République algérienne démocratique et populaire
Ministère de l'enseignement supérieur et de la recherche scientifique

Université Saad Dahleb Blida

Faculté des sciences
Département Informatique

Détection d'attaques de déni de service basé sur l'apprentissage automatique

Mémoire entrant dans le cadre de l'obtention du diplôme de Master en informatique

---

Master Sécurité des Systèmes d'Informations

Réalisé par : Assam Samy *Encadré par :* Dr. Bacha

Septembre 2021

Résumé

Les attaques de déni de service représentent une réelle menace pour les systèmes informatiques, car, d'une part elles surgissent d'une manière soudaine et d'une autre part, elles sont relativement faciles à mettre en oeuvre. Dans cette étude nous cherchons une solution pour détecter ces attaques en se basant sur des modèles et des techniques d'apprentissage automatiques.

L'usage de l'apprentissage automatique en cybersécurité apporte un réel avantage, quand il s'agit de traitement complexe à mettre en oeuvre, tel que la détection d'agent malveillant dans un fichier, de classer des e-mails ou encore de classifier un trafic réseau.

Notre système de protection est composé de trois modules. Le premier module concerne *l'acquisition des données*, notre choix se porte sur l'utilsation d'un programme pour extraire des données statistiques depuis des flux réseaux en format Comma-separated values (CSV). Le deuxième module intitulé *Transformation des données*, où nous verrons plusieurs techniques de transformation des données acquises et prétraitées du module précédent. Le troisième module, est le module de *classification*. Nous y verrons deux classificateurs d'apprentissage automatique et des modèles basés sur des réseaux de neurones artificiels. Forêts aléatoires, arbres de décision, réseau de neurones denses et auto-encodeur seront mis à l'épreuve sur une suite de tests, qui détermineront les paramètres et modèles à utiliser pour l'évaluation globale. Nous utiliserons la base de données CIC IDS 2018 qui comporte une grande quantité de données de trafic réseau d'attaque et normal. Certaines combinisations (entre modèles et paramètres d'entrées) se verront plus performant que d'autres, et avoisinneront 0.99 en exactitude.

Mots clés :

Réseau - Machine Learning - Classification - Deep Learning - Auto-encodeur - DNN - Déni de service - DDoS

Abstract

Distributed Deny Of Service (DDoS) attacks represents a real threat for computer systems, on the one hand they appear suddenly and on the other hand they are relatively easy to implement. In this study we are looking for a solution to detect these attacks based on machine learning models and techniques, such as detecting malicious agents in a file, e-mails spam classification or classifying network traffic. Our protection system is made up of 3 essential modules and 1 optional module. The first module concerns data acquisition, our choice is the use of a program to extract statistical data from network streams in CSV format. Module 2 called Data Transformation, where we will see several techniques for transforming acquired data and preprocessing it. Module 3, being the last, is the most important module. We will see two machine learning classifiers and two models based on artificial neural networks. Random forests, decision trees, dense neural network and auto-encoder will be compared in a suite of tests, which will determine the parameters and models to be used for the overall assessment. We will use the CICIDS 2018 database [Sharafaldin et al., 2018] which has a large amount of data extracted from captured Packet Capture packets (PCAP) using the CIC FlowMeter program [Sharafaldin et al., 2019a] to train models to distinguish between normal and malicious network flow. We will approach 0.99 in precision for the selected models including the parameters on the data namely if the data is normalized, encoded or even selected.

Keywords :

Netowrk - Machine Learning - Classification - Deep Learning - Auto encoder - DNN - Denial of service - DDoS

ملخص

تشكل هجمات رفض الخدمة DOS ، تهديدا حقيقيا لأنظمة الكمبيوتر. تتمثل خطورتها في سهولة تنفيذها من جهة ، و في حدوثها المفاجئ من جهة أخرى. الهدف من هذه الدراسة هو البحث عن حل يسمح بكشف هذه الهجمات باستعمال نماذج التعلم الآلي .

التعلم الآلي يضيف إلى الأمن السيبراني تسهيلات كثيرة ، خصوصا حين يتعلق الأمر بمعالجات معقدة مثل كشف برمجية خبيثة في ملف، تصنيف رسائل البريد الإلكتروني ، تصنيف التدفق الشبكي. نظام الحماية لدينا يتكون من ثلاث وحدات.

الوحدة الأولى متعلقة بجمع البيانات، و هذا باستخدام برنامج لاستخراج البيانات الإحصائية من تدفقات الشبكة على شكل قيم مفصولة بفواصل (comma separated values - CSV)

الوحدة الثانية هي تحويل البيانات المكتسبة و المعالجة مسبقا و هذا باستغلال عدة تقنيات .

الوحدة الثالثة تشتمل على التصنيف (classification) . سيتم الاعتماد على مصنفي التعلم الآلي (arbres de décisions - forêts aléatoires) و كذلك على نماذج شبكات الخلايا العصبية الاصطناعية (réseaux de neurones - denses - auto encoders) . تخضع جميع هذه النماذج إلى مجموعة اختبارات متتالية بهدف ضبط الاعدادات لكل نموذج (configuration des paramètres) . و تحديد النماذج الأكثر فعالية من أجل التقييم النهائي. قاعدة البيانات المستعملة هي CIC IDS 2018 . التي تشمل كم هائل من بيانات التدفق الشبكي بما فيه الهجمات و التدفقات العادية.

الكلمات الدالة

الشبكة ـ التعلم الآلي ـ التصنيف ـ التعلم العميق ـ المشفر التلقائي ـ DNN ـ رفض الخدمة ـ DDoS

Remerciements

Je tiens à exprimer toute ma reconnaissance à ma promotrice de mémoire, Madame Sihem Bacha. Je la remercie de m'avoir encadré, encouragé, orienté, aidé, conseillé et surtout de m'avoir inspiré depuis la licence.

J'adresse mes sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de répondre à mes questions durant mes recherches.

Je remercie mes parents et ma famille pour leurs encouragements.

Enfin, je remercie mes amis Mamadou, Dymia, Abdelmoumene, Amine, Louise, et Zineddine qui ont toujours été là pour moi.

Leur soutien inconditionnel et leurs encouragements ont été d'une grande aide.

À tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.

Table des matières

1 Introduction générale **13**

1.1 Contexte 14

1.2 Problèmatique 15

1.3 Objectifs 15

1.4 Organisation du mémoire 15

2 Revue de la littérature **16**

2.1 Réseau Internet 16

2.1.1 Notions globales 16

2.1.2 Paquets et flux réseaux 17

2.2 Déni de service 19

2.3 Types d'attaques de déni de services 19

2.3.1 Attaques basées sur le volume 19

2.3.2 Attaques protocolaire 21

2.3.3 Attaques de la couche d'application 21

2.4 Solutions et techniques de détections des attaques de déni de service 23

2.4.1 Données en entête 23

2.4.2 Inpsection appronfondie des paquets 24

2.4.3 Cybersécurité et apprentissage automatique 24

3 Apprentissage automatique **35**

3.1 Apprentissage automatique 35

3.1.1 Présentation et définition 35

3.1.2 Types d'apprentissage automatique 36

3.1.3 Classfication 37

3.2 Codage des données 38

3.2.1 One Hot Encoding . . . 38
3.2.2 Binary Encoding . . . 40
3.3 Normalisation . . . 40
3.3.1 MinMaxScaler . . . 41
3.3.2 StandardScaler . . . 41
3.4 Deep learning . . . 42
3.4.1 Couches . . . 43
3.4.2 Fonctions d'activations . . . 44
3.4.3 Fonctions d'erreurs . . . 47
3.4.4 Auto encodeur . . . 47
3.4.5 Détection d'anomalies . . . 47
3.5 Métriques de performances . . . 48

4 Conception **50**
4.1 Architecture . . . 50
4.2 Module 1 : Acquisition des données . . . 51
4.2.1 Type de données . . . 51
4.2.2 Prétaitements des données . . . 52
4.2.3 Sélection des caractéristiques . . . 52
4.3 Module 2 : Transformation des données . . . 56
4.3.1 Encodage . . . 56
4.3.2 Normalisation . . . 60
4.4 Module 3 : Classification . . . 60
4.4.1 Fôrets aléatoires et Arbres de décision . . . 61
4.4.2 Réseau de neurones profond . . . 61
4.4.3 Auto-encodeur . . . 63
4.5 Module 4 : Déploiement . . . 64

5 Implémentation et évaluation **65**
5.1 Environnement et outils de travail . . . 65
5.1.1 Matériels . . . 65
5.1.2 Langages de programmation et modules . . . 65
5.2 Base de données . . . 66
5.2.1 Base de données de tests . . . 69
5.2.2 CSE CIC IDS 2018 . . . 69
5.3 Tests des paramètres . . . 70

5.4 Évaluation du modèle 74

6 Conclusion générale **76**

Table des figures

2.1 Modèle OSI et protocole TCP/IP - Source : [Sery, 2014] 17
2.2 Adresse IPv4 - Source : [Picaxe, 2014] 18
2.3 Dissection du datagramme IPv4 [Anjum, 2011] 18
2.4 Déni de service [DNSstuff, 2019] 20
2.5 Inondation UDP - [ion, 2020] 20
2.6 Inondation requête Synchronize (SYN), extraite de [ion, 2020] 22
2.7 Inondation Hypertext Transfer Protocol (HTTP), extraite de CloudFlare.com 23
2.8 Performances Datanet, extraite de [Pan et al., 2018] 29
2.9 Performances CAPC, extraite de [Kai-Cheng et al., 2020] 30
2.10 Matrice de confusion de RF 32
2.11 Matrice de confusion de DeepMAL, extraite de [Marìn et al., 2020] 33

3.1 Intelligence artificielle 35
3.2 Exemple d'arbre de décision 37
3.3 Encoding des caractéristiques 38
3.4 Exemple de base de données 39
3.5 Exemple de valeur unique du protocole 39
3.6 Exemple d'encodage de l'attribut protocole 40
3.7 Comparaison de la normalisation des données 41
3.9 Illustration représentante d'un neurone artificiel 42
3.8 Apprentissage automatique et apprentissage profond 42
3.10 Réseau de neurones denses 43
3.11 Cellule Long Short Term Memory (LSTM) 43
3.12 Effet de la couche Dropout, image extraite de [Baeldung, 2019] 44
3.13 Graphe fonction ReLU 45
3.14 Graphe fonction Sigmoïde 46
3.15 Graphe fonction Tangente Hyperbolique 46

3.16 Schéma globle d'un auto-encodeur 48
3.17 Détection d'anomalies basé sur l'erreur de reconstruction 49

4.1 Architecture de la solution proposée 50
4.2 Corrélation de Pearson - voir annexe 54
4.3 Valeur P - voir annexe 55
4.4 Type de valeurs catégorielles 57
4.5 Transformation du protocole en vecteur de 16 bits 59
4.6 Module de classification 60
4.7 DNN proposé 62
4.8 Auto-encodeur proposé 63

5.1 Distribution des données de CIC IDS 2018 70
5.2 Classes d'attaques de CIC IDS 2018 71

Liste des abréviations

UDP	User Datagram Protocol
SQL	Structured Query Language
HTTP	Hypertext Transfer Protocol
CIA	Confidentiality, Integrity, Availability
ICMP	Internet Control Message Protocol
DDoS	Distributed Deny Of Service
DoS	Deny Of Service
CNN	Convulsional Neural Network
RNN	Recurrent Neural Network
RBM	Restricted Boltzmann Machine
DBN	Deep Belief Networks
DBM	Deep Boltzmann Machine
DA	Deep Auto encoder.
CAE	Convulsional Auto Encoder
TCP	Transmission Control Protocol
IDS	Intrusion Detection System
IoT	Internet of Things
ML	Machine Learning
DPI	Deep Packet Inspection
SAE	Stacked Auto Encoder
VAE	Varitional Auto Encoder
DNN	Deep Neural Network
LLC	Latent Layer Classification

LBB	Loss Based Detection
AE	Auto Encoder
NTP	Network Time Protocol
SYN	Synchronize
SYN-ACK	Synchronize Acknowledgment
ACK	Acknowledgment
DNS	Domain Name Server
DHCP	Dynamic Host Configuration Protocol
APIPA	Automatic Private Internet Protocol Addressing
API	Application Programming Interface
MLP	Multi-layer Perceptron
OSI	Open Systems Interconnection
PCAP	Packet Capture
CSV	Comma-separated values
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
NAT	Network address translation
IP	Internet Protocol
IPS	Intrusion Prevention Systems
DT	Decision Tree
RF	Random Forest
LSTM	Long Short Term Memory
LOIC	Low Orbit Ion Cannon
Conv	Convulsional
ReLU	Rectified Linear Unit

Liste des algorithmes

1 Prétraitement des données 52
2 One Hot Encoding 57
3 One Hot Encoding - 3b 58
4 Transformation en binaire 58
5 Encodage de l'adresse IP en vecteur de 32 bits 59
6 Normalisation des données numérique à l'aide de MinMaxScaler 60
7 Random Forest (RF) et Decision Tree (DT) proposés 61
8 Classificateur Deep Neural Network (DNN) proposé 62
9 Auto Encoder (AE) proposé 64

Chapitre 1

Introduction générale

Les attaques de déni de services, aussi populaires qu'elles soient, sont très redoutables depuis l'aube d'Internet. Aussi connu sous le nom de DOS (Denial Of Service attack), ce type d'attaques a pour but de perturber un service, un serveur, et de le rendre indisponible aux utilisateurs légitimes. Ce type d'attaque consiste à inonder le système cible avec des requêtes inutiles. Autrefois, les attaques de déni de service étaient menées par des botnets - réseaux d'ordinateurs zombis, une grande quantité de postes compromis - qui submergeait la cible, ce qui engendre un dépassement des ressources disponibles de la cible, la rendant inopérante.

Les utilisateurs d'un service victime de ces attaques peuvent ressentir des lenteurs ou voir celui-ci non disponible. Ces mêmes utilisateurs, parfois, possèdent un système (smartphone, ordinateur) compromis et sont eux même en train d'attaquer le service sans en avoir connaissance. Selon le rapport Baromètre du numérique [CGE et al., 2019], 77% des Français sont équipés d'un smartphone en 2019 contre 17% en 2011. Ce pourcentage est en évolution permanente. Une bonne hygiène informatique est essentielle pour limiter un grand nombre d'attaques et empêcher un attaquant de compromettre son système. [ANSSI, 2015]

Avec la croissance d'Internet, nos smartphones y sont connectés en permanence, des objets connectés Internet of Things (IoT) ; toute machine connectée à Internet, peut être exploitée pour mener une attaque de déni de service Deny Of Service (DoS). Certes, les géants du web comme Google, Microsoft, Amazon ou Apple font en sorte que leurs systèmes soient le plus sécurisés et de les rendre compliqués à compromettre. Un développeur mal intentionné peut écrire une fonction qui exécute une simple requête HTTP/GET ; une fonction qui représentera un code malveillant indétectable qui mettra à sa disposition un nombre important d'équipements tels que des smartphones ou des objets connectés afin d'attaquer un système comme une Application Programming Interface (API) de météo.

Les systèmes de détection d'intrusion sont conçus pour empêcher un trafic d'entrer ou de sortir d'un

système d'informations en sondant celui-ci. Ces équipements fonctionnent avec des règles qui sont vérifiées lorsqu'un paquet les traverse. On pourra citer Snort, le système de détection d'intrusion open source le plus utilisé qui permet d'analyser en temps réel les paquets basés sur des règles. Snort et plein d'autres systèmes de détection d'intrusion se limitent par leur méthode d'analyse.
Les attaques de déni de services représentent une famille d'attaques, certaines distribuées DDoS. Certaines des ces attaques visent une faiblesse dans un protocole. On citera l'attaque par inondation de requête SYN, où le but est de surcharger le serveur avec des connexions ouvertes. Tandis que d'autres visent la couche d'application comme l'inondation de requêtes HTTP.
Suscitant l'intérêt de la communauté scientifique, des approches basées sur l'apprentissage automatique ont fait surface. Certaines de ces approches analysent le flux réseau ou en d'autres termes une disscusion entre deux machines, tandis que d'autres analysent les paquets là où celles-ci representent les messages.

Ces attaques augmentent en fréquence et en volume à une vitesse assez inquiétante. De plus, elles surviennent d'une manière inattendue, et d'appareils a priori légitimes, analyser en temps réel le trafic pour déterminer si un système est victime d'une attaque de déni de service est ce qui semble être une contre-mesure efficace.

1.1 Contexte

La triade Confidentiality, Integrity, Availability (CIA) en français confidentialité, intégrité, et disponibilité, est le pilier de la cyber-sécurité. La confidentialité peut se résoudre à l'aide de droit d'accès par exemple, l'intégrité par des moyens cryptographiques quant à la disponibilité d'un système c'est bien plus complexe.
Un système d'informations peut être atteint sur sa disponibilité depuis plusieurs volets. Cela peut être une panne de courant, rendant le système inaccessible, ou à cause d'une panne d'un composant primaire du système. Plusieurs techniques existent afin d'y remedier tel que le cloud, ou une architecture répartie. Seulement les ressources ne sont pas infinies.
Qu'en est-il des attaques perturbantes la disponibilité d'un système d'informations ?
La première phase que l'attaquant fera, c'est le footprinting ; dans laquelle il découvrira le système d'exploitation, les versions de l'application du serveur ; tout un tas d'informations pour mener son attaque. La seconde sera de déterminer la fenêtre d'attaque ainsi que sa stratégie d'attaque, suite à cela, il mettra en place son attaque. Plusieurs outils tout public et techniques existent pour mener des attaques sur un système. La difficulté de détecter les attaques de déni de services se trouve dans leur manière soudaine de surgir et leur facilité de mise en place. Elles peuvent être de différent niveau d'impact.

1.2 Problèmatique

Comment détecter les attaques de déni de services afin de permettre à un système de protection tel qu'un pare-feu ou à un Intrusion Prevention Systems (IPS) de bloquer les requêtes malicieuses et ainsi garantir une meilleure disponibilité.

1.3 Objectifs

Dans ce mémoire, nous proposons une solution qui permet de classifier des flux réseaux à l'aide de techniques d'apprentissage automatique. Cette solution se greffe à la capture des paquets, en extrayant les caractéristiques jusqu'à l'analyse des données. La base de données CSE CIC IDS 2018 [Sharafaldin et al., 2018] sera utilisée pour l'apprentissage et la mesure des performances du modèle. Initialement, nous allons comparer plusieurs modèles entrainés à classifier des observations ou à reconstruire des instances légitimes. Nous procéderons à une sélection ; le modèle le plus adéquat capable de détecter non seulement des attaques de déni de services, mais aussi d'autres anomalies dues aux attaques comme les injections SQL ou encore les attaques de brutes forces sur le système d'informations.

1.4 Organisation du mémoire

Dans un premier temps nous allons introduire l'aspect sécurité de ce mémoire, à savoir définir ce qu'un est un réseau, définir certaines vulnérabilités et attaques de déni de services. Puis, nous verrons plusieurs solutions existantes en rempart contre les attaques de déni de services permettant de limiter l'impact de celles-ci ou de détecter une potentielle offensive lancée dans un système informatique. Certaines de ses solutions étant basées sur l'apprentissage automatique, nous essayerons de couvrir un ensemble diversifié de méthodes et d'approches. Dans le chapitre subséquent à la revue de la littérature, l'apprentissage automatique sera introduit où nous décrivons le fonctionnement de plusieurs algorithmes que nous utiliserons dans nos expérimentations. Certains principes sont aussi décrits tel que l'encodage des données, et la normalisation ainsi que les fonctions d'activation. Dans le chapitre 4, nous révèlerons l'architecture de notre solution et nous expliquerons en détails les modules composants la solution. Puis dans le chapitre 5, intitulé Expérimentation nous expliquerons notre choix de dataset, et effectuerons maintes tests afin d'affiner les performances de notre modèle sélectionnés parmi d'autres. Nous considérons aussi, dans ce chapitre, le déploiement de notre solution dans un environnement exposé. Ci-après et finalement, le chapitre de conclusion où nous présenterons les résultats de l'expérimentation et une conclusion générale.

Chapitre 2

Revue de la littérature

2.1 Réseau Internet

Internet est un réseau mondial qui permet aux équipements réseaux (ordinateurs, serveurs, smartphones, routeurs, etc..) de communiquer grâce au protocole de communication TCP/IP. Un système informatique peut être sujet d'évenements ou d'attaques empêchant son fonctionnement normal. Dans le cas d'attaques de déni de service, il s'agira de nui à sa disponibilité et ce, s'il est exposé à Internet ; il est nécessaire de rappeler les notions globales du domaine de la réseautique. Cette section décrit donc sommairement le réseau, la notion des couches ainsi que certains protocoles.

2.1.1 Notions globales

Le modèle OSI est une référence publiée en 1984, le sigle d'Open System Interconnections, conçue pour standardiser les communications entre deux équipements réseaux. Composé de 7 couches empilées, chacune d'entre elles remplit une fonction bien précise. Le modèle TCP/IP, le sigle de Transmission Control Protocol / Internet Protocol est un protocole client-serveur utilisé pour établir une connexion. Il est composé de 4 couches : Réseau, Internet, Transport et Application. OSI est un modèle conceptuel, TCP/IP est un protocole utilisé par tous les réseaux.

La version 4 du protocole IP demeure la version la plus utilisée, publiée le premier janvier 1981.
L'une des principales spécifications du protocole TCP/IP est l'adressage. Une adresse IP est une chaîne de caractères qui sert à identifier une hôte dans son réseau. Elle est attribuée automatiquement grâce au protocole DHCP (Dynamic Host Configuration Protocol) , soit configurée statiquement ou plus encore avec le protocole APIPA (Automatic Private Internet Protocol Addressing) permettant d'attribuer une adresse IP lorsque le serveur DHCP n'est pas disponible.
Deux versions publiées de ce protocole existent actuellement, IPv4 et IPv6. Une adresse IPv4, figure

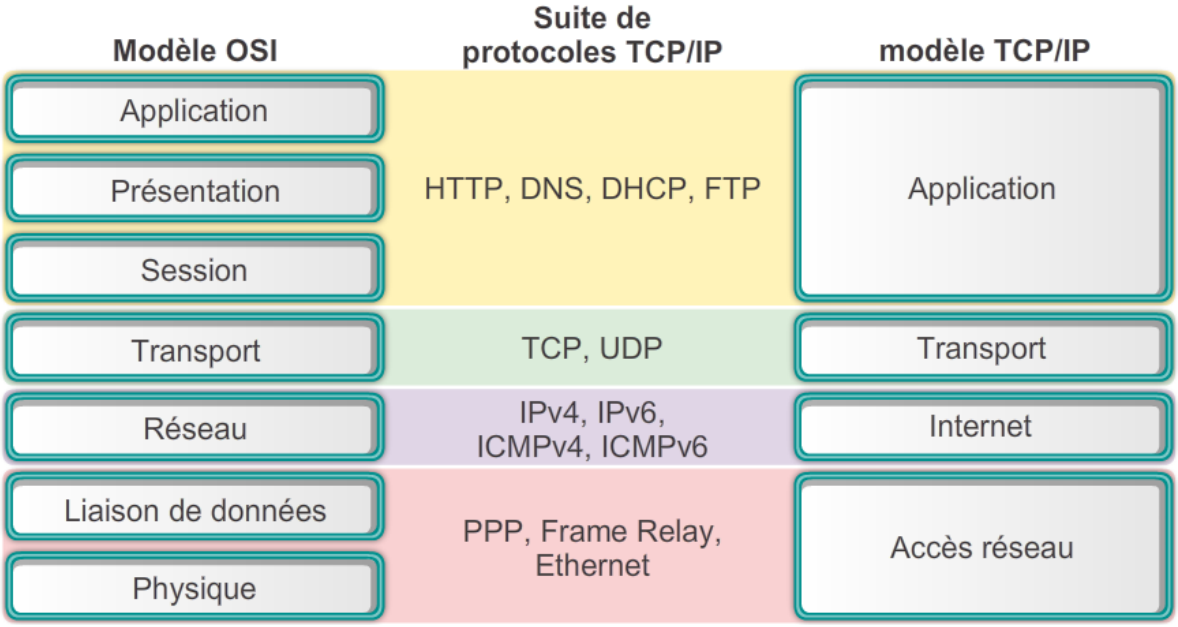


FIGURE 2.1 – Modèle OSI et protocole TCP/IP - Source : [Sery, 2014]

2.2, possède une longueur fixe de 4 octets séparés par un point (32 bits), écrits en nombre entiers de 0 à 255. Ce qui laisse à cette version du protocole l'attribution au maximum 2^{32} adresses. Depuis RFC 791 [rfc, 1986].

A l'instar, une adresse IPv6 est composée de 128 bits, écrite en hexadécimal. Cette version peut identifier au maximum 2128 adresses uniques. La version 4 est nettement plus utilisée de nos jours, nous allons donc nous concentrer sur l'IPv4, a noter qu'une adresse IPv4 peut être convertie en IPv6.

2.1.2 Paquets et flux réseaux

Paquets

Une communication entre deux hôtes ne peut être établie qu'après la couche Internet suite à l'attribution d'une adresse IP. Un datagramme ou paquet est une entité de cette couche, il transporte des données dans le réseau. Il comporte une partie en-tête grâce à laquelle les paquets peuvent être acheminés - l'en-tête contient l'adresse IP, le port - et une partie données. Chaque couche encapsule les paquets en y ajoutant des données propres à la couche.

L'illustration 2.3 ci-dessus représente l'en-tête d'un datagramme IPv4. Certaines approches se basent sur un test effectué sur l'en-tête. Dans les faits, les systèmes de filtrage du trafic réseau basés sur des règles ACL "liste de contrôle d'accès" prennent en considération uniquement l'en-tête du datagramme,

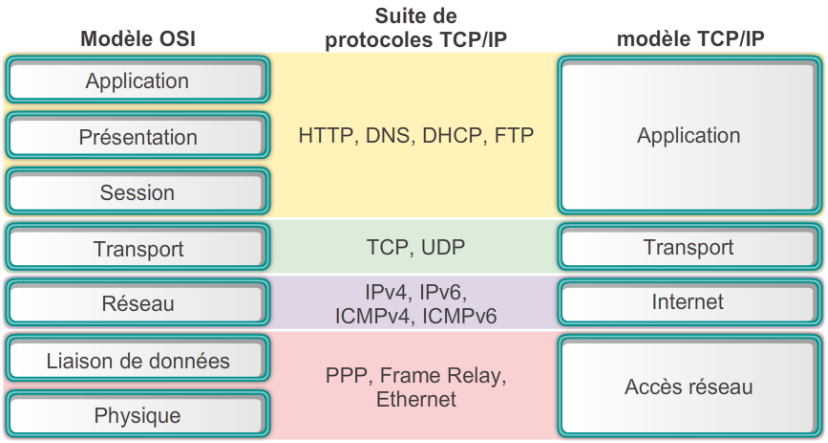


FIGURE 2.2 – Adresse IPv4 - Source : [Picaxe, 2014]

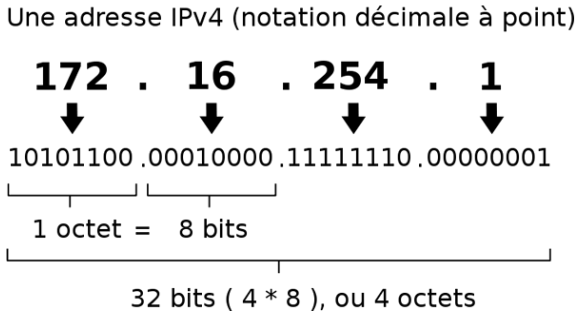


FIGURE 2.3 – Dissection du datagramme IPv4 [Anjum, 2011]

c'est le cas de Snort [Snort, 2021].
Nous ne pouvons nous limiter à l'étude des paquets uniquement. En effet, un datagramme ne peut définir en soi une menace à lui tout seul. Il faut analyser un ensemble des paquets transitant dans le réseau sujet à la surveillance pour émettre un verdict. Cette contrainte ou désavantage donne naissance à des techniques allant de l'analyse sur les flux réseau jusqu à la classification des données du paquets.

Flux réseau

Les flux réseaux permettent d'avoir une représentation intéressante et plus riche pour du network monitoring. Le but est de construire une conversation complète entre les deux hôtes et d'en déduire des propriétés qui permettront d'effectuer une analyse.

Un flux réseau est composé de plusieurs paquets. Il peut être vu comme une séquence, une série temporelle. On peut scinder un flux selon plusieurs critères : par la suite on peut choisir soit de les grouper selon l'adresse IP source et destination, ou par rapport à leur session. La représentation en flux de conversation réseau est plus intéressante pour effectuer une analyse, elle offre principalement un avantage quant à l'extraction de caractéristiques. En effet, des statistiques peuvent être déduites du flux.
« Ajouter le nombre de paquets transmis, le nombre de paquets reçu, le débit de paquets etc.. »
Nous découvrirons dans la section suivante les attaques de déni de service liées à l'exploitation des vulnérabilités du protocole IP.

2.2 Déni de service

2.3 Types d'attaques de déni de services

Toutes les attaques de déni de service se rejoignent dans leur objectif de perturber la disponibilité d'un système et dérouter son utilisation légitime, la figure 2.4 illustre un trafic réseau massif malicieux qui epuise les ressources du serveur victime. Il existe trois classes d'attaques de déni de services : les attaques basées sur le volume, les attaques protocolaires et sur la couche d'application.

2.3.1 Attaques basées sur le volume

L'objectif de l'attaque est de saturer la bande passante disponible entre la cible et Internet. Cette attaque se base sur la quantité de données envoyées à la cible, créant un trafic massif. Ce type inclut les inondations UDP, l'amplification Domain Name Server (DNS) ou l'amplification Network Time Protocol (NTP).

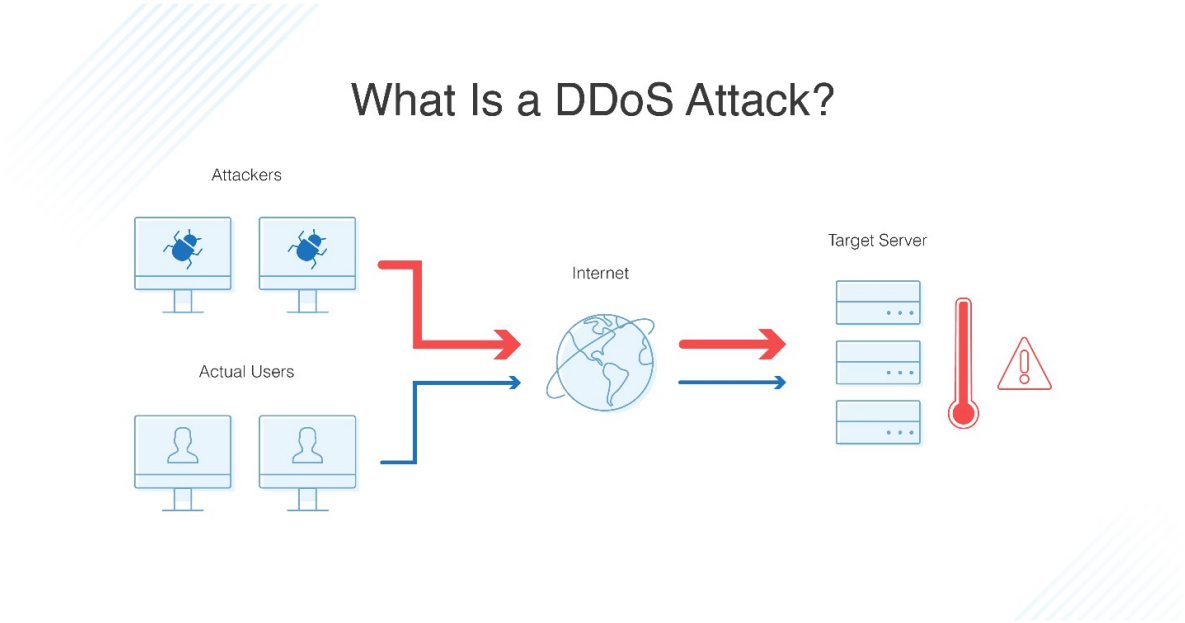


FIGURE 2.4 – Déni de service [DNSstuff, 2019]

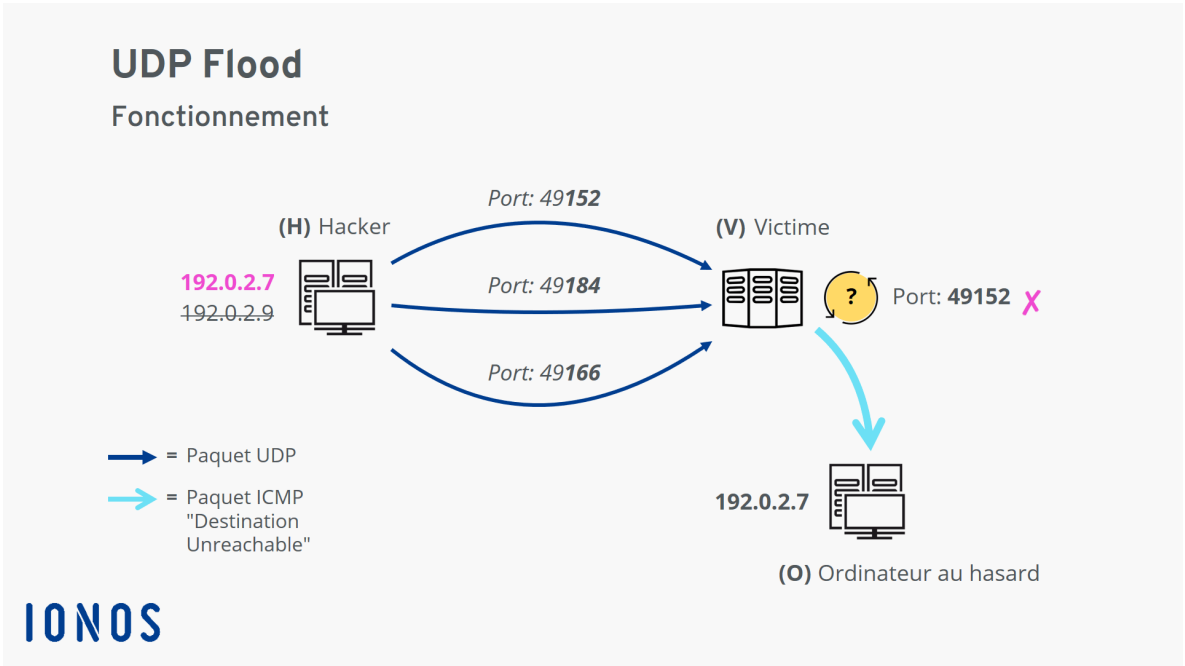


FIGURE 2.5 – Inondation UDP - [ion, 2020]

Cas de l'UDP Flood

Cette attaque se repose sur les spécificités du protocole User Datagram Protocol (UDP). Lorsqu'un paquet UDP atteint un serveur, le système d'exploitation vérifie si le port spécifié a des applications d'écoute. Si aucune application n'est détectée, le serveur doit en informer l'expéditeur. Comme UDP est un protocole sans connexion, le serveur utilise le protocole ICMP (Internet Control Message Protocol) pour informer l'expéditeur que le paquet n'a pas été remis. [ion, 2020] L'attaquant utilise une adresse IP usurpée, souvent réalisé à l'aide de botnet. La victime à la réception de ces paquets, envoie l'accusé de réception à l'adresse IP source, mais n'obtient aucune réponse à son tour et continue à l'attendre 2.5. Enfin, lorsque la victime abandonne la communication, toutes ses les ressources ont été consommées, ce qui a entraîné un crash du système.[S. and J., 2019]

Ces outils permettent de mener des attaques UDP Flood :

— Low Orbit Ion Cannon (LOIC)

— XOIC

.

2.3.2 Attaques protocolaire

Ce type d'attaque consomme les ressources réelles du serveur, ou celles des équipements de communication intermédiaires, tels que les pare-feu et les équilibreurs de charge. Elles exploitent les faiblesses des couches 3 et 4 du modèle OSI. L'inondation SYN ou Ping of Death font partie de ce type d'attaques. Ci-dessous, le fonctionnement de l'attaque par inondation de requête SYN, .

Cas de l'attaque SYN Flood

Afin d'initier une connexion Transmission Control Protocol (TCP), le client envoie une requête SYN au serveur, le serveur lui répondra par une requête SYN-Ack puis le client lui renvoie une requête Acknowledgment (ACK). Ce processus est similaire dont le protocole DHCP fonctionne. L'attaque d'inondation SYN utilise cette vulnérabilité du protocole ; l'attaquant envoie des requêtes SYN au serveur se qui aura pour effet de saturer le serveur avec des connexions actives. Cette attaque est contrable en limitant le nombre de connexion entrante au serveur, comme illustré dans la figure 2.6.

2.3.3 Attaques de la couche d'application

Ces attaques ciblent la couche d'application (7 du modèle Open Systems Interconnection (OSI)). Elles représentent le type d'attaque les plus complexes à détecter. Composées de requêtes de toute apparence légitimes et innocentes, l'objectif de ces attaques est de faire planter le serveur d'application en épuisant les ressources de calculs.

L'attaquant visera essentiellement un fragment d'une application, gourmand en ressources pour le

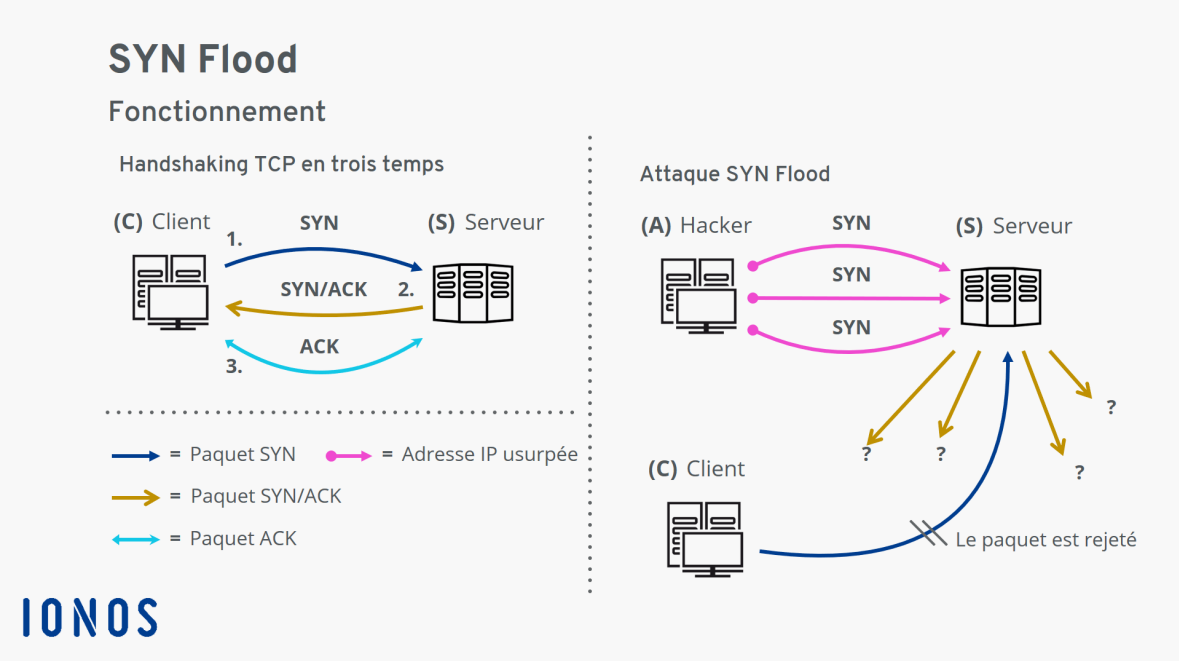


FIGURE 2.6 – Inondation requête SYN, extraite de [ion, 2020]

serveur. Cela peut être une page web menant à une requête complexe à la base de données (jointures dans une base de données Structured Query Language (SQL)) ou des requêtes utilisant des fonctions telles que la cryptographie, où des milliers de calculs seront exécutés inutilement jusqu'à ce que le service ne puisse plus traiter davantage de requêtes . Voici un exemple de page d'application web susceptible d'être visée par un attaquant :

Page de connexion- requête HTTP/POST :

Les requêtes pour se connecter devront être vérifiées (token csrf, échappements des entrées) puis le mot de passe entré par l'utilisateur devra être hashé ou crypté pour le comparer aux enregistrements de la base de données

Page de produits - requête HTTP/GET : Des requêtes complexes pour récupérer une ressource disponible comme un vol, une chambre d'hôtel, un produit (impliquant plusieurs jointures avec des bases de données distribuées par exemple).

Ce type d'attaque peut présenter différents niveaux de complexité. Les versions complexes peuvent utiliser un grand nombre d'adresses IP malveillantes, et cibler des URL aléatoires en utilisant des référents et des agents utilisateurs aléatoires. [clo, 2021a]

Exemple d'attaques : inondation HTTP, attaques sur les services DNS. Ces outils permettent de mener des attaques d'inondation HTTP :

— LOIC [clo, 2021b]

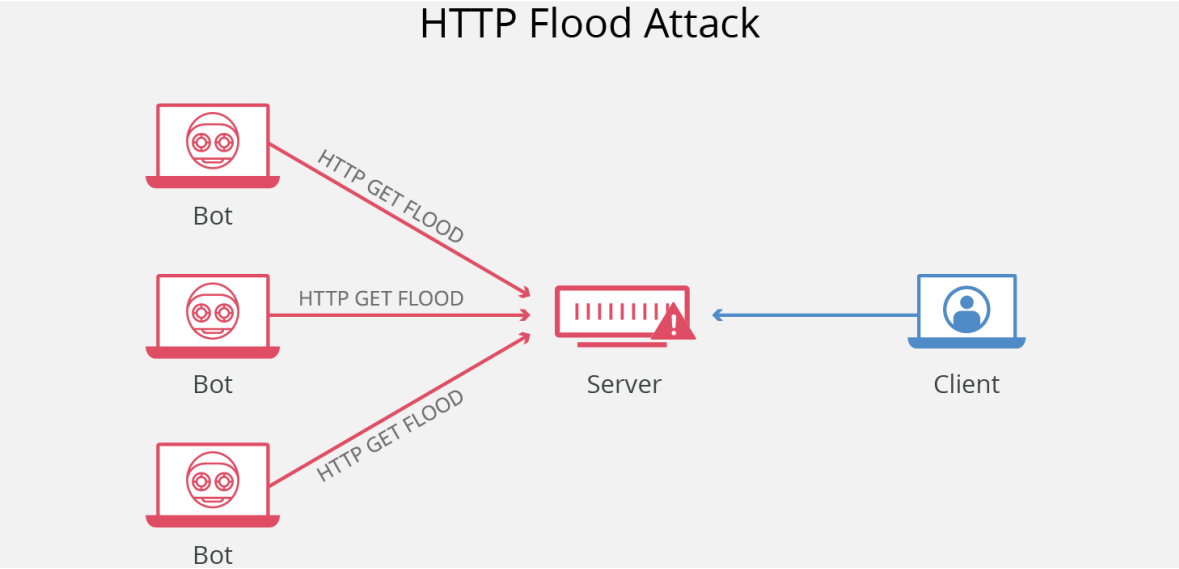


FIGURE 2.7 – Inondation HTTP, extraite de CloudFlare.com

— XOIC [XOIC, 2021]

— Slowloris [clo, 2021c]

.

2.4 Solutions et techniques de détections des attaques de déni de service

2.4.1 Données en entête

Les données en en-tête d'un paquet réseau peuvent être utilisées afin de limiter le trafic réseau entrant ou sortant, il s'agira entre autre de délimiter le périmetre d'attaque. Certains équipements réseaux de types pare-feu, permettent de filtrer le trafic circulant dans un système d'information en fonction des données en en-tête - adresse ip , port, protocole. C'est le cas des IDS tel que Snort [Snort, 2021] et de certains système de détection d'instrusion, par-feux voire application qui utilisent des règles pour interdire des protocoles, des ports ou encore des adresses IP voire des adresses mac (liste blanche, liste noire) ou à l'inverse d'autoriser uniquement certaines règles. Des données supplémentaires peuvent être déduites à partir des données en en-tête dont les règles pourront se baser : nombre de requêtes autorisées, temps de session et permettront de renforcer la sécurité et de limiter certaines attaques. Cette approche ne permet pas de se prémunir contre les attaques applicatives. La cible reste vulnérable aux attaques telles que le port-spoofing ou encore l'inondation HTTP. [ANSSI, 2015]

Tâche	Description
1	Identification d'application et caractérisation du trafic réseau
2	Détection d'anomalies du trafic réseau
3	Détection d'intrusion
4	Détection d'attaque DDoS

TABLE 2.1 – Désignation des tâches

2.4.2 Inpsection appronfondie des paquets

L'inspection approfondie des paquets offre un réel avantage par rapport aux autres approches. Au lieu de vérifier les données en en-tête, le contenu du paquet "payload" est aussi analysé. Le paquet peut être analysé en comparant sa signature avec une base de données de signature d'attaques. La base de données doit être suffisamment grande pour couvrir un ensemble considérable d'attaques. Toutefois calculer la signature d'un paquet peut s'avérer coûteux en termes de performances. La base de données des signatures d'attaques peut représenter un désavantage lorsque celle-ci n'est pas mise à jour régulièrement. [Kai-Cheng et al., 2020] Cette approche donne naissance à plusieurs tâches comme «l'identification d'application» *(Email, YouTube, FTPS, etc...)* et «la caractérisation du trafic réseau» *(VPN, Non-VPN, VoIP, etc...)*. [Lotfollahi et al., 2018].

L'inspection appronfondie des paquets est souvent réalisé à l'aide de méthode d'apprentissage automatique.

2.4.3 Cybersécurité et apprentissage automatique

L'apprentissage automatique a démontré son efficacité dans la cybersécurité et est utile pour détecter des anomalies du trafic réseau ou encore identifier une application d'un paquet. Ces tâches peuvent limiter les dégâts d'une attaque de déni de services, et même s'en protéger efficacement. Voici des travaux ayant utilisé l'apprentissage automatique pour résoudre des problèmes de sécurité *(intrusions - détection d'anomalies - identification du réseau)* ; tableau extrait de [Leevy and Khoshgoftaar, 2020]

Approche	**Tâche2.1**	**Données**	**Source**
SAE+CNN	1	ISCX VPN-nonVPN	[Lotfollahi et al., 2018]
Voir 2.5	3	CSE CIC IDS2018	[Ferrag et al., 2019]
DataNet	1	ISCXVPN-nonVPN	[Pan et al., 2018]
CAPC : AE + Classifier	1	ISCXVPN-nonVPN	[Kai-Cheng et al., 2020]
Autoencoder	4	CIC DDOS 2019	[Li, 2020]
DeepMal	3	[Stratosphere, 2015]	[Marìn et al., 2020]
LLC -Varitionnal Autoencoder	3	CSE CIC IDS 2018	[Bårli, 2019]
ML	3	CSE CIC IDS 2018	[de Lima Filho et al., 2019]
D2PI	3	ISCX IDS 2012	[Cheng and Watson, 2017]
Conv-AE IDS	3	CSE CIC IDS 2018	[Khan and Kim, 2020]

TABLE 2.2: Travaux similaires, reproduit de [Leevy and Khoshgoftaar, 2020]

Deep Packet [Lotfollahi et al., 2018] Deep packet est un framework basé sur un auto-encodeur -SAE- et un réseau de neuronne à convulsion -CNN-, permettant d'identifier la classe d'application d'un paquet. Deep packet est une forme d'inspection appronfondie, il analyse le payload ainsi que quelques informations en en-tête du paquet. Les paquets du dataset ISCX-VPN-nonVPN ont été utilisé afin de détecter correctement l'application et les caractéristiques du réseau. Un score F1 de 0.98 pour le CNN et 0.95 pour l'autoencodeur. [Lotfollahi et al., 2018].
La table 2.3 représente les résultats de la classification des paquets, la table 2.4 réprésente la caractérisation du réseau.

Application	**CNN F1**	**SAE F1**
AIM chat	**0.81**	0.70
Email	0.89	**0.97**
Facebook	**0.96**	0.95
FTPS	**1.00**	0.86
Gmail	**0.96**	0.94
Hangouts	0.97	0.97
ICQ	0.76	0.69
Netflix	**1.00**	0.99
SCP	0.98	**1.00**
SFTP	**1.00**	0.81
Skype	**0.97**	0.94
Spotify	0.98	0.98
Torrent	**1.00**	0.99
Tor	1.00	1.00
VoipBuster	0.99	0.99
Vimeo	**0.99**	0.98
YouTube	0.99	0.99

TABLE 2.3 – Résultats identification d'application de Deep Packet, extrait de [Lotfollahi et al., 2018]

Classe	CNN F1	SAE F1
Chat	0.77	**0.74**
Email	0.91	**0.95**
FileTransfer	0.99	0.99
Streaming	**0.90**	0.83
Torrent	**1.00**	0.98
VoIP	0.74	**0.75**
VPN :Chat	**0.98**	0.94
VPN :FileTransfer	**0.99**	0.97
VPN :Email	**0.99**	0.95
VPN :Streaming	**1.00**	0.99
VPN :Torrent	**1.00**	0.98
VPN :VoIP	**1.00**	0.99

TABLE 2.4 – Résultats de la caractérisation du réseau de Deep Packet, extrait de [Lotfollahi et al., 2018]

L'étude de [Ferrag et al., 2019] est une étude comparative des techniques de deep learning pour la detection d'intrusion. Plusieurs modèles ont été proposé en les comparant selon leur classe ; modèles génératifs et modèles discriminatifs. Les modèles discriminatifs sont : Recurrent Neural Network (RNN), DNN, Convulsional Neural Network (CNN) Les modèles génératifs sont : Restricted Boltzmann Machine (RBM), Deep Belief Networks (DBN), Deep Boltzmann Machine (DBM) Deep Auto encoder. (DA) Les modèles opèrent sur le dataset CIC IDS 2018.

Classe	**DNN**	**RNN**	**CNN**	**RBM**	**DBN**	**DBM**	**DA**
Benign	96.915	98.112	98.914	97.316	98.212	96.215	98.101
SSH-Bruteforce	100	100	100	100	100	100	100
FTP-BruteForce	100	100	100	100	100	100	100
BruteForce-XSS	83.265	92.182	92.101	83.164	92.281	92.103	95.223
BruteForce-Web	82.223	91.322	91.002	83.164	92.281	92.103	95.223
SQL Injection	100	100	100	100	100	100	100
DoS Attacks-Hulk	93.333	94.912	94.012	91.323	91.712	93.072	92.112
DoS Attacks-SlowHTTPTest	94.513	96.123	96.023	93.313	95.273	95.993	94.191
DoS Attacks-Slowloris	98.140	98.220	98.120	97.040	97.010	97.112	97.120
DoS Attacks-GoldenEye	92.110	98.330	98.221	92.010	97.130	97.421	96.222
DDOS Attack-HOIC	98.640	98.711	98.923	97.541	97.211	97.121	96.551
DDOS Attack-LOIC-UDP	97.348	97.118	97.888	96.148	96.122	96.654	96.445
DDOS Attack-LOIC-HTTP	97.222	98.122	98.991	96.178	97.612	97.121	97.102
Botnet	96.420	98.101	98.982	96.188	97.221	97.812	97.717
Infilteration	97.518	97.874	97.762	96.411	96.712	96.168	97.818

TABLE 2.5 – Comparaison des modèles discriminatifs, extrait de [Ferrag et al., 2019]

Datanet de [Pan et al., 2018] Le projet datanet est de classifier un trafic réseau. Trois modèles ont été proposés pour comparer leur performances : MLP, SAE, CNN. Les paquets de la base ISCX VPN-nonVPN ont été utilisées en format PCAP.

Le prétraitement consiste à supprimer l'entête du paquet, puis le tronquer pour que tout les paquets aient la même taille pour finalement les normaliser. La figure 2.8 représente le tableau des performances des trois modèles extrait de [Pan et al., 2018].

	MLP		**SAE**		**CNN**	
	Full	Balanced	Full	Balanced	Full	Balanced
Precision						
Maximum	0.9714	0.9420	0.9914	0.9741	0.9930	0.9754
Average	0.9657	0.9342	0.9883	0.9692	0.9847	0.9696
Minimum	0.9595	0.9254	0.9851	0.9642	0.9628	0.9633
Recall						
Maximum	0.9717	0.9403	0.9915	0.9737	0.9920	0.9746
Average	0.9653	0.9309	0.9881	0.9678	0.9842	0.9685
Minimum	0.9582	0.9226	0.9847	0.9619	0.9613	0.9623
F1-Score						
Maximum	0.9694	0.9375	0.9905	0.9723	0.9891	0.9732
Average	0.9653	0.9308	0.9882	0.9678	0.9843	0.9685
Minimum	0.9603	0.9235	0.9855	0.9641	0.9656	0.9634

FIGURE 2.8 – Performances Datanet, extraite de [Pan et al., 2018]

[Kai-Cheng et al., 2020] CAPC est un modèle composé d'un auto encoder concaténé a un classificateur. Les paquets sont utilisés en format PCAP. Les adresses sont traitées comme des valeurs fixes. Testé sur deux base différentes l'une étant notée publique et l'autre privée ; il s'agit de ISCX VPN-nonVPN et d'un trafic capturé par les auteurs pour l'expérimentation respectivement. La figure 2.9 montrent les résultats de CAPC face à d'autres modèles.

	DNN		**DAE**		**1D CNN**		**CAPC**	
	private	public	private	public	private	public	private	public
Accuracy	0.9471	0.9436	0.9975	0.9679	0.9995	0.9603	0.9998	0.9742
Precision	0.9329	0.9509	0.9975	0.9661	0.9995	0.9609	0.9998	0.9736
Recall	0.9145	0.9418	0.9975	0.9664	0.9995	0.9588	0.9998	0.9733

FIGURE 2.9 – Performances CAPC, extraite de [Kai-Cheng et al., 2020]

Type d'attaque	DNN - F1	AE - F1	PCC AE - F1
SNMP	0.750	**0.882**	0.876
NetBIOS	**0.996**	0.994	0.993
MSSQL	0.940	**0.941**	0.939
UDP	**0.993**	0.986	0.981
SSDP	**0.991**	0.987	0.979
LDAP	0.558	**0.567**	0.558
SYN	**0.999**	**0.999**	0.998
NTP	**0.995**	0.992	0.982
DNS	**0.717**	0.711	0.665
WebDDoS	0.330	0.330	**0.932**
UDP-Lag	0.937	0.865	**0.939**
TFTP	**0.999**	0.992	0.997

TABLE 2.6 – Résultats détéction d'attaque DDoS, extrait de [Ferrag et al., 2019]

[Li, 2020] Ce projet a pour but de détecter les attaques DDoS. Les données proviennent de la base de données CIC DDOS 2019. L'auteur a d'abord utilisé la correlation de pearson sur le trafic pour sélectionner 37 features des 86 features disponibles du trafic générer par CICFlowMeter depuis la base de données puis normaliser ces mêmes features. Trois modèles ont été proposé : AE, DNN, un AE avec le correlation de pearson. Le tableau 2.6 représente les scores F1 de la détection d'attaque DDoS des modèles.

Classificateur	Score
KNN	0.987
LR	0.987
RF	0.986

TABLE 2.7 – ML Performances, extrait de [Dharshini, 2021]

Botnet	Protocole	Activité
Neris	IRC	spam, click fraud
Rbot	IRC	DDoS
Virut	HTTP	spam, port scan

TABLE 2.8 – Classe

Le projet mis en oeuvre par [Dharshini, 2021] compare 3 algorithmes de machine learning : KNN, RF, LR sur le dataset CSE CIC IDS 2018. Le tableau 2.7 réprésente les scores moyens de l'expérimentation.

DeepMal de [Marìn et al., 2020] DeepMal est un modèle proposé pour analyser le contenu des paquets et peut fonctionné sur deux répresentations : paquet et trafic ; ce qui compose l'avantage de DeepMal. Pour son expérimentation, le modèle est comparé avec un classificateur RF.
Le tableau 2.8 représente les attaques exécutées par les botnets. Les figures 2.11 et 2.10 répresentent les matrices de confusions de DeepMAL et du RF.

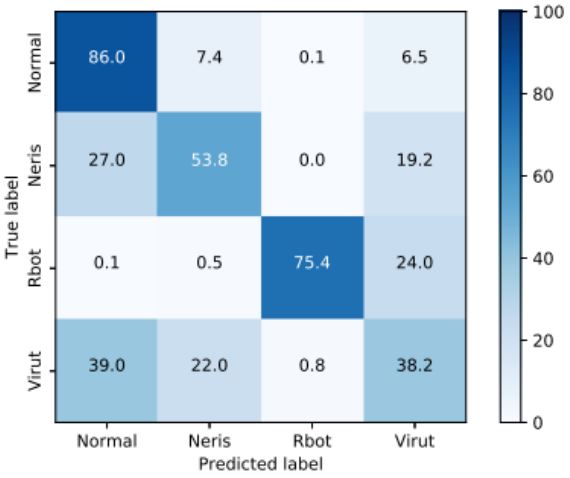


FIGURE 2.10 – Matrice de confusion de RF

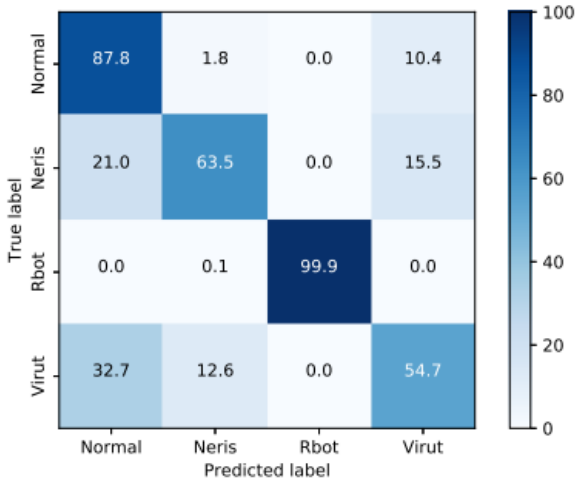


FIGURE 2.11 – Matrice de confusion de DeepMAL, extraite de [Marìn et al., 2020]

Base	Métrique	k-NN	SVM	Naïve Bayes	Desicion Tree	RF	LR
A	F1	0.99	0.99	0.99	1	0.99	0.99
A	µ(s)	148.2	16.8	2.7	5.3	120.8	10.8
B	F1	0.99	0.99	0.99	1	0.99	0.99
B	µ(s)	148.2	16.8	2.7	5.3	120.8	10.8

TABLE 2.9 – Performances mesurées par [Kiourkoulis, 2020]

[Khan and Kim, 2020] Le système de détection d'intrusion proposé par [Khan and Kim, 2020] est composé de 4 parties. La première étape consiste à préparer les données, secondement, une détection des anomalies est faite à l'aide des classificateurs traditionnels en utilisant Spark MLlib. L'avant dernière étape consiste à déceler les abus grâce a un Conv-AE pour finalement à la dernière étape déclencher l'alarme. Atteignant **0.9827 en score F1**, cette approche se distincte par l'extraction des caractéristiques grâce à l'approche CNN et AE ; 14 attributs ont été sélectionnés des 86 de la base CSE CIC IDS 2018.

Tandis que le système proposé par **[de Lima Filho et al., 2019]** est constitué d'une base de données de signatures et d'un algorithme d'apprentissage automatique. Il s'agit d'un classificateur RF sélectionné en comparaison avec Adaboost, SGC, LR ... sur 20 attributs sélectionnés de la base. La moyenne du score F1 est supérieure à 0.99 sur les bases de données de référence à savoir CSE CIC IDS 2018, CIC DDoS 2019 et CIC IDS 2017.

Par ailleurs l'étude de **[Kiourkoulis, 2020]** compare différents algorithmes sur plusieurs bases. Le temps d'exécution a été mesuré sur une machine Linux dotée de 8GB de ram et d'un processeur i5.

A	CSE CIC DDOS 2018
B	CIC IDS 2017
µ(s)	Temps d'execution en secondes

TABLE 2.10 – Définition des symboles de table 2.9

Deux modèles proposés par **[Bårli, 2019]** basé sur un AE variationnel. Le premier modèle LLC-VAE [1] et le second LBD-VAE [2] Sur les bases de références (CSE CIC IDS 2018 et CIC IDS 2017) sur les 86 attributs, 40 ont été sélectionnés après l'utilisation de LIME. Les meilleures performances ont été obtenues par modèle LLC-VAE, pouvant classer les trafics légitimes avec une précision de 0.97 et les trafics malicieux avec une précision de 0.93.
Les travaux énoncés démontrent que l'utilisation de l'apprentissage automatique pour résoudre les problèmes de sécurité est très prometteuse. Quelque soit l'algorithme utilisé, et quelle que soit la base de données sur laquelle les modèles ont été entraînés, les performances sont convaincantes.

La capacité des algorithmes d'apprentissage automatique à prendre des décisions, à retenir des informations et à acquérir de nouvelles compétences conduira très certainement à l'émergence de solutions de sécurité basées sur l'apprentissage automatique.

1. LLC-VAE : Latent Layer Classification on a Variational Autoencoder
2. LBB-VAE : Loss Based Detection on a Variational Autoencoder

Chapitre 3

Apprentissage automatique

3.1 Apprentissage automatique

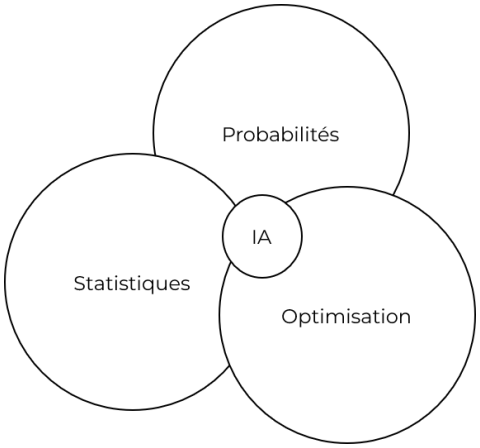


FIGURE 3.1 – Intelligence artificielle

3.1.1 Présentation et définition

L'apprentissage automatique est une discipline de l'intelligence artificielle qui rassemble les statistiques, les probabilités, et l'optimisation, figure 3.1. En apprentissage automatique, on tente de résoudre un problème permettant à un programme d'apprendre à partir des données. Essentiellement, ce domaine peut être vu comme une discipline cherchant à trouver une fonction optimale à partir des observations (X) et dans certains cas en ayant les valeurs de ces observations ou étiquettes (Y) tout en minimisant la valeur d'erreur notée e [Ah-Pine, 2019].

$$F : X = Y + e$$

Un modèle d'apprentissage automatique possède deux primitifs élémentaires :

— ***model.fit(x, y)*** ou ***model.fit(x)***
pour entraîner le modèle avec l'essemble d'observations d'apprentissage x, et les valeurs ou étiquettes optionnelles y associées.
— ***model.predict(x)***
pour prédire les valeurs de x

Autrement dit, il s'agit de découvrir une relation entre les observations, en commettant le moins d'erreurs possibles, tout en faisant de bonnes prédictions pour des valeurs de X non encore observées.

Dans la prochaine section, nous verrons deux types d'apprentissage et le processus nécessaire pour créer un modèle qui sera capable de traiter des flux réseaux et de détecter les anomalies. On représentera X par notre base d'instances de taille n et Y le vecteur des valeurs des instances associées. Ainsi que x, y une instance quelconque suivi de sa valeur tel que

$$x \subset X, y \subset Y$$

3.1.2 Types d'apprentissage automatique

Apprentissage supervisé

Dans ce type d'apprentissage, nous disposons d'un ensemble d'instances où à chacune d'elles est associé sa valeur cible ou étiquette. Entre autres, notre base de données est composée de X, Y . Le modèle devra être capable de prédire la bonne valeur cible d'une nouvelle instance.
Lorsque la valeur cible à prédire est continue on parlera de problème de régression, un exemple très courant est la prédiction du prix d'un bien immobilier.
Tandis que lorsqu'il s'agit de prédire une valeur discrète, c'est un problème de classification, par exemple : prédiction de cellule maligne ou bénigne ou classification de e-mail comme spam .

Apprentissage non supervisé

Contrairement à l'apprentissage supervisé, nous disposons uniquement d'observations sans les étiquettes, on parle d'apprentissage non supervisé ou clustering.
Le modèle devra identifier des groupes, tels que les observations doivent être similaires au sein de leur groupes tout en ayant une hétérogénéité intergroupes. [Bacha, 2019]

Autres type d'apprentissage

Il est à noté que d'autres types d'apprentissage existent, comme :

— Apprentissage semi-supervisé

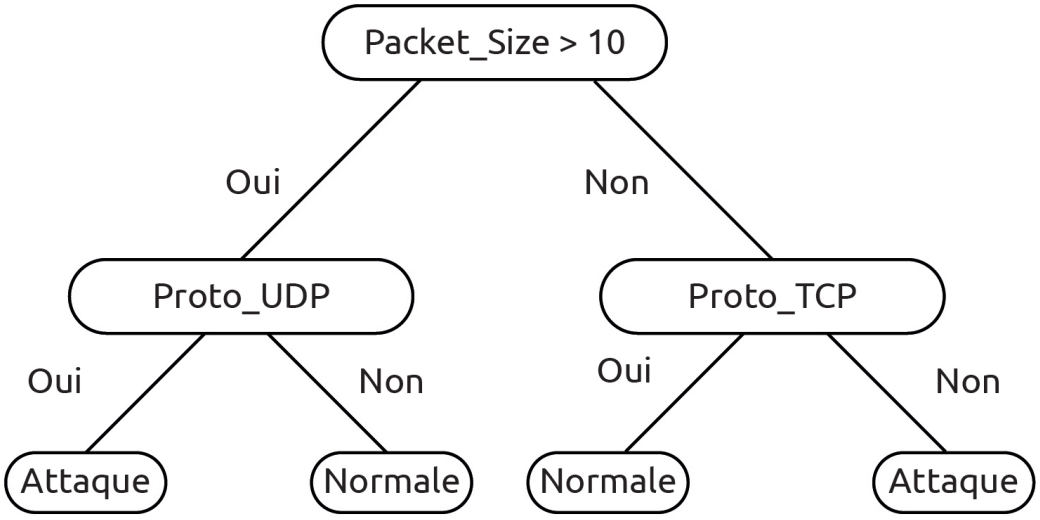


FIGURE 3.2 – Exemple d'arbre de décision

— Apprentissage par renforcement
— et bien d'autres

Le choix de la méthode dépend fortement des tâches que l'algorithme devra effectuer par la suite. Pour les problèmes de régression et de classification, l'apprentissage supervisé est à privilégier par rapport aux autres méthodes.

3.1.3 Classfication

Pour notre étude, nous avons des données étiquetées ; des données statistiques extraites depuis des datagrammes ainsi que leur valeur - trafic normal ou d'attaque - voici deux algorithmes de classification utilisés pour l'évaluation.

Arbre de décision

Un arbre de décision, figure 3.2, est un avant tout une structure hiérachique, un graphe connexe, sous forme des séquences de décisions en vue de la prédiction d'un résultat ou d'une classe. Chaque observation, qui doit être attribuée à une classe, est décrite par un ensemble de variables qui sont testées dans les nœuds de l'arbre. [Cnam., a]

FIGURE 3.3 – Encoding des caractéristiques

Forêts aléatoires

Un classificateur forêts aléatoires représente un ensemble d'arbre de décision dont la classification se fait par agrégation par vote majoritaire. Ce classificateur est facile à mettre en place, donne de bons résultats en grande dimensions [Cnam., b].

3.2 Codage des données

Un modèle d'apprentissage automatique, ne peut apprendre des données telles qu'elles sont représentées réellement. En d'autres termes, appliquer une fonction mathématique sur des lettres, des images, ou des sons serait sans doute une erreur.
Admettons par exemple, que la tâche demandée est de classifier un trafic réseau. Nous disposons d'un fichier PCAP : une base de données de paquets réseaux IPv4 capturés avec un Wireshark. Notre modèle n'est bien évidemment pas conçu pour comprendre ce que le protocole ou le port signifie. Il faut encoder les caractéristiques en valeurs numériques. Manifestement, le port est une valeur numérique, nous verrons par la suite pourquoi certaines valeurs numériques auront aussi besoin d'être encodées.
Rappelons que nos ordinateurs, smartphones, objets connectés etc.. convertissent en permanence les données pour que nous puissions interagir. Il est donc tout à fait possible de tout encoder.
Plusieurs techniques existent pour encoder des données selon leurs types (catégories, séries, textuelles, etc ..) Notre intérêt se porte plus exactement sur l'encodage de caractéristiques "catégorielles" . Selon [Enchun, 2019], 2 méthodes sont plus avantageuses que d'autres pour encoder des adresses IP.

3.2.1 One Hot Encoding

Le One Hot Encoding est une technique très populaire. Elle consiste à représenter les valeurs de l'attribut en tant que vecteur de dimension d correspondante à la cardinalité de l'attribut. Le vecteur est parsemé de 0 excepté l'indice de la valeur référencée par un 1.

Exemple : La figure 3.4 est une représentation d'une base de données :

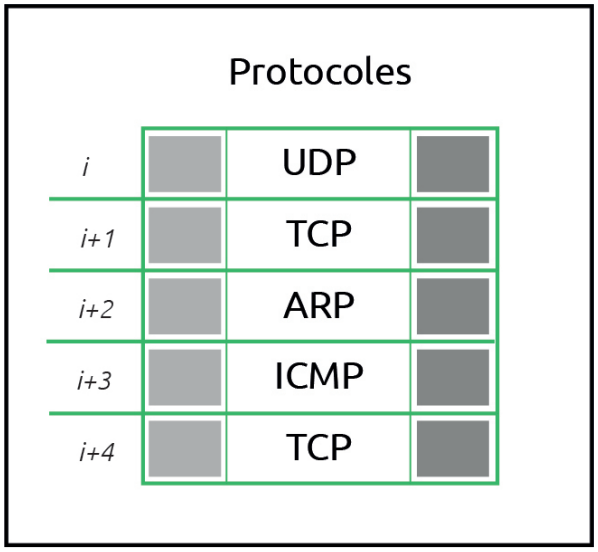


FIGURE 3.4 – Exemple de base de données

Nous voulons encoder l'attribut protocole avec la technique du One Hot Encoding. Nous récupérons d'abord les valeurs unique du protocole (figure 3.5) :

FIGURE 3.5 – Exemple de valeur unique du protocole

Puis nous remplaçons le protocole par un vecteur rempli de 0 avec uniquement la valeur référencée en 1, cela augmentera la dimension de l'instance selon la taille du vecteur résultant (figure 3.6).

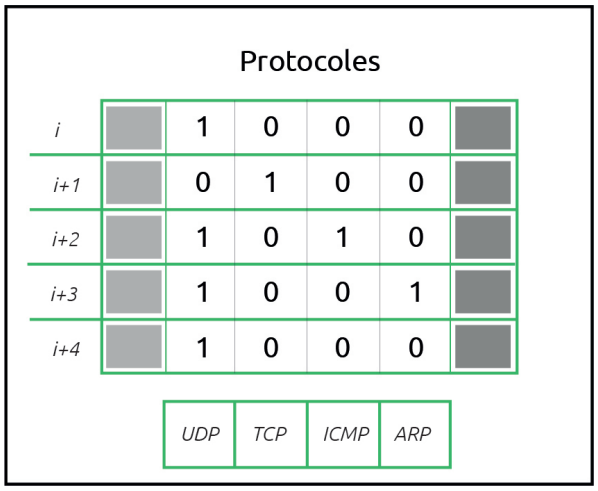


FIGURE 3.6 – Exemple d'encodage de l'attribut protocole

3.2.2 Binary Encoding

L'encodage en binaire consiste à encoder l'information en binaire. Très utilisée pour encoder du texte, des images ou d'autres types de données complexes.

3.3 Normalisation

La normalisation dans le domaine de l'apprentissage automatique est technique qui transforme les caractéristiques adaptant chaque attribut à une plage donnée. Cela permet à notre modèle d'être plus précis.

[Jaitley, 2018] La figure 3.7 illustre les résultats d'une étude comparative menée par [Urvashi Jaitley]. Une mise en épreuve de deux modèles identifques d'apprentissage profond est éffectuée. Le premier modèle reçoit des données non normalisées, le second est entraîné avec des données normalisées à l'aide de l'algorithme StandardScaler.

Depuis la figure 3.7, nous constatons que sur le diagramme de gauche, la précision du modèle ne dépasse pas 0.489 et reste stable sur 25 epochs, tandis que la courbe de droite montre que les performances augmentent au fil des itérations, tout en avoisinant 0.89 en accuracy.

Dans la partie implémentation, nous allons tester les classificateurs avec des données normalisées et non normalisées, en guise de recherche de meilleures performances.

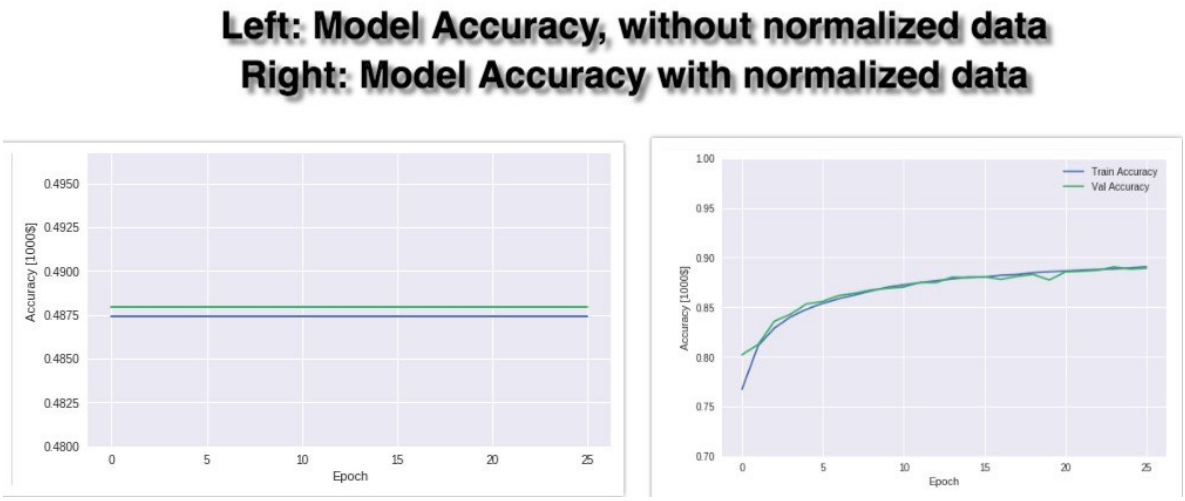


FIGURE 3.7 – Comparaison de la normalisation des données

3.3.1 MinMaxScaler

Cet estimateur met à l'échelle et transforme chaque caractéristique individuellement de telle sorte qu'elle se situe dans la plage donnée entre 0 et 1 sur l'ensemble de données.

La transformation est donnée par :

$$x' = \frac{x - min(x)}{max(x) - min(x)}$$

3.3.2 StandardScaler

Le centrage et la mise à l'échelle se produisent indépendamment sur chaque caractéristique. L'estimateur transforme la distribution des données de chaque caractéristique de tel que sorte à sa moyenne soit 0 et son écart-type soit 1.

La transformation est donnée par :

$$z = \frac{x - \mu(x)}{\sigma(x)}$$

Où $\mu(x)$ représente la moyenne des X_i et $\sigma(x)$ représente l'écart-type des X_i

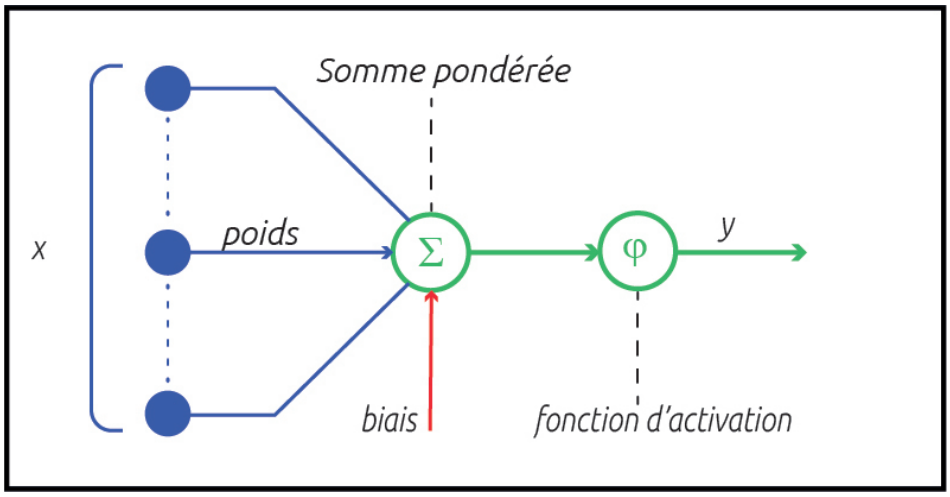


FIGURE 3.9 – Illustration représentante d'un neurone artificiel

3.4 Deep learning

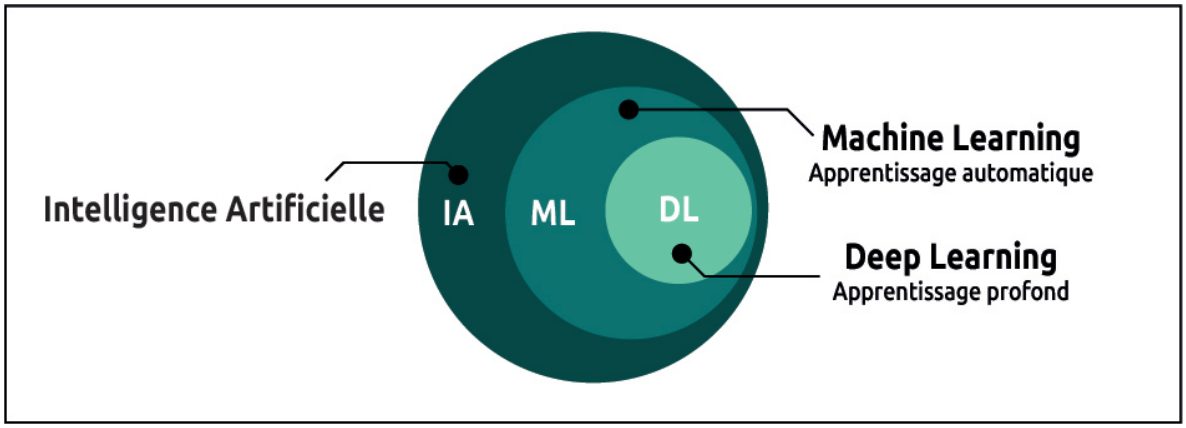


FIGURE 3.8 – Apprentissage automatique et apprentissage profond

L'apprentissage profond, en anglais Deep Learning une catégorie d'apprentissage automatique. L'apprentissage se fait via des réseaux de neurones artificiels dont le fonctionnement est inspiré des neurones biologiques. Il existe une panoplie de réseaux de neurones (récurrents, à convulsion etc) pour résoudre des problèmes spécifiques.

Chaque neurone (figure 3.9) transforme les entrées, en apprenant le poid et le biais notés w et b respectivement.

$$y : \phi(w * x + b)$$

ϕ, appelée fonction d'activation permet de changer la représentation des données, elle est spécifique

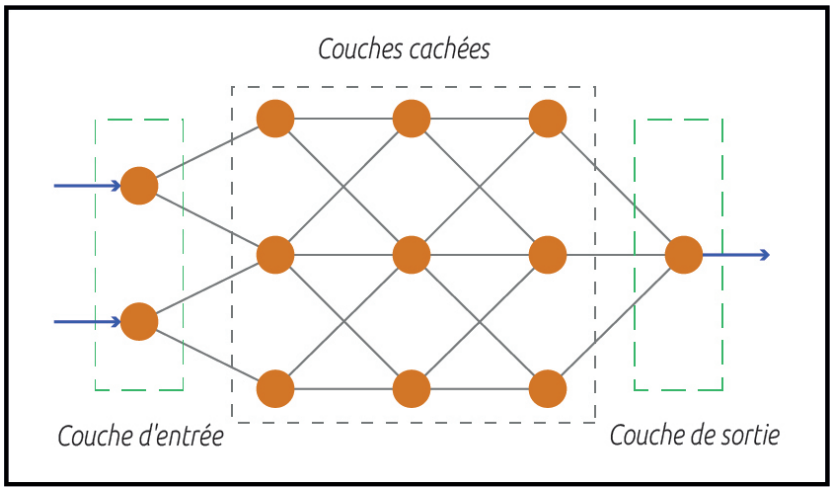
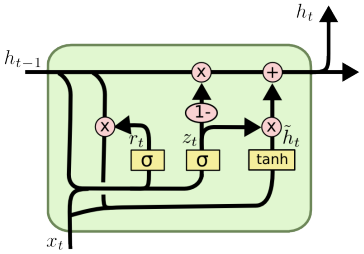


FIGURE 3.10 – Réseau de neurones denses



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

FIGURE 3.11 – Cellule LSTM

à chaque couche. Chaque neurone d'une couche va appliquer la fonction d'activation de la couche sur les données. Cette transformation sera différente selon chaque neurone car chacun possède un poids différent. [Keldenich, 2021]

3.4.1 Couches

Dans cette partie, nous verrons sommairement quelques couches : Dense, LSTM qui reviennent souvent dans la littérature pour la détection d'anomalies du trafic réseau.

Dense

Cette couche non linéaire est connectée entièrement à la couche précédente ou chaque neurone est connecté avec ceux de la couche précédente. Le réseau de neurones illustré ci-dessus est inspiré du site Tibco.com, il est constitué de couches Dense.

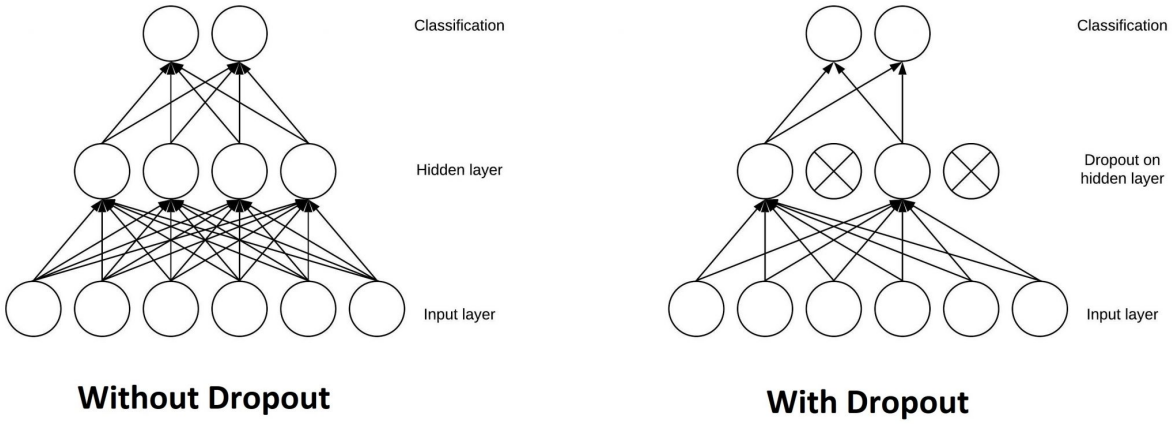


FIGURE 3.12 – Effet de la couche Dropout, image extraite de [Baeldung, 2019]

LSTM

Utilisé dans les réseaux de neurones récurrents, LSTM, sont surtout utilisées pour faire des prédictions sur des séries temporelles. Dans un contexte de détection d'anomalies, il s'agira de prédire si la prochaine instance est une anomalie en se basant sur un intervalle défini.

Batch Normalisation

La normalisation par lots est une méthode utilisée pour accélérer le processus d'apprentissage des réseaux de neurones artificiels : en appliquant une transformation qui maintient la sortie moyenne proche de 0 et l'écart type de sortie proche de 1. [Tensorflow, 2021]

Dropout

La couche Dropout (figure 3.12), définit de manière aléatoire les unités d'entrée sur 0 avec une fréquence de taux à chaque étape pendant le temps d'entraînement, ce qui permet d'éviter le surapprentissage [Keras, 2021]

3.4.2 Fonctions d'activations

Les fonctions d'activation servent à introduire une non-linéarité au données. Selon le problème à résoudre, on utilise des fonctions d'activations différentes, nous verrons trois fonctions d'activations les plus utilisées.

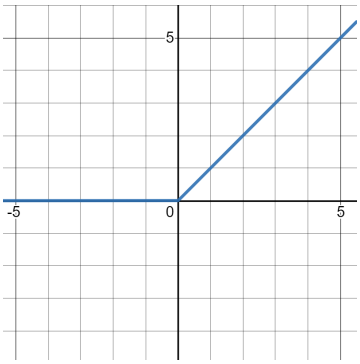


FIGURE 3.13 – Graphe fonction ReLU

Unité Linéaire Rectifiée - ReLU

Rectified Linear Unit (ReLU), figure 3.13 est une fonction d'activations qui retourne le max en l'entrée et 0

$$relu(x) = max(x, 0)$$

Sigmoïde

Cette fonction d'activation, graphe dans la figure 3.14, est utilisée dans de la classification binaire, elle retourne une valeur entre 0 et 1. Nous utiliserons cette fonction d'activation dans les couches de sorties.

$$sigmoid(x) = \frac{1}{(1 + e^{-x})} :\subset [0, 1]$$

Ainsi, si la fonction retourne un résultat > 0.5, nous associerons l'instance à la classe 1, ou à la classe 0 inversement.

Tangente Hyperbolique - tanh

La fonction de la tangente hyperbolique 3.15 retourne un résultat dans l'invervale $[-1, 1]$. Son utilisation et son fonctionnement sont très similaires à Simoid. Elle est également utilisée dans de la classification binaire.

$$tanh(x) = \frac{sinh(x)}{cosh(x)} :\subset [-1, 1]$$

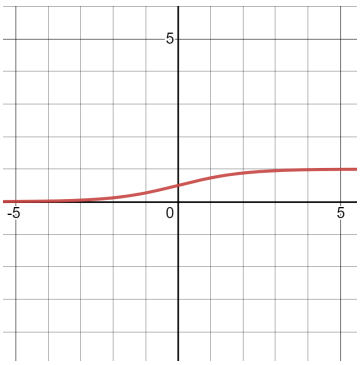


FIGURE 3.14 – Graphe fonction Sigmoïde

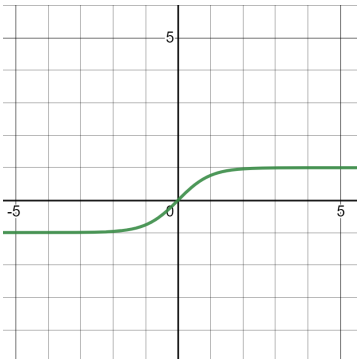


FIGURE 3.15 – Graphe fonction Tangente Hyperbolique

3.4.3 Fonctions d'erreurs

Les fonctions d'erreurs ou loss functions représentent la différence entre les données et les prédictions. Dans le cas d'un auto encodeur, ça sera l'erreur de reconstruction. Cette fonction que l'on cherche a minimiser devra être choisie selon le problème. Ci dessous deux fonctions d'erreurs que nous utiliserons.

— **Mean Absolute Error**

$$mae = \frac{1}{n} \sum |\epsilon_i|$$

— **Binary Cross Entropy** Cette fonction est très utilisée pour de la classification binaire.

$$bce = -(p(x) * log(q(x)) + (1 - p(x)) * log(1 - q(x))$$

3.4.4 Auto encodeur

Un auto encodeur est un type réseau de neurones artificiel, conçu à l'origine pour réduire la dimension, il est également utilisé pour de la détection d'anomalies. La forme d'un auto encodeur la plus simple est composée de couches entièrement connectées ; (voir figure 3.16).
Composé de deux parties : une encodeur et un décodeur concaténées. Les couches visibles à savoir la couche d'entrée et celle de la sortie partagent la même dimension.

L'encodeur est composé d'au moins 3 couches : une couche d'entrée, une autre couche qui compresse les caractéristiques et donc la dimension des données entrées et enfin une couche de sortie qui sera connectée au décodeur. Le décodeur décompresse les caractéristiques, il fait une expansion de la dimension.
On peut avoir différente architecture d'auto encodeur :

— des couches LSTM pour de la détection d'anomalies sur des séries temporelles,
— des couches à convulsion, Convulsional Auto Encoder (CAE)
— un réseau d'auto encodeurs empilés, Stacked Auto Encoder (SAE)

$$model.fit(\mathbf{x}, \mathbf{x})$$

Le but est alors de prédire une nouvelle valeur x' se rapprochant au maximum de x, en d'autres termes reconstruire x .

3.4.5 Détection d'anomalies

Précédemment, nous avons vu qu'un auto encodeur reconstruit les entrées tout en minimisant l'erreur de reconstruction. Le principe de la détection d'anomalies est dans les faits, d'entrainer un

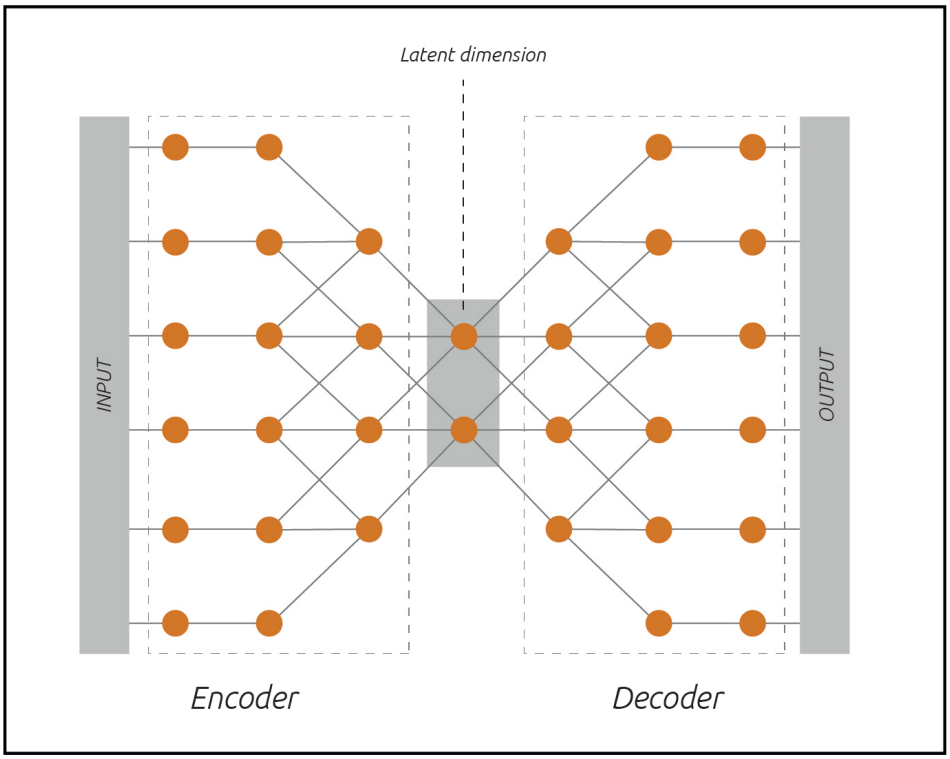


FIGURE 3.16 – Schéma globle d'un auto-encodeur

auto encodeur avec suffisamment d'instances légitime uniquement puis l'utiliser sur l'ensemble des données . En supposant que les instances malignes auront une erreur de reconstruction plus élevée. L'instance est classifiée selon l'erreur de reconstruction, si elle dépasse un seuil fixé noté t threshold, elle sera classifié comme anomalie.

Voici un exemple Le seuil étant défini, les instances ayant une erreur de reconstruction supérieur à 0.049 seront classées comme attaques.

3.5 Métriques de performances

Les performances des modèles sont calculées selon ces trois métriques :

La précision, Le rappel et l'exactitude, voici les formules associées dans le cadre d'une classification mono-classe. Soit (TP, TN) des instances correctement classifiées et (FN, FP) représentent des observations mal-classées [K, 2020].

— True Positive (TP) : vrai positif

— True Negative (TN) : vrai négatif

— False Positive (FP) : faux positif

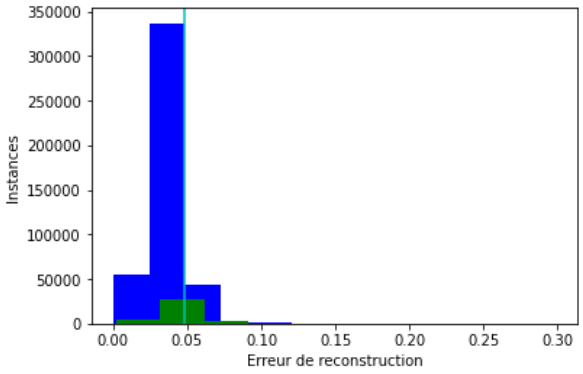


FIGURE 3.17 – Détection d'anomalies basé sur l'erreur de reconstruction

— False Negative (FN) : faux négatif

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

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

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

Nous ajouterons à cela le temps de prédiction noté μ_p, le temps d'apprentissage noté μ_t

Chapitre 4

Conception

4.1 Architecture

L'architecture de la solution proposée est illustrée dans la figure 4.1. Le premier module est l'acquisition des données, qui permet de charger les données en un format exploitable, puis le second module nommé Traitement des données effectue une transformation et une normalisation des données chargées. Le troisième module se repose sur les deux précedents et il se traduit par la classification des données ; qu'elles soient bénines ou malignes. Enfin, le module de déploiement qui permet s'étendre avec des fonctionnalités tel qu'enregistrer les adresses IP qui peut servir comme liste noire ou liste de blocage pour les pare-feux notamment.

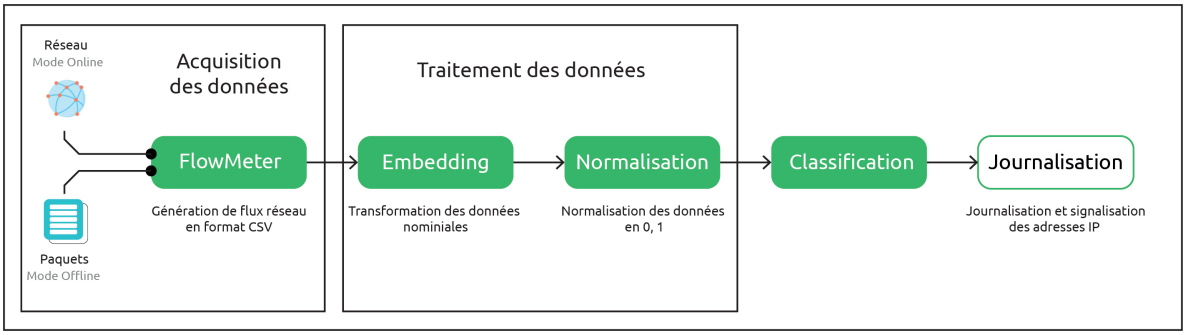


FIGURE 4.1 – Architecture de la solution proposée

4.2 Module 1 : Acquisition des données

Pour récolter les données, nous avons opté pour une approche exacte à celle qui est réalisée lors de la création des bases de données [Sharafaldin et al., 2018] et [Sharafaldin et al., 2019b] par l'institut Canadienne de Cybersécurité, ainsi il s'agit d'utiliser le programme FlowMeter pour recueillir les données et en extraire les caractéristiques des paquets réseaux.
A noter que le programme FlowMeter génére des données statistiques sur les paquets, de par sa nature même il ne permet donc pas de faire de l'inspection approfondie des paquets - Deep Packet Inspection (DPI).
Ceci permet de nous aligner avec les études similaires effectuées sur ces bases de données.

D'autres techniques peuvent s'avérer pertinentes et dépendront des données dont nous voulons en sortir et traiter, par exemple nous pouvons capturer et utiliser les datagrammes en tant que tels en format héxadécimal, puis normaliser ces derniers afin d'en tirer des matrices en deux dimensions (2D). La classification est alors faite similairement à la classification d'images, c'est à dire à l'aide de réseau neuronal convolutif CNN. Cette technique permet l'inspection approfondie des paquets ce qui peut résulter à la

- détection d'application
- détection du trafic VPN
- détection du protocole
- etc..

4.2.1 Type de données

Les données récoltées pour l'entraînement et l'évaluation du modèle sont en format CSV (voir Chapitre Implémentation et évaluation section 5.2). Ce sont des flux de trafic réseaux capturés et générés par CIC FlowMeter [Sharafaldin et al., 2019a].
Les données sont chargées dans l'environnement de développement en segment de 100 000 instances. Nous faisons alors une analyse exploratoire pour découvrir la composition des bases de données. Notre but étant de détecter les attaques de déni de service, nous nous débarrassons donc des noms de classes d'attaques ; nous remplaçons les classes d'attaques par 0 et la classe normale par 1 pour toutes les bases. Nous sauvegardons les données pour les utiliser ultériement.

Dans la phase de déploiement, nous utiliserons essentiellement, le mode en ligne, c'est-à-dire que le programme capture les datagrammes du réseau, extrait les caractéristiques en temps réel, traite les données pour enfin les classifier.
Nous avons fait le choix de ne pas traiter les données (sélection, encodage, normalisation) en même

temps que le programme effectue l'extraction des caractéristiques, le traitement est séquentiel, mais modulaire. En effet, cette dissociation entraînera une consommation des ressources. Cependant cette modularité apporterera une facilité débogage et de personnalisation des paramètres.

4.2.2 Prétaitements des données

Les données subissent un nettoyage en plusieurs niveau. Le traitement (1) consiste d'abord à encoder les valeurs des étiquettes ; la valeur 1 est associée à la classe normale, le reste des valeurs (SYN, DoS, etc..) auront tous la même valeur qui est 0 représentant la classe anormale. Ensuite (2) on supprime les lignes ayant les noms de colonnes à la place de valeurs numériques - c'est une erreur de fusion que nous avons rencontré dans les bases de données. L'instruction (3) aura pour effet de caster les données en float64. Les instructions (4, 5) traitent les instances ayant des valeurs [inf, nan] qui seront aussi supprimées. Et enfin, les attributs ayant des valeurs qui ne changent jamais seront écartés. Pour ces derniers, par la sélection des caractéristiques.

Algorithm 1 Prétraitement des données

Require: x

Require: $target$

1: $x[target] = replace(x[target], with : 1, else : 0)$

2: $x = select(x, where : x['DstPort']! =' DstPort')$

3: $x.astype('float64')$

4: $indices = \ x.isin([np.nan, np.inf, -np.inf]).any(1)$

5: $x = x[indices]$

4.2.3 Sélection des caractéristiques

Sur les +80 attributs des bases générés par le programme CIC FlowMeter, certains sont inutiles et d'autres sont redondants, la liste des attributs se trouve dans section 5.2 du chapitre Implémentation et évaluation. Nous utilisons la corrélation de Pearson [Benesty et al., 2009] pour sélectionner les attributs les plus important en écartant les attributs ayant une forte corrélation entre eux. En déterminant un coefficient égale à (0.95), on s'assure que seul les attributs ayant corrélation au-delas de 0.95 sont retirer de l'ensemble des attributs. Cette méthode est utilisé pour réduire la dimension de la donnée en supprimant les redondances. [Nasir et al., 2020] Dans la thèse de Juhong [Li, 2020], l'auteur utilise la corrélation de Pearson afin d'établir une liste d'attributs fortement corrélés pour les exclure de la base de données.

Dst Port	Protocol	Flow Duration
Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts
TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min
Fwd Pkt Len Mean	Fwd Pkt Len Std	Bwd Pkt Len Max
Bwd Pkt Len Min	Bwd Pkt Len Mean	Bwd Pkt Len Std
Flow Byts/s	Flow Pkts/s	Flow IAT Mean
Flow IAT Std	Flow IAT Max	Fwd IAT Std
Bwd IAT Tot	Bwd IAT Mean	Bwd IAT Std
Bwd IAT Max	Bwd IAT Min	Fwd PSH Flags
Bwd PSH Flags	Fwd URG Flags	Bwd URG Flags
Bwd Pkts/s	Pkt Len Max	Pkt Len Mean
Pkt Len Std	Pkt Len Var	FIN Flag Cnt
RST Flag Cnt	PSH Flag Cnt	ACK Flag Cnt
URG Flag Cnt	Down/Up Ratio	Fwd Byts/b Avg
Fwd Pkts/b Avg	Fwd Blk Rate Avg	Bwd Byts/b Avg
Bwd Pkts/b Avg	Bwd Blk Rate Avg	Init Fwd Win Byts
Init Bwd Win Byts	Fwd Seg Size Min	Active Mean
Active Std	Active Min	Idle Std

TABLE 4.1 – Attributs seléctionnés

— **Application du coefficient de corrélation de Pearson** [Biesiada and Duch, 2007] Nous utilisons la corrélation de Pearson pour sélectionner les attributs sur un échantillon de 1.7 millions d'instances dont 50% normales et 50% attaques. Nous avons écarté les attributs ci-dessous dont le but est de faire apparaître la corrélation entre les attributs numériques uniquement.

— Identifiant du flux

— Horodatage

Le tableau 4.1 représente les attributs séléctionnés. Dans la partie tests, nous comparons les performances des modèles face à différentes variantes :

1. Sans sélection en ignorant le port source et destination
2. Avec séléction en prennant compte du port de destination uniquement

Ci dessous la matrice de la corrélation de Pearson résultantes : Si la corrélation entre 2 caractéristiques distinctes est supérieur ou égale à 0.95 alors nous la retirons de la liste des caractéristiques. Ainsi nous nous baserons sur cela pour effectuer la sélection des caractéristiques. Ci-dessous le résultat

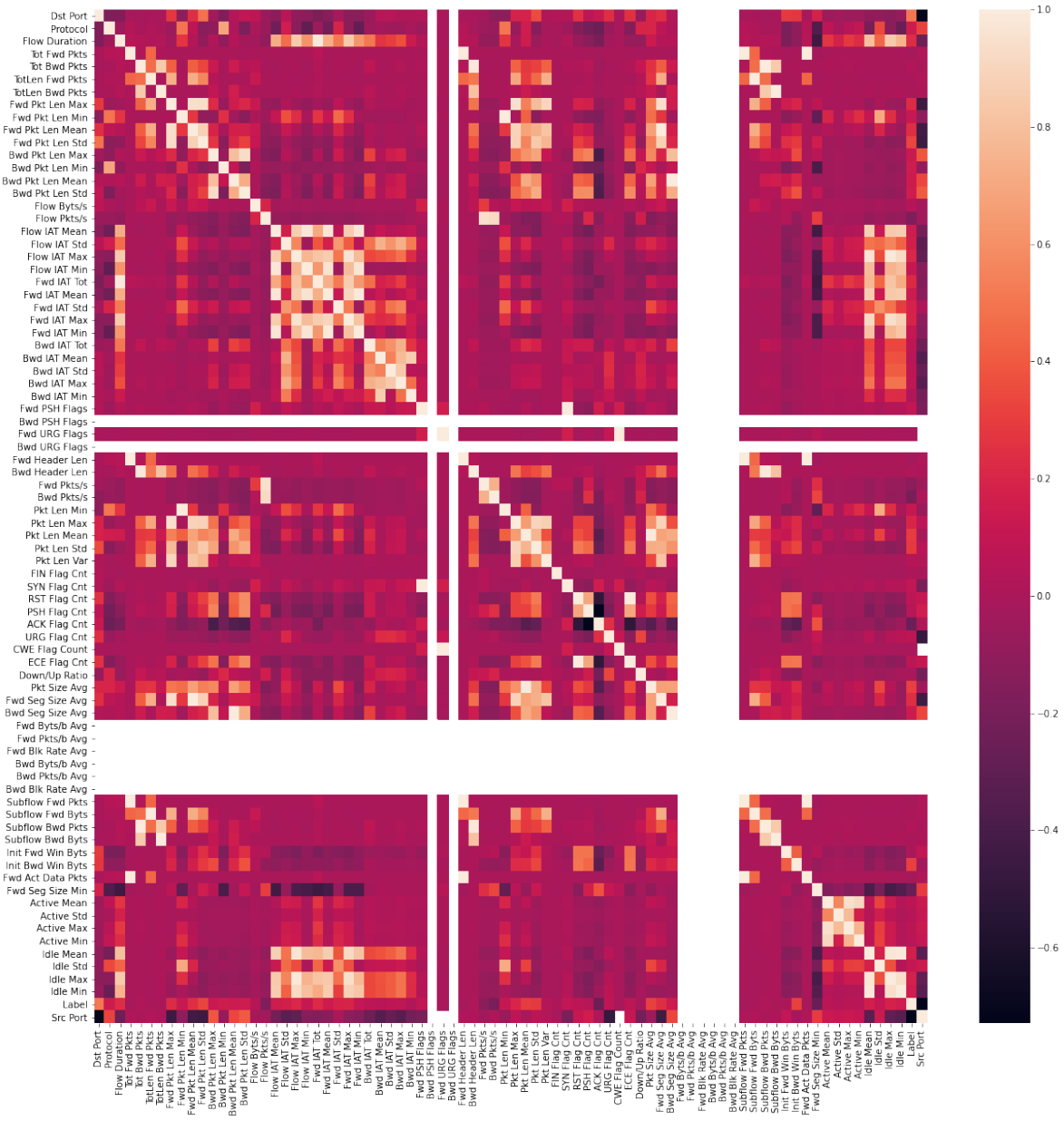


FIGURE 4.2 – Corrélation de Pearson - voir annexe

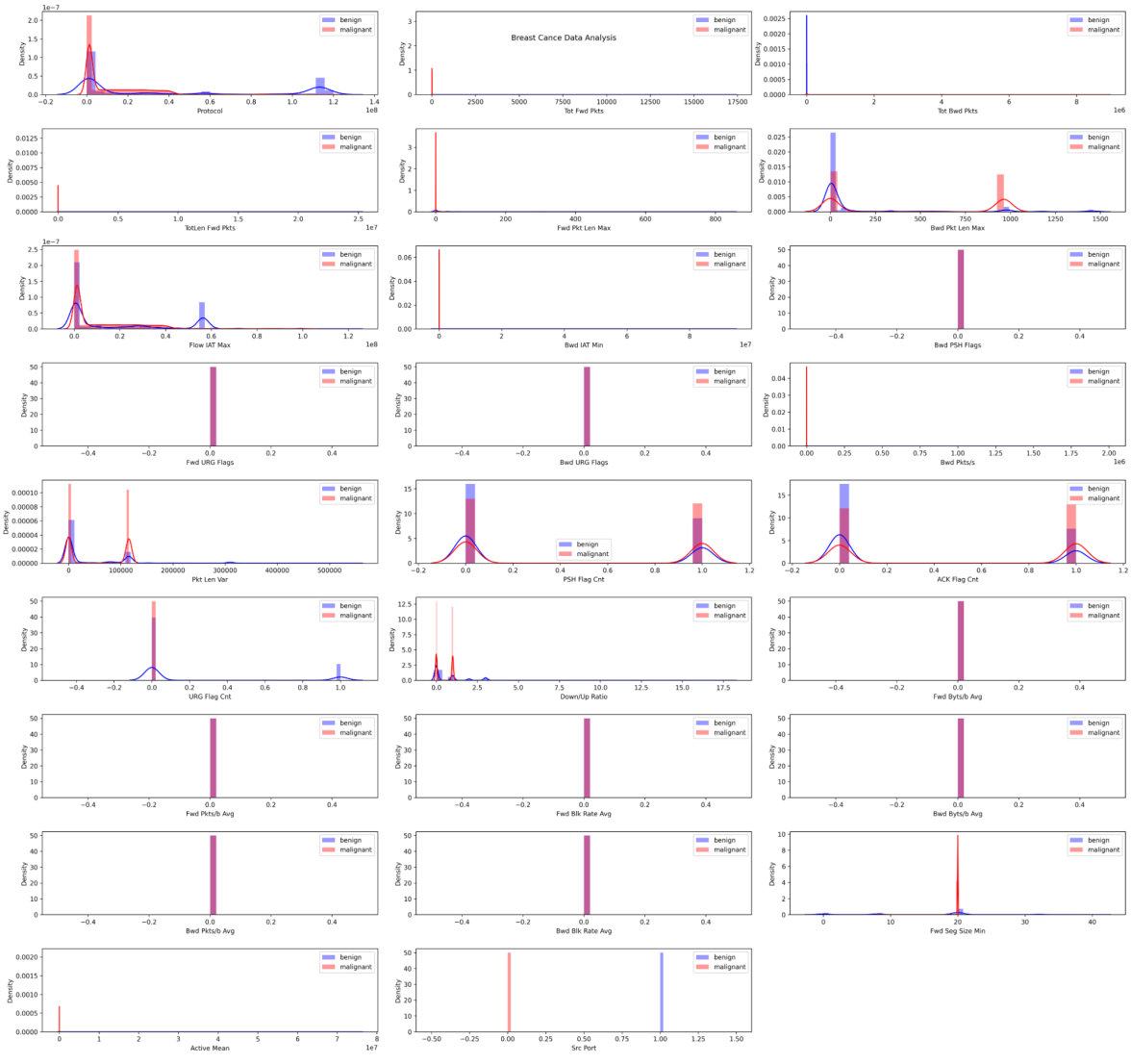


FIGURE 4.3 – Valeur P - voir annexe

du calcul de la valeur P des attributs sélectionnés.
A noter que nous avons écarté l'adresse IP pour ce traitement.
Dans 1 Le port source et destination ont été écarté, car d'une manière générale, le port est translaté grâce au protocole Network address translation (NAT), il est aléatoire et ne servira donc pas dans l'analyse.

4.3 Module 2 : Transformation des données

4.3.1 Encodage

De nombreux algorithmes d'apprentissage automatique sont incapables de traiter les variables catégorielles. Il est donc important de coder les données sous une forme appropriée afin de pouvoir prétraiter ces variables. Ainsi, grâce a cette tâche le modèle sera en mesure de comprendre et d'extraire les informations générant le résultat souhaité.
Les variables catégorielles contiennent différentes valeurs, et chaque valeur représente une catégorie distincte. Par exemple, le port est une variable, et il comprend différentes valeurs telles que HTTP, SSH. De même, un protocol est une variable, mais les protocoles TCP et UDP sont des valeurs distinctes représentant des catégories différentes.
Nous allons nous intéresser aux attributs catégoriels : adresses ip, ports, protocole - voir 5.2 - Chapitre Implémetation et évaluation partie : Base de données.

Encodage du Port et du Protocole

Nous savons que le port et le protocole sont des nombres entiers. Les utiliser sans encoding serait naïf. Prenons par exemple le port, qui est compris entre 0 à 65535. Si nous laissons le port comme tel (c'est à dire en tant que valeur entière), notre modèle fera des calculs qui sont incohérents ;
De toute évidence, 80 est supérieur 21, seulement il n'y a pas de relation d'ordre, il s'agit d'applications différentes ; il est nécessaire de traiter les ports et le protocole de communication comme des valeurs catégorielles nominales.

Approche 1 : One Hot Encoding

La méthode la plus facile de "one hot encoder" les caractéristiques catégorielles est d'utiliser la bibliothèque Pandas. Voici une fonction qui permet d'effectuer ce traitement :

Hélas, cette approche consomme beaucoup de ram ; en effet le nombre de port maximal est 2^{16}. Même si nous avons au maximum 30 ports dans notre base, c'est-à-dire que nous obtenons un vecteur

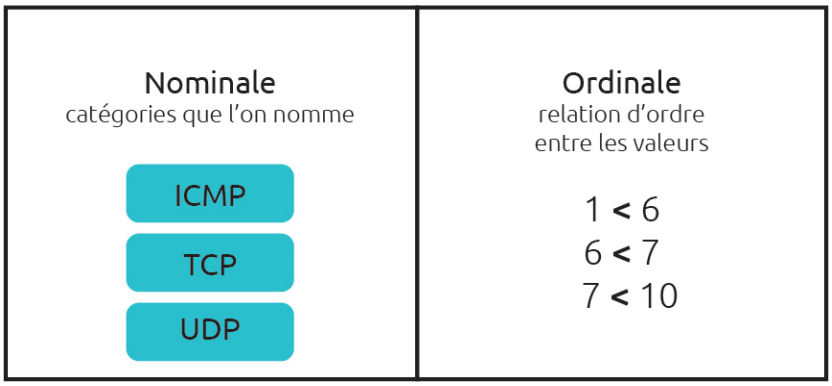


FIGURE 4.4 – Type de valeurs catégorielles

Algorithm 2 One Hot Encoding

Require: x

Require: $column$

1: $dummies = pd.get_dummies(x[column])$

2: **for** col in dummies.columns **do**

3: $\quad nn = f"name - col"$

4: $\quad x[nn] = dummies[col]$

5: **end for**

6: $x.drop(column, axis = 1, inplace = True)$

de 30 entiers à replacer à chaque instance. De plus, les instances non testées auront surement des ports ou protocoles jusqu'à la non-découvertes. Il est clair que les ports inconnus doivent aussi pouvoir être traités par notre modèle. En tant que solution et par conséquent nous pouvons enrichir le vecteur afin de placer les ports inconnus dans des classes. (port privé, etc..). Le deuxième inconvénient que représente cette approche, est le temps d'exécution ; si nous partons dans la méthode exhaustive, c'est-à-dire ajouter un vecteur de 2^{16} (65536) pour chacun des ports et du protocole, la dimension devient alors plus grande, et conséquemment le temps de classification.

Approche 2 : One hot Encoding 3b

Afin de pouvoir profiter des avantages du One Hot Encoding, nous avons proposé un schèma qui consiste à encoder le protocole de communication en vecteur de taille 3. Il s'agira d'encoder les deux protocoles les plus fréquents dans les bases à savoir UDP et TCP, avec la troisième valeur sera les autres protocoles. Voir l'algorithme 3

Algorithm 3 One Hot Encoding - 3b

Require: x ▷ Dataframe à encoder

1: $x['proto\_udp'] = np.where(x['Protocol'] == 17, 1, 0)$

2: $x['proto\_tcp'] = np.where(x['Protocol'] == 6, 1, 0)$

3: $x['proto\_other'] = np.where((x['Protocol']! = 6)\ \&\ (x['Protocol']! = 17), 1, 0))$

4: $x.drop('Protocol', axis = 1, inplace = True)$

Approche 3 : Transformation en binaire

Nous voulons gagner en performances, encoder une grande base de données nécessite beaucoup de ressources. Et parallèlement, le traitement devra être en temps réel lorsque le modèle sera déployé. Nous avons donc pensé à un moyen d'encoder les attributs catégoriels comme le port, et le protocole en vecteur de taille fixe : 16 bits et 8 bits respectivement [rfc, 1986]. La figure 4.5 illustre la transformation du protocole en vecteur de 16 bits. Nous utiliserons cette méthode dans l'expérimentation.

Algorithm 4 Transformation en binaire

Require: x ▷ L'attribut à encoder

Require: $size$ ▷ Taille du vecteur (16 par défaut)

1: $encode = lambdax, f = list, size = ": 016b" : list(size.format(x))$

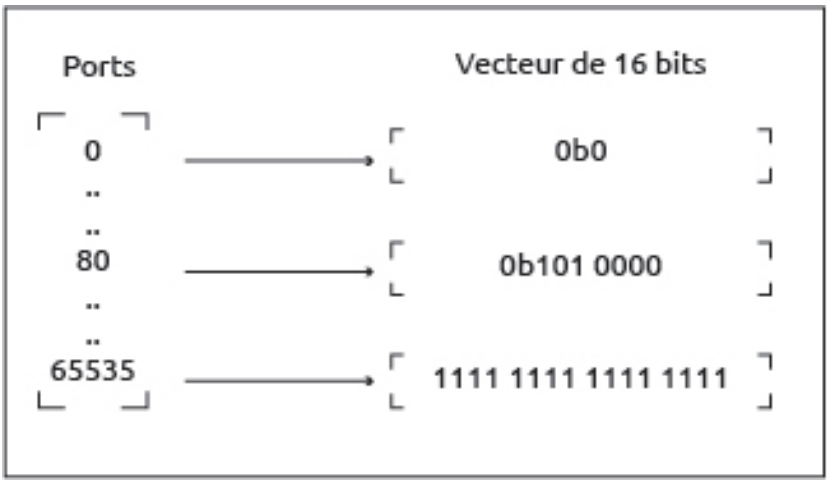


FIGURE 4.5 – Transformation du protocole en vecteur de 16 bits

Encoding de l'adresse IP

De la même manière, les adresses IP peuvent être traitées en tant que catégories. Encore une fois, l'encodage en one hot encoding est à proscrire, exhaustivement la taille du vecteur sera de $2 * 2^{32}$. Avec l'approche du One hot encoding, nous allons allouer de l'espace qui sera très probablement utilisé qu'une seule fois, voire jamais. Les travaux de [Enchun, 2019] expliquent plusieurs méthodes pour utiliser une adresse IP comme feature pour du machine learning. Nous proposons une approche inspirée de l'étude [Enchun, 2019], et des précédentes méthodes encoding (port, protocole). Notre approche est décrite comme telle :

— (1) Conversion de l'adresse IP en entier
— (2) Nous utilisons la même fonction pour encoder le port et le protocole pour encoder l'adresse IP en vecteur de 32 bits

Algorithm 5 Encodage de l'adresse IP en vecteur de 32 bits

Require: ip

1: $transform = lambda(ip) : reduce(lambda(a, b) : int(a) * 256 + int(b), ip.split('.'))$
2: $encode(transform(ip), size = ": 032b")$

Ce traitement est associé à la suppression de l'adresse IP du dataset puis la concaténation de celui-ci avec le vecteur de 32 bits produit. Une autre approche intéressante consiste à écarter l'adresse IP et utiliser une API externe pour obtenir la réputation de l'adresse IP. Par la suite, l'ensemble des données des différents systèmes sera confronté à un vote.

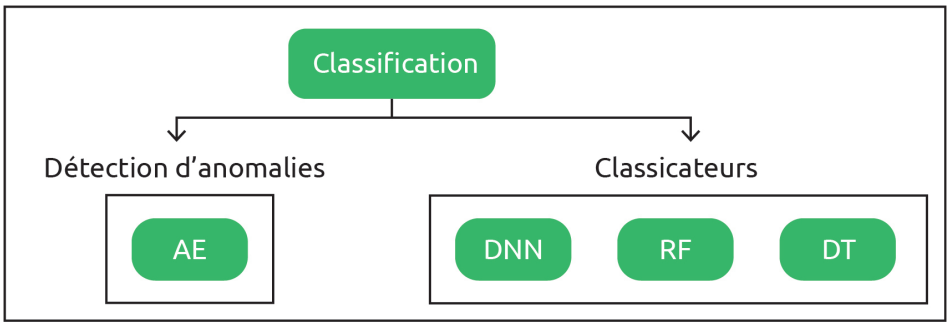


FIGURE 4.6 – Module de classification

4.3.2 Normalisation

Pour cette partie, nous avons opté pour MinMax Scaler pour normaliser les données dans un intervalle entre 0 et 1. Le module Scikit-Learn propose cet algorithme. Une fois le scaler instancié, deux fonctions, nous intéresse : fit et transform

Algorithm 6 Normalisation des données numérique à l'aide de MinMaxScaler

Require: ip

1: $transform = lambda(ip) : reduce(lambda(a, b) : int(a) * 256 + int(b), ip.split('.'))$

2: $encode(transform(ip), size = ": 032b")$

A noter que nous utilisons la fonction fit dans l'ensemble d'apprentissage uniquement et nous appliquons la transformation sur l'ensemble des données. La transformation sera donc basée sur les valeurs de l'ensemble d'apprentissage. Dans la partie Expérimentation, nous comparons les performances des classifieurs avec et sans normalisation.

4.4 Module 3 : Classification

Dans cette partie, nous présenterons plusieurs modèles en comparaison. La figure 4.6 réprésente deux classes, la première étant la classe des classificateurs séléctionnés à savoir :

— Réseau de neurones denses DNN

— Arbre de décisions DT

— Fôret aléatoire RF

Et la seconde classe, celle de la détection d'anomalies avec le modèle séléctionné :

— Un auto-encodeur composé de couches denses AE

De la, nous utilisons des classificateurs d'apprentissage automatique à savoir un arbre de décision et une forêt aléatoire et de la classification avec des réseaux de neurones artificiels avec un réseau de

neurones denses, un auto encodeur composé de couches denses. Ce module fera la distinction entre une entité légitime, ou une attaque. Il est donc très essentiel. Nous savons qu'avoir un modèle qui correspond à la problématique posée peut s'avérer exhaustif. Par nécessité, nous allons confronter plusieurs modèles à une comparaison.

Dans un premier temps, les modèles seront soumis à plusieurs tests sur une base de données équilibrées de 1.7 millions d'instances, puis nous sélectionnerons le modèle et paramètres les plus performants et adéquats. Ces paramètres sont les suivants :

— Utilisation de la normalisation des données

— Utilisation de l'encodage des données

— Utilisation de l'approche 3B - One Hot Encoding pour l'encodage des données

4.4.1 Fôrets aléatoires et Arbres de décision

Nous utiliserons la bibliothèque Scikit-learn pour instancier les différents arbres de décisions avec les paramètres par défaut, et les forêts aléatoires avec une profondeur maximale de 3 en paramètre.
L'instruction (1) instancie un classificateur RF depuis $fromsklearn.ensembleimportRandomForestClassifier$.
L'instruction (2) via $fromsklearn.treeimportDecisionTreeClassifier$.

Algorithm 7 RF et DT proposés

1: $rf = RandomForestClassifier(max\_depth = 3)$

2: $dt = DecisionTreeClassifier()$

4.4.2 Réseau de neurones profond

Les réseaux de neurones sont très performants quand il s'agit de résoudre des problèmes de classifications. L'architecture du classificateur basé sur un réseau de neurones denses, que nous utiliserons pour classifier les flux réseaux générés est décrite ci dessous 4.7.
Constitué d'une première couche dense de 64 unités, d'une couche de Batch Normalisation qui aura pour effet de recentrer et remettre à l'échelle les données entrées par lots. Puis en troisième couche dense de 32 unités puis une couche Dropout puis une couche dense de 16 unités et enfin une couche dense d'une unité pour la classification binaire.
Les fonctions d'activations sont ReLU pour toutes couches cachées exceptées la dernière couche dont la fonction d'activations est Sigmoid.

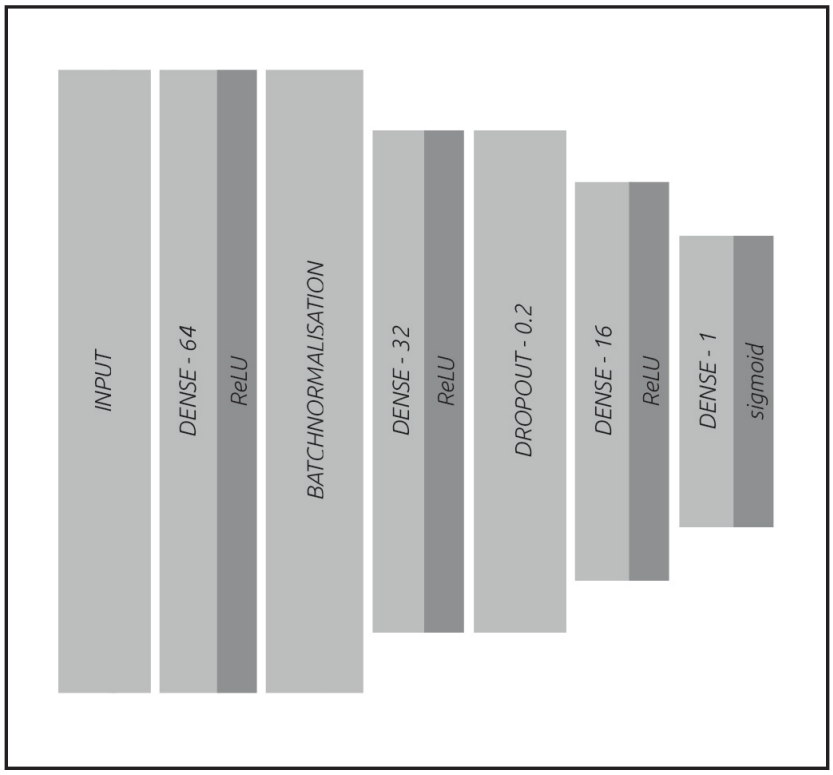


FIGURE 4.7 – DNN proposé

Algorithm 8 Classificateur DNN proposé

Require: n

1: $model = tf.keras.Sequential()$

2: $model.add(layers.InputLayer(n))$

3: $model.add(layers.Dense(64, activation =' relu'))$

4: $model.add(layers.BatchNormalization())$

5: $model.add(layers.Dense(32, activation = "relu"))$

6: $model.add(layers.Dropout(0.2))$

7: $model.add(layers.Dense(16, activation =' relu'))$

8: $model.add(layers.Dense(1, activation =' sigmoid'))$

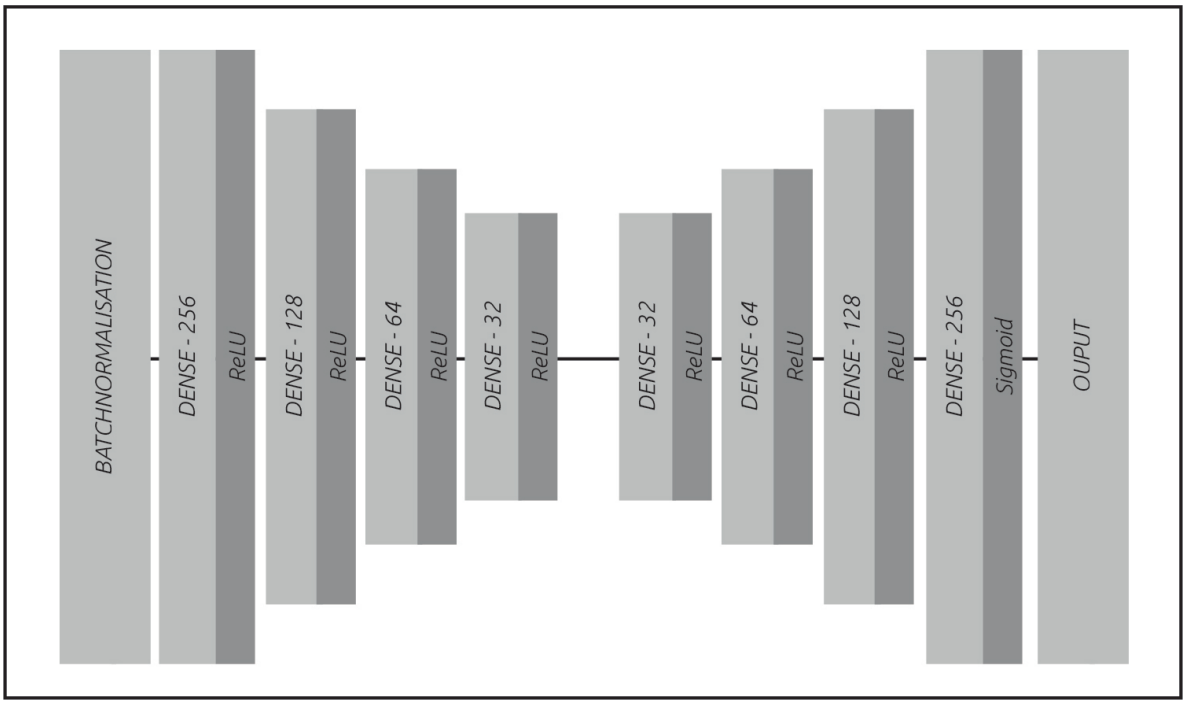


FIGURE 4.8 – Auto-encodeur proposé

4.4.3 Auto-encodeur

Notre proposition d'un auto encodeur se fonde sur la capacité de ce modèle a détecter les anomalies. En outre, nous voulons entraîner un auto encodeur avec des données normales (légitimes) puis calculer le seuil d'erreur de reconstruction des données d'apprentissage. Ce seuil d'erreur fera office de classificateur, et est calculer selon la formule suivante :

$$threshold = \mu(loss) + 2 * \sigma(loss)$$

μ : fonction moyenne

σ : écart type

loss : l'erreur de reconstruction de l'ensemble des données d'apprentissage

L'architecture 4.8 de notre auto-encodeur est la suivante : Pour l'encodeur, on commence par une couche de Batch Normalisation puis une couche dense de 256 unités, c'est alors que le nombre d'unités des couches denses qui suivront est divisé par 2 jusqu'à atteindre 32 unités pour dernière couche (256 - 128 - 64 - 32) .

Pour le décodeur, cette fois on multiplie le nombre d'unités jusqu'à atteindre les 256 unités de base (32 - 64 - 128 - 256). Comme pour le modèle précédent, seule la dernière couche a une fonction Sigmoid ;

ReLu pour les autres des couches cachées.

Algorithm 9 AE proposé

Require: n

1: $encoder = tf.keras.Sequential()$
2: $encoder.add(layers.BatchNormalization())$
3: $encoder.add(layers.Dense(256, input\_dim = n))$
4: $encoder.add(layers.Dense(128, activation =' relu'))$
5: $encoder.add(layers.Dense(64, activation = "relu"))$
6: $encoder.add(layers.Dense(32, activation = "relu"))$
7: $decoder = tf.keras.Sequential()$
8: $decoder.add(layers.Dense(32, input\_dim = n))$
9: $decoder.add(layers.Dense(64, activation =' relu'))$
10: $decoder.add(layers.Dense(128, activation = "relu"))$
11: $decoder.add(layers.Dense(256, activation = "sigmoid"))$
12: $decoder.add(layers.Dense(n))$
13: $model = encoder(x)$
14: $model = decoder(model)$

4.5 Module 4 : Déploiement

Nous avons jusqu'ici les modules essentiels pour classifier des datagrammes prétraités. Le dernier module représente le déploiement du classificateur afin de pouvoir faire des prédictions dans un environnement exposé. Afin de mener les tests, nous avons inclus tout les modules dans un pogramme Python. Ceci dans le but de limiter les latences dues aux requêtes à l'API, mais aussi d'interchanger les modèles, de les entraîner ou re-entraîner lorsqu'on le souhaite, et essentiellement de paramétrer le traitement des donnéees. Le modèle sélectionné, ainsi que les paramètres du traitement des données seront chargé lors du démarrage du programme.

Chapitre 5

Implémentation et évaluation

5.1 Environnement et outils de travail

5.1.1 Matériels

— Google Collaboratory
— Pycharm 2020
— i5 7200 3.1 GHZ
— 16GB DDR4

5.1.2 Langages de programmation et modules

— Python 3.7.10
— **Tensorflow 2.5** [Tensorflow, 2021]
TensorFlow offre la possibilité de créer des modèles d'apprentissage automatique complets et robustes . Tensorflow sera utilisé pour implémenter les différents modèles d'apprentissage profond AE et DNN via Keras.
— **Scikit-learn** [Pedregosa et al., 2011]
Cet outils robuste et complet d'apprentissage automatique regroupe plusieurs algorithmes dont les arbres de décision, les forêts aléatoires, ainsi que des fonctions applicables sur les ensembles de données (découpage, scores etc..)
— **Pandas** [Scapy, 2021]
Pandas est sous aucun doute la bibliothèque la plus utilisé pour manipuler des fichiers CSV. Elle fournit des fonctions puissantes pour le calcul et la manipulation des données. [Scapy, 2021]
— **Numpy** [Numpy, 2021]

Numpy est un module pour le calcul scientifique sous Python. Il s'agit d'une bibliothèque Python qui fournit un objet de type *ndarray* qui est une structure de données qui stocke et accède efficacement tableaux multidimensionnels [Charles R. Harris1 et al., 2020]. Il fournit une large variété de calculs scientifiques.

— **Scapy**[Pandas, 2021]

Scapy est un programme pour la manipulation de paquets réseaux. Il est capable de forger ou de décoder des paquets d'un grand nombre de protocoles, de les envoyer sur le fil, de les capturer, de faire correspondre les demandes et les réponses, et bien plus encore. Il peut facilement gérer la plupart des tâches classiques comme l'analyse, le traçage, le sondage, les tests unitaires, les attaques ou la découverte de réseau (il peut remplacer hping, 85% de nmap, arpspoof, arp-sk, arping, tcpdump, tshark, p0f, etc.).

— **FlowMeter** [Prc-hsv,]

FlowMeter est un programme open-source codé en Python qui reproduit les mêmes fonctionalités que le programme CIC FlowMeter [Sharafaldin et al., 2019a] ; c'est à dire extraire les features des paquets réseaux.

5.2 Base de données

Les bases de données citées ci-dessous ont les mêmes attributs. Elles contiennent des statistiques des flux réseaux qui ont été généré à partir de capture de paquets dans un environnement de simulation puis converti en format csv grâce au programme CIC FlowMeter. Voici la liste des attributs que nous retrouvons :

TABLE 5.1: Liste des attributs résumée

Attribut	Description
flow duration	Flow duration
tot fw pk	Total packets in the forward direction
tot bw pk	Total packets in the backward direction
tot l fw pkt	Total size of packet in forward direction
fw pkt l max	Maximum size of packet in forward direction
fw pkt l min	Minimum size of packet in forward direction
fw pkt l avg	Average size of packet in forward direction
fw pkt l std	Standard deviation size of packet in forward direction
Suite à la page suivante	

Attribut	Description
Bw pkt l max	Maximum size of packet in backward direction
Bw pkt l min	Minimum size of packet in backward direction
Bw pkt l avg	Mean size of packet in backward direction
Bw pkt l std	Standard deviation size of packet in backward direction
fl byt s	flow byte rate that is number of packets transferred per second
fl pkt s	flow packets rate that is number of packets transferred per second
fl iat avg	Average time between two flows
fl iat std	Standard deviation time two flows
fl iat max	Maximum time between two flows
fl iat min	Minimum time between two flows
fw iat tot	Total time between two packets sent in the forward direction
fw iat avg	Mean time between two packets sent in the forward direction
fw iat std	Standard deviation time between two packets sent in the forward direction
fw iat max	Maximum time between two packets sent in the forward direction
fw iat min	Minimum time between two packets sent in the forward direction
bw iat tot	Total time between two packets sent in the backward direction
bw iat avg	Mean time between two packets sent in the backward direction
bw iat std	Standard deviation time between two packets sent in the backward direction
bw iat max	Maximum time between two packets sent in the backward direction
bw iat min	Minimum time between two packets sent in the backward direction
fw psh flag	Number of times the PSH flag was set in packets travelling in the forward direction
bw psh flag	Number of times the PSH flag was set in packets travelling in the backward direction
fw urg flag	Number of times the URG flag was set in packets travelling in the forward direction
bw urg flag	Number of times the URG flag was set in packets travelling in the backward direction
fw hdr len	Total bytes used for headers in the forward direction
bw hdr len	Total bytes used for headers in the forward direction
fw pkt s	Number of forward packets per second
bw pkt s	Number of backward packets per second
pkt len min	Minimum length of a flow
pkt len max	Maximum length of a flow
pkt len avg	Mean length of a flow
pkt len std	Standard deviation length of a flow

Suite à la page suivante

Attribut	Description
pkt len va	Minimum inter-arrival time of packet
fin cnt	Number of packets with FIN
syn cnt	Number of packets with SYN
rst cnt	Number of packets with RST
pst cnt	Number of packets with PUSH
ack cnt	Number of packets with ACK
urg cnt	Number of packets with URG
cwe cnt	Number of packets with CWE
ece cnt	Number of packets with ECE
down up ratio	Download and upload ratio
pkt size avg	Average size of packet
fw seg avg	Average size observed in the forward direction
bw seg avg	Average size observed in the backward direction
fw byt blk avg	Average number of bytes bulk rate in the forward direction
fw pkt blk avg	Average number of packets bulk rate in the forward direction
fw blk rate avg	Average number of bulk rate in the forward direction
bw byt blk avg	Average number of bytes bulk rate in the backward direction
bw pkt blk avg	Average number of packets bulk rate in the backward direction
bw blk rate avg	Average number of bulk rate in the backward direction
subfl fw pk	The average number of packets in a sub flow in the forward direction
subfl fw byt	The average number of bytes in a sub flow in the forward direction
subfl bw pkt	The average number of packets in a sub flow in the backward direction
subfl bw byt	The average number of bytes in a sub flow in the backward direction
fw win byt	Number of bytes sent in initial window in the forward direction
bw win byt	# of bytes sent in initial window in the backward direction
Fw act pkt	# of packets with at least 1 byte of TCP data payload in the forward direction
fw seg min	Minimum segment size observed in the forward direction
atv avg	Mean time a flow was active before becoming idle
atv std	Standard deviation time a flow was active before becoming idle
atv max	Maximum time a flow was active before becoming idle
atv min	Minimum time a flow was active before becoming idle
idl avg	Mean time a flow was idle before becoming active

Suite à la page suivante

Classe	Nombre
Normale (1)	899955
Attaque (0)	899955
Total	1 799 910

TABLE 5.2 – Répartition des observations de la base de tests des paramètres

Attribut	Description
idl std	Standard deviation time a flow was idle before becoming active
idl max	Maximum time a flow was idle before becoming active
idl min	Minimum time a flow was idle before becoming active

5.2.1 Base de données de tests

Nous avons créé une base de données dédiée aux tests des paramètres, échantillonnée de la base CIC IDS 2018. Elle comporte 2 classes ; normale étiquetée 1 et attaque étiquetée 0, regroupant toutes les classes d'attaques. La répartition des observations dans les deux classes est équivalente 5.2 ; ainsi c'est une base équilibrée pour un total de 1.7 millions d'instances (899555 par classe).

5.2.2 CSE CIC IDS 2018

Cette base de données est d'autant plus intéressante dans le cadre de cette étude car elle contient une grande variété dans le type d'instances. Elle fut conçue par [Sharafaldin et al., 2018]. Les attaques sont réparties en 10 jours.

— Attaques de force brutes
— Attaques DoS
— Attaques Web
— Infiltration
— Attaques Botnet
— Attaques DDoS
— Scan de port

Une base de données plus récente nommée CIC DDoS 2019 [Sharafaldin et al., 2019b] issue de la même organisation, qui comporte les attaques DDoS par réflexion et d'exploitation de vulnérabilité tout aussi intéressante ; attaques d'innondation SYN, UDP etc.. Mais, notre sélection de la base de

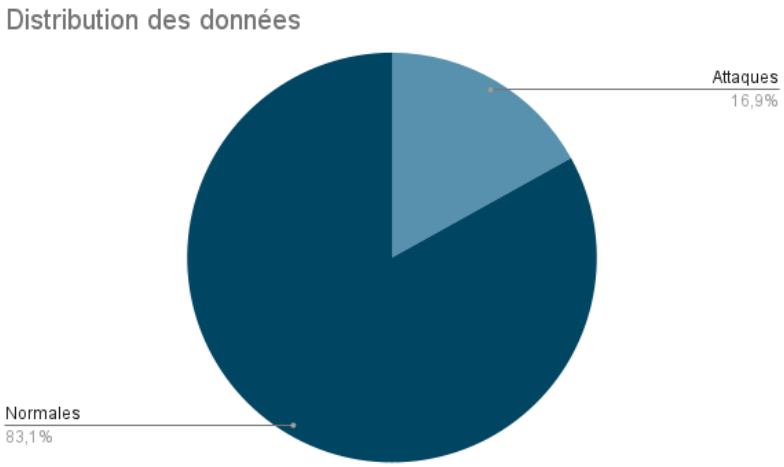


FIGURE 5.1 – Distribution des données de CIC IDS 2018

Modèle	Loss	Optimizer
DNN	Binary Cross Entropy	Adam
AE	Mean Absolute Error	Adam

TABLE 5.3 – Hyperparamètres des modèles AE et DNN

données s'est orientée vers la moins récente en vue de richesse de la quantité d'instances bénines.

Analyse exploratoire

Les figures 5.1 et 5.2 représentent la distribution des données et répartition des données dans la classe d'attaque.

5.3 Tests des paramètres

Dans cette partie nous testons nos classificateurs sur la base de données de tests 5.2.1 composée de 1.7m d'observations étiquetées . 20 % des données sont dédiées à la prédiction 5.4. Les modèles DNN et AE ont un essemble de validation 30% divisé de l'essemble d'apprentissage.

Les tables 5.5 et tables 5.7 représentent les modèles annontés Type-Variante, le score d'exactitude de la classification, le temps d'apprentissage en seconde noté $\mu_l(s)$, le temps de prédiction en seconde $\mu_p(s)$, si la normalisation ait été utilisée, si l'encodage ait été utilisé, et si l'approche 3b pour l'encodage ait été utilisée par exemple :

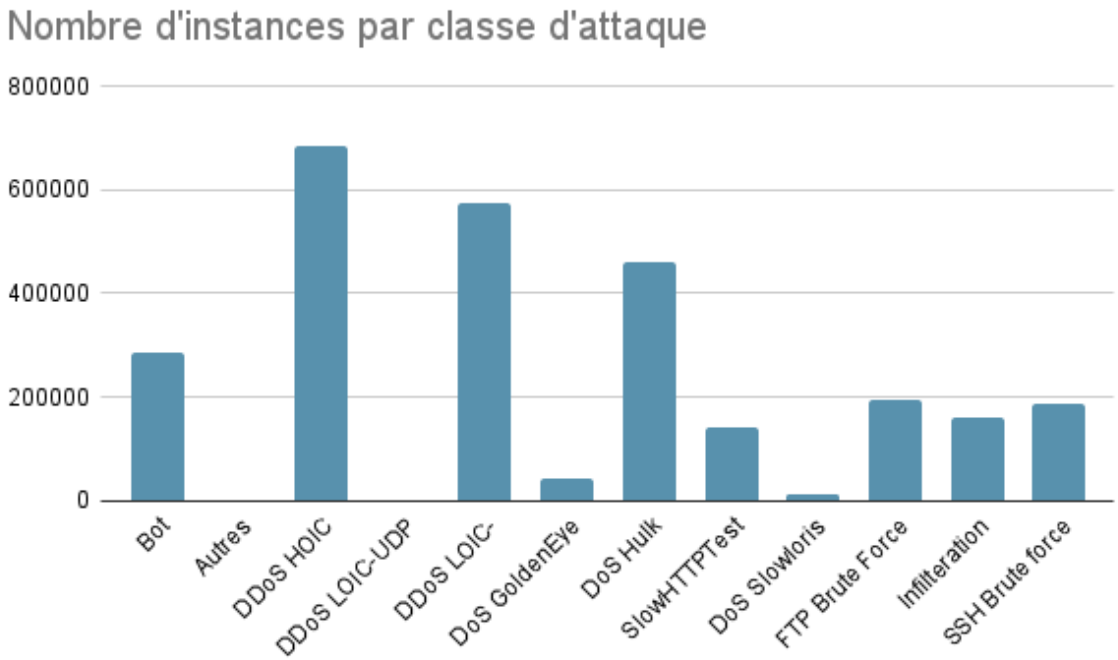


FIGURE 5.2 – Classes d'attaques de CIC IDS 2018

Essemble	Quantité
Apprentissage	80 %
Tests	20 %

TABLE 5.4 – Distribution des données des tests des paramètres

Modèles	**Accuracy**	$\mu_l(s)$	$\mu_p(s)$	**Normalisation**	**Encoding**	**3b**
RFA	0.89	161.04	2.54	Oui	Non	Non
RFB	0.88	**149.73**	2.51	Oui	Oui	Non
RFC	0.89	172.84	2.50	Non	Non	Non
RFD	0.90	155.12	2.52	Non	Oui	Non
RFE	0.89	167.47	2.54	Non	Oui	Oui
RFF	0.89	**145.64**	2.55	Oui	Oui	Oui
DTA	0.72	57.71	0.46	Oui	Non	Non
DTB	0.73	69.00	0.49	Oui	Oui	Non
DTC	**0.95**	54.07	0.49	Non	Non	Non
DTD	0.95	70.61	0.50	Non	Oui	Non
DTE	0.95	68.62	0.50	Non	Oui	Oui
DTF	0.72	68.85	0.47	Oui	Oui	Oui
DnnA	**0.94**	142.58	**7.91**	Oui	Non	Non
DnnB	**0.94**	113.22	8.14	Oui	Oui	Non
DnnC	0.81	142.79	7.71	Non	Non	Non
DnnD	0.87	114.69	7.92	Non	Oui	Non
DnnE	0.84	113.41	10.76	Non	Oui	Oui
DnnF	**0.94**	116.84	**7.53**	Oui	Oui	Oui
AAA	**0.68**	**202.80**	**2.31**	Oui	Non	Non
AAB	0.52	211.19	2.51	Oui	Oui	Non
AAC	0.48	197.53	2.45	Non	Non	Non
AAD	0.48	209.43	2.79	Non	Oui	Non
AAE	0.48	209.80	2.75	Non	Oui	Oui
AAF	0.52	262.77	2.69	Oui	Oui	Oui

TABLE 5.5 – Tests avec la séléction des données

— DnnA pour DNN variante A, le score d'exactitude est de 0.94 avec un temps d'apprentissage de 142.58 secondes et de temps de prédiction (classification) de 7.91 secondes. Cette variante du modèle DNN traite des données **normalisées** mais **non encodées**.

Les modèles sont :

— RF**X** pour RF, forêt aléatoire, le symbole **X** réprésente la variante.

— DT**X** pour DT, arbre de décision, le symbole **X** réprésente la variante.

— Dnn**X** pour DNN, réseau de neurones denses, le symbole **X** réprésente la variante.

— AA**X** pour AE, auto-encodeur, le symbole **X** réprésente la variante.

Les variantes sont :

— A : Données normalisées seulement.

— B : Données normalisées et encodées.

— C : Données brutes.

— D : Données encodées seulement.

— E : Données encodées et usage de l'approche 3B

— F : Données normalisées, encodées et usage de l'approche 3B

Les résultats figurants dans le table 5.5 représentent la comparaison entre 4 groupes de modèles sur les données sélectionnées à l'aide de la corrélation de pearson. Les modèles RF partagent globalement les mêmes performances avec 0.89 en accuracy et 2.50 (s) en temps de prédiction, la différence demeure dans le temps d'apprentissage lorsque les données sont encodées et normalisées, avec une légère avance sur la précision et le temps d'apprentissage pour le RFF dont l'encodage du protocole est en 3b 3.

Pour le groupe des arbres de décision DT, la différence est plus flagrante. Avec le DTC marquant les meilleures performances tout groupe confondu, avec 0.95 en accuracy. Cette fois ci les données sont telles qu'elles. Nous remarquons que la normalisation affecte négativement les performances des modèles. Nous constatons que dans le groupe 3, groupe des DNN, la normalisation impacte positivement les modèles avec un avantage pour le modèle DnnB ayant jusqu'ici la meilleure sensibilité (Recall) de tout les groupes confondus. Nous constatons que l'encodage données améliore le temps d'apprentissage et légèrement l'exactitude (Accuracy). Nous noterons le modèle DnnF classifie les données le plus rapidement au sein de son groupe. Nous noterons une très légère différence entre le modèle DnnB et DnnA, l'écart se trouve dans le temps de prédictions ; une métrique tout aussi importante. Et finalement le dernier groupe AE sujet au tests avec des données sélectionnées, là où les performances sont les moins bonnes, notament dans le temps d'apprentissage et du score d'exactitude (Accuracy). Toutefois nous remarquons encore une fois l'impact positive sur les performances des modèles d'apprentissage profond sur des données normalisées. L'auto encodeur AAA prend tout de même l'avantage en temps de prédiction sur l'entièreté du groupe des DNN, nous rappelons qu'il s'agit de deux groupes de modèles d'apprentissage profond. Le modèle qui répond le plus au critères est sans aucun doute le modèle DTC

avec des features sélectionnées, non normalisées et non encodées.
Pour le prochain tests nous verrons l'impact de la selection des données sur les différents modèles.

Le tableau 5.6 montre clairement que la séléction des features à l'aide du coéficient de corrélation de Pearson, améliore le temps d'execution des différents algorithmes et les performances des modèles ; la réduction de dimension. Hormis pour certains modèles montrant de meilleures performances sans la selection, c'est le cas de l'arbre de décision et certains variants du réseau de neurones denses. Globalement les performances ont été meilleures dans le premier test. Ainsi, nous baserons la selection sur le résultats des premiers tests sur les données séléctionnées.

5.4 Évaluation du modèle

Les deux modèles sélectionnés sont l'arbre de décision C et le réseau de neurones A selon leurs paramètres avec des données séléctionnées. Le modèle DnnA prend le dessus sur le modèle DnnB sur le temps de prédiction, il est donc plus valorisé et c'est pour cela qu'il été sélectionné pour l'évaluation du modèle sur l'entièreté de la base CIC IDS 2018.
En raison de la taille du dataset, nous l'avons chargé en mémoire en tronçons, puis nous entrainons les deux modèles sur 70% du tronçons, nous utilisons les 30% restants pour l'évaluation. Le tableau 5.7 représente les résultats de l'entrainement et l'évaluation sur l'entièrté de la base de données. Nous constons cette fois que le modèle basé sur un réseau de neurones artificiels est plus performants en termes d'exactitude sur la classification des instances.

Modèles	**Accuracy**	$\mu_l(s)$	$\mu_p(s)$	**Normalisation**	**Encoding**	**3b**
RF A	0.87	222.65	2.72	Oui	Non	Non
RF B	0.87	190.40	2.84	Oui	Oui	Non
RF C	0.88	196.21	2.77	Non	Non	Non
RF D	**0.89**	206.46	2.79	Non	Oui	Non
RF E	0.88	252.21	2.72	Non	Oui	Oui
RF F	0.87	220.62	2.69	Oui	Oui	Oui
DT A	0.84	103.36	0.51	Oui	Non	Non
DT B	0.89	107.48	0.49	Oui	Oui	Non
DT C	**0.95**	126.18	0.53	Non	Non	Non
DT D	0.95	126.97	0.58	Non	Oui	Non
DT E	0.95	130.56	0.54	Non	Oui	Oui
DT F	0.88	110.86	0.85	Oui	Oui	Oui
DnnA	0.93	165.75	10.85	Oui	Non	Non
DnnB	0.84	164.07	10.81	Oui	Oui	Non
DnnC	0.92	167.48	10.83	Non	Non	Non
DnnD	0.79	164.22	10.79	Non	Oui	Non
DnnE	0.82	202.68	9.28	Non	Oui	Oui
DnnF	**0.94**	202.68	10.42	Oui	Oui	Oui
AAA	**0.58**	323.03	2.98	Oui	Non	Non
AAB	0.55	323.02	5.31	Oui	Oui	Non
AAC	0.46	315.41	2.98	Non	Non	Non
AAD	0.46	322.90	3.72	Non	Oui	Non
AAE	0.46	304.71	3.44	Non	Oui	Oui
AAF	0.58	322.89	3.25	Oui	Oui	Oui

TABLE 5.6 – Tests sans la séléction des données

Modèle	Accuracy	Normalisation	Encoding	3b	Selection
DnnA	**0.99089**	Oui	Non	Non	Oui
DTC	0.99023	Non	Non	Non	Oui

TABLE 5.7 – Résultats finaux

Chapitre 6

Conclusion générale

Dans cette étude, nous avons exposé une solution pour détecter des attaques de déni de services appliquable dans un environement exposé. La première étape était de découvrir les moyens de classification ou de détection d'anomalies. Nous avons vu au cours de l'expérimentation 4 modèles dont 2 d'apprentissage automatique et deux d'apprentissage profond, respectivement forêts aléatoires, arbres de décision, réseau de neurones simple et un auto encodeur simple. Les trois premiers modèles sont des classificateurs, ils sont entrainés a prédire une valeur booléenne significative d'une des deux classes ; si l'instance prédite est maligne ou bénine. Quant à l'auto encodeur, il est entrainé à reproduire des instances bénines et ainsi utilisé l'erreur de reconstruction pour détecter si l'instance reproduite est une attaque.
La comparaison des paramètres sur les modèles a déterminé que les classificateurs présentés étaient plus performants que l'auto-encodeur proposé. Nous avons séléctionné les paramètres et les deux modèles les plus prometteur à savoir un arbre de décision (DT C) avec un score de 0.95 ainsi que le réseau de neurones artificiels (DnnA) avec un score de 0.94 sur des données séléctionnées à l'aide de la corrélation de Pearson. Nous avons remarqué que le fait de supprimer les données redondantes n'impacte pas uniquement le temps d'execution, mais aussi l'exactitude des modèles. L'évaluation des classificateurs sur toute la base de données CSE CIC IDS 2018 montre cette fois que le modèle basé sur le réseau de neurones est légèrement plus performant que l'arbre de décision avec 0.99089 contre 0.99023 en accuracy. La méthode employée derrière cette étude motivante, peut s'exporter sur la base CIC DDoS 2019.

Les travaux menés dans cette thèse peuvent être poursuivis dans différentes directions. Tout d'abord, nous pouvons améliorer les modèles en utilisant des bases de données plus récentes comme nous l'avons énoncé dans le précédent paragraphe. La classification pourra être alors plus performante. Parallèlement, nous pouvons envisager d'adopter un autre raisonnement qui consiste a considérer le

paquet comme une matrice et ainsi permettre l'inspection approfondie des paquets DPI, ce qui ajoute une couche de sécurité supplémentaire lorsqu'elle est déployée correctement autour d'équipements de sécurité. En règle générale, il n'y a pas de limitations concernant le déploiement. Celui-ci dépendra d'un choix fonctionnel, il peut être sous forme d'image Docker, de machine virtuelle, ou encore de fonction serverless exécutée dans le cloud.

La collecte du trafic réseau, en tant que telle, peut se faire à différent niveau, soit à travers des outils de gestion de l'information et d'événements de sécurité (SIEM) par exemple la suite ELK (Elasticsearch, Logstash et Kibana), soit en intégrant un module à des IPS/IDS tel que Snort [Snort, 2021], à des pare-feux, et autres équipements etc..

En effet, les travaux autour de l'utilisation de l'intelligence artificielle pour régler des problèmes de cybersécurité sont très intéressant. La détection est la première étape dans la voie de la sécurisation. De plus, les travaux actuels portent sur des données statistiques sur des paquets, et nous pourrions apporter la possibilité de passer en dimension supérieure, celle de l'inspection approfondie des paquets.

Bibliographie

[rfc, 1986] (1986). Survey of data representation standards. RFC 971.

[ion, 2020] (2020). Udp flood.

[clo, 2021a] (2021a). Http flood.

[clo, 2021b] (2021b). Loic.

[clo, 2021c] (2021c). Slowloris.

[Ah-Pine, 2019] Ah-Pine, J. (2019). Apprentissage automatique.

[Anjum, 2011] Anjum, W. (2011). Performance eveluation of ipv4 and ipv6 networks in absence of link layer protection.

[ANSSI, 2015] ANSSI (2015). *Comprendre et anticiper les attaques DDoS*.

[Bacha, 2019] Bacha, S. (2019). *Cours Intelligence artificielle, Licence Informatique USDB*.

[Baeldung, 2019] Baeldung (2019). How relu and dropout layers work in cnns.

[Benesty et al., 2009] Benesty, J., Chen, J., Huang, Y., and Cohen, I. (2009). *Pearson Correlation Coefficient*, pages 1–4. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Biesiada and Duch, 2007] Biesiada, J. and Duch, W. (2007). *Feature Selection for High-Dimensional Data — A Pearson Redundancy Based Filter*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Bårli, 2019] Bårli, E. (2019). Ddos and dos mitigation using a variational autoencoder.

[CGE et al., 2019] CGE, ARCEP, and numérique, A. (2019). Baromètre du numérique.

[Charles R. Harris1 et al., 2020] Charles R. Harris1, K. J. M., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., and al. (2020). Array programming with numpy.

[Cheng and Watson, 2017] Cheng, R. and Watson, G. (2017). D^{2}pi : Identifying malware through deep packet inspection with deep learning.

[Cnam., a] Cnam., C. Arbres de décision.

[Cnam., b] Cnam., C. Forêts aléatoires.

[de Lima Filho et al., 2019] de Lima Filho, F. S., Silveira, F. A. F., Junior, A. M. B., Vargas-Solar, G., and Silveira, L. F. (2019). Smart detection : An online approach for dos/ddos attack detection using machine learning.

[Dharshini, 2021] Dharshini, R. (2021). Towards enhancement of machine learning techniques using cse-cic-ids2018 cybersecurity dataset.

[DNSstuff, 2019] DNSstuff (2019). How to stop, prevent, and protect yourself from a ddos attack.

[Enchun, 2019] Enchun, S. (2019). *Encoding IP address as feature for network intrusion detection.*

[Ferrag et al., 2019] Ferrag, M. A., Janicke, H., L., M., and Smith, R. (2019). Deep learning techniques for cyber security intrusion detection : A detailed analysis.

[Jaitley, 2018] Jaitley, U. (2018). Why data normalization is necessary for machine learning models.

[K, 2020] K, R. (2020). Problèmes de classification.

[Kai-Cheng et al., 2020] Kai-Cheng, C., Chien-Chang, L., and Li-der, C. (2020). Capc : Packet-based network service classifier with convolutional autoencoder.

[Keldenich, 2021] Keldenich, T. (2021). Fonction d'activation.

[Keras, 2021] Keras (2021). Dropout layer.

[Khan and Kim, 2020] Khan, M. A. and Kim, J. (2020). Toward developing efficient conv-ae-based intrusion detection system using heterogeneous dataset.

[Kiourkoulis, 2020] Kiourkoulis, S. (2020). Ddos datasets use of machine learning to analyse intrusion detection performance.

[Leevy and Khoshgoftaar, 2020] Leevy, J. L. and Khoshgoftaar, T. M. (2020). A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data.

[Li, 2020] Li, J. (2020). *Detection of DDoS attacks based on dense neural networks, autoencoders and Pearson correlation coefficient.*

[Lotfollahi et al., 2018] Lotfollahi, M., Siavoshani, J. M., Shirali, R. H. Z., and Saberian, M. (2018). Deep packet : A novel approach for encrypted traffic classification using deep learning.

[Marìn et al., 2020] Marìn, G., Casas, P., and G, C. (2020). Deepmal - deep learning models for malware traffic detection and classification.

[Nasir et al., 2020] Nasir, I. M., Khan, M. A., Yasmin, M., Shah, J. H., Gabryel, M., Scherer, R., and Damaševičius, R. (2020). Pearson correlation-based feature selection for document classification using balanced training. *Sensors*, 20(23).

[Numpy, 2021] Numpy (2021). Numpy, the fundamental package for scientific computing with python.

[Pan et al., 2018] Pan, W., Feng, Y., Xuejiao, C., and Yi, Q. (2018). Datanet : Deep learning based encrypted network traffic classification in sdn home gateway.

[Pandas, 2021] Pandas (2021). pandas - python data analysis library.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn : Machine learning in python. *Journal of machine learning research*, 12(Oct) :2825–2830.

[Picaxe, 2014] Picaxe, M. (2014). Les adresses ip.

[Prc-hsv,] Prc-hsv. Prc flowmeter v0.2.0.

[S. and J., 2019] S., A. and J., D. (2019). Hagent based preventive measure for udp flood attack in ddos attacks.

[Scapy, 2021] Scapy (2021). Scapy, packet crafting for python2 and python3.

[Sery, 2014] Sery, N. (2014). Le modèle osi et le modèle tcp/ip.

[Sharafaldin et al., 2018] Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization.

[Sharafaldin et al., 2019a] Sharafaldin, I., Lashkari, A. H., Hakak, S., and Ghorbani, A. A. (2019a). Cicflowmeter is an open source tool that generates biflows from pcap files, and extracts features from these flows.

[Sharafaldin et al., 2019b] Sharafaldin, I., Lashkari, A. H., Hakak, S., and Ghorbani, A. A. (2019b). Developing realistic distributed denial of service (ddos) attack dataset and taxonomy.

[Snort, 2021] Snort (2021). Snort open source intrusion detection system, instrusion prevention system.

[Stratosphere, 2015] Stratosphere (2015). Stratosphere laboratory datasets. Retrieved March 13, 2020.

[Tensorflow, 2021] Tensorflow (2021). Tensorflow, une plate-forme open source de bout en bout dédiée au machine learning.

[XOIC, 2021] XOIC (2021). Xoic – launch dos attack with tcp/http/udp/icmp message.