2019.

[35] O. Almomani A. Almomani, M. Alauthman and F. Albalas. A proposed framework for botnet spam-email filtering using neucube. *IEEE Transactions on Information Forensics and Security, Vol. 10, No. 8, pp. 1745-1755, 2015l*, 2017.

[36] M. Alkasassbeh A. Rawashdeh and M. Al-Hawawreh. An anomaly-based approach for ddos attack detection in cloud environment. 57(4) :312–324, 2018.

[37] R. C. G. Pinto S. S. C. Silva, R. M. P. Silva and R. M. Salles. Botnets : A survey. 57 :378–403, 2013.

[38] C. Nunnery B. Kang J. Grizzard, V. Sharma and D. Dagon. Peer-to-peer botnets : overview and case study. 2007.

[39] https://www.mersenne.org/various/history.php, consulté le 22-04-2024.

[40] D. Anderson J. Cobb E. Korpela, D. Werthimer and M. Leboisky. Seti@home-massively distributed computing for seti. 3 :78–83, 2001.

[41] D. Spitz and S. D. Hunter. Contested codes : The social construction of napster. 21 :169–180, 2005.

[42] European Union Agency for Cybersecurity (EU body or agency). Enisa threat landscape report 2018 : 15 top cyber-threats and trends. 2019.

[43] Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. 31 :211–235, 2017.

[44] M. Alauthman. P2p bot detection using deep learning with traffic reduction schema. *Journal of Theoretical and Applied Information Technology, JATIT*, 98(15), 2020.

[45] F. Jahanian X. Yunjing M. Bailey, E. Cooke and M. Karir. A survey of botnet technology and defenses. pages 299–304, 2009.

[46] J. Kok and B. Kurz. Analysis of the botnet ecosystem. pages 1–10, 2011.

[47] J. Zhang G. Gu, R. Perdisci and W. Lee. Botminer : Clustering analysis of network traffic for protocol and structure independent botnet detection. pages 139–154, 2008.

[48] C. Hung-Chang K. Tung-Ming and W. Guo-Quan. Construction p2p firewall http-botnet defense mechanism. pages 33–39, 2011.

[49] P. Jun-Hyung K. Minsoo L. Jae-Seo, J. HyunCheol and N. Bong-Nam. The activity analysis of malicious http-based botnets using degree of periodic repeatability. pages 83–86, 2008.

[50] J. Zhang G. Gu and W. Lee. Botsniffer : Detecting botnet command and control channels in network traffic. 2008.

[51] J. Hyun-chul J. Dae-il, K. Minsoo and N. Bong-Nam. Analysis of http2p botnet : Case study waledac. pages 409–412, 2009.

[52] H. Y. Liao T. T. Lu and M. F. Chen. An advanced hybrid p2p botnet 2.0. 81 :595–597, 2011.

[53] D. Harley G. Evron T. Bradley C. Willems C. Schiller, J. Binkley and M. Cross. *Botnets : The Killer Web Application.* Syngress, 2007.

[54] G. G. Granadillo N. Hachem, Y. Ben Mustapha and H. Debar. Botnets : Lifecycle and taxonomy. pages 1–20, 2011.

[55] J. Govil. Examining the criminology of bot zoo. pages 1–6, 2007.

[56] H. Lee H. Choi and H. Kim. Botgad : Detecting botnets by capturing group activities in network traffic. pages 1–8, 2009.

[57] K. Minsoo J. Hyun-chul J. Dae-il, C. Kang-yu and N. Bong-Nam. Evasion technique and detection of malicious botnet. pages 1–5, 2010.

[58] Y. Xiang Y. Zhou Y. Zhang, W. Li. Detecting stealthy p2p botnets using statistical traffic fingerprints. 10(8) :1745–1755, 2015.

[59] J. Goebel and T. Holz. Rishi : Identify bot contaminated hosts by irc nickname evaluation. pages 8–20, 2007.

[60] R. K. Sehgal J. S. Bhatia and S. Kumar. Honeynet based botnet detection using command signatures. 154 :69–78, 2011.

[61] J. Wei L. Chao and Z. Xin. Botnet : Survey and case study. pages 1184–1187, 2009.

[62] Sunny Behal, Amanpreet Brar, and Krishan Saluja. Signature-based botnet detection and prevention.

[63] Ric Messier. *Network Forensics.* John Wiley Sons, Inc., 2017.

[64] M. Grill J. Stiborek K. Bartos M. Rehak, M. Pechoucek and P. Celeda. Adaptive multiagent system for network traffic monitoring. 24 :16–25, 2009.

[65] D. Drapeau D. Burke A. Caglayan, M. Toothaker and G. Eaton. Behavioral analysis of botnets for threat intelligence. 2011.

[66] Mohammed Anbar Selvakumar Manickam Arkan Hammoodi Hasan, Achmad Husni Thamrin and Shankar Karuppayah. Multi-layer perceptron to detect peer-to-peer botnet. *PeerAmbush*, Symmetry 14(12) :2483, 2022.

[67] https://blent.ai/blog/a/rnn-on-vous-explique-tout,consulté le 30-04-2024.

[68] https://www.mdpi.com/2076-3417/11/17/7868, HIKARI-2021 dataset : consulté le 07-03-2024.

[69] https://www.stratosphereips.org/datasets-ctu13TU13 Dataset : consulté le 07-03-2024 .

[70] https://www.wireshark.org/,consulté le 30-05-2024.

[71] https://www.anaconda.com/blog/anaconda-training-a-learning-path-for-data-scientis consulté le 09-05-2024.

[72] https://numpy.org/, consulté le 09-05-2024.

[73] https://pandas.pydata.org/, consulté le 09-05-2024.

[74] https://keras.io/, consulté le 09-05-2024.

[75] https://www.tensorflow.org/?hl=fr, consulté le 09-05-2024.

[76] https://scikit-learn.org/stable/, consulté le 09-05-2024.

[77] https://matplotlib.org/, consulté le 09-05-2024.