الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE YAHIA FARES - MEDEA

FACULTE DES SCIENCES

Département Mathématiques et Informatique



Projet de fin d'études

Présenté pour l'obtention du Diplôme de

Master en Informatique

Option : Ingénierie des Systèmes et Technologies du Web

THEME

Approche basée sur l'apprentissage profond pour la détection d'intrusion réseau

Proposé et dirigé par :

Mr. SAHMADI Brahim

Préparé et présenté par :

Mr. BABAALI Baligh
Mr. LAIB Hamza

Année Universitaire 2018-2019

Dédicaces

Je dédie ce modeste travail
A tous ceux qui me connaissent, en particulier,
A mon père et à ma mère pour leurs sacrifices et leur soutien.
A ma femme pour sa grande patience,
son interminable soutien et encouragement
A mes filles:
Kawthar, Nour, Chaimaa, Hafsa, Walaa et Jouri.
A mes frères et ma sœur.
A ma belle nièce Aicha et son frère Mohamed pour leur encouragement.
A mon ami Cherif pour son soutien et sa présence.
A tous mes amis et collègues.
Sans oublier tous les professeurs qui ont contribué à ma formation de l'enseignement primaire, moyen et secondaire jusqu'à l'enseignement supérieur.

Baligh

Dédicaces

Je dédie ce modeste travail
A mon père Ammar et à ma mère Fatma
pour leurs sacrifices, leur patience et leur soutien.
A ma femme Manel pour sa patience, soutien et encouragement
A mes enfants:
Hadil, Mohamed Aman-allah et Assil Minat-allah.
A tous mes amis et collègues.
Sans oublier tous les professeurs qui ont contribué à ma formation de l'enseignement primaire, moyen et secondaire jusqu'à l'enseignement supérieur.

Hamza

Remerciements

En préambule à ce mémoire nous remercions ALLAH qui nous aide et nous donne la patience et le courage tout au long de notre vie et durant notre année d'étude.

Nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire.

Nous tenons à exprimer notre profonde reconnaissance et la preuve de notre grande gratitude pour Mr. Brahim SAHMADI, pour nous avoir proposé ce thème, pour son aide, ses conseils, sa disponibilité et le suivi qu'il nous a apporté tout au long de notre travail.

Nous tenons à remercier, également, les membres du jury pour avoir bien voulu examiner et avoir accepter de juger ce travail.

Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont toujours soutenu et encouragé au cours de la réalisation de ce mémoire.

Merci à toutes et à tous.

ملخص:

في الوقت الحاضر، تستخدم أنظمة كشف التسلل (IDS)على نطاق واسع في مجال أمن أنظمة الكمبيوتر. وقد استخدمت مختلف تقنيات التعلم الآلي بغرض تطوير هذه الأنظمة. مع ظهور تقنيات التعلم العميق ونجاحاتها في العديد من المجالات، قمنا بتطوير طريقة تستند على هذه الأساليب.

تم اختبار الطريقة المقترحة على قاعدة بيانات عامة CICIDS2017. لقد قمنا بتقليص حجم قاعدة البيانات باستخدام برنامج تشفير تلقائي مكدس، حيث حققنا معدل خسارة منخفض للغاية (10^{-5} إلى 10^{-6}). ثم قمنا بإجراء تصنيف (متعدد الطبقات) مع شبكة عصبية تلافيفية. حققنا دقة مشجعة للغاية (99.20 ٪). لقد أعطى النموذج الذي تم اختياره (SAE/CNN) نتائج مرضية للغاية مقارنةً بنتائج أعمال سابقة حيث تم استخدام طرق أخرى.

كلمات مفتاحية: الأمن المعلوماتي، كشف التسلل، التعلم العميق، CICIDS2017.

Résumé :

Les systèmes de détection d'intrusion sont largement répandus de nos jours pour la sécurité de systèmes informatiques. Diverses techniques d'apprentissage automatique ont été utilisées pour développer les IDS. Avec l'apparition des technologies d'apprentissage profond et ses succès dans plusieurs domaines, nous avons élaboré une approche basée sur ces méthodes.

L'approche proposée a été testée sur la base publique CICIDS2017. Nous avons effectué la réduction la dimensionnalité de la base à l'aide d'un auto-encodeur empilé. Nous avons atteint un taux de perte très réduit (10^{-5} à 10^{-6}). Ensuite, nous avons effectué une classification (multi-classes) avec un réseau de neurones convolutionnel. Nous avons atteint une précision très encourageante (99.20%). Le modèle choisi (SAE/CNN) a donné des résultats très satisfaisants en comparaison avec ceux des travaux antérieurs où d'autres approches ont été utilisées.

Mots clés : Sécurité informatique, Détection d'intrusion, Apprentissage profond, CICIDS2017.

Abstract:

Intrusion Detection Systems are widely used today for the security of computer systems. Various machine learning techniques have been used to develop IDS. With the advent of deep learning technologies and its successes in many areas, we have developed an approach based on these methods.

The proposed approach has been tested on the public database CICIDS2017. We reduced the dimensionality of the database using a stacked auto-encoder. We achieved a very low loss rate (10^{-5} to 10^{-6}). Then we performed a classification (multi-classes) with a convolutional neural network. We achieved a very encouraging accuracy (99.20%). The chosen model (SAE/CNN) gave very satisfactory results in comparison with those of previous work where other approaches were used.

Keywords: Information security, Intrusion Detection, Deep Learning, CICIDS2017.

Table des matières

LISTE DES FIGURES ..... **I**

LISTE DES TABLEAUX ..... **II**

LISTE DES ACRONYMES ..... **III**

INTRODUCTION GENERALE **1**

CHAPITRE I : SECURITE INFORMATIQUE **4**

I.1. Introduction **5**

I.2. Objectifs de la sécurité informatique **5**

I.3. Protection des systèmes informatiques **6**

I.4. Taxonomie des cyber-intrusions **8**

I.5. Quelques attaques courantes **9**

I.5.1. Attaque par balayage ICMP 9
I.5.2. Attaque par balayage TCP 9
I.5.3. Attaque par balayage semi-ouvert TCP 10
I.5.4. IP spoofing 11
I.5.5. Virus 11
I.5.6. Déni de service (DoS) / Déni de service distribué (DDoS) 11

I.6. Nécessité de la détection d'intrusions **12**

I.7. Conclusion **13**

CHAPITRE II : SYSTEME DE DETECTION D'INTRUSIONS **14**

II.1. Introduction **15**

II.2. Définition **15**

II.3. Architecture fonctionnelle des IDS **16**

II.3.1. Capteur 16
II.3.2. Analyseur 17
II.3.3. Manager 17

II.4. Fonctions de l'IDS **17**

II.5. Types des IDS **18**

II.5.1. IDS réseau 18
II.5.2. IDS hôte 19
II.5.3. IDS hybride 20

II.6. Classification des IDS **20**

II.6.1. Source des données à analyser 22

II.6.2. Méthode de détection utilisée. 22
II.6.3. Lieu de l'analyse des données. 22
II.6.4. Fréquence de l'analyse. 23
II.6.5. Comportement en cas de détection d'une attaque. 23
II.7. Efficacité des IDS 24
II.8. Limites et vulnérabilités des IDS 24
II.9. Conclusion 28
CHAPITRE III : APPRENTISSAGE PROFOND 29
III.1. Introduction 30
III.2. Définition 30
III.3. Principe 31
III.4. Classification de l'apprentissage profond 32
III.5. Quelques méthodes d'apprentissage profond 33
III.5.1. Machines de Boltzmann Restreintes 33
III.5.2. Perceptrons multicouches 34
III.5.3. Réseaux de croyance profonde (DBN) 36
III.5.4. Réseaux de neurones convolutionnels 38
III.5.5. Réseaux de neurones récurrents 44
III.5.6. Mémoire longue à court terme 46
III.5.7. Auto-encodeurs 48
III.5.8. Mesures d'évaluations des modèles 49
III.5.9. Conclusion 52
CHAPITRE IV : CONCEPTION DE LA METHODE PROPOSEE 53
IV.1. Introduction 54
IV.2. Schéma du modèle proposé 54
IV.3. Méthode proposée pour la réduction de dimensionnalité 55
IV.3.1. L'encodage 56
IV.3.2. Le décodage 57
IV.3.3. La reconstruction 58
IV.4. Classification des données 59
IV.4.1. Méthode de réseau de neurones convolutionnel (CNN) 59
IV.5. Conclusion 61
CHAPITRE V : TESTS ET RESULTATS 62
V.1. Introduction 63
V.2. Matériel et logiciels utilisés 63
V.3. Description de la base de données utilisée 64
V.4. Correction des données manquantes 65

V.5. Réduction de la dimensionnalité 66
V.5.1. Paramètre d'évaluation de l'auto-encodeur 66
V.5.2. Résultats de la réduction de la dimensionnalité 67
V.5.3. Analyse des résultats de la réduction de la dimensionnalité 67
V.6. Classification 70
V.6.1. Paramètre d'évaluation du réseau CNN 71
V.6.2. Résultats de la classification 72
V.6.3. Analyse des résultats de la classification 72
V.6.4. Réglage fin des hyper-paramètres du réseau CNN 82
V.7. Comparaison du modèle avec les modèles des recherches antérieures 84
V.8. Conclusion 85
CHAPITRE VI : CONCLUSION 86
REFERENCES BIBLIOGRAPHIQUES

LISTE DES FIGURES

Figure I.1. *Taxonomie des cyber-intrusions [7]* 8
Figure I.2. *Fonctionnement de la commande ping.* 9
Figure I.3. *Attaque par balayage TCP.* 10
Figure I.4. *Attaque IP spoofing.* 11
Figure I.5. *Attaque par déni de service distribué (DDoS).* 12
Figure II.1. *Schéma fonctionnel du Système de détection d'intrusion (IDS)* 16
Figure II.2. *Exemple d'une architecture d'un NIDS* 18
Figure II.3. *Modèle de système de détection d'intrusion* 19
Figure II.4. *Exemple d'une architecture d'un HIDS* 20
Figure II.5. *Classification des systèmes de détection d'intrusion* 21
Figure III.1. *Classification des méthodes d'apprentissage profond* 32
Figure III.2. *Machine de Boltzman restreinte* 34
Figure III.3. *Perceptron multicouches* 35
Figure III.4. *Flux de signal d'un MLP* 35
Figure III.5. *Schéma d'un réseau DBN* 37
Figure III.6. *Schéma d'un réseau de neurones convolutionnel* 38
Figure III.7. *Exemple d'une carte des caractéristiques en entrée (Input feature map)* 40
Figure III.8. *Exemple de filtre convolutif* 40
Figure III.9. *Exemple de max-pooling réalisé sur une carte de caractéristiques 4x4 avec un filtre 2x2 et un pas de 2* 43
Figure III.10. *Schéma d'un réseau de neurones récurrents à une unité reliant l'entrée et la sortie du réseau* 45
Figure III.11. *Effet de l'application répétée d'une fonction sigmoïde* 46
Figure III.12. *Schéma d'un réseau LSTM* 48
Figure III.13. *Structure schématique d'un auto-encodeur avec 3 couches cachées entièrement connectées* 48
Figure III.14. *Auto-Encodeurs (1) et Auto-Encodeurs Variationnels (2)* 49
Figure III.15. *Matrice de confusion* 50
Figure III.16. *Interprétation graphique de la mesure ROC-AUC* 50
Figure IV.1. *Schéma de la méthode de conception* 55
Figure IV.2. *Auto-encodeur avec plusieurs couches cachées* 56
Figure IV.3. *Illustration de l'encodage avec trois couches cachées* 57
Figure IV.4. *Illustration de décodage du réseau précédent* 58
Figure IV.5. *Illustration du réseau convolutionnel (CNN 1D) adopté* 60
Figure V.1. *Graphes de la fonction loss du SAE avec les différents fichiers de tests* 70
Figure V.2. *Graphes de la fonction loss et accuracy et matrices de confusion combinées du CNN avec les différents fichiers de tests* 81
Figure V.3. *Distribution des données du jeudi* 82
Figure V.4. *Réponse du modèle en fonction du nombre des cartes de filtres* 83
Figure V.5. *Réponse du modèle en fonction de la taille du kernel* 84

LISTE DES TABLEAUX

Tableau V.1. *Tableau des bibliothèques python utilisées* 63
Tableau V.2. *Tableau des caractéristiques physiques de la base CICIDS2017* 64
Tableau V.3. *Tableau des caractéristiques de l'auto-encodeur empilé SAE* 66
Tableau V.4. *Tableau des résultats de la réduction de la dimensionnalité* 67
Tableau V.5. *Tableau des caractéristiques du réseau de neurones CNN 1D* 71
Tableau V.6. *Tableau des résultats de la réduction de la classification* 72
Tableau V.7. *Comparaison de l'accuracy/loss de la prédiction de la classification avec les études antérieures utilisées avec CICIDS2017* 84
Tableau V.8. *Comparaison de la précision de la prédiction de la classification avec les études antérieures utilisées avec CICIDS2017* 85

LISTE DES ACRONYMES

AE: Auto Encoder
ANN: Adversarial Neural Network
CD: Contrastive Divergence
CM: Confusion Matrix
CNN: Convolutional Neural Network
CPU: Central Processor Unit
DAE: Deep Auto-Encoder
DBN: Deep Belief Network
DDoS: Distributed Deny Of Service
DL: Deep Learning
DNN: Deep Neural Network
DoS: Deny Of Service
FN: False Negative
FP: False Positive
FNR: False Negative Rate
FPR: False Positive Rate
GHz: Giga Hertz
HIDS: Host Intrusion Detection System
IDMEF: Intrusion Detection Message Exchange Format
IDS: Intrusion Detection System
Inf: Infinity
LSTM: Long Short-Term Memory
ML: Machine Learning
MLP: Multi-Layer Perceptrons
MSE: Mean Squared Error
NaN: Not a Number
NIDS: Network Intrusion Detection System
NLP: Natural Language Processing
PC: Personal Computer
PDF: Probability Density Function
RAM: Random Acces Memory
ReLU: Rectified Linear Unit
RNN: Recurrent Neural Network
ROC-AUC: Receiver Operating Characteristic-Area Under Curve
SAE: Stacked Auto Encoder
SNMP: Simple Network Management Protocol
SMS: Short Message System
TN: True Negative
TP: True Positive

TPU: Tensor Processing Unit
TNR: True Negative Rate
TPR: True Positive Rate
VAE: Variational Auto Encoder

INTRODUCTION GENERALE

“Of all men’s miseries, the bitterest is this :
to know so much and have control over nothing.”

Herodote

La sécurité des systèmes informatiques est un problème sensible et préoccupant. Le progrès spectaculaire des technologies de l'information et de communication offre actuellement des facilités incontournables en matière de transfert de fichiers, messageries et bien d'autres formes d'échange d'informations. Le développement de l'informatisation des échanges s'est accompagné malheureusement du développement d'activités malveillantes dont les motivations sont aussi nombreuses que dangereuses et évoluent dans le temps.

Profitant de la connectivité croissante des systèmes informatiques, à l'internet notamment, les possibilités d'attaques à distance sont plus grandes et plus menaçantes. La connectivité croissante des systèmes informatiques aux différents réseaux (filaires ou sans fils) et les vulnérabilités et failles continuellement découvertes dans les systèmes informatiques (systèmes d'exploitation, applications, protocoles de communication, etc.), augmentent les risques d'attaques à distance.

Afin de détecter toute tentative de violation des mécanismes de la sécurité, une surveillance permanente ou régulière des systèmes peut être mise en place : ce sont les Systèmes de Détection d'Intrusions (IDS).

La détection d'intrusions consiste à scruter le trafic réseau, collecter tous les événements, les analyser et générer des alarmes en cas d'identification de tentatives malveillantes. Ces systèmes sont devenus très largement déployés dans les systèmes informatiques et ils ont gagné une place importante dans la conception de la stratégie de sécurité. Ils sont généralement utilisés pour surveiller l'accès et le flux d'information, dans le but de déterminer tout comportement malicieux, que ce soit de l'intérieur ou de l'extérieur de système d'informations, et rendre cette information disponible aux administrateurs de la sécurité. En option, les systèmes de détection d'intrusions peuvent réagir contre ces comportements malicieux et prendre des contre-mesures.

Il existe actuellement deux principales approches mises en œuvre pour élaborer des systèmes de détection d'intrusions : l'approche par scénario et l'approche comportementale. La première se base sur des signatures d'attaques déjà connues et qui peuvent être fournies par l'expert du domaine ou extraites à l'aide de techniques inductives à partir de données intrusives. L'inconvénient majeur de cette approche est son incapacité à détecter les nouvelles attaques (dont les signatures sont encore inconnues). L'approche comportementale suppose que l'activité normale est différente de l'activité intrusive. Il suffit alors d'élaborer un profil pour l'activité normale et un mécanisme permettant de comparer l'activité courante au profil établi pour détecter des écarts sensibles qui seront considérés des éventuelles intrusions.

Pratiquement, toutes les méthodes de réduction de la dimensionnalité et de classification (y-compris celles de l'apprentissage profond) ont été développées et testées pour la détection d'intrusion, mais avec des bases de données anciennes (KDD99, DARPA ou NSL-KDD, etc...). Bien que les résultats obtenus sont très satisfaisants, mais ces bases présentent quelques lacunes car elles sont nettement anciennes (fin des années 90), leurs nombres d'instances est maigres (quelques dizaines de milliers) et leurs attaques sont pratiquement très anciennes et dépassées.

Dans ce mémoire, nous proposons d'implémenter une méthode de détection d'intrusions basée sur les signatures d'attaques en utilisant l'approche apprentissage profond qui a donné des résultats dans plusieurs autres secteurs. Le système proposé sera élaboré et testé à l'aide de la base de données *CICIDS2017* (publiée en 2018), qui est une base de connexions TCP/IP, et qui comporte deux types de données, les données d'apprentissage qui sont utilisées pour l'élaboration des modèles de détection d'intrusions, et les données de test qui sert à évaluer les performances des systèmes élaborés. Notre contribution est de proposer et implémenter un système de détection d'intrusions basé sur l'apprentissage profond, puis nous donnons des résultats expérimentaux du système élaboré.

Notre mémoire est organisé comme suit :

La première partie de notre travail est consacré à la présentation des différents aspects de la sécurité informatique ainsi que les concepts relatifs aux réseaux d'une manière générale.

La deuxième partie comprend une description bien détaillée des systèmes de détection d'intrusions: leurs différents types, leurs principes de fonctionnement leurs avantages et inconvénients.

La troisième partie est réservée à la présentation des différents types d'apprentissage automatique et spécialement l'apprentissage profond: principe, fonctionnement, apprentissage, test et évaluation.

La dernière partie consiste à détailler la méthode d'apprentissage élaborée, les modules et dépendances à prendre en compte dans l'application de cette approche pour la détection d'intrusions, ainsi que tous les résultats obtenus.

Chapitre I : **SECURITE INFORMATIQUE**

> ***There are only two types of companies :***
> ***those that have been hacked,***
> ***and those that will be.***
>
> **Robert Mueller**

I.1. Introduction

La sécurité informatique est l'ensemble des moyens matériels et logiciels mis en œuvre pour minimiser la vulnérabilité d'un système contre des menaces accidentelles ou intentionnelles [9] (externes ou internes), permettant ainsi au système de fonctionner normalement et d'assurer la disponibilité des services attendus. Elle consiste, aussi, à s'assurer que celui qui modifie ou consulte les données du système en a l'autorisation et qu'il peut le faire car le service sollicité est disponible.

I.2. Objectifs de la sécurité informatique

Sécuriser les données, c'est garantir : [9]

- **L'authentification :** Le but de l'authentification est de garantir l'origine:
  - **D'une information :** Prouver qu'une information provient de la source annoncée (auteur, émetteur).
  - **D'une personne (ou machine, groupe ou organisation):** Prouver que l'identité est bien celle annoncée.
- **L'intégrité :** L'intégrité est d'assurer que les données n'ont pas été modifiées et empêcher toute modification (intentionnelle ou accidentelle) non explicitement requise par une entité habilitée. Cela permet, par exemple, au récepteur d'un message d'être raisonnablement assuré que le message reçu est le même que le message envoyé. Donc, l'intégrité des données est la propriété qui assure qu'une information n'est modifiée que dans des conditions prédéfinies (selon des contraintes précises).
- **La confidentialité :** La confidentialité est de garder secret le contenu de l'information et empêcher (ou prévenir) sa divulgation à des entités (sites, organisation, personnes, etc.) non habilitées à le connaître. Seuls les destinataires prédéterminés doivent être capables de lire le contenu du message.
- **La non-répudiation :** pour éviter la contestation par l'émetteur de l'envoi de données, la non répudiation est une propriété qui assure que l'auteur d'un acte ne peut ensuite nier l'avoir effectué (signature de l'acte) et que le récepteur ne peut ultérieurement dénier avoir reçu un message (exemple exécution d'un ordre boursier, d'une commande...).

Définir une politique de sécurité informatique revient à élaborer l'organisation et les mécanismes à mettre en place et l'ensemble des mesures à prendre en vue de :

- Réduire dans la mesure du possible l'utilisation frauduleuse, l'altération et l'accès non autorisé au système et ses ressources (machines, réseau, etc.).
- Détecter aussi rapidement que possible toute activité, malveillante ou accidentelle, pouvant porter atteinte à la confidentialité, intégrité et disponibilités des ressources et services.
- Prendre aussi efficacement que possible des contre-mesures afin de limiter les conséquences et poursuivre éventuellement l'auteur du forfait.

I.3. Protection des systèmes informatiques

Pour atteindre les objectifs de la sécurité, une stratégie de sécurité doit minutieusement évaluer ce qui est à protéger dans le système d'informations, les risques potentiels, les vulnérabilités du système à sécuriser, pour dégager de cette analyse les mécanismes et moyens nécessaires, tenant compte bien entendu d'autres considérations telles que le coût de déploiement et la complexité de cette stratégie. Entre autres mécanismes utilisés pour la sécurité des systèmes d'informations, ci-après les plus répandus :

- **Authentification et contrôles d'accès:** il s'agit en premier lieu de la sécurité physique des équipements et installations. L'authentification est un mécanisme permettant de prouver l'identité d'un utilisateur (mots de passe, cartes à puce, méthodes biométriques, etc.) et de lui accorder uniquement les privilèges nécessaires pour l'accomplissement de ses tâches.
- **Scanners de vulnérabilités:** Les scanners de vulnérabilités automatisent la découverte des failles de sécurité. Ils sont utilisés par les attaquants pour localiser les faiblesses du réseau cible. De plus les administrateurs peuvent en tirer profit pour corriger les vulnérabilités de leurs systèmes informatiques. Cependant, malgré le grand nombre de vulnérabilités détectées, les scanners d'aujourd'hui sont inaptes à déterminer toutes les faiblesses possibles. De plus, la mise à jour de ces produits ne suit pas le rythme de la découverte des nouvelles vulnérabilités.
- **La cryptographie:** les informations sensibles et confidentielles sont souvent cryptées pour empêcher leur lecture même si elles étaient dérobées ou accédées frauduleusement.

La cryptographie garantit la confidentialité, l'intégrité, la non répudiation et l'authenticité des données mais elle ne constitue pas une solution unique et suffisante de sécurité. Effectivement, diverses implémentations des protocoles de sécurité se sont révélées vulnérables. De plus la sécurité peut être rompue via plusieurs types d'attaques (par exemple : l'homme du milieu (MITM), les courts et les simples mots de passes utilisés sont facilement cassables, le risque de vol des clés privées).

- **Les pare-feu (*Firewall*):** un pare-feu (ou coupe-feu) est un dispositif matériel et logiciel d'interfaçage entre le réseau à sécuriser et un autre réseau externe, jugé moins sûr, et potentiellement source d'attaques. Un pare-feu assure d'abord une fonction de filtrage des flux entrants et sortants puisqu'il est un passage obligé pour tout échange. Le risque d'attaques distantes est alors réduit puisque le pare-feu ne laisse passer que le trafic d'une certaine plage d'adresses IP et relatif à certains ports et services. Malgré leurs grands intérêts, les pare feux présentent quelques lacunes. En effet, un attaquant peut exploiter les ports laissés ouverts pour pénétrer au réseau local. Les scripts constituent aussi des sources d'intrusion que les pare feux échouent à détecter. Ainsi l'opération supplémentaire d'encapsulation/décapsulation des données permet à l'attaquant de contourner le pare feu.
- **Protection anti-virus:** les logiciels anti-virus sont largement utilisés pour les stations de travail et ordinateurs personnels. Ils détectent et protègent contre les virus pouvant se propager à travers des fichiers, courrier électronique, etc.
- **Audit de sécurité et détection d'intrusions :** la plupart des systèmes d'exploitations et applications sont dotés de mécanismes d'audit permettant d'enregistrer tout ou une partie des événements (exécutions de commandes, utilisation de ressources, etc.) ayant lieu sur le système. Ces données peuvent faire l'objet d'une analyse en temps réel ou en différé pour détecter des activités malveillantes, suspectes ou toute activité pouvant violer la politique de sécurité. Cependant, ces données contiennent beaucoup d'informations normales et anormales. La taille énorme des fichiers contenant ces données pose souvent des problèmes de stockage et d'exploration du contenu. Les administrateurs fournissent aussi l'effort pour localiser les activités anormales, comprendre les objectifs des attaquants et déterminer les vulnérabilités exploitées du système.

I.4. Taxonomie des cyber-intrusions

Dans cette section nous résumons les différentes cyber-intrusions existantes par la figure suivante, en présentant une taxonomie basée sur le type, la cible de l'attaque ainsi que la méthode utilisée.

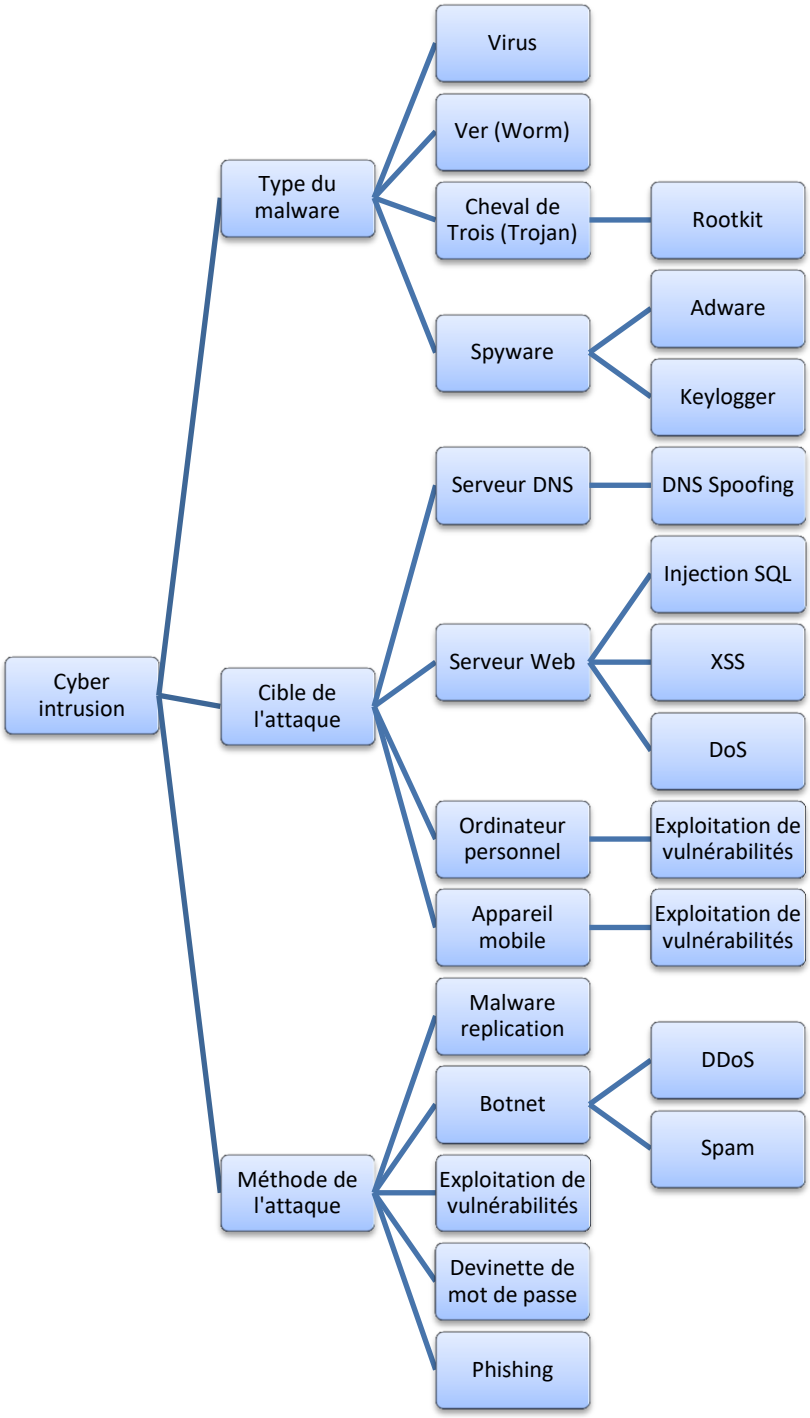


Figure I.1. *Taxonomie des cyber-intrusions [7]*

I.5. Quelques attaques courantes

Il existe plusieurs attaques, pour cela nous pouvons citer:

I.5.1. Attaque par balayage ICMP

Elle consiste à ce que le client envoie un paquet avec le protocole ICMP qui utilise la fonction ***request*** connue sous le nom ***ping***, vers le serveur qui lui répond avec un paquet ICMP ***echo-reply***, comme l'illustre la figure II.2.

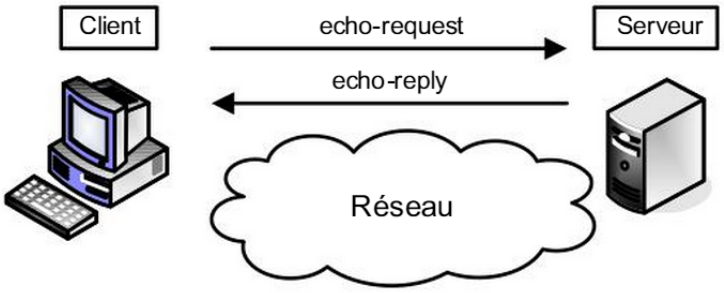


Figure I.2. *Fonctionnement de la commande* ***ping****.*

Il existe deux méthodes pour cartographier le réseau par cette technique :

- En balayant (*scanning*) le réseau et en interrogeant chaque adresse IP possible, ce qui n'est pas très discret.
- En visant une seule fois l'adresse de broadcaste du réseau, ce qui fait répondre toutes les machines présentes. Une seule demande permet ainsi d'engendrer l'envoi de toutes les réponses.

Cependant, du fait de l'accroissement constant de l'insécurité, nombre d'administrateurs de pare-feu ont pris l'initiative de ne pas laisser passer les réponses à de telles demandes. [14]

I.5.2. Attaque par balayage TCP

C'est en partant du principe que le flux réseau toujours accessible au pirate est celui qui est destiné à être accessible au public que la technique du balayage TCP a été inventée. Similaire au balayage ICMP, sa spécificité est de s'appuyer sur le protocole TCP. Le client envoie un paquet SYN vers un port réseau particulier de l'adresse IP du serveur. Si le port est en écoute,

un paquet SYN/ACK est reçu en retour. Sinon, la réception d'un paquet RST signifie qu'il n'y a pas de service en écoute sur le port. Le client envoie en réponse un paquet RST pour terminer la connexion, comme l'illustre la figure II.3. [14]

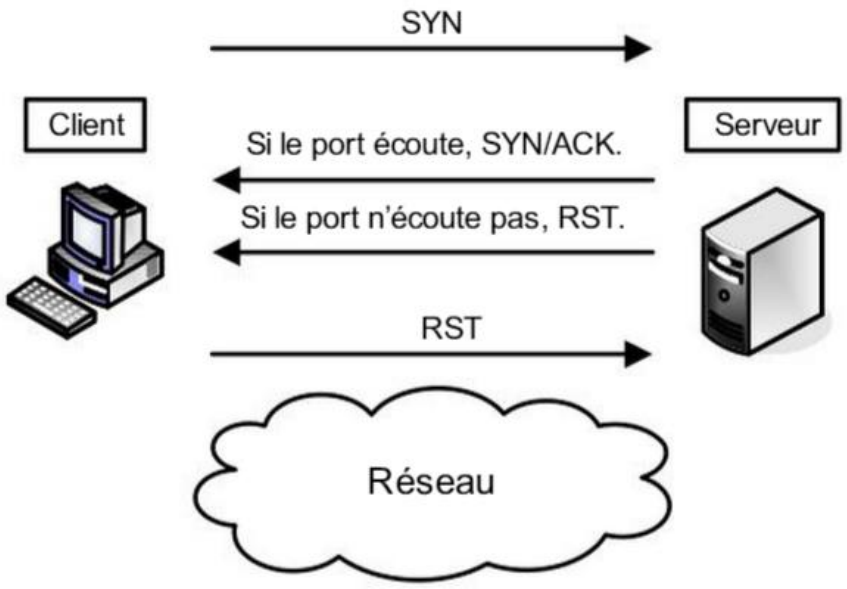


Figure I.3. *Attaque par balayage TCP.*

Si aucune réponse n'est reçue en retour, c'est qu'il existe un équipement filtrant entre le serveur et le client ou qu'il n'y a aucune machine derrière l'adresse IP visée. Cette technique est cependant si peu discrète, que des variantes ont été élaborées pour améliorer le balayage en jouant sur le principe de fonctionnement de la pile TCP/IP. [14]

I.5.3. Attaque par balayage semi-ouvert TCP

Les variantes à cette technique du balayage TCP reposent sur le non-respect de la définition du protocole TCP/IP. Nous venons de voir qu'il existait une séquence lors de l'établissement d'une session TCP. Lorsque cette séquence n'est pas respectée, le serveur TCP se comporte différemment, ainsi que les équipements filtrants présents sur le chemin. La variante dite de balayage semi-ouvert consiste en un balayage dans lequel le client envoie son paquet SYN et reçoit les paquets prévus en retour, comme l'illustre la figure II.3. Contrairement au balayage TCP normal, le client n'envoie pas de paquet RST pour rompre la session. Il note simplement la réponse et passe au port suivant. Par ce procédé, la session TCP

n'est pas ouverte, puisque le *handshake* ne s'est pas terminé, et le serveur ne trace pas cet échange de données. [11]

I.5.4. IP spoofing

Le pirate commence par choisir le système qu'il veut attaquer pour qu'il se fasse passer pour un autre système en falsifiant son adresse IP pour obtenir le maximum de détails sur le système cible, et il détermine les systèmes ou adresses IP autorisés à se connecter au système cible. Le pirate ensuite attaque le serveur cible en utilisant l'adresse IP falsifiée.

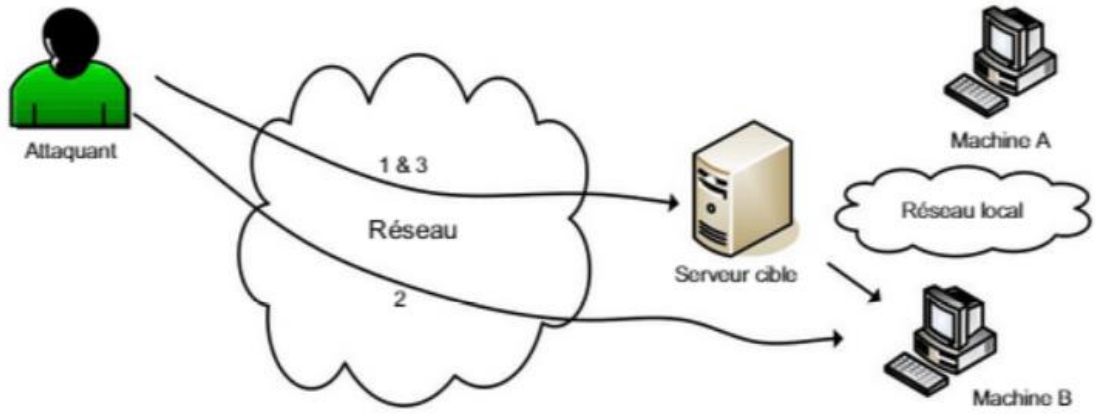


Figure I.4. *Attaque IP spoofing.*

I.5.5. Virus

C'est un petit programme qui a la faculté de se reproduire automatiquement. Il va recopier son propre code tel quel, ou en le modifiant, dans des éléments qui sont déjà dans l'ordinateur. Le plus souvent son but est de nuire. [12]

I.5.6. Déni de service (DoS) / Déni de service distribué (DDoS)

Ces attaques ne visent pas l'accès aux informations sur une machine à distance mais de paralyser un service ou un réseau complet, et l'utilisateur ne peut plus accéder au ressources, Les deux exemples principaux, sont le *ping-flood* ou l'envoi massif de courrier électroniques pour saturer une boîte aux lettres (*mailbombing*). [12]

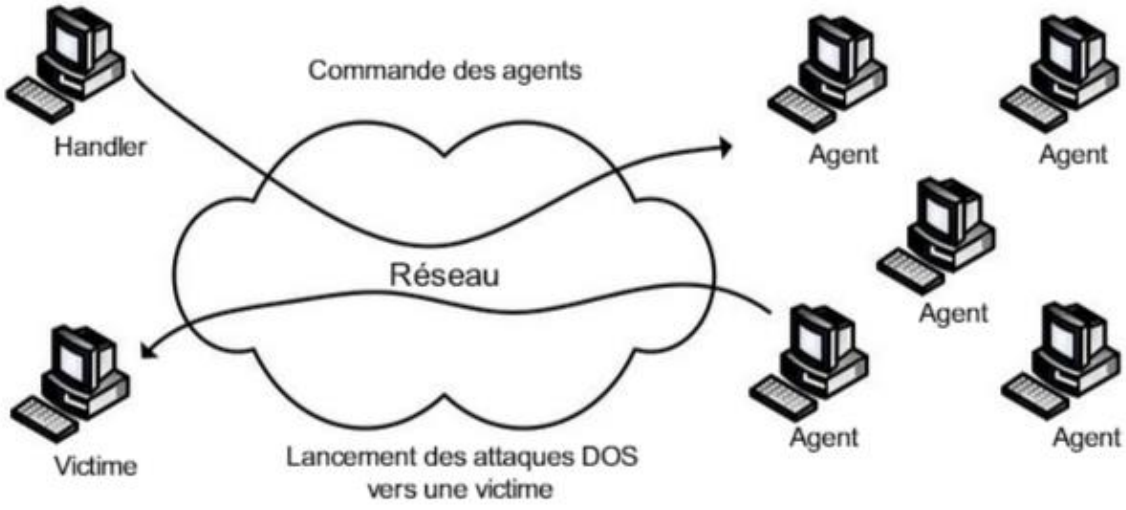


Figure I.5. *Attaque par déni de service distribué (DDoS).*

I.6. Nécessité de la détection d'intrusions

Des attaques exploitant les failles de ces systèmes et contournant leurs mécanismes de sécurité sont toujours possibles. Il n'est donc pas toujours possible d'agir préventivement, c'est-à-dire de définir une politique de sécurité en termes de confidentialité, d'intégrité et de disponibilité des données et ressources du système à protéger [3]. Afin de détecter toute tentative de violation de la sécurité, une surveillance permanente ou régulière des systèmes peut être mise en place par des Systèmes de Détection d'Intrusions (IDS). L'importance particulière qu'il convient selon [16] d'apporter à la détection d'intrusions tient à trois raisons:

- De nombreux systèmes existants sont vulnérables aux intrusions externes (attaques), et internes (abus de la part d'utilisateurs habilités à se servir du système).
- Les systèmes existants ne sont pas toujours remplaçables par des systèmes plus sécurisés, soit pour des raisons de coût (développer un système hautement sécurisé est une tâche difficile et couteuse), soit car l'apport de sécurité se fait au détriment du confort d'utilisation du système (voire de sa philosophie, c'est notamment le cas du système UNIX).
- Même les systèmes les plus sécurisés sont vulnérables aux abus de la part des utilisateurs habilités et aux canaux cachés. Il est important de noter que ces deux risques, ne violant aucune des règles de la politique de sécurité, ne sont détectables que par audit.

I.7. Conclusion

Les techniques de protection contre les attaques permettent de réaliser les bases de la sécurité : confidentialité, intégrité, authentification, disponibilité. Mais malgré toutes ces techniques utilisées pour empêcher les attaques perpétrées contre les systèmes d'information, un système n'est jamais totalement sûr. Dans le deuxième chapitre, nous sommes intéressés par la présentation du système de détection d'intrusion.

Chapitre II : SYSTEME DE DETECTION D'INTRUSIONS

« To know your Enemy,
you must become your Enemy. »

Sun Tzu

II.1. Introduction

L'augmentation du nombre de réseaux locaux dans le monde a conduit à une évolution continue et importante des données Internet, en même temps, la disponibilité de quantités massives de données a favorisé le développement des technologies d'information, ce qui nécessite une attention particulière du point de vue sécurité. Par conséquence, cette évolution a entrainé une augmentation de la vulnérabilité des systèmes à des menaces différentes [2].

Selon un consortium mondial de télécommunication METIS (*Mobile and Wireless Communications Enablers for the Twenty-Twenty Information Society*) [18], une prolifération de la 5G et des réseaux Wi-Fi devrait se produire au cours des prochaines décennies. Ils croient que le développement rapide de la société nécessitera une avalanche de volume de trafic mobile et sans fil, et les applications telles que l'apprentissage en ligne (*e-learning*), la banque en ligne (*e-banking*) et la santé en ligne (*e-health*) se répandraient et deviendraient plus mobiles. D'ici 2020, le trafic des réseaux sans fil devrait représenter les deux tiers du trafic Internet total, 66% du trafic IP devant être généré par le Wi-Fi et les appareils cellulaires uniquement [12].

Chaque intrusion peut avoir des conséquences désastreuses, par exemple, destruction, altération ou corruption des données personnelles et accès illégal à des données confidentielles. De ce fait la sécurité des réseaux informatiques est devenue un outil prometteur pour fournir des canaux sécurisés aux données. Parmi ces outils prometteurs qui ont pu imposer leur présence dans le domaine de la sécurité des réseaux informatiques, nous trouvons les systèmes de détection d'intrusion (*IDS*) qui permettent la détection des attaques. Les infrastructures modernes de la cyber-sécurité emploient ces IDS comme étant un composant essentiel pour la prévention et la protection contre différentes menaces.

II.2. Définition

Un système de détection d'intrusion (ou IDS: *Intrusion Detection System*) est un mécanisme destiné à repérer des activités anormales ou suspectes sur la cible analysée (un réseau ou un hôte). Ce système de surveillance permanent est incorporé dans la topologie du réseau, ou appliqué à un ordinateur simple, afin de surveiller les données qui transitent sur un système ou un réseau ciblé. Il permet aussi la collecte des données d'audit, l'analyse des informations collectées pour déterminer des activités inhabituelles ou suspectes et d'établir des plans de réponse. L'IDS doit modéliser tout type d'attaque ou évènement anormal

pouvant affecter le réseau (basé sur la signature) ou créer un modèle général décrivant le trafic normal sur ce réseau (basé sur le comportement).

L'IDS est composé d'un ensemble de capteurs à usage unique ou d'ordinateurs hôtes positionnés à des endroits différents d'un réseau ou d'un ensemble de réseaux interconnectés pour surveiller le trafic transmis. Ces unités examinent les paquets de trafic réseau en temps réel, effectuent une analyse locale pour un paquet capturé et prennent enfin des mesures telles que la notification de l'administrateur [11].

Par définition, un IDS n'a pas de vocation préventive ou réactive dans la mesure où il n'empêche pas une intrusion de se produire. Il se contente plutôt d'analyser certaines informations en vue de détecter d'éventuelles activités malveillantes qu'il aura à notifier dans les plus brefs délais au responsable de la sécurité du système. C'est pour cette raison que la majorité des IDS opèrent en temps réel. Toutefois, il y'a des IDS qui réagissent suite à la détection d'une intrusion en mettant fin par exemple à une connexion suspecte [9].

II.3. Architecture fonctionnelle des IDS

Nous décrivons dans cette section les trois composants qui constituent classiquement un système de détection d'intrusion. La Figure ci-dessous illustre les interactions entre ces trois composants [4].

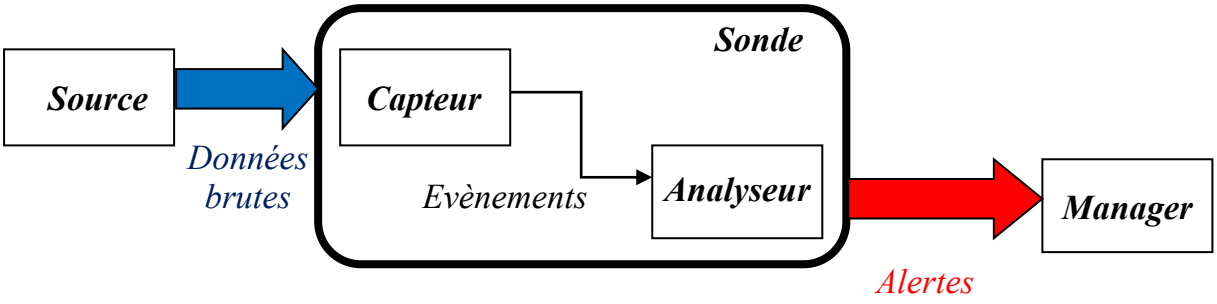


Figure II.1. *Schéma fonctionnel du Système de détection d'intrusion (IDS)*

II.3.1. Capteur

Le capteur observe l'activité du système par le biais d'une source de donnée et fournit à l'analyseur une séquence d'événements qui renseignent de l'évolution de l'état du système. Le capteur peut se contenter de transmettre directement ces données brutes, mais en général un prétraitement est effectué. Et pour cela nous distinguons trois types de capteurs en fonction

des sources de données utilisées pour observer l'activité du système : les capteurs système, les capteurs réseau et les capteurs applicatifs.

II.3.2. Analyseur

L'objectif de l'analyseur est de déterminer si le flux d'événements fourni par le capteur contient des éléments caractéristiques d'une activité malveillante.

II.3.3. Manager

Le manager collecte les alertes produites par le capteur, les met en forme et les présente à l'opérateur (ou la personne responsable de la sécurité). Eventuellement, le manager est chargé de la réaction à adopter qui peut être :

- Isolement de l'attaque, qui a pour but de limiter les effets de l'attaque.
- Suppression d'attaque, qui tente d'arrêter l'attaque.
- Recouvrement, qui est l'étape de restauration du système dans un état sain.
- Diagnostic, qui est la phase d'identification du problème.

II.4. Fonctions de l'IDS

Les fonctions principales d'un IDS peuvent être résumées dans les points suivants :

- Journaliser l'événement source d'information et vision des menaces courantes.
- Comparer les données collectées avec des données de référence qui correspondent à des opérations interdites ou autorisées.
- Avertir un système avec un message (Exemple: appel *SNMP* [1]).
- Avertir un humain avec un message (Courrier électronique, SMS, interface web,…).
- Amorcer certaines actions sur un réseau ou hôte (Exemple: mettre fin à une connexion réseau, ralentir le débit des connexions,…).
- Appliquer des mesures correctives en cas de détection d'une intrusion [11].

[1] ***Simple Network Management Protocol*** (abrégé **SNMP**), en français « protocole simple de gestion de réseau », est un protocole de communication qui permet aux administrateurs réseau de gérer les équipements du réseau, de superviser et de diagnostiquer des problèmes réseaux et matériels à distance. SNMP est utilisé pour administrer les équipements et/ou surveiller le comportement des équipements.

II.5. Types des IDS

Suivant l'emplacement de l'IDS dans l'architecture du réseau informatique à surveiller, nous distinguons trois types d'IDS :

II.5.1. IDS réseau

L'IDS réseau (*Network IDS* ou *NIDS*) est situé sur un réseau isolé et ne voit qu'une copie du trafic, c'est-à-dire des paquets qui circulent sur le réseau. Et en cas, de détection d'une menace, le NIDS peut lever des alertes et ordonner les actions pour le blocage d'un flux.

En termes d'architecture, le NIDS est situé sur un réseau isolé et analyse une copie du trafic du réseau à surveiller, entre ses points d'entrées et les terminaux du réseau. A noter qu'il est entièrement passif et n'est pas capable de dialoguer avec le réseau surveillé.

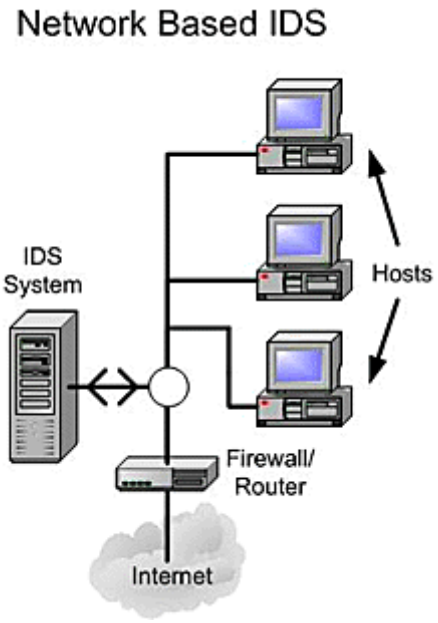


Figure II.2. *Exemple d'une architecture d'un NIDS*

Il est fréquent de trouver plusieurs IDS sur les différentes parties du réseau. Nous trouvons souvent une architecture composée d'une sonde placée à l'extérieur du réseau afin d'étudier les tentatives d'attaques et d'une sonde en interne pour analyser les requêtes ayant traversé le pare-feu [1].

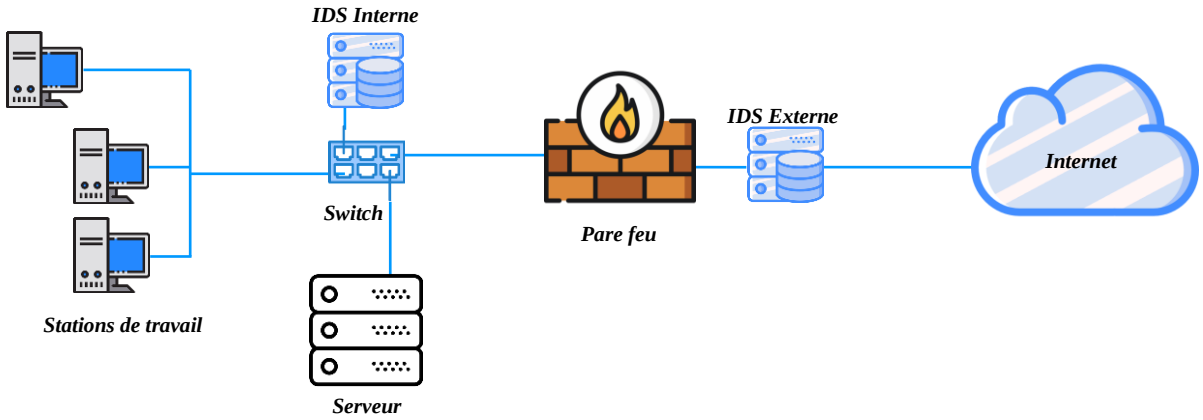


Figure II.3. *Modèle de système de détection d'intrusion*

II.5.2. IDS hôte

Il y a ensuite les IDS hôte (*Host IDS* ou *HIDS*) ou IDS système. Les HIDS (*Host Intrusion Detection System*), surveillent l'état de la sécurité des hôtes selon différents critères :

- Activité de la machine (comme par exemple le nombre et listes de processus, le nombre d'utilisateurs, ressources consommées, etc.).
- Le second critère de surveillance est l'Activité de l'utilisateur sur la machine : horaires et durée des connexions, commandes utilisées, programmes activés, etc.
- Et évidemment le HIDS analyse toute activité potentielle liée à l'activité d'un ver, d'un virus ou cheval de Troie.

En termes d'architecture, les HIDS sont, généralement, placés sur des machines sensibles, susceptibles de subir des attaques et possédantes des données sensibles pour l'entreprise. Les serveurs web et applicatifs, peuvent notamment être protèges par un HIDS. Le HIDS master récupère les informations remontées par une machine sur laquelle un client HIDS est installé. Ensuite, le HIDS master va analyser ces informations sur le fonctionnement et l'état des machines afin de détecter les menaces.

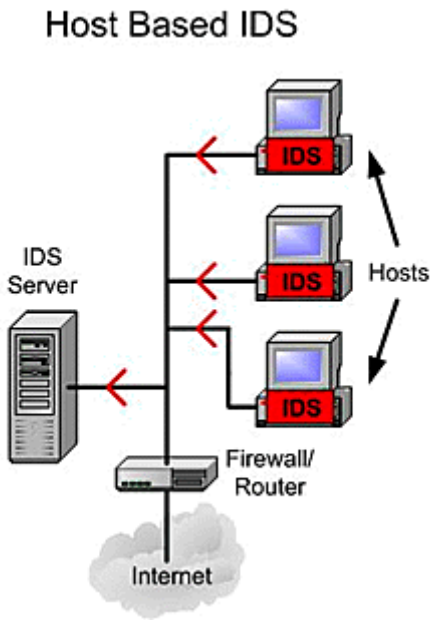


Figure II.4. *Exemple d'une architecture d'un HIDS*

II.5.3. IDS hybride

Les IDS hybrides sont, généralement, utilisés dans un environnement décentralisé, ils permettent de réunir les informations de diverses sondes placées sur le réseau, et agissent comme NIDS et/ou HIDS suivant leurs emplacements. Leur appellation « hybride » provient du fait qu'ils sont capables de réunir aussi bien des informations provenant d'un système HIDS qu'un NIDS. Toutes ces sondes HIDS et NIDS remontent alors les alertes à une machine qui va centraliser le tout, et agréger/lier les informations d'origines multiples. Ainsi, nous comprenons que les IDS hybrides sont basés sur une architecture distribuée, où chaque composant unifie son format d'envoi (par exemple *IDMEF*[2]). Cela permet de communiquer et d'extraire des alertes plus pertinentes [1].

II.6. Classification des IDS

La classification adoptée selon différents critères qui ne sont pas forcément mutuellement exclusifs n'est pas hiérarchique (par exemple, un IDS peut mettre en œuvre deux analyseurs utilisant des méthodes différentes), elle présente tour à tour et au même niveau les catégories caractérisant chaque IDS, et utilise les critères suivants (figure III.5) [16]:

[2] ***Intrusion Detection Message Exchange Format*** (abrégé **IDMEF**), en français « format d'échange de message de détection d'intrusion » est une norme qui spécifie le format des messages d'alertes échangés entre les différents IDS.

- La source des données à analyser.
- La méthode de détection utilisée.
- Le lieu de l'analyse des données.
- La fréquence de l'analyse.
- Le comportement en cas de détection d'une attaque.

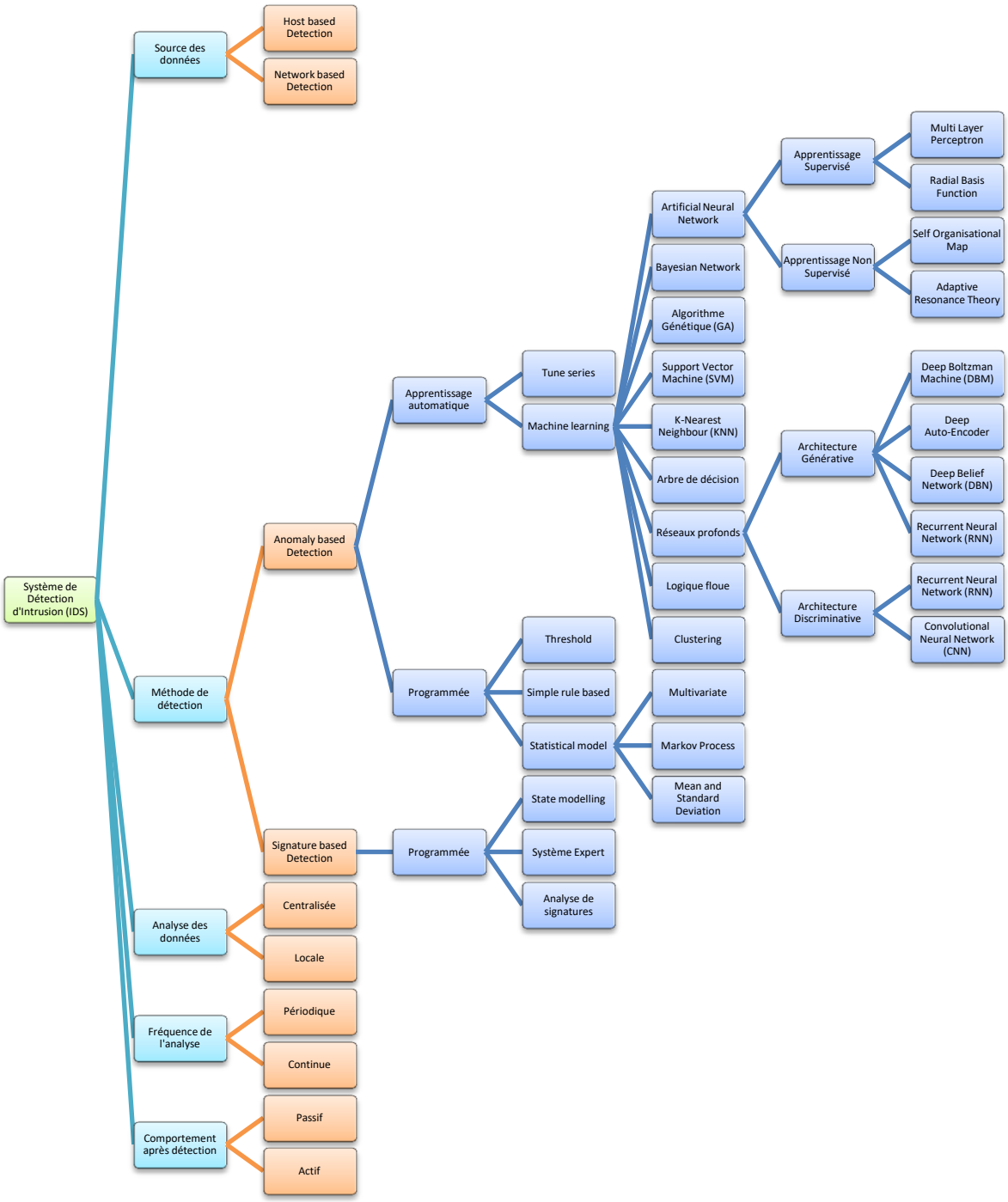


Figure II.5. *Classification des systèmes de détection d'intrusion*

II.6.1. Source des données à analyser

Parmi les caractéristiques essentielles des systèmes de détection d'intrusions, les sources de données à analyser constituent la matière première du processus de détection. Ces données proviennent soit de logs (journaux) générés par le système d'exploitation, soit de logs des applications, soit d'informations provenant du réseau, soit encore d'alertes générées par d'autres IDS. [16]

II.6.2. Méthode de détection utilisée.

L'IDS implique deux techniques de détection principales, à savoir, la détection basée sur une mauvaise utilisation (*misuse-based detection* ou *knowledge-based detection*) et l'approche comportementale où la détection se base sur une anomalie (*anomaly-based detection*). Les systèmes de détection de mauvaise utilisation définissent le comportement d'un intrus et analysent les données d'attaque prédéterminées pour faire correspondre les signatures. Les systèmes de détection d'anomalies définissent uniquement le comportement normal (typique) ou attendu de l'entité à surveiller (utilisateur, service, application. . .) et signalent tout écart par rapport à ce comportement.

Cependant, ces deux systèmes de détection possèdent des faiblesses. Les systèmes de détection des utilisations abusives ne peuvent pas détecter les nouvelles formes d'attaque car leurs signatures ne sont pas encore disponibles pour la correspondance des modèles. Les systèmes de détection d'anomalies produisent un taux de fausses alarmes élevé lorsqu'ils sont adoptés dans la plupart des approches existantes, car il est difficile de créer un comportement normal pour protéger les systèmes ciblés.

II.6.3. Lieu de l'analyse des données.

Les systèmes de détection d'intrusions peuvent être classés aussi en se basant sur la localisation réelle de l'analyse des données :

- **Analyse centralisée :** Certains IDS ont une architecture multi-capteurs (ou multisondes). Ils centralisent les événements (ou alertes) pour analyse au sein d'une seule machine. L'intérêt principal de cette architecture est de faciliter la corrélation entre événements puisque nous disposons alors d'une vision globale. Par contre, la charge des calculs (effectués sur le système central) ainsi que la charge réseau (due à la collecte des événements ou des alertes) peuvent être lourdes et risquent de surcharger le trafic réseau.

- **Analyse locale :** Si l'analyse des événements est effectuée au plus près de la source de données (généralement en local sur chaque machine disposant d'un capteur), nous minimisons le trafic réseau et chaque analyseur séparé dispose de la même puissance de calcul. En contrepartie, il est impossible de croiser des événements qui sont traités séparément et nous risquons de passer à côté de certaines attaques distribuées.

II.6.4. Fréquence de l'analyse.

Une autre caractéristique des systèmes de détection d'intrusions est leur fréquence d'utilisation, dans ce cas nous distinguons deux (2) types :

- **IDS online (continue)**: Ce sont des IDS qui font leur analyse des fichiers d'audit ou des paquets réseau de manière continue ou en permanence afin de détecter une attaque au moment de sa production, c'est une détection en temps réel. Ce type d'IDS consomme un taux élevé de ressources systèmes car il faut analyser à la volée tout ce qui se passe sur le système et ce qu'il le rend non préférable en cas de ressources précieuses telle que les serveurs de messagerie par exemple.
- **IDS offline (périodique)**: Ce type d'IDS fait l'analyse dans des durées périodiques afin de détecter des traces d'attaques au but de modéliser des signatures d'attaques pour la base du système, l'avantage de ce type est qu'il ne consomme pas beaucoup de ressources système. Cela peut être suffisant dans des contextes peu sensibles (nous ferons alors une analyse journalière, par exemple). L'inconvénient majeur de ce type est sa détection tardive des attaques ce qui risque de provoquer des dégâts dangereux.

II.6.5. Comportement en cas de détection d'une attaque.

Nous pouvons également faire une distinction entre les IDS en se basant sur le type de réaction lorsqu'une attaque est détectée :

- **Passive:** Généralement, la plupart des systèmes de détection d'intrusions n'apportent qu'une réponse passive à l'intrusion ; c'est à dire lorsqu'une attaque est détectée, ils génèrent une alarme et notifient l'administrateur système par e-mail, message dans une console, voire même par beeper ou SMS. C'est alors l'opérateur qui devra prendre les mesures qui s'imposent.
- **Active:** d'autres systèmes de détection d'intrusions peuvent, en plus de la notification à l'opérateur, prendre automatiquement des mesures pour stopper l'attaque en cours. Par

exemple, ils peuvent couper les connexions suspectes ou même, pour une attaque externe, reconfigurer le pare-feu pour qu'il refuse tout ce qui vient du site incriminé. Toutefois, il apparait que ce type de fonctionnalité automatique est potentiellement dangereux car il peut mener à des dénis de service provoqués par l'IDS. Un attaquant déterminé peut, par exemple, tromper l'IDS en usurpant des adresses du réseau local qui seront alors considérées comme la source de l'attaque par l'IDS. Il est préférable de proposer une réaction facultative à un opérateur humain (qui prend la décision finale). [16]

II.7. Efficacité des IDS

L'efficacité d'un système de détection d'intrusions est déterminée par les mesures suivantes : [10]

- **Exactitude :** Le système de détection d'intrusions n'est pas exact s'il considère les actions légitimes des utilisateurs comme atypiques ou intrusives (faux positif).
- **Performance :** Effectuer une détection en temps réel.
- **Tolérance aux pannes :** Un système de détection d'intrusions doit être résistant aux attaques.
- **Rapidité :** Un système de détection d'intrusions doit exécuter et propager son analyse d'une manière prompte pour permettre une réaction rapide dans le cas d'existence d'une attaque pour permettre à l'agent de sécurité de réagir.
- **Complétude :** La complétude est la capacité d'un système de détection d'intrusion de détecter toutes les attaques [30].

II.8. Limites et vulnérabilités des IDS

Malgré l'importance des IDS dans la sécurité des systèmes informatiques, et comme tout système informatique, les IDS ont des limites et vulnérabilités. Parmi ces limites, nous pouvons citer :

- **Pollution/surcharge :** Les IDS peuvent être pollués ou surchargés, par exemple par la génération d'un trafic très important (le plus difficile et lourd possible à analyser). Une quantité importante d'attaques peut également être envoyée afin de surcharger les alertes de l'IDS. Des conséquences possibles de cette surcharge peuvent être la saturation de ressources (disque, CPU, mémoire), la perte de paquets, le déni de service partiel ou total.

- **Consommation de ressources :** outre la taille des fichiers de logs (de l'ordre du Go), la détection d'intrusion est excessivement gourmande en ressources. En effet un système NIDS doit générer des journaux de comptes-rendus d'activité anormale ou douteuse sur le réseau.
- **Perte de paquets (limitation des performances) :** les vitesses de transmission sont parfois telles qu'elles dépassent largement la vitesse d'écriture des disques durs, ou même la vitesse de traitement des processeurs. Il n'est donc pas rare que des paquets ne soient pas traités par l'IDS, et que certains d'entre eux soient néanmoins reçus par la machine destinataire.
- **Vulnérabilité aux dénis de service :** un attaquant peut essayer de provoquer un déni de service au niveau du système de détection d'intrusion, ou pire au niveau du système d'exploitation de la machine supportant l'IDS. Une fois l'IDS désactivé (« hors service »), l'attaquant peut tenter tout ce qui lui convient.

Pour ce qui est des vulnérabilités des IDS, les attaquants ont développé plusieurs techniques afin de contourner l'IDS, ces techniques peuvent être répertoriées en trois (3) grandes catégories : [15]

1. **Insertion :** Un paquet peut être soigneusement construit de sorte qu'un NIDS l'accepte pour traitement, mais ignoré par l'hôte cible ; Dans cette méthode englobe plusieurs techniques, parmi celles-ci nous pouvons citer :
   - **Encodage URL (*URL Encoding*) :** C'est le nom donné à la technique de substitution des caractères d'une adresse URL par leurs équivalents hexadécimaux.
     `/cgi-bin/test.cgi` ⇨ `/cg%69-b%69n/t%65st.cg%69`
   - **Traversée Inversée (*Reverse Traversal*) :** Un autre stratagème consiste à essayer de confondre le NIDS en compliquant la demande avec des références de répertoire supplémentaires. La demande sera toujours résolue correctement sur l'hôte cible mais l'IDS éliminera le paquet.
     `/cgi-bin/test-cgi` ⇨ `/cgi-bin/redherring/../test.cgi`
   - **Répertoires auto-référencés (*Self Referencing Directories*) :** Une technique similaire pour inverser la traversée est celle des répertoires à auto-référencement. L'insertion de «./» dans n'importe quelle demande n'aura aucun effet, car cela signifie «répertoire courant».

     `/cgi-bin/test-cgi` ⇨ `/./cgi-bin/./test.cgi`
   - **Masquage de paramètres (*Parameter Hiding*) :** Une demande peut contenir des informations supplémentaires utilisées pour créer un contenu de page dynamique. Ces

informations supplémentaires sont appelées paramètres. Les paramètres sont généralement utilisés lors de demandes de recherche ou de sélections et prennent cette forme:

```
/anypage.php?attack=paramhiding&evasion=blackhat&success...
```

Les paramètres sont spécifiés après le «?», Et un NIDS pseudo-intelligent ignorera probablement les données après le ? pour améliorer les performances de traitement. Cependant, l'indicateur de paramètre peut être utilisé pour masquer éventuellement d'autres données pertinentes.

```
GET /index.htm%3fparam=/../cgi -bin/test.cgi
```

deviens

```
GET /index.htm?param=/../cgi -bin/test.cgi
```

- **URLs longues (*Long URLs*) :** Il est évident qu'une trame d'une longueur donnée ne peut en lire et analyser qu'une partie. Si un paquet est conçu avec un rembourrage aussi long, alors tout contenu malveillant en souffrira sans contrôle.
- **Barres obliques multiples (*Multiple Slashes*) :** Il est possible d'envoyer une demande à un serveur Web qui substitue des barres obliques uniques à plusieurs barres obliques et le serveur Web interprétera toujours correctement la demande. L'attaquant peut espérer que le NIDS ne parvienne pas à correspondre à la chaîne d'attaque.

  `/cgi-bin/test.cgi` ⇨ `//cgi-bin//test.cgi` ou `///cgi -bin///test.cgi`

2. **Elimination :** Cette technique consiste à pénétrer le NIDS et le rendre inutile en le saturant par les flux, ou l'empêcher d'effectuer une analyse ;
3. **Evasion :** Les paquets soigneusement construits peuvent être acceptés par le système d'extrémité, mais ignorés par un NIDS ;
   - **Balayage lent (*Slow Scans*) :** Le NIDS détecte l'activité de balayage du réseau en surveillant la fréquence du trafic à partir d'une adresse IP donnée. Si un outil d'analyse peut répartir artificiellement l'activité d'analyse sur une période prolongée, le NIDS peut ne pas détecter l'activité.
   - **Correspondance de méthode (*Method Matching*) :** Dans le HTTP RFC, il est tout à fait légitime d'envoyer une méthode alternative à GET qui était à l'origine la seule méthode disponible. Des méthodes alternatives sont utiles à un attaquant car elles permettent de détecter la présence sur un serveur Web de scripts ou de fichiers CGI utiles ayant des faiblesses ou des vulnérabilités connues.

     Les méthodes alternatives sont: HEAD, POST, PUT, DELETE, PATCH, PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK.

- **Fin de demande prématurée (*Premature Request Ending*) :** Une technique conçue pour tromper un NIDS pseudo Intelligent consiste à insérer une demande de fin avant la fin réelle de la demande et, plus important encore, avant les données malveillantes. Une demande authentique typique ressemblerait à ceci:

```
GET   /some.file HTTP/1.0 \r\n
Header: blah \r\n
Header: blah \r\n
Header: blah \r\n
Header: blah \r\n
\r\n
```

Un NIDS pseudo Intelligent n'analysera généralement pas les en-têtes d'une requête car il ne sert à rien. Par conséquent, l'IDS peut cesser de s'occuper du "HTTP / 1.0 \ r \ n". Cependant, l'exemple suivant montre le danger de cette méthode:

```
GET  /%20HTTP/1.0%0d%0aHeader:%2  0/../../cgi  -bin/test.cgi  HTTP/1.0
\r\n\r\n
```

deviens :

```
GET / HTTP/1.0 \r\nHeader: /../../cgi -bin/test.cgi HTTP/1.0 \r\n\r\n
```

- **Mauvaise mise en forme HTTP (*HTTP Mis-Formating*)** : Bien qu'il existe une structure clairement définie pour toute requête HTTP, de nombreux serveurs Web accepteront une requête qui ne correspond pas exactement à cette spécification. Une demande qui se conforme exactement au RFC prend la forme de:

```
Method <space> URI <space> HTTP/ Version CRLF CRLF
```

Certains serveurs Web autoriseront l'utilisation d'un séparateur alternatif, par exemple:

```
Method <tab> URI <tab> HTTP/ Version CRLF CRLF
```

- **Syntaxe de répertoire DOS (*DOS directory syntax*) :** Les serveurs Web basés sur DOS doivent traduire de manière transparente les barres obliques '\' des demandes en barres obliques '/'.

```
/pages/login/password.lst  ⇨  /pages\login\password.lst
```

- **Sensibilité à la casse (*Case sensitivity*) :** Une autre différence entre les systèmes DOS et Unix est leur interprétation du cas. Sous Unix, les fichiers suivants sont différents:

```
password
PASSWORD
PassWord
```

Un système basé sur DOS interpréterait chacune des chaînes ci-dessus comme le même fichier.

- **Fragmentation** : La fragmentation des paquets peut cacher quelques attaques afin de ne pas les détecter. Les NIDS n'implémentent pas le réassemblage de fragments en raison des charges de travail importantes engendrées par cette approche. Par conséquent, tout paquet contenant du contenu malveillant fragmenté passera inaperçu.
- **Épissage de session (*Session Splicing*) :** Le fractionnement de session diffère de la fragmentation décrite ci-dessus, car il ne s'agit que d'envoyer la charge HTTP des données en morceaux dans le seul but d'empêcher un NIDS d'analyse brute de détecter avec succès une correspondance de chaîne.
- **Traitement de la méthode NULL (*NULL Method Processing*) :** Cette technique repose sur le fait que les bibliothèques de chaînes C utilisent le caractère NULL pour indiquer la fin d'une chaîne. L'IDS analysera une requête et, en tant que tel, interprétera la requête de manière incorrecte, ignorant potentiellement le contenu malveillant.

```
GET%00 /cgi-bin/test.cgi HTTP/1.0
```

II.9. Conclusion

Ce chapitre nous a permis de constater que les IDS sont de plus en plus fiables, d'où le fait qu'ils soient souvent intégrés dans les solutions modernes de sécurité. Les avantages qu'ils présentent par rapport aux autres outils de sécurité les favorisent. Il nous a également permis de comprendre que ces derniers sont indispensables aux entreprises afin d'assurer leur sécurité informatique. Dans le chapitre qui suit nous présenterons les différentes techniques de l'apprentissage profond ou *Deep Learning*.

Chapitre III : **APPRENTISSAGE PROFOND**

"هذا الكون العظيم كتاب مطوي،
لم يقرأ العلماء إلا كلمات من صفحة غلافه"

III.1. Introduction

Les systèmes d'apprentissage automatique (*Machine Learning* ou *ML*) déployés sont capables d'apprendre les comportements voulus à partir de bases de données d'échantillons. En outre, de tels systèmes peuvent être recyclés régulièrement au fur et à mesure que de nouvelles données apparaissent. Les systèmes logiciels très sophistiqués, dopés par l'apprentissage automatique, sont capables de modifier leur comportement de manière radicale sans apporter de grands changements à leur code (juste à leurs données d'apprentissage).

L'apprentissage profond (*Deep Learning* ou *DL*) a révolutionné les industries technologiques. La traduction automatique moderne, les moteurs de recherche et les assistants informatiques sont tous alimentés par un apprentissage profond. Cette tendance ne va faire que se poursuivre, alors que le deep learning étend sa toile à la robotique, aux produits pharmaceutiques, à l'énergie et à tous les autres domaines de la technologie contemporaine [20]. Les modèles d'apprentissage profond tentent d'imiter au maximum les modes de traitement de l'information et de communication observés dans le système nerveux biologique, tels que le codage neuronal, qui tente de définir et de décrire les relations existant entre plusieurs stimuli et les réponses neuronales associées dans le cerveau. [3]

III.2. Définition

Nous définissons l'apprentissage profond (*DL*) comme une classe de techniques d'apprentissage automatique (*ML*), dans lesquelles les informations sont traitées en couches hiérarchiques pour comprendre les représentations et les caractéristiques des données à des niveaux de complexité croissants. L'apprentissage profond est appelé, également, apprentissage hiérarchique ou apprentissage profond structuré [3].

L'apprentissage des représentations de données pourrait se faire par le biais d'approches semi-supervisées, supervisées ou non supervisées. En pratique, tous les algorithmes d'apprentissage profond sont des réseaux de neurones (*Neural networks*), qui partagent certaines propriétés de base communes. Ils sont tous constitués de neurones interconnectés organisés en couches. Ce qui les différencie, c'est l'architecture du réseau (ou la manière dont les neurones sont organisés dans le réseau) et parfois la manière dont ils sont formés. [13]

III.3. Principe

L'apprentissage profond est constitué d'une variété de techniques issues du domaine de l'apprentissage automatique qui utilisent un déluge de nœuds non linéaires disposés en plusieurs couches qui extraient et convertissent des valeurs de variables d'entité à partir du vecteur d'entrée [4]. Les couches individuelles d'un tel système ont en entrée les sorties des couches précédentes, à l'exception de la couche initiale d'entrée qui reçoit les signaux ou les vecteurs d'entrée de l'environnement extérieur.

De plus, lors de la formation des systèmes, des techniques non supervisées ou supervisées pourraient être appliquées. Cela entraîne l'application possible de ces modèles à des tâches d'apprentissage supervisées telles que la classification et à des tâches non supervisées telles que l'analyse des modèles. Les modèles d'apprentissage approfondi reposent également sur l'extraction des entités de niveau supérieur des entités de niveau inférieur afin d'obtenir une représentation stratifiée des données d'entrée via une approche d'apprentissage non supervisée sur les différents niveaux des entités [4]. Un classement des notions et des théories est obtenu en apprenant différentes couches de représentations des données représentant différents niveaux d'absorption des données. Certains des cadres d'apprentissage profond de la littérature sont les réseaux de croyances profondes (*DBN*) [10], les réseaux d'auto-encodeurs profonds / empilés (*DAE* / *SAE*) et les réseaux de neurones convolutionnels (*CNN*). Ces cadres d'apprentissage profond ont été utilisés dans divers secteurs, tels que le traitement du langage naturel, la reconnaissance de la parole, la reconnaissance audio, la reconnaissance et la détection d'objets et la vision par ordinateur. L'apprentissage profond est une branche des algorithmes d'apprentissage automatique qui : [3]

- utilise plusieurs couches de nœuds de traitement non linéaires pour l'extraction et la transformation d'entités. Les couches successives utilisent les sorties des couches précédentes en entrée.
- apprends de manière non supervisée (analyse de modèle, par exemple) et / ou supervisée (classification, par exemple).
- apprends plusieurs niveaux de représentations liés à différents niveaux d'abstraction. Ces niveaux représentent une hiérarchie de concepts.

III.4. Classification de l'apprentissage profond

L'apprentissage profond peut être classé en trois classes principales en fonction des objectifs pour lesquels elles ont été conçues: [6]

- **Les réseaux profonds d'apprentissage non supervisé ou génératif** visent à capturer la corrélation d'ordre élevé des données d'entrée non étiquetées à des fins de reconnaissance ou de synthèse de modèles. Lorsqu'ils sont utilisés pour caractériser les distributions statistiques communes des données observées et de leurs classes associées, les réseaux disposent d'un mode génératif et pourraient être transformés en réseaux discriminants pour un apprentissage plus approfondi.
- **Les réseaux profonds d'apprentissage supervisé** sont utilisés lorsque des données d'étiquette cible sont disponibles, les modèles pouvant directement fournir un pouvoir discriminant aux fins de la classification.
- **Les réseaux profonds hybrides** sont la combinaison des deux types de réseaux mentionnés ci-dessus, de sorte que les réseaux profonds non supervisés pourraient fournir une excellente initialisation sur la base de laquelle la discrimination pourrait être examinée.

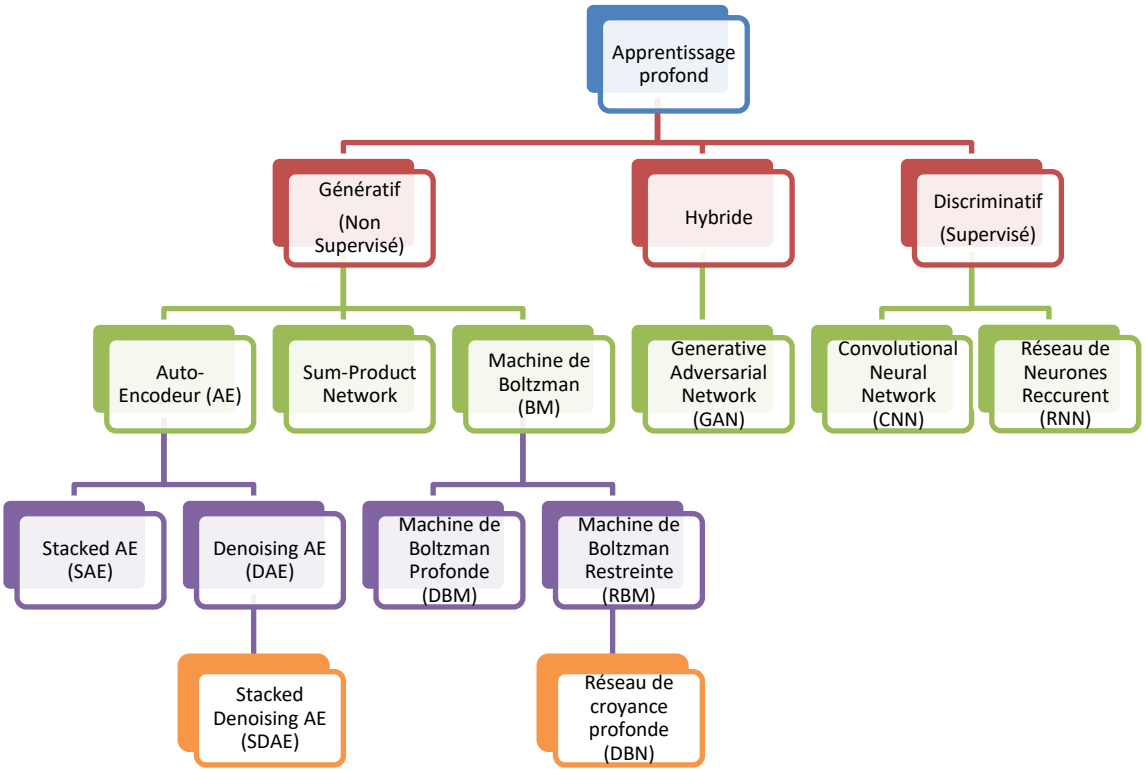


Figure III.1. *Classification des méthodes d'apprentissage profond*

III.5. Quelques méthodes d'apprentissage profond

En pratique, tous les algorithmes d'apprentissage profond sont des réseaux de neurones, qui partagent certaines propriétés de base communes. Ils sont tous constitués de neurones interconnectés organisés en couches. Ce qui les différencie, c'est l'architecture du réseau (ou la manière dont les neurones sont organisés dans le réseau) et parfois la manière dont ils sont formés.

Dans cet esprit, nous présentons les principales méthodes d'apprentissage profond. La liste suivante n'est pas exhaustive, mais elle représente la grande majorité des algorithmes utilisés aujourd'hui:

III.5.1. Machines de Boltzmann Restreintes

Dans sa forme la plus simple, une machine de Boltzmann est composée d'une couche de neurones qui reçoit l'entrée, ainsi que d'une couche de neurones cachée. Si nous supposons que les neurones d'une même couche sont indépendants entre eux, nous appelons cette configuration une machine de Boltzmann restreinte (*RBM*) [26]. Les machines de Boltzman restreintes sont un type particulier de réseau de neurones génératifs, où les neurones sont organisés en deux couches, à savoir visible et masquée. Contrairement aux réseaux à retransmission directe, les données d'une RBM peuvent circuler dans les deux sens - des unités visibles aux unités cachées, et inversement. La RBM est l'un des outils d'apprentissage en profondeur les plus populaires en raison de sa capacité à connaître la distribution de la probabilité des entrées de manière supervisée et non supervisée. Elle a été introduite par Paul Smolensky en 1986 avec le nom *Harmonium*. Cependant, elle est popularisée par Hinton en 2002 avec l'avènement de l'algorithme d'entraînement amélioré pour RBM. Après cela, elle a eu une large application dans diverses tâches telles que l'apprentissage de la représentation, la réduction de la dimensionnalité, les problèmes de prédiction [22], la classification, la régression, le filtrage collaboratif (*collaborative filtering*), l'apprentissage des fonctionnalités (*feature learning*) et modélisation des sujets (*topic modeling*). En 2002, le professeur Hinton a introduit la divergence contrastive (*CD*), un algorithme non supervisé pour la formation des RBM. [13]

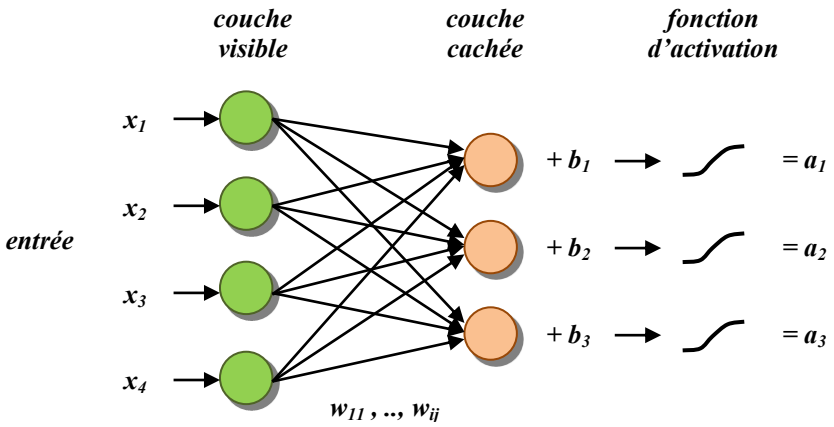


Figure III.2. *Machine de Boltzman restreinte*

L'énergie (ou fonction) d'activation pour une Machine de Boltzmann Restreinte est définie de la manière suivante:

$$E = -\left[\sum_{i,j}(w_{ij}x_i h_j) + \sum_i(b_i x_i) + \sum_j(c_j h_j)\right] \quad \textbf{(III.1)}$$

Avec:

- w_{ij} la matrice de poids entre le neurone j et le neurone i ;
- x_i est l'état du neurone visible i, $x_i \in \{0, 1\}$;
- h_j est l'état du neurone caché j ;
- b_i et sont c_j respectivement les biais des neurones x_i et h_j.

III.5.2. Perceptrons multicouches

Des réseaux neuronaux organisés en plusieurs couches (au moins une couche cachée) au sein desquelles une information circule de la couche d'entrée vers la couche de sortie uniquement ; il s'agit donc d'un réseau à propagation directe (*feedforward*) avec propagation anticipée, couches entièrement connectées. Chaque couche est constituée d'un nombre variable de neurones, les neurones de la dernière couche (dite « de sortie ») étant les sorties du système global.

Le concept de base du perceptron singulier a été introduit par Rosenblatt en 1958. Le perceptron calcule une sortie unique à partir de multiples entrées à valeurs réelles en formant une combinaison linéaire en fonction de ses poids d'entrée, puis en plaçant éventuellement la sortie via une fonction d'activation non linéaire. Mathématiquement, cela peut être écrit comme

$$y = \varphi(\sum_{i=1}^{n} w_i x_i + b) = \varphi(W^T X + b) \quad \textbf{(III.2)}$$

Avec

- ***W*** désigne le vecteur des poids ;
- ***X*** est le vecteur des entrées ;
- ***B*** le biais ;
- ***φ*** représente la fonction d'activation.

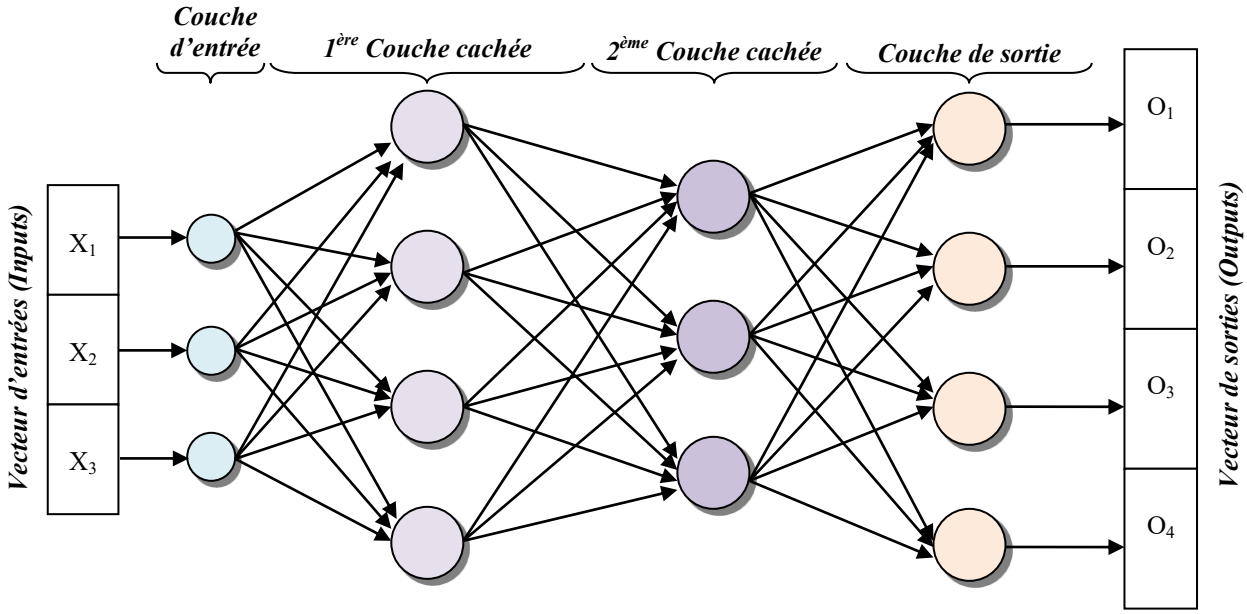


Figure III.3. *Perceptron multicouches*

Un réseau typique de perceptron multicouche (*MLP*) comprend un ensemble de nœuds sources formant la couche d'entrée, une ou plusieurs couches cachées de nœuds de calcul et une couche de sortie de nœuds. Le signal d'entrée se propage couche par couche sur le réseau. Le flux de signaux d'un tel réseau avec une couche cachée est illustré par la Figure III.4.

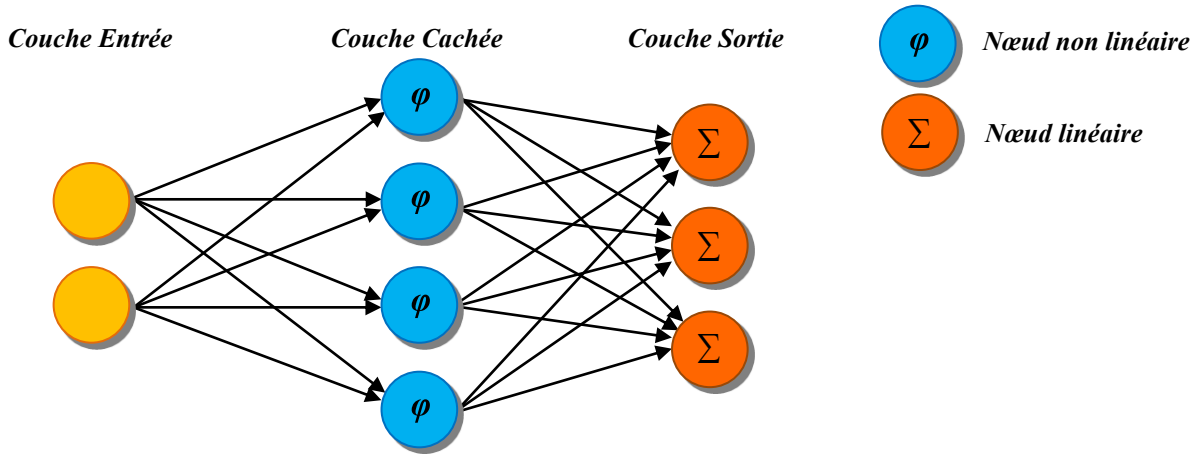


Figure III.4. *Flux de signal d'un MLP*

Les calculs effectués par un tel réseau à anticipation avec une seule couche cachée avec des fonctions d'activation non linéaires et une couche de sortie linéaire peuvent être écrits de manière mathématique.

$$\boldsymbol{x = f(s) = B\varphi(As + a) + b} \quad \textbf{(III.3)}$$

où

- $\boldsymbol{s}$ est un vecteur d'entrées et
- $\boldsymbol{x}$ un vecteur de sorties.
- $\boldsymbol{A}$ est la matrice des poids de la première couche,
- $\boldsymbol{a}$ est le vecteur de polarisation de la première couche.
- $\boldsymbol{B}$ et $\boldsymbol{b}$ sont, respectivement, la matrice de pondération et le vecteur de polarisation de la deuxième couche.
- $\boldsymbol{\varphi}$ dénote une non-linéarité par élément.

Les réseaux MLP sont généralement utilisés dans les problèmes d'apprentissage supervisé. Cela signifie qu'il existe un ensemble d'apprentissage entrée/sortie et que le réseau doit apprendre à modéliser la dépendance entre elles. La formation ici signifie l'adaptation de tous les poids et biais à leurs valeurs optimales.

III.5.3. Réseaux de croyance profonde (DBN)

Un réseau DBN est un type de réseau de neurones profonds qui est essentiellement un modèle génératif probabiliste comprenant plusieurs couches de variables cachées. Ces réseaux ont à la fois des bords dirigés et non dirigés. Il est formé à l'aide d'une série de RBM, souvent d'auto-encodeurs, avec une couche supplémentaire formant un réseau bayésien. L'utilisation de RBM signifie la présence d'aucune connexion intra-couche. De plus, les performances d'un DBN dépendent en grande partie de l'initialisation des nœuds. Par conséquent, les couches utilisent une formation préalable non supervisée à l'aide de la procédure d'empilement de RBM, qui intègre une divergence contraste (*CD*). Un réseau de croyance (BN) est un graphe acyclique dirigé constitué de couches d'unités binaires stochastiques, chaque couche connectée ayant une pondération. Ces unités binaires stochastiques ont l'état 0 ou 1 et la probabilité d'être activé (devenant 1) est déterminée par un biais et une entrée pondérée provenant d'autres unités, représentée par: [13]

$$\boldsymbol{p(u_i = 1) = \frac{1}{1+e^{-\left(b_i + \sum_j u_j w_{ji}\right)}}} \quad \textbf{(III.4)}$$

Où :

- ***u*** est une unité stochastique,
- ***b*** le biais associé à cette unité
- ***w*** le paramètre pondéré.

Certaines de ces unités sont des unités visibles. Celles-ci posent deux problèmes principaux qui doivent être résolus. Ce sont :

1. **Le problème d'inférence**: où les états des unités non observées doivent être inférés.
2. **Le problème d'apprentissage**: les interactions entre les unités sont ajustées pour que le réseau soit capable de générer les données observées dans les unités visibles.

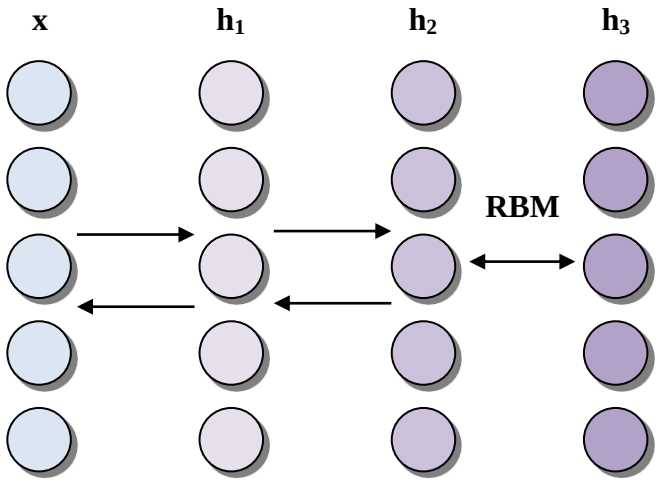


Figure III.5. *Schéma d'un réseau DBN*

Un réseau DBN est un réseau de croyances multicouches dans lequel chaque couche est un RBM empilé les uns sur les autres pour former le réseau DBN. La première étape de l'apprentissage d'un DBN en utilisant une approche de base consiste à apprendre une couche de caractéristiques des unités visibles à l'aide de l'algorithme CD. L'étape suivante consiste à traiter les activations des caractéristiques précédemment formées comme des unités visibles, puis à en apprendre davantage à partir des caractéristiques des couches cachées suivantes. La dernière étape consiste à former l'ensemble du réseau DBN de manière supervisée, ce qui permet d'affiner les paramètres du réseau. Comme indiqué précédemment, les RBM peuvent être empilés et formés de manière gourmande pour former ces DBN qui sont des modèles graphiques permettant d'apprendre à extraire des représentations hiérarchiques profondes des

données d'entraînement en entrée. Ces *frameworks* modélisent comme suit la distribution commune entre le vecteur de données observé $\boldsymbol{x}$ et les $\boldsymbol{l}$ couches cachées $\boldsymbol{h^k}$: [13]

$$\boldsymbol{P(x, h^1, \dots, h^l)} = \left(\prod_{k=0}^{l-2} \boldsymbol{P(h^k | h^{k+1})}\right) \boldsymbol{P(h^{l-1}, h^l)} \qquad \textbf{(III.5)}$$

Où

$\boldsymbol{x} = \boldsymbol{h^0}$ et $\boldsymbol{P(h^{k-1} | h^k)}$ représente la distribution conditionnelle de l'unité visible conditionnée par les unités cachées de la RBM au niveau $\boldsymbol{k}$. De plus, la distribution conjointe visible-cachée dans le niveau supérieur du RBM est $\boldsymbol{P(h^{l-1} | h^l)}$, comme schématisé par la Figure IV.3.

III.5.4. Réseaux de neurones convolutionnels

Un CNN est un réseau de neurones à propagation directe (*feedforward*) avec plusieurs types de couches spéciales. Par exemple, les couches convolutives appliquent un filtre à l'image d'entrée (ou au son) en faisant glisser ce filtre sur tout le signal entrant, afin de produire une carte d'activation (*activation map*) à $\boldsymbol{n}$ dimensions. Il existe certaines preuves que les neurones dans les CNN sont organisés de la même manière que les cellules biologiques organisées dans le cortex visuel du cerveau. Aujourd'hui, les CNN surpassent tous les autres algorithmes ML pour un grand nombre de tâches dans lesquelles des motifs répétitifs peuvent être trouvés - par exemple, cela peut être une image avec des bords répétitifs et d'autres motifs, ou comme le traitement de vision par ordinateur et le traitement du langage naturel (*NLP*). Les réseaux de convolution sont particulièrement efficaces pour capturer les modèles locaux de données en raison de la structure des couches de convolution.

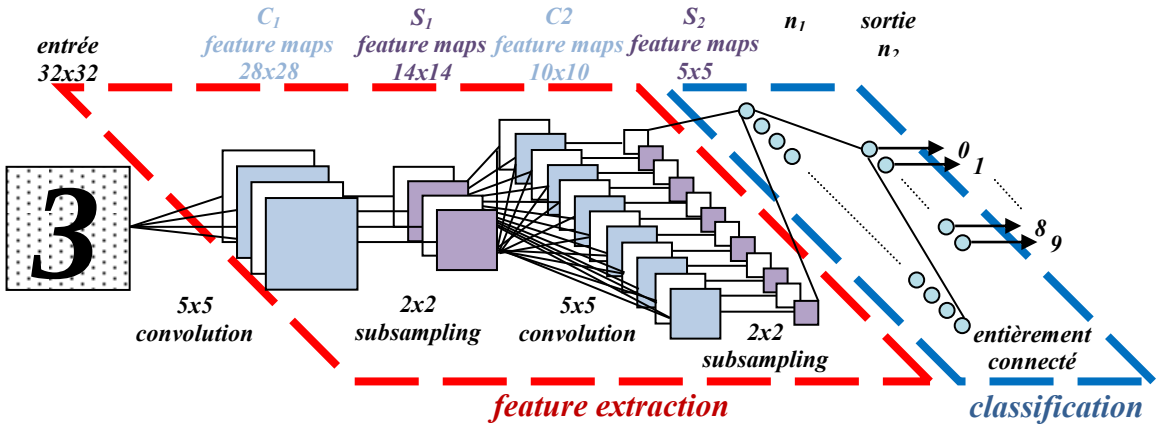


Figure III.6. *Schéma d'un réseau de neurones convolutionnel*

Actuellement, le processus de reconnaissance d'image est mis en œuvre principalement en appliquant des CNN. L'emploi des CNN pour la reconnaissance d'image dans les années 90 a été le premier succès concret des réseaux de neurones artificiels. Les CNN ont déjà dépassé les performances humaines pour certaines tâches de reconnaissance d'image. Ces réseaux sont également utilisés pour une grande variété d'autres tâches, et ils constituent le type de réseaux de neurones profonds le plus utilisé de nos jours.

Les réseaux convolutifs ont généralement plusieurs couches cachées qui, en général, sont organisées en blocs (CONV→ ReLU → POOL). Mais certains réseaux convolutifs ont des centaines de couches cachées. Les calques masqués plus proches du calque d'entrée détectent des caractéristiques très simples telles que les bords et les dégradés de couleurs dans les images. Ensuite, les couches supérieures combinent ces caractéristiques simples dans des motifs plus complexes. Enfin, les couches proches de la sortie combinent ces motifs plus complexes.

III.5.4.1. Convolution

Une convolution extrait les mosaïques de la carte des caractéristiques (*feature map*) en entrée et leur applique des filtres pour calculer de nouvelles fonctionnalités, en produisant une carte en sortie ou une fonctionnalité convolution (dont la taille et la profondeur peuvent être différentes de celles de la carte des caractéristiques en entrée). Les convolutions sont définies par deux paramètres:

- Taille des mosaïques extraites (généralement 3x3 ou 5x5 pixels).
- La profondeur de la carte des caractéristiques en sortie, qui correspond au nombre de filtres appliqués.

Lors d'une convolution, les filtres (matrices de la même taille que la taille de la mosaïque) glissent sur la grille de la carte d'entités en entrée horizontalement et verticalement, pixel par pixel, en extrayant chaque mosaïque correspondante (voir les figures suivantes).

Supposons que nous avons à l'entrée la matrice suivante :

1	-1	2	3
3	-1	3	1
0	-1	-1	3
3	3	1	0

Figure III.7. *Exemple d'une carte des caractéristiques en entrée (Input feature map)*

Ainsi que le filtre suivant (à appliquer sur la carte des caractéristiques en entée):

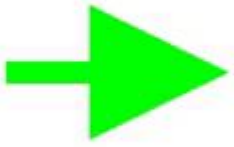
1	0
1	0

Figure III.8. *Exemple de filtre convolutif*

Appliquons maintenant les opérations de convolution à cette entrée en utilisant ce filtre et un pas égal à 1 pixel:

Mosaïque 1 :

1 x 1	-1x 0	2	3
3 x 1	-1x 0	3	1
0	-1	-1	3
3	3	1	0



4

Mosaïque 2 :

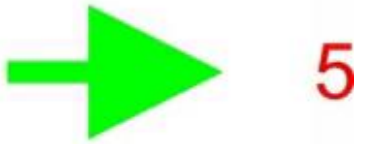
1	-1x 1	2 x 0	3
3	-1x 1	3 x 0	1
0	-1	-1	3
3	3	1	0



-2

Mosaïque 3 :

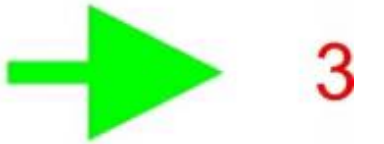
1	-1	2 x 1	3 x 0
3	-1	3 x 1	1 x 0
0	-1	-1	3
3	3	1	0



5

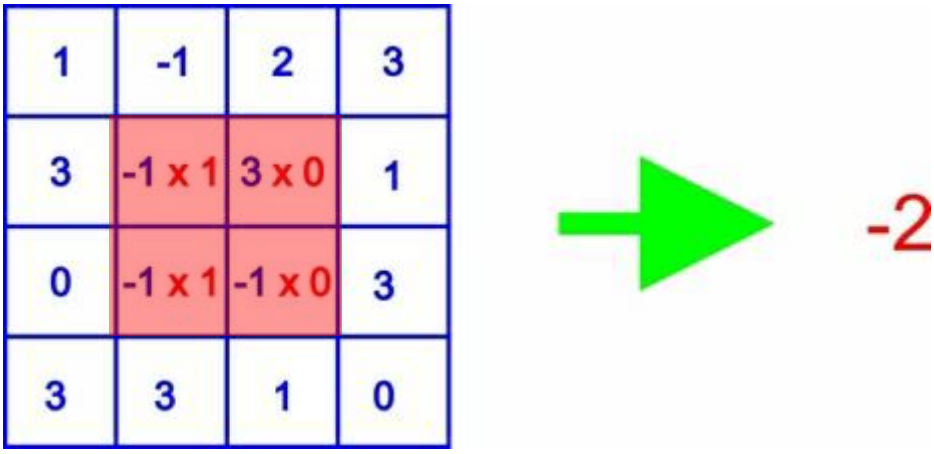
Mosaïque 4 :

1	-1	2	3
3 x 1	-1 x 0	3	1
0 x 1	-1 x 0	-1	3
3	3	1	0



3

Mosaïque 5 :



Et ainsi de suite jusqu'à la fin. À la fin, nous obtenons le résultat suivant (avant d'appliquer une fonction d'activation):

4	-2	5
3	-2	2
3	2	0

En déplaçant le filtre dans toutes les positions possibles, nous avons créé une carte de caractéristiques (*feature map*) de ce filtre. Chaque filtre représente une certaine fonctionnalité. Une carte de caractéristiques montre à quel point cette caractéristique est évidente à une certaine position de l'entrée.

Dans un CNN typique, la dernière couche de convolution est aplatie (*flattened*) dans un vecteur et généralement introduite dans une ou plusieurs couches entièrement connectées (*fully-connected*). La majorité des paramètres pouvant être appris dans la plupart des CNN résident en réalité dans leurs couches finales entièrement connectées, et non pas dans les couches de convolution. Par exemple, dans le célèbre réseau AlexNet (2012), 59 millions sur 62 millions de paramètres étaient situés dans ses trois dernières couches entièrement connectées. Cependant, cette tendance change progressivement avec les nouvelles architectures de CNN. [12]

IV.5.4.2. ReLU

Après chaque opération de convolution, le CNN applique une transformation ReLU[3] (Unité Rectifiée Linéaire) à l'entité convolue, afin d'introduire la non-linéarité dans le modèle. La fonction ReLU, $F(x) = max(0, x)$, renvoie x pour toutes les valeurs de $x > 0$ et renvoie 0 pour toutes les valeurs de $x \leq 0$.

III.5.4.3. Pooling

Après la transformation ReLU vient une étape de mise en commun, dans laquelle CNN sous-échantillonne la fonctionnalité résolue (pour gagner du temps de traitement), réduisant ainsi le nombre de dimensions de la carte des fonctionnalités, tout en préservant les informations les plus critiques. Un algorithme commun utilisé pour ce processus est appelé *max-pooling*[4].

Le *max-pooling* fonctionne de manière similaire à la convolution. Nous glissons sur la carte des fonctionnalités et extrayons les carreaux d'une taille spécifiée. Pour chaque mosaïque, la valeur maximale est sortie dans une nouvelle carte de fonctions et toutes les autres valeurs sont ignorées. Les opérations de *max-pooling* prennent deux paramètres: [8]

- Taille du filtre de *max-pooling* (généralement 2x2 pixels)
- La distance, en pixels, séparant chaque mosaïque extraite.

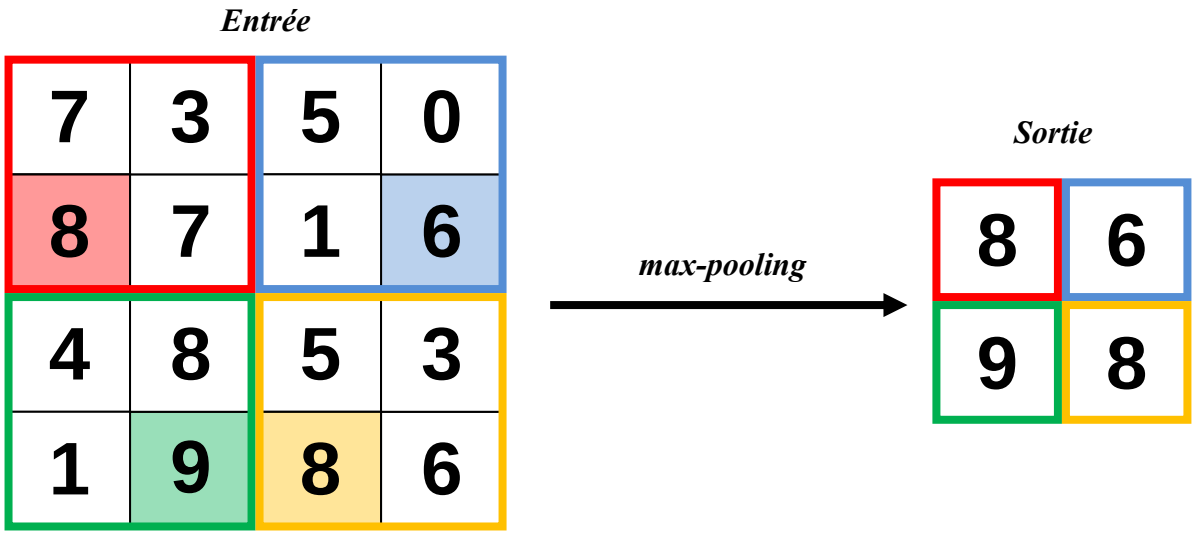


Figure III.9. *Exemple de max-pooling réalisé sur une carte de caractéristiques 4x4 avec un filtre 2x2 et un pas de 2*

[3] ReLU est utilisé comme fonction d'activation dans divers réseaux de neurones.
[4] Il existe différents types de couches de pooling: *max-pooling*, *average-pooling*, etc. Le plus courant et le plus utilisé est le *max-pooling*.

Contrairement à la convolution, dans laquelle les filtres glissent sur la carte de fonctions pixel par pixel, en *max-pooling*, la foulée détermine les emplacements d'extraction de chaque mosaïque. Pour un filtre 2x2, une foulée de 2 indique que l'opération de regroupement maximal extraira toutes les mosaïques 2x2 non chevauchantes de la carte des fonctionnalités (voir la figure IV.9).

Une couche de regroupement n'a aucun paramètre pouvant être appris. De ce fait, les couches de *pooling* ne sont généralement pas incluses dans le nombre total de couches de réseaux de convolution. Par exemple, si un réseau de neurones comporte sept couches convolutives, trois couches entièrement connectées, une couche soft-max et quatre couches de *pooling*, il s'agit d'un réseau neuronal profond de 11 couches (7 + 3 + 1), car nous ne comptons que les couches avec des paramètres pouvant être appris

III.5.4.4. Couches entièrement connectées (Fully conneced layers)

À la fin d'un réseau CNN, il y a une ou plusieurs couches entièrement connectées (lorsque deux couches sont "entièrement connectées", chaque nœud de la première couche est connecté à chaque nœud de la couche suivante). Leur travail consiste à effectuer une classification basée sur les caractéristiques extraites des convolutions.

En général, la couche finale, entièrement connectée, contient une fonction d'activation *Softmax*, qui génère une valeur de probabilité de ***0*** à ***1*** pour chacune des étiquettes de classification que le modèle tente de prédire. [8]

Dans certaines architectures de réseaux neuronaux convolutifs récentes, les couches entièrement connectées à l'extrémité des réseaux sont remplacées par plusieurs couches de pooling moyennes (*average-pooling*). Cela permet à ces réseaux de réduire considérablement le nombre total de paramètres pouvant être appris, ce qui permet une meilleure prévention des sur-ajustements. [12]

III.5.5. Réseaux de neurones récurrents

Les réseaux entièrement connectés et les réseaux convolutionnels appartiennent à des réseaux de neurones à *feed-forward*. Cela signifie que les signaux de ces réseaux se déplacent sans faire de boucle à l'intérieur des réseaux. Lorsque nous travaillons avec des réseaux de neurones, nous avons généralement des entrées et des sorties de taille fixe. Cependant, des entrées et des sorties de taille fixe entravent l'exécution de certaines tâches. Un exemple classique d'une telle tâche est la traduction. Pour effectuer une traduction adéquate, nous

devons avoir des entrées et des sorties de longueur variable. De plus, les réseaux convolutifs et les réseaux entièrement connectés ne tiennent pas compte de l'ordre de leurs entrées, ce qui pose également un problème sérieux pour certaines tâches.

Un réseau de neurones récurrent (*RNN*) est un type de réseau ayant un état interne (ou mémoire), basé sur tout ou partie des données d'entrée déjà fournies au réseau. La sortie d'un réseau récurrent est une combinaison de son état interne (mémoire d'entrées) et du dernier échantillon d'entrée. En même temps, l'état interne change pour intégrer les nouvelles données saisies. En raison de ces propriétés, les réseaux récurrents sont de bons candidats pour les tâches qui traitent des données séquentielles, telles que des données textuelles ou temporelles. Il existe de nombreux types de données séquentielles, par exemple, la parole, la vidéo, diverses formes de séries chronologiques.

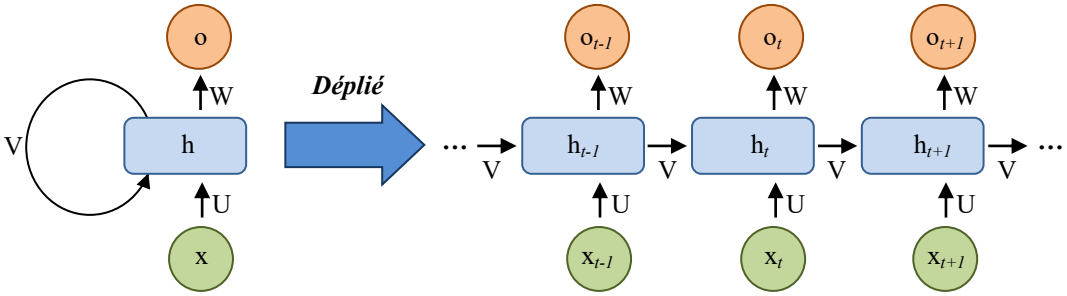


Figure III.10. *Schéma d'un réseau de neurones récurrents à une unité reliant l'entrée et la sortie du réseau*

La description mathématique du processus de transfert de mémoire est comme suit:

$$\boldsymbol{h_t = \varphi(W x_t + U h_{t-1})} \qquad \textbf{(III.6)}$$

où

- $\boldsymbol{h_t}$ est l'état caché au temps $\boldsymbol{t}$;
- $\boldsymbol{W}$ est la matrice de pondération (comme celle utilisée pour les réseaux *feed-forward*) ;
- $\boldsymbol{x_t}$ est l'entrée au même temps $\boldsymbol{t}$;
- $\boldsymbol{h_{t-1}}$ est l'état caché au temps $\boldsymbol{t\text{-}1}$;
- $\boldsymbol{U}$ est la matrice d'état caché à état caché, autrement connue comme une matrice de transition et similaire à une chaîne de Markov;
- $\boldsymbol{\varphi}$ est une fonction (soit une fonction sigmoïde logistique ou tanh) qui est un outil standard pour condenser des valeurs très grandes ou très petites dans un espace logistique, ainsi que pour rendre les gradients exploitables pour la rétro-propagation.

Comme cette boucle de rétroaction se produit à chaque pas de temps, chaque état caché $\boldsymbol{h_t}$ contient des traces non seulement de l'état masqué précédent, mais également de tous ceux qui ont précédé $\boldsymbol{h_{t-1}}$ aussi longtemps que la mémoire peut persister.

Au début des années 90, le problème du gradient disparaissant (*vanishing gradient problem*) est devenu un obstacle majeur à la performance des réseaux RNN. Tout comme une ligne droite exprime un changement de x parallèlement à un changement de y, le gradient exprime le changement de toutes les pondérations par rapport au changement d'erreur. Si nous ne pouvons pas connaître le gradient, nous ne pouvons pas ajuster les poids dans une direction qui réduira l'erreur, et notre réseau cessera d'apprendre. Comme les couches et les pas de temps des réseaux de neurones profonds sont reliés les uns aux autres par la multiplication, les dérivés sont susceptibles de s'effacer ou d'exploser.

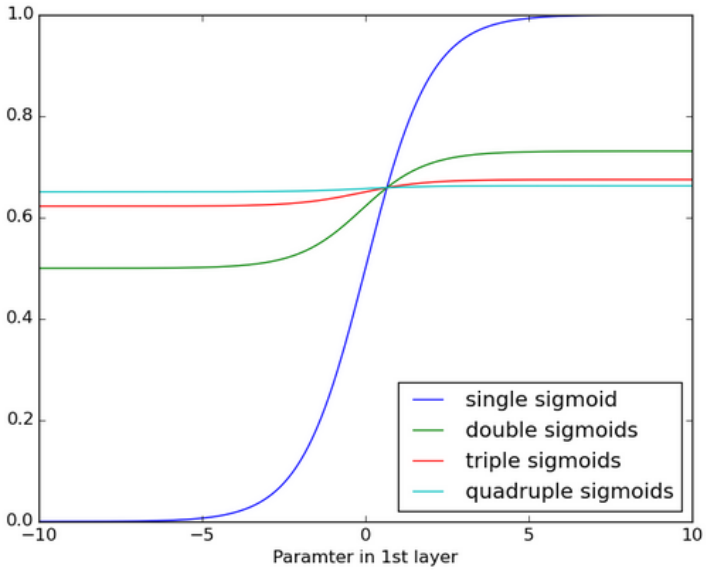


Figure III.11. *Effet de l'application répétée d'une fonction sigmoïde*

Ci-dessus, nous pouvons voir les effets de l'application répétée d'une fonction sigmoïde. Les données sont aplaties jusqu'à ce que, pour les grandes étendues, elles n'aient pas de pente détectable. Ceci est analogue à un gradient qui disparaît lorsqu'il passe à travers plusieurs couches.

III.5.6. Mémoire longue à court terme

Parfois, il suffit de regarder les informations récentes pour effectuer la tâche actuelle. Par exemple, considérons un modèle de langage essayant de prédire le mot suivant en fonction des mots précédents. Si nous essayons de prédire le dernier mot dans "les nuages sont dans le

ciel", nous n'avons pas besoin de contexte supplémentaire - il est assez évident que le mot suivant sera ciel. Dans de tels cas, où l'écart entre les informations pertinentes et l'endroit dont ils ont besoin est faible, les RNN peuvent apprendre à utiliser les informations antérieures.

Mais il y a aussi des cas où nous avons besoin de plus de contexte. Essayez de prédire le dernier mot du texte «J'ai grandi au Japon… je parle couramment le japonais». Des informations récentes suggèrent que le mot suivant est probablement le nom d'une langue, mais si nous voulons préciser quelle langue, il faut le contexte de ''Japon'', de plus en plus. Il est tout à fait possible que la fosse entre les informations pertinentes et le moment où elles soient nécessaires devienne très important. Malheureusement, à mesure que cet écart se creuse, les RNN deviennent incapables d'apprendre à connecter les informations.

Au milieu des années 90, les chercheurs allemands Sepp Hochreiter et Juergen Schmidhuber ont proposé une variante du réseau récurrent avec des unités dites de mémoire longue à court terme ou LSTM, comme solution au problème du gradient disparaissant.

Les réseaux LSTM sont un type particulier de réseaux de neurones récurrents, capables d'apprendre des dépendances à long terme. Ils diffèrent des réseaux RNN en ce que ces derniers sont simplement capables d'apprendre des dépendances à court terme.

La clé de la méthode LSTM est l'état de la cellule. La LSTM a la capacité de supprimer ou d'ajouter des informations à l'état de la cellule, soigneusement régulé par des structures appelées portes. Les portes pourraient être une fonction sigmoïde où une valeur de 1 signifie que toutes les informations passent et une valeur de 0 signifie le contraire. Un LSTM a trois de ces portes, comme illustré à la figure 2. Les LSTM sont particulièrement utiles pour les données de différentes périodes. Un exemple de ceci est le suivi de la santé, où les données d'un patient pourraient arriver en temps réel (c.-à-d. d'un appareil portable), pourraient correspondre à des données annuelles (c.-à-d. Après un examen médical approfondi) ou à plusieurs années (c.-à-d. Depuis l'enfance à l'âge adulte).

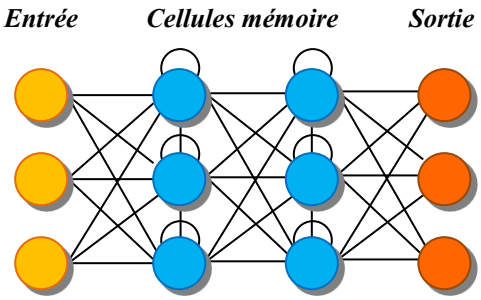


Figure III.12. *Schéma d'un réseau LSTM*

III.5.7. Auto-encodeurs

Une classe d'algorithmes d'apprentissage non supervisés de caractéristiques discriminantes, dans lesquels la forme de sortie est la même que l'entrée ce qui permet au réseau de mieux apprendre les représentations de base. Récemment, le concept d'auto-encodeur est devenu plus largement utilisé pour l'apprentissage de modèles génératifs.

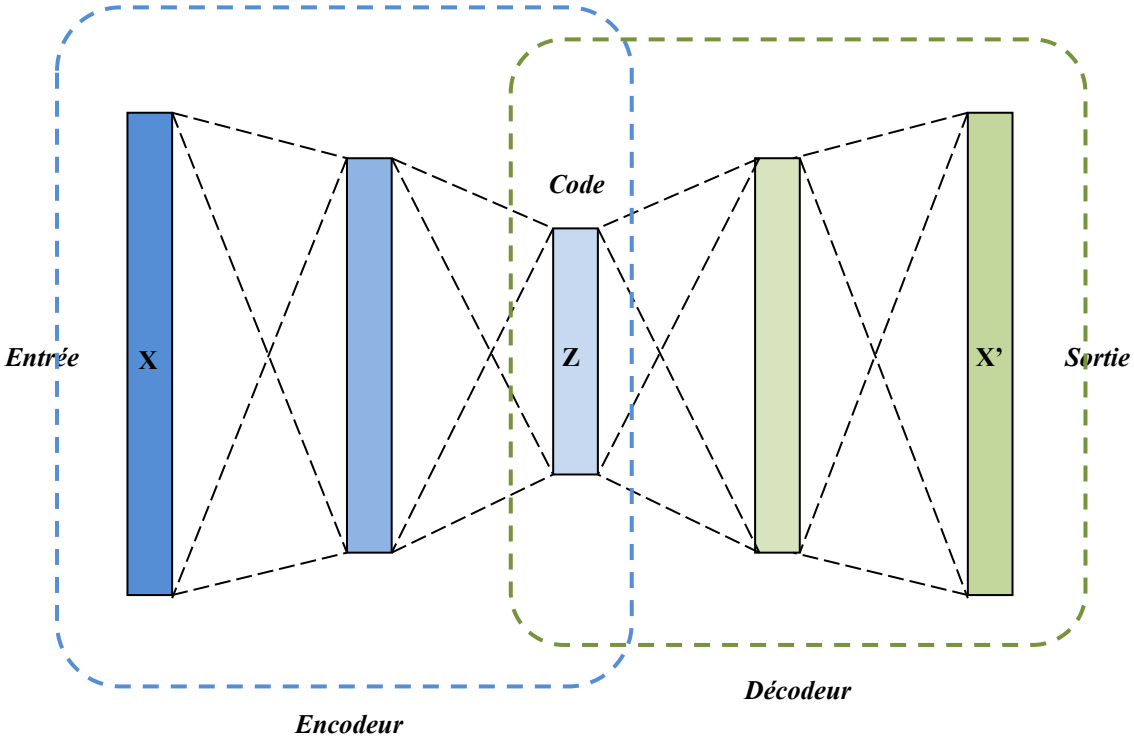


Figure III.13. *Structure schématique d'un auto-encodeur avec 3 couches cachées entièrement connectées*

Les auto-encodeurs variationnels (VAE) sont similaires aux AE, mais ils ajoutent du bruit gaussien pour sélectionner les couches cachées lors du décodage et estimer la quantité d'informations contenues dans cette couche. AE et VAE conviennent particulièrement à la détection d'anomalies. Ils pourraient trouver des anomalies qui ne sont pas facilement détectées par les médecins, par exemple. Cela pourrait être utilisé, par exemple, pour rechercher les caractéristiques d'une nouvelle épidémie dans une région spécifique du monde.

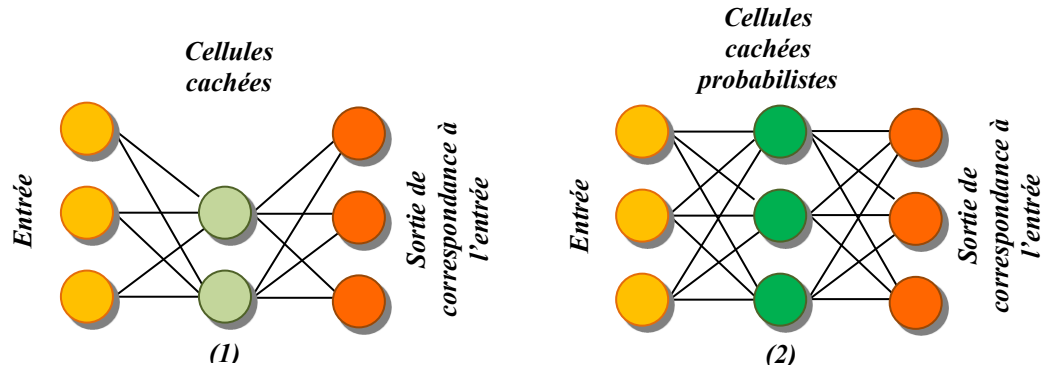


Figure III.14. *Auto-Encodeurs (1) et Auto-Encodeurs Variationnels (2)*

III.5.8. Mesures d'évaluations des modèles

En termes d'évaluation des modèles, plusieurs mesures ont été définies dans le but de valoriser la précision et la robustesse des différents algorithmes d'apprentissage automatique ou profond. En général, positif = identifié et négatif = rejeté. Donc:

- **Vrai positif :** ***TP*** = correctement identifié ;
- **Faux positif :** ***FP*** = identifié de manière incorrecte ;
- **Vrai négatif :** ***TN*** = correctement rejeté ;
- **Faux négatif :** ***FN*** = rejeté de manière incorrecte ;
- **Rappel ou Sensibilité ou TPR (Taux Vrai Positif) :** Nombre d'éléments correctement identifiés comme positifs par rapport au total des vrais positifs. Cette mesure est donnée par la formule ***Recall = TPR = TP / (TP + FN)*** ;
- **Spécificité ou TNR (Taux Vrai Négatif) :** nombre d'éléments correctement identifiés comme négatifs par rapport au total des négatifs. ***TNR = TN / (TN + FP)*** ;
- **Précision :** Nombre d'éléments correctement identifiés comme positifs sur le total des éléments identifiés comme positifs. ***Precision = TP / (TP + FP)*** ;
- **Taux de faux positifs ou erreur de type I :** nombre d'éléments incorrectement identifiés comme positifs par rapport au total des véritables négatifs. ***FPR = FP / (FP + TN)*** ;

- **Faux taux négatif ou erreur de type II :** nombre d'éléments identifiés à tort comme négatifs par rapport au total des vrais positifs. ***FNR = FN / (FN + TP)*** ;
- **Matrice de confusion :** également connue sous le nom de matrice d'erreur, est une disposition de tableau spécifique permettant de visualiser les performances d'un algorithme, généralement un algorithme d'apprentissage supervisé (dans le cas d'un apprentissage non supervisé, il s'agit généralement d'une matrice correspondante).

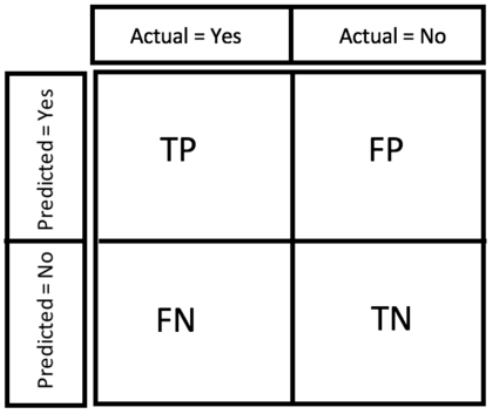


Figure III.15. *Matrice de confusion*

- **F1 Score:** C'est une moyenne harmonique de précision et de rappel donnée par ***F1 = 2*Precision*Recall / (Precision + Recall)*** ;
- **Exactitude (Accuracy) :** pourcentage du nombre total d'éléments classés correctement. ***Accuracy = (TP + TN) / (N + P)*** ;
- **ROC-AUC Score**

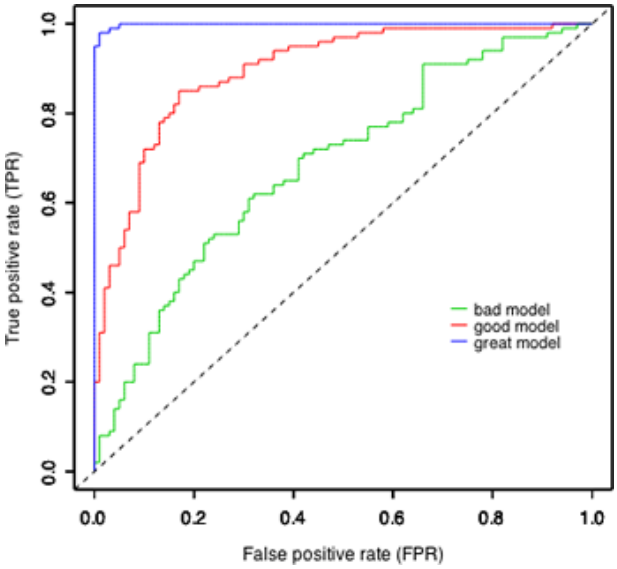


Figure III.16. *Interprétation graphique de la mesure ROC-AUC*

Selon l'interprétation probabiliste du score ROC-AUC, si nous choisissons au hasard un cas positif et un cas négatif, la probabilité que le cas positif l'emporte sur le cas négatif selon le classificateur est donnée par l'AUC. Ici, le rang est déterminé en fonction de l'ordre des valeurs prédites.

Mathématiquement, il est calculé par la surface sous la courbe de sensibilité (TPR) en fonction de FPR (1-spécificité). Dans le cas idéal, nous aimerions avoir une sensibilité et une spécificité élevées, mais dans des scénarios réels, il existe toujours un compromis entre sensibilité et spécificité.

Certaines caractéristiques importantes de ROC-AUC sont:

- La valeur peut aller de ***0*** à ***1***. Toutefois, le score de AUC d'un classifieur aléatoire pour des données équilibrées est de ***0.5***.
- Le score ROC-AUC est indépendant du seuil défini pour la classification car il ne prend en compte que le rang de chaque prédiction et non sa valeur absolue. Il n'en va pas de même pour le F1 score qui nécessite une valeur seuil en cas de sortie de probabilités.

III.5.9. Conclusion

Les Réseaux de neurones sont un domaine de recherche très actif, qui ne cesse de progresser afin d'améliorer les performances des résultats.

L'apprentissage profond (*DL*) a pu s'imposer et révolutionner plusieurs domaines technologiques. Traduction automatique moderne, moteurs de recherche, assistants informatiques et plusieurs applications de notre vie quotidienne sont tous alimentés par un apprentissage profond. Les modèles d'apprentissage profond tentent d'imiter au maximum les modes de traitement de l'information et de communication observés dans le système nerveux biologique.

Ce chapitre a présenté les architectures des réseaux de neurones les plus connues et les plus utilisées ainsi que les nouvelles architectures qui semblent avoir un avenir prometteur dans différents domaines d'application de la technologie contemporaine.

Chapitre IV : CONCEPTION DE LA METHODE PROPOSEE

« Intellectuals solve problems, geniuses prevent them. »

Albert Einstein

IV.1. Introduction

Dans ce chapitre nous allons présenter d'une façon détaillée la conception de notre méthode proposée pour la réduction de la dimensionnalité et la classification des données volumineuses de grande dimension, issues du flux réseau, pour le but de la détection d'intrusion.

La première étape consiste à faire une réduction de dimension de la base de données qui contient un grand nombre d'attributs, et ceci en utilisant une méthode très répandue dans le domaine de réduction de dimension: la sélection et l'extraction d'attributs non supervisée par un réseau de neurones profond de type auto-encodeur empilé (*SAE*: *Stacked Auto-Encoder* ou encore *DAE*: *Deep Auto-Encoder*).

Pour la deuxième étape, qui consiste à prédire les classes des nouvelles instances (qui contiennent beaucoup moins d'attributs, cela dit que les attributs qui ont été réduits sont moins pertinents que ceux qui ont été gardés), nous avons eu recours à une classification basée sur un algorithme d'apprentissage automatique profond du type CNN.

IV.2. Schéma du modèle proposé

Pour la concrétisation de notre approche, nous avons adopté le schéma suivant du modèle de la méthode de conception (voir figure ci-dessous) :

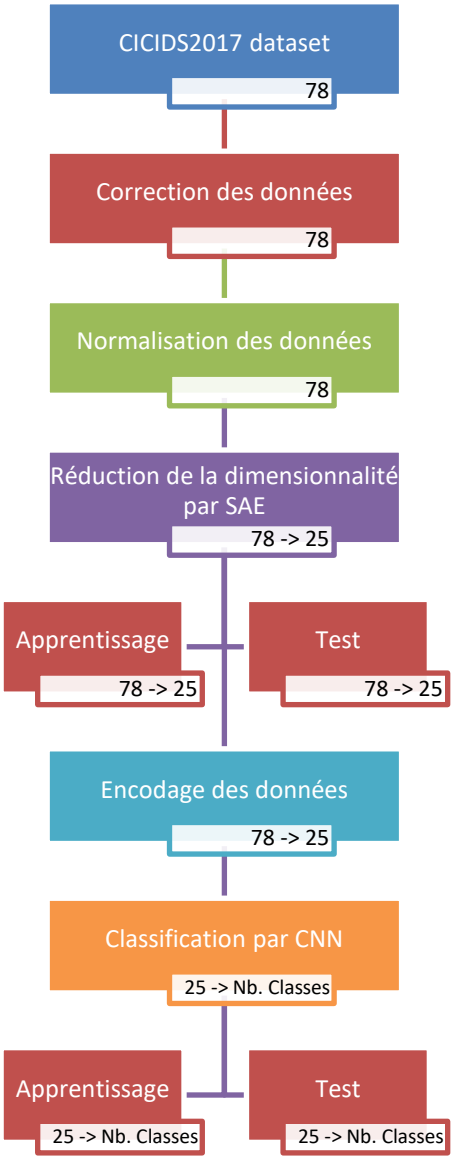


Figure IV.1. *Schéma de la méthode de conception*

IV.3. Méthode proposée pour la réduction de dimensionnalité

La classification des instances rencontre plusieurs problèmes tels que le grand nombre des attributs qui caractérise les instances de la base de données. Afin d'améliorer la précision de la classification et réduire les temps de calcul, plusieurs méthodes de réduction de dimensions ont été conçues dans ce but. Dans le cadre de notre travail nous proposons une réduction de la dimensionnalité des données par un réseau de neurones profonds de type auto-encodeur empilé.

Les auto-encodeurs sont des réseaux de neurones composés de trois types de couches :

1. La couche des entrées

2. Les couches cachées
3. La couche des sorties

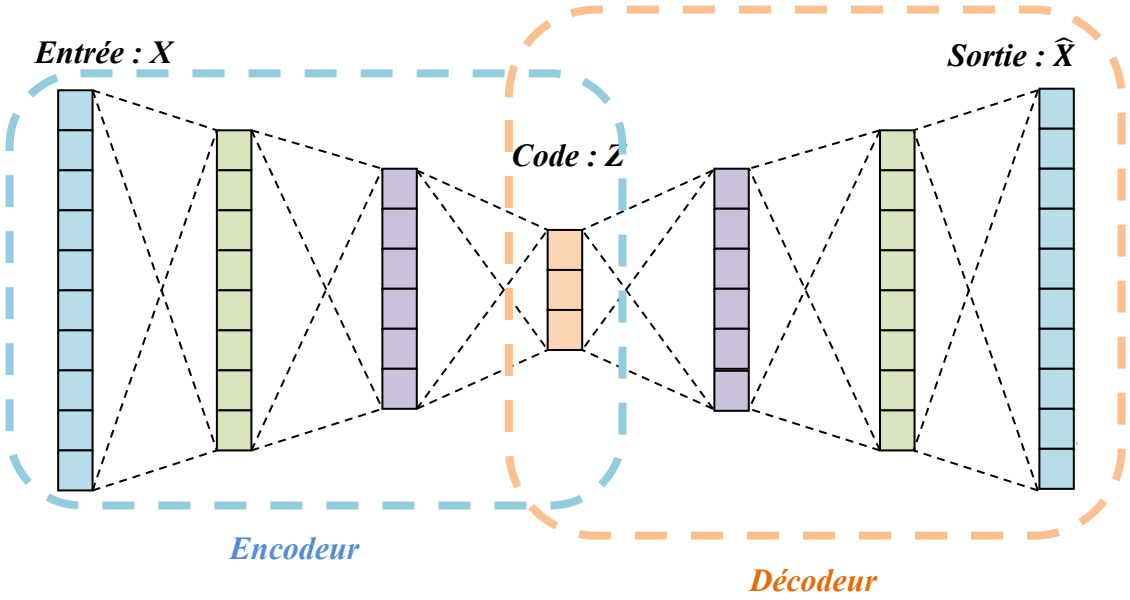


Figure IV.2. *Auto-encodeur avec plusieurs couches cachées*

Les couches cachées relient la couche des entrées avec celle des sorties. Elles sont empilées l'une après l'autre pour former l'encodeur avec la couche des entrées, et le décodeur avec la couche des sorties.

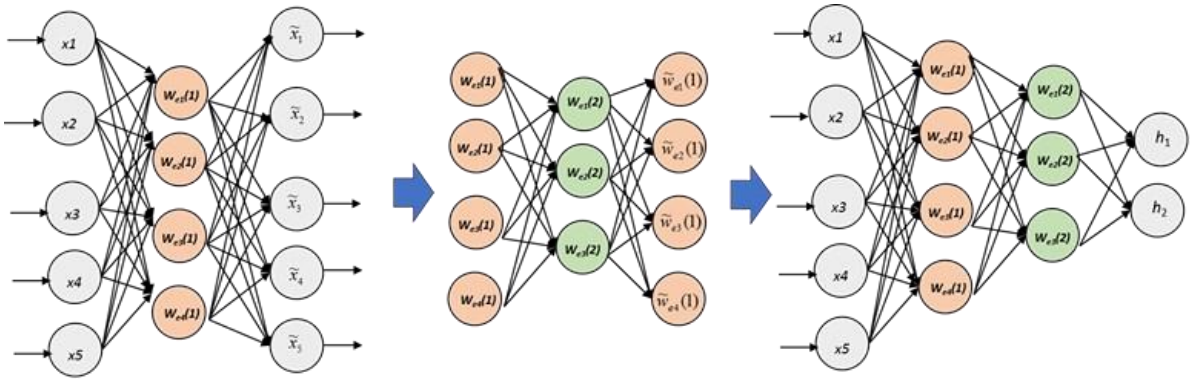
Afin d'extraire les attributs les plus forts et les plus importants pour une bonne classification, l'auto-encodeur passe par trois phases principales :

IV.3.1. L'encodage

L'étape d'encodage se passe, comme noté précédemment, entre la couche d'entrée ***X*** et les couches cachées jusqu'à la couche du code ***Z***. Les nœuds de la première couche ne sont d'autres que les attributs initiaux de notre base de données qui seront combinés aléatoirement pour produire de nouveaux attributs avec peut-être plus de précision. Le nombre de neurones cachés a été initialisé au début de l'apprentissage pour chacune des couches cachées.

Dans notre architecture ont a opté pour trois couches cachées, pour l'étape d'encodage des attributs. Les poids seront mis automatiquement a fur et à mesure de l'apprentissage, couche par couche et non pas d'un seul coup dès le début.

Il faut préciser que la dernière couche cachée de cette étape est la première pour l'étape suivante c'est à dire le décodage.



Etape1: Couche 1 de préapprentissage **Etape2 :** Couche 2 de préapprentissage **Etape3 :** Ajustement fin du model

Figure IV.3. *Illustration de l'encodage avec trois couches cachées*

L'encodeur calcule à partir de $\boldsymbol{X}$, le vecteur $\boldsymbol{h}$ de taille $\boldsymbol{m}$ qui est le nombre de neurones cachés tel que :

$$\boldsymbol{h} = !\left(\boldsymbol{W^{(1)}X} + \boldsymbol{b^{(1)}}\right) \quad \textbf{(IV.1)}$$

où :

- $\boldsymbol{W^{(1)}}$ est une matrice de taille *m*n*
- $\boldsymbol{b^{(1)}}$ un vecteur de biais de taille *m*
- ! (.) est une fonction d'activation de type sigmoïde hyperbolique définie comme suit:

$$\boldsymbol{f(y) = 1/\left(1 - exp^{(-x)}\right)} \quad \textbf{(IV.2)}$$

IV.3.2. Le décodage

Le décodage se fait entre les couches cachées (à partir de la couche code $\boldsymbol{Z}$) et la couche de sortie $\overline{\boldsymbol{X}}$ qui possède le même nombre de nœuds que la couche d'entrée $\boldsymbol{X}$, les couches cachées dans cette étapes sont symétriques aux couches cachées de l'étape précédente avec une en commun (celle du milieu c'est-à-dire la couche code $\boldsymbol{Z}$).

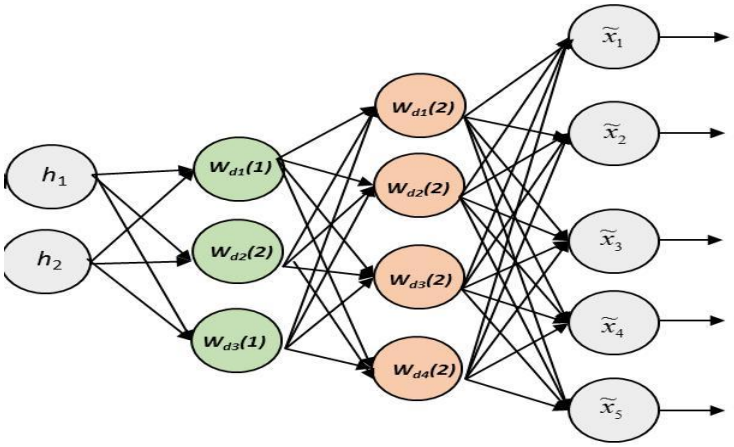


Figure IV.4. *Illustration de décodage du réseau précédent*

L'auto-encodeur empilé tente d'avoir à partir de la couche caché du milieu la représentation la plus proche possible des attributs initiaux (ceux de la couche d'entrée $\boldsymbol{X}$), en minimisant la fonction de perte *loss* (dans notre cas nous avant choisi une fonction *loss* égale à la erreur quadratique moyenne *MSE*).

$$\boldsymbol{loss = MSE = \frac{1}{n}\sum_{i=1}^{n}(X_i - \widehat{X}_i)^2} \quad \textbf{(IV.3)}$$

Le décodeur cherche à reconstruire le vecteur $\boldsymbol{X}$ à partir de la couche cachée $\boldsymbol{h}$, le résultat de cette reconstruction est le vecteur $\widehat{\boldsymbol{X}}$ tel que :

$$\widehat{\boldsymbol{X}} = \left(\boldsymbol{W^{(2)}h + b^{(2)}}\right) \quad \textbf{(IV.4)}$$

où :

- $\widehat{\boldsymbol{X}}$ est un vecteur de taille *m*
- $\boldsymbol{W^{(2)}}$ est une matrice de poids de taille *n*m*
- $\boldsymbol{b^{(2)}}$ est un vecteur de biais de taille n.

IV.3.3. La reconstruction

Une erreur quadratique est calculée entre les nœuds de la couche d'entrée et ceux de la couche de sortie à la fin du décodage et ceci afin de trouver l'erreur minimale entre elles, pour bien choisir l'ensemble de nœuds le plus représentatif réduit de l'ensemble initial.

Cette ensemble n'est d'autre que celui de la couche cachée qui se trouve au milieu du grand réseau de neurone c'est à dire la fameuse couche cachée commune entre l'étape d'encodage et celle du décodage.

Durant l'apprentissage, l'auto-encodeur tente de réduire une erreur de reconstruction entre $\boldsymbol{X}$ et $\widehat{\boldsymbol{X}}$. Pour cela, il utilise l'erreur quadratique moyenne de manière à minimiser l'erreur de reconstruction totale avec l'ensemble de paramètres$\{\boldsymbol{W^{(1)}}, \boldsymbol{b^{(1)}}, \boldsymbol{W^{(2)}}, \boldsymbol{b^{(2)}}\}$.

Plusieurs essais ont été effectués sur les multiples base de tests afin de trouver la meilleure représentation réduite, les essais variaient entre le changement de nombre de nœuds dans les couches cachées ou bien en changeant carrément le nombre de couche cachée utilisée, car nous pouvons bien évidemment les changer selon notre besoin pour améliorer les résultats.

IV.4. Classification des données

Après la réduction, d'une manière remarquable, du nombre d'attributs (*features*) de la base de données originale, il est temps maintenant de prédire les classes des nouvelles instances, pour cela nous avons choisi d'appliquer la prédiction des classes en utilisant un réseau de neurones convolutionnel (*CNN*).

IV.4.1. Méthode de réseau de neurones convolutionnel (CNN)

Le réseau de neurones convolutionnel (*CNN*) est une variante du réseau de neurones, dont le but est d'apprendre des représentations de caractéristiques appropriées des données d'entrée. Un réseau CNN présente deux différences principales avec les réseaux MLP, notamment le partage du poids (*weight sharing*) et la mise en commun (*pooling*). Chaque couche du réseau CNN peut être composée de nombreux noyaux de convolution utilisés pour générer différentes cartes de caractéristiques (*feature map*). Chaque région de neurones voisins est connectée à un neurone de la carte de caractéristiques de la couche suivante. De plus, pour générer la carte de caractéristiques, tous les emplacements spatiaux de l'entrée partagent le noyau. Après quelques couches de convolution et de regroupement, une ou plusieurs couches entièrement connectées sont utilisées pour la classification. [17]

En raison de l'utilisation de poids partagés dans le CNN, le modèle peut apprendre le même motif (*pattern*) se produisant à différentes positions des entrées, sans nécessiter l'apprentissage de détecteurs séparés pour chaque position. Par conséquent, le modèle peut être robuste à la traduction des entrées.

Les couches de regroupement (*pooling layers*) réduisent la charge de calcul, car elles réduisent le nombre de connexions entre les couches de convolution. De plus, les couches de

regroupement augmentent les propriétés d'invariance de la traduction et améliorent le champ de réception des couches convolutives suivantes. [17]

Généralement, une ou plusieurs couches entièrement connectées (*fully connected layers*) sont ajoutées à la fin du flux de convolution du réseau, et une fonction de perte (*loss function*) est utilisée pour mesurer les erreurs à des fins d'apprentissage.

A chaque couche CNN, un ensemble de $\boldsymbol{n}$ noyaux (*kernel*) $\boldsymbol{W} = \{\boldsymbol{w_1}, \boldsymbol{w_2}, \dots, \boldsymbol{w_n}\}$ et leurs biais $\boldsymbol{B} = \{\boldsymbol{b_1}, \boldsymbol{b_2}, \dots, \boldsymbol{b_n}\}$, est convolutionné avec les données d'entrée. La convolution entre les données et chaque noyau produit une nouvelle carte de fonctionnalités $\boldsymbol{x_k}$. Pour chaque couche convolutive $\boldsymbol{l}$, la transformation est définie par: [17]

$$\boldsymbol{x_k^l = \sigma\left(w_k^{l-1} * x^{l-1} + b_k^{l-1}\right)} \quad \textbf{(IV.5)}$$

Au cours du processus d'apprentissage de CNN, une petite fenêtre est glissée sur les entrées et les valeurs de biais et de pondération via cette fenêtre peuvent être optimisées à partir de diverses caractéristiques des données d'entrée sans leur position dans les données d'entrée.

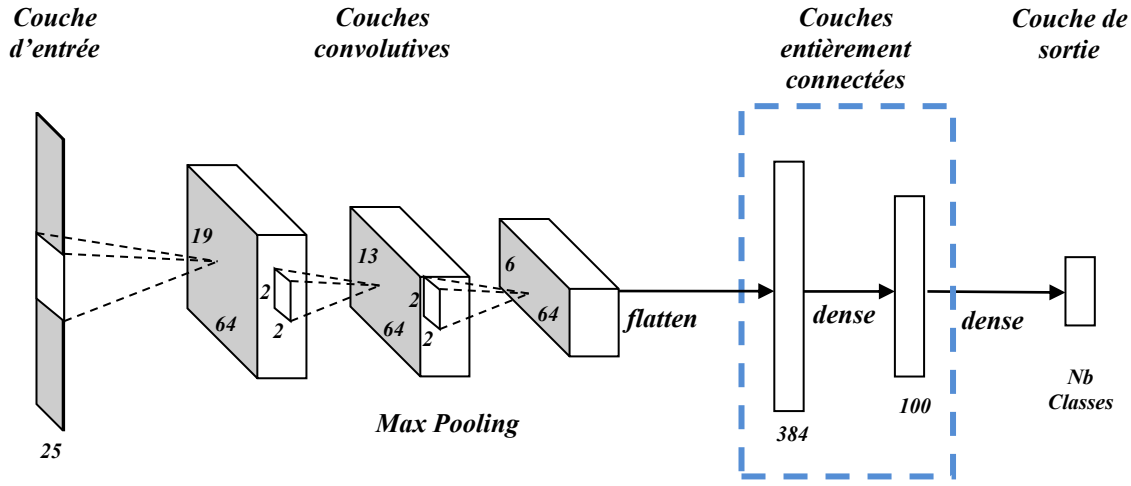


Figure IV.5. *Illustration du réseau convolutionnel (CNN 1D) adopté*

Après avoir choisi la meilleure représentation réduite des données de la base initiale, l'algorithme de classification ne prendra que le résultat de la dernière couche $\boldsymbol{Z}$ de l'encodeur. Le type de fonction d'activation adéquat à notre problème de classification est bien le type "non linéaire" et nous avons choisi la fonction "*softmax*" qui quand elle est utilisée, la somme des sorties doit être égale à 1 ce qui signifie que la procédure donnera en fin d'exécution 1 pour un des nœuds et forcément 0 pour le reste des nœuds, le nœuds prenant 1 est donc la classe prédite pour cette série d'attributs. Une évaluation de performance de ce système sera

effectuée afin de juger le taux de précision des classes prédites par rapport aux classes originales.

IV.5. Conclusion

Dans ce chapitre, nous avons présenté en détail toutes les étapes de la méthode proposée dans ce travail afin de réduire la dimension des différentes bases de tests, et pour pouvoir mieux classifier ou prédire les classes des nouvelles instances issues de la réduction de dimensionnalité.

Chapitre V : **TESTS ET RESULTATS**

"الحكم السليم هو نتيجة التجربة..
والتجربة تكون ناتجة عن حكم خاطئ."

V.1. Introduction

Après une étude détaillée des IDS et des différentes méthodes d'apprentissage profond, et après avoir expliqué en détail la conception de notre choix pour la réalisation d'une approche basée sur l'apprentissage profond, nous sommes enfin arrivés à la phase finale de notre travail qui consister à implémenter les différentes phases de notre architecture et tous les modules et bibliothèques nécessaires au bon fonctionnement de notre modèle, et enfin, les résultats obtenus en appliquant les méthodes déjà citées dans le chapitre précédent.

V.2. Matériel et logiciels utilisés

Pour mener à bien notre travail, nous avons choisi de travailler dans un environnement expérimental caractérisé par un PC doté de 8 Go de RAM, 1 CPU i3-3110M cadencé à 2.40GHz, tournant sous *Windows 10* 64 bits. En ce qui concerne le coté logiciel, notre développement est fait sur *Anaconda* qui est doté de *Python* 3.7 ainsi que l'éditeur *Spyder* 3.3.4 qui a été choisi pour sa simplicité et efficacité. Plusieurs bibliothèques ont été utilisées : *Tensorflow/keras, scipy, numpy, pandas, matplotlib, sklearn*.

Tableau V.1. *Tableau des bibliothèques python utilisées*

Bibliothèque	**Version utilisée**
keras	2.2.4
Tensorflow	1.13.1
matplolib	3.0.3
numpy	1.16.3
pandas	0.24.2
scipy	1.2.1
sklearn	0.21.1
System (python)	3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)]

Parmi toutes ces bibliothèques utilisées dans ce travail, la plus importante est bien *Tensorflow*. Cette dernière est une bibliothèque de logiciels open source destinée au calcul numérique de haute performance. Son architecture flexible permet un déploiement facile du calcul sur diverses plates-formes (processeurs *CPU*, processeurs graphiques *GPU*, processeurs tenseurs *TPU*), et des ordinateurs de bureau aux clusters de serveurs, aux

périphériques mobiles et périphériques. La bibliothèque *Tensorflow*[5] a été développée à l'origine par des chercheurs et des ingénieurs de l'équipe *Google Brain* au sein de l'organisation *Google AI*, elle prend en charge l'apprentissage automatique et l'apprentissage profond et le cœur de calcul numérique flexible est utilisé dans de nombreux autres domaines scientifiques

V.3. Description de la base de données utilisée

Nous allons présenter dans ce qui suit les caractéristiques des jeux de données utilisées dans la phase expérimentale. Pour accomplir notre travail, nous avons choisi de travailler avec l'ensemble de données (*dataset*) le plus récent qui est nommé *CICIDS2017*[6]. Ce dataset couvre l'ensemble des onze critères nécessaires avec des attaques mises à jour communes telles que le déni de service (*DoS*), le DoS distribué (*DDoS*), la force brute, le script inter-site (*XSS*), l'injection SQL, l'infiltration, Balayage de port et *botnet*. L'ensemble de données est entièrement étiqueté et plus de 78 entités de trafic réseau sont extraites et calculées pour tous les flux bénins et intrusifs à l'aide du logiciel *CICFlowMeter*, disponible sur le site Web de l'Institut canadien de la cyber-sécurité[7], ainsi qu'un 79ème et dernier attribut qui décrit le type du paquet (bénin ou attaque avec le nom explicite de l'intrusion).

Tableau V.2. *Tableau des caractéristiques physiques de la base CICIDS2017*

Journée	Fichiers	Taille (KO)	Nb. Instances	Activités/Attaques
Lundi	1	172.782	529.918	• Activité normale (bénine)
Mardi	1	131.914	445.909	• Activité normale (bénine) • FTP-Patator • SSH-Patator
Mercredi	1	220.558	692.703	• Activité normale (bénine) • DoS slowloris • DoS slowhttptest • DoS Hulk • DoS GoldenEye • Heartbleed
Jeudi	2	131.958	458.968	• Activité normale (bénine) • Web Attack-Brute Force • Web Attack-XSS

[5] Définition de tensorflow consulté le 14 juin 2019 récupérée de [https://www.tensorflow.org/]
[6] https://www.unb.ca/cic/datasets/ids-2017.html.
[7] https://www.unb.ca

				• Web Attack-SQL Injection • Infiltration
Vendredi	3	207.369	703.245	• Activité normale (bénine) • Botnet ARES • Port Scan • DDoS LOIT
Global	**8**	**864.580**	**2.830.743**	• **Activité normale (bénine) +** • **14 types d'attaques**

Contrairement aux jeux de données existants tels que les jeux de données *DARPA*, *KDD Cup 99* et *CAIDA*, le jeu de données *CICIDS2017* fournit 14 méthodes d'attaque différentes auxquelles les techniques de détection d'attaque peuvent être appliquées. En particulier, il inclut divers types d'attaques *DoS* qui épuisent les ressources du serveur, telles que *DoS Hulk*, *GoldenEye*, *Slowloris* et *Slowhttptest*, ainsi qu'une analyse de port, qui est une attaque réseau typique, et d'attaques brutales visant à capturer une autorité tels que *FTP-Patator* et *SSH-Patator*. [19]

La base de données *CICIDS2017* est composée de 78 attributs et 2830743 enregistrements, dont 80% sont utilisés pour l'apprentissage (*training set*), 20% pour les tests (*testing set*), que ce soit dans la partie réduction de dimensionnalité ou classification. Cette base est contenue dans 7 fichiers au format CSV et représente la capture du trafic réseau du lundi 3 juillet 2017 à 9 h du matin jusqu'au jeudi 7 juillet 2017 à 17h. Le tableau V.2 résume les caractéristiques de la base *CICIDS2017*.

V.4. Correction des données manquantes

Lors de l'utilisation de la base CICIDS2017, nous avons constaté qu'elle contient quelques données obsolètes qui peuvent nuire au bon fonctionnement de notre système. Cette lacune réside dans la présence des valeurs vide, *NaN* et *Inf* dans les différents fichiers de la base de données. Ces deux valeurs empêchent l'apprentissage des deux réseaux profond implémentés. De ce fait, une étape de correction de ces valeurs s'impose, et pour cela, nous avons développé un algorithme qui permet le remplacement l'absence d'un nombre *NaN* par la valeur 0, l'infini *Inf* par la valeur maximale et les vides par la valeur moyenne de l'attribut en question. Pour la valeur *Inf* nous avons tenté de la remplacer par la valeur moyenne de l'attribut mais les résultats obtenus n'étaient pas convainquant, alors nous avons opté pour le choix de la valeur maximale. Avant la sauvegarde de la copie corrigée des données, toutes les

valeurs numériques de la base de données sont converties en *float64* pour un calcul plus précis.

V.5. Réduction de la dimensionnalité

Comme nous l'avons déjà mentionné dans le chapitre précédent, nous avons choisi un auto-encodeur empilé (*SAE*) comme module de réduction de la dimensionnalité non supervisée. Plusieurs tests ont été effectués pour trouver la bonne valeur optimale du nombre de couches cachées, afin de définir modèle optimal pour notre auto-encodeur empilé, la valeur retenue était **2** couches cachés entre la couche des entrées $\boldsymbol{X}$ qui est un vecteur de taille **78**, et la couche du code $\boldsymbol{Z}$ dont la taille est **25**. L'apprentissage de notre auto-encodeur a été fixé à 20 époques (*epochs*) et un lot (*batch*) de 320 (après plusieurs tests d'évaluation). Le tableau suivant résume les caractéristiques de notre modèle SAE :

Tableau V.3. *Tableau des caractéristiques de l'auto-encodeur empilé SAE*

		Layer	(type)	Output Shape	Param #
Encodage		input_1	(InputLayer)	(None, 78)	0
		dense_1	(Dense)	(None, 64)	5056
		dense_2	(Dense)	(None, 50)	3250
	Décodage	dense_3	(Dense)	(None, 25)	1275
		dense_4	(Dense)	(None, 50)	1300
		dense_5	(Dense)	(None, 64)	3264
		dense_6	(Dense)	(None, 78)	5070
Total params: 19,215 Trainable params: 19,215 Non-trainable params: 0					

V.5.1. Paramètre d'évaluation de l'auto-encodeur

L'auto-encodeur empilé a été choisi comme module de réduction de la dimensionnalité, son principe de base est de fournir un vecteur de sortie $\widehat{\boldsymbol{X}}$ « presque » identique au vecteur d'entrée $\boldsymbol{X}$. De ce fait, le meilleur paramètre d'évaluation du modèle est de minimiser la fonction l*oss* qui est égale à la *MSE*.

V.5.2. Résultats de la réduction de la dimensionnalité

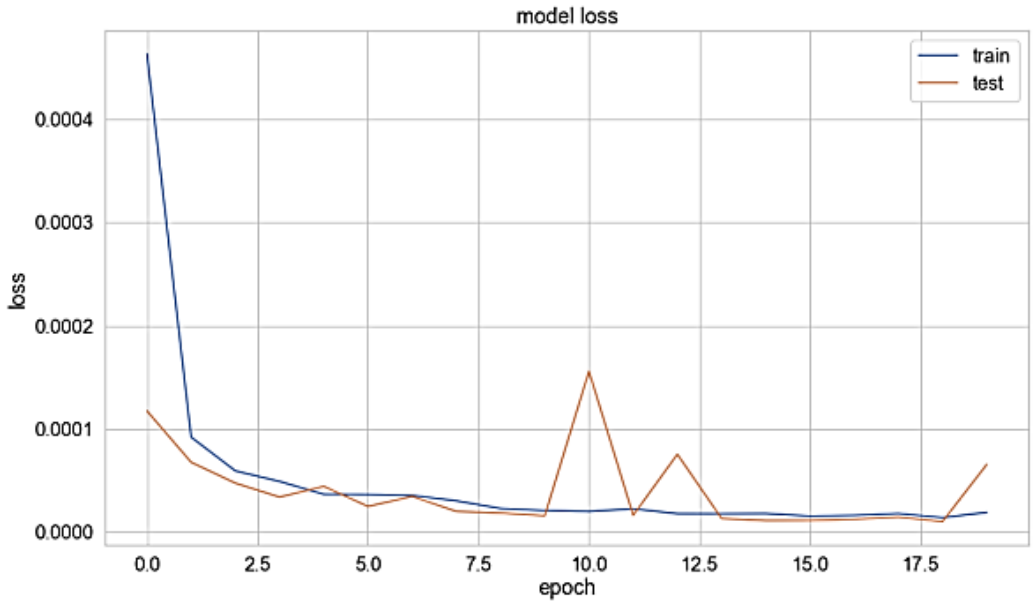
Le SAE proposé nous a permis de réduire le nombre d'attributs de 78 à 25 (c'est-à-dire un facteur de compression égal à 1:3.12). Le tableau suivant donne les résultats obtenus en appliquant les étapes précédentes sur les différents fichiers de la base de données:

Tableau V.4. *Tableau des résultats de la réduction de la dimensionnalité*

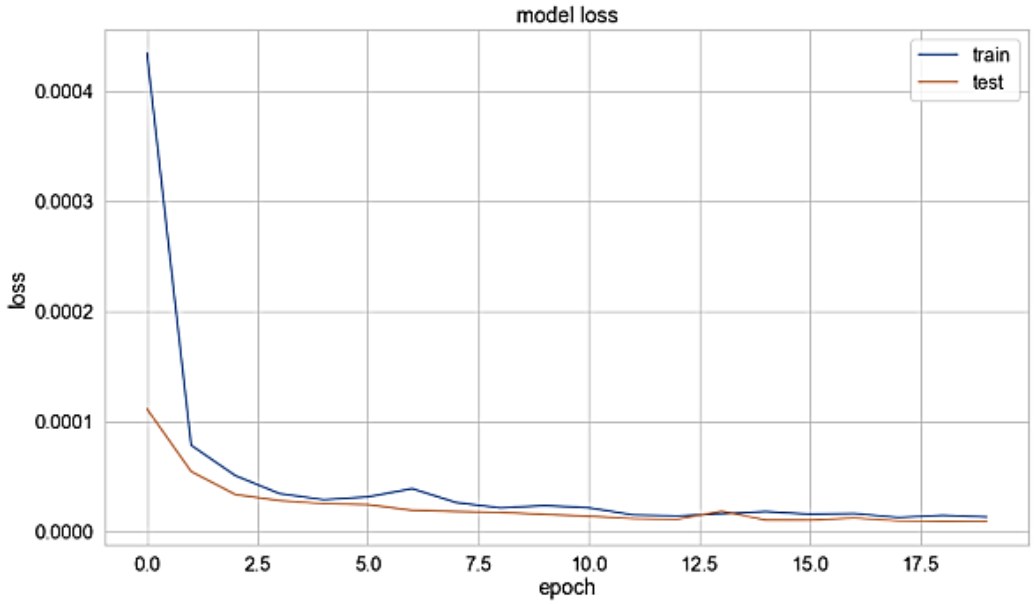
Fichier	**Nb. Instances**	**Train**		**Test**
		Temps	**Loss %**	**Loss %**
Lundi	529.918	0:08:39.18	0. 00101	0. 00101
Mardi	445.909	0:07:36.79	0. 00088	0. 00094
Mercredi	692.703	0:06:38.59	0. 00086	0. 00094
Jeudi	458.968	0:09:48.43	0. 00069	0. 00076
Vendredi (1^{er} fichier)	225.745	0:03:56.07	0. 00122	0. 00127
Vendredi	703.245	0:15:45.61	0. 00066	0. 00061
Global	**2.830.743**	**0:46:14.55**	**0. 00040**	**0. 00039**

V.5.3. Analyse des résultats de la réduction de la dimensionnalité

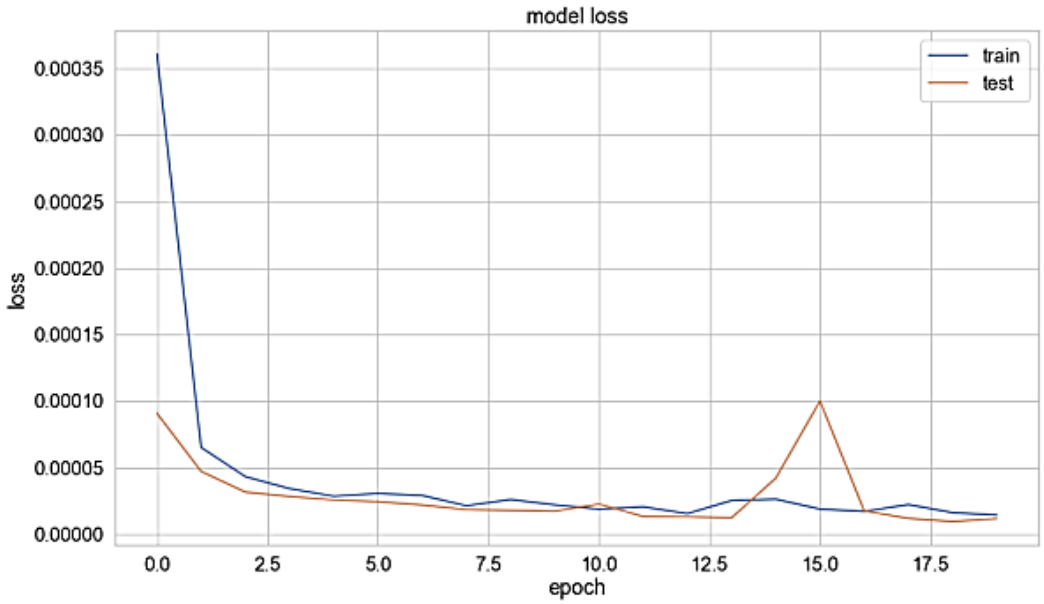
Comme nous pouvons le constater, l'auto-encodeur empilé proposé a réussi son objectif qui est la réduction de la dimensionnalité de notre base de données de 78 attributs à seulement 25, tout en gardant un taux de perte nettement réduit (de l'ordre de 10^{-5} à 10^{-6}). La figure suivante présente le graphe de la fonction *loss* pour les différents fichiers de la base de données ainsi que la base globale, pour un nombre d'époques égal à 20 et ceci avec les deux ensembles de données: apprentissage et test (*train* et *test*).


model loss
train
test
loss
epoch
0.0004
0.0003
0.0002
0.0001
0.0000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

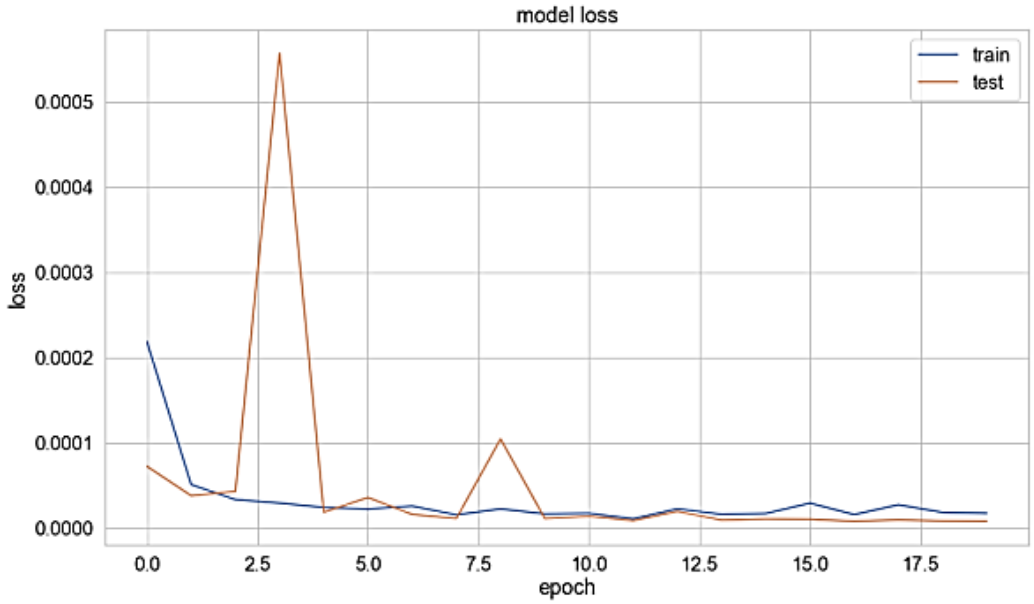
Lundi


model loss
train
test
loss
epoch
0.0004
0.0003
0.0002
0.0001
0.0000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

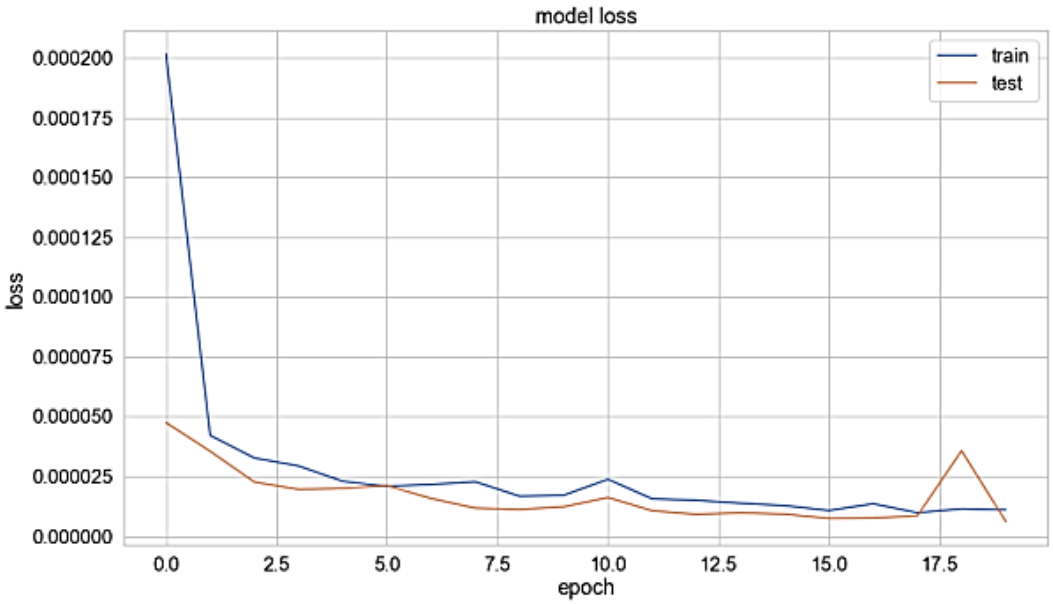
Mardi



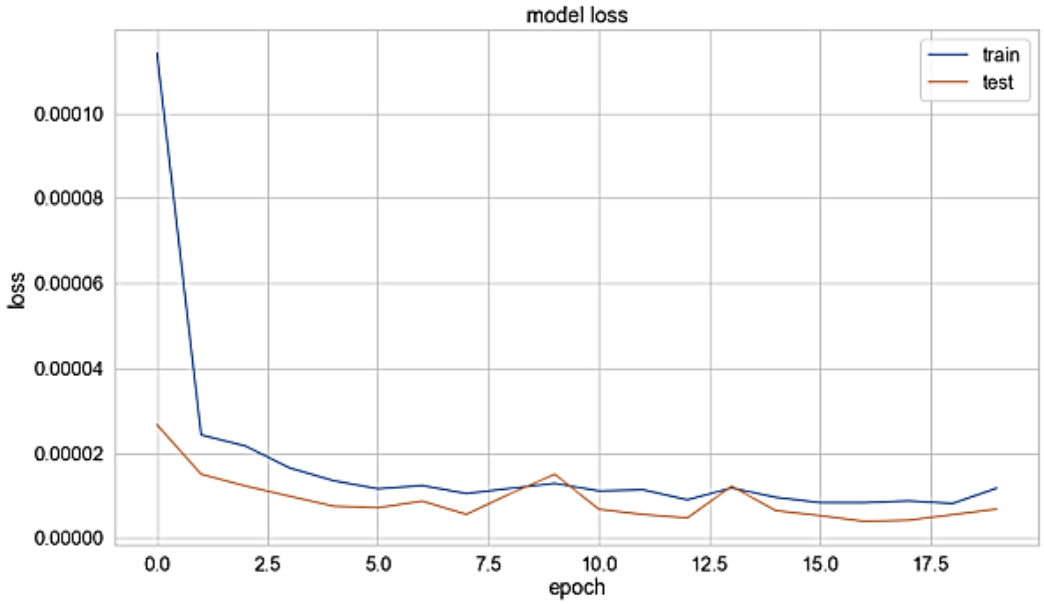
Mercredi



Jeudi



Vendredi



Global

Figure V.1. *Graphes de la fonction loss du SAE avec les différents fichiers de tests*

V.6. Classification

Pour la classification, nous avons choisi un réseau de neurones convolutionnel (*CNN*) comme module de classification supervisée. Plusieurs tests ont été effectués pour choisir la bonne valeur optimale du nombre de couches du réseau. L'apprentissage de notre réseau *CNN*

a été fixé à 5 époques (*epochs*), un lot (*batch*) de 100 instances, 7 pour la taille de la fenêtre de convolution 1D (*kernel size*), 64 pour le nombre de filtres de sortie dans la convolution (après plusieurs tests d'évaluation) et la valeur de 2 pour la taille du pool de la couche du *max pooling*. La fonction d'activation choisie pour les couches de convolution et *ReLU* et pour de la dernière couche est *softmax* (déjà décrite dans le chapitre précédent). Le nombre de classes de la dernière couche Nb_Classes est une variable dont la valeur diffère d'un jeu de tests à un autre suivant le dataset utilisé. Le tableau suivant résume les caractéristiques de notre modèle *CNN*:

Tableau V.5. *Tableau des caractéristiques du réseau de neurones CNN 1D*

Layer	**(type)**	**Output Shape**	**Param #**
conv1d_1	(Conv1D)	(None, 19, 64)	512
conv1d_2	(Conv1D)	(None, 13, 64)	28736
dropout_1	(Dropout)	(None, 13, 64)	0
max_pooling1d_1	(MaxPooling1D)	(None, 6, 64)	0
flatten_1	(Flatten)	(None, 384)	0
dense_1	(Dense)	(None, 100)	38500
dense_2	(Dense)	(None, Nb_Classes)	404
Total params: 68,152 Trainable params: 68,152 Non-trainable params: 0			

V.6.1. Paramètre d'évaluation du réseau CNN

Le réseau CNN choisi comme module de classification, a pour but la classification du vecteur ***Z*** issu de la partie précédente (réduction de la dimensionnalité par *SAE*). De ce fait, le meilleur paramètre d'évaluation du modèle est de maximiser la précision dite *accuracy*. La fonction *loss* choisie pour notre modèle est *categorical_crossentropy* puisque nous avons choisi de faire une classification multi classes et non pas binaire. L'optimiseur qui convient à notre configuration est défini par 0.01 pour le taux d'apprentissage (*learning rate*) et un élan (*momentum*) égal à 0.9. Entre autres, nous avons essayé d'utiliser la matrice de confusion (*confusion matrix*) et la matrice de confusion normalisée (*normalized confusion matrix*) comme paramètre d'évaluation du modèle, ainsi que l'aire sous la courbe *AUC*.

V.6.2. Résultats de la classification

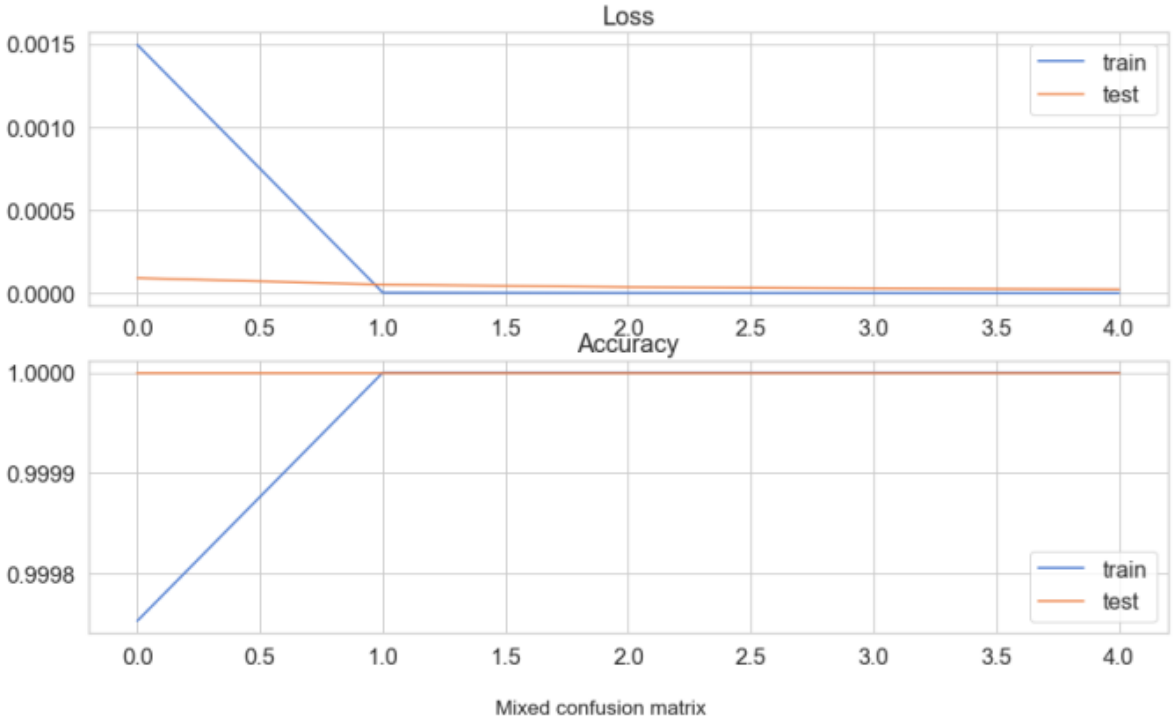
Le réseau CNN proposé nous a permis de faire la classification des instances de la base données avec une précision (*accuracy*) très satisfaisante. Le tableau suivant donne les résultats obtenus en appliquant les étapes précédentes sur les différents fichiers de la base de données:

Tableau V.6. *Tableau des résultats de la réduction de la classification*

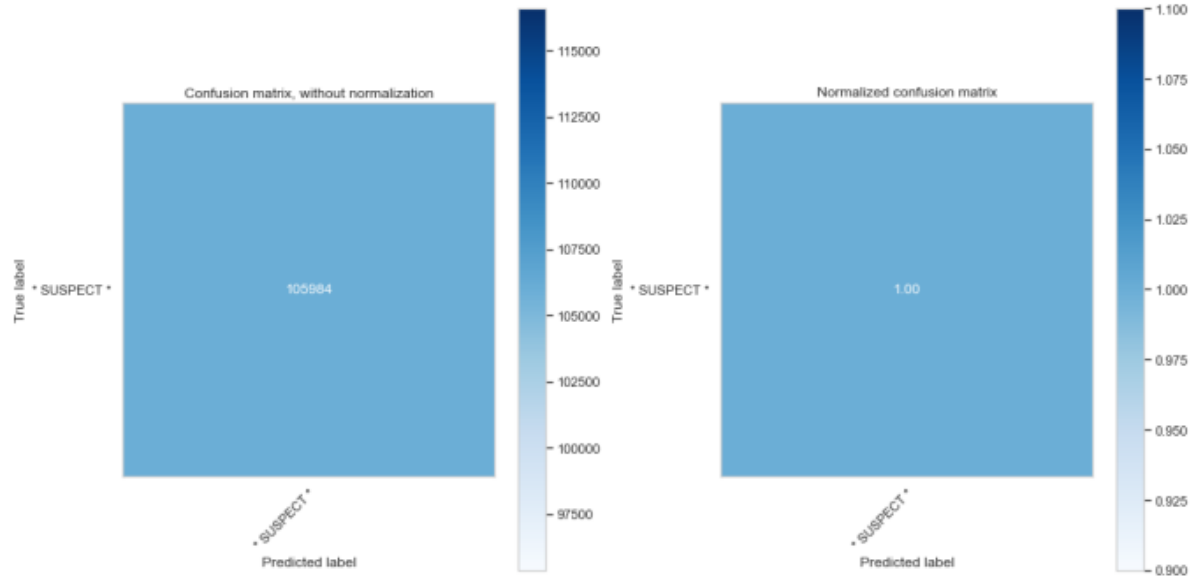
Fichier	Nb. Instances	Train			Test	
		Temps	Accuracy %	Loss %	Accuracy %	Loss %
Lundi	529.918	0:06:33.01	100.0	0.0	100.0	0.0
Mardi	445.909	0:05:31.47	98.5	3.2	98.6	3.1
Mercredi	692.703	0:07:06.09	98.0	5.9	97.9	5.8
Jeudi	458.968	0:06:24.91	99.5	1.5	99.5	1.4
Vendredi (1^{er} fichier)	225.745	0:02:59.43	99.0	2.1	99.0	2.1
Vendredi	703.245	0:08:38.08	99.2	2.7	99.2	2.7
Global	**2.830.743**	**0:38:40.68**	**96.2**	**9.5**	**96.2**	**9.5**

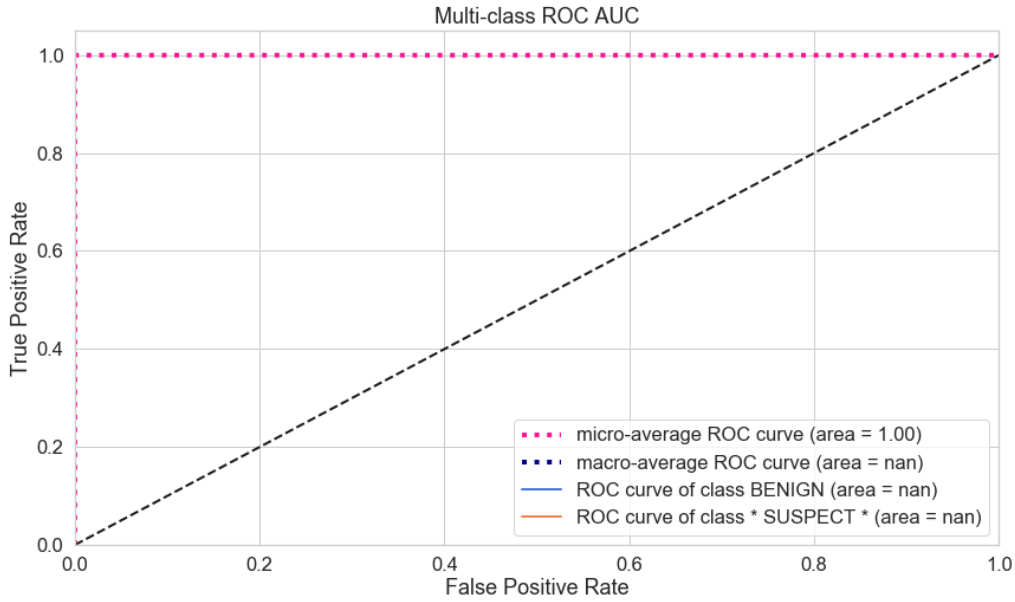
V.6.3. Analyse des résultats de la classification

Comme nous pouvons le constater clairement, notre réseau CNN proposé a réussi son objectif qui est la classification des instances de notre base de données, tout en gardant un taux de précision nettement élevé et une perte réduite. La figure suivante présente le graphe de la fonction *loss* ainsi que l'*accuracy* obtenu pour les différents fichiers de la base de données ainsi que la base globale, pour un nombre d'époques égal à 5 et ceci avec les deux ensembles de données: apprentissage et test (*train* et *test*).

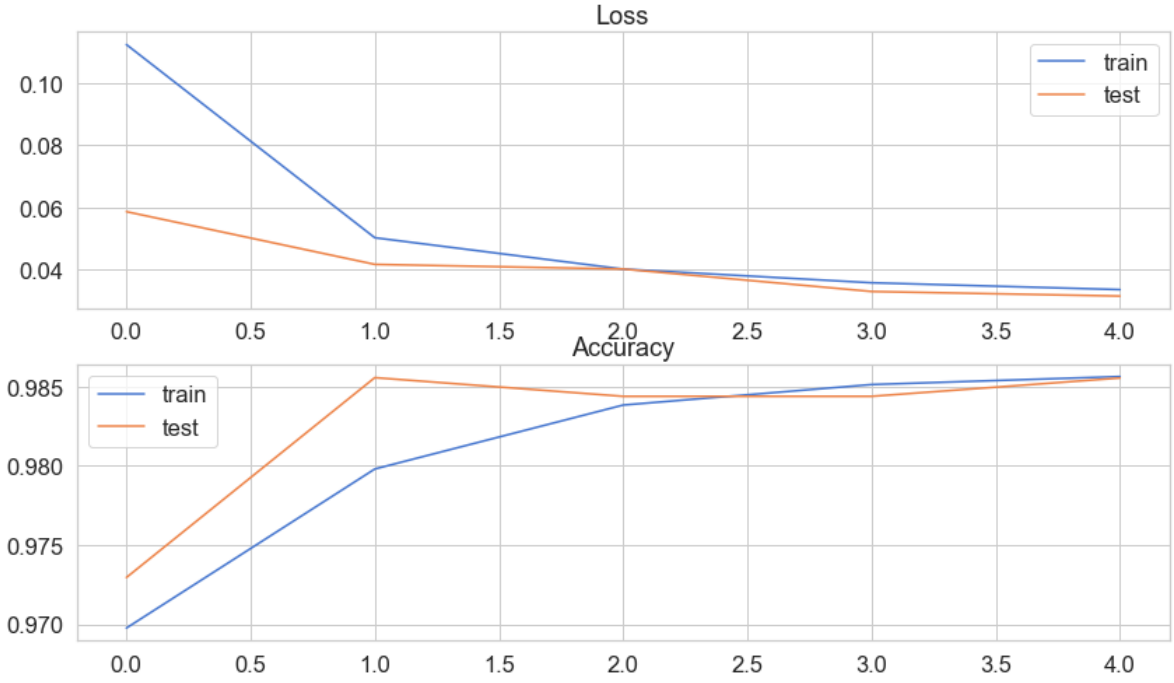


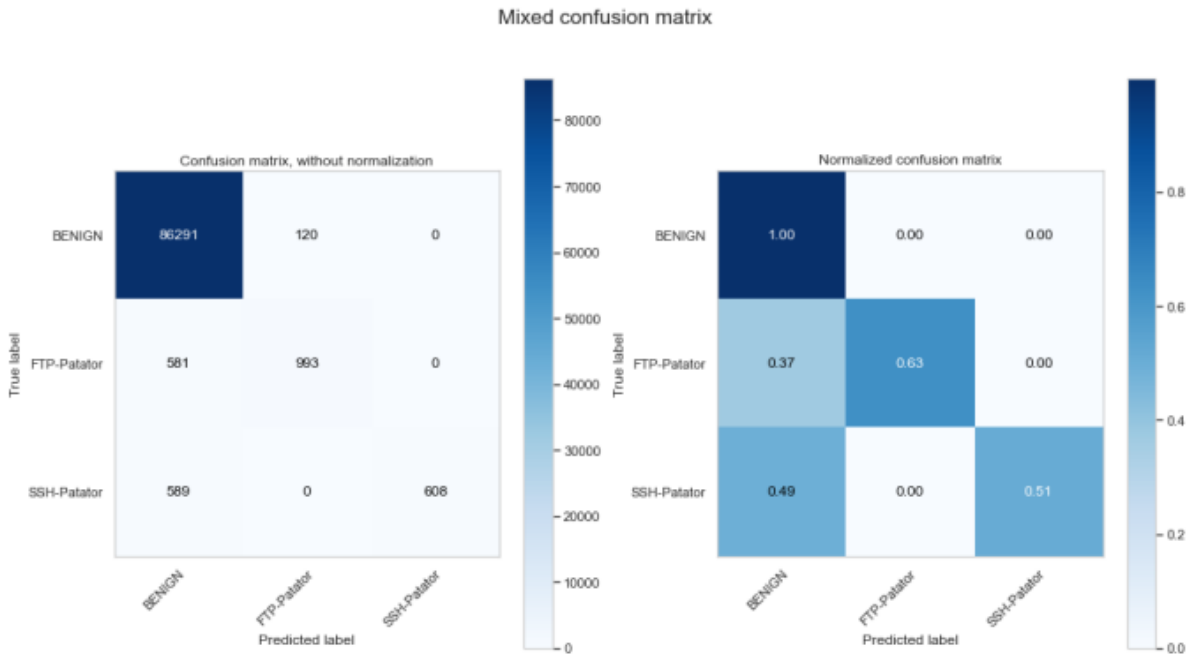
Mixed confusion matrix

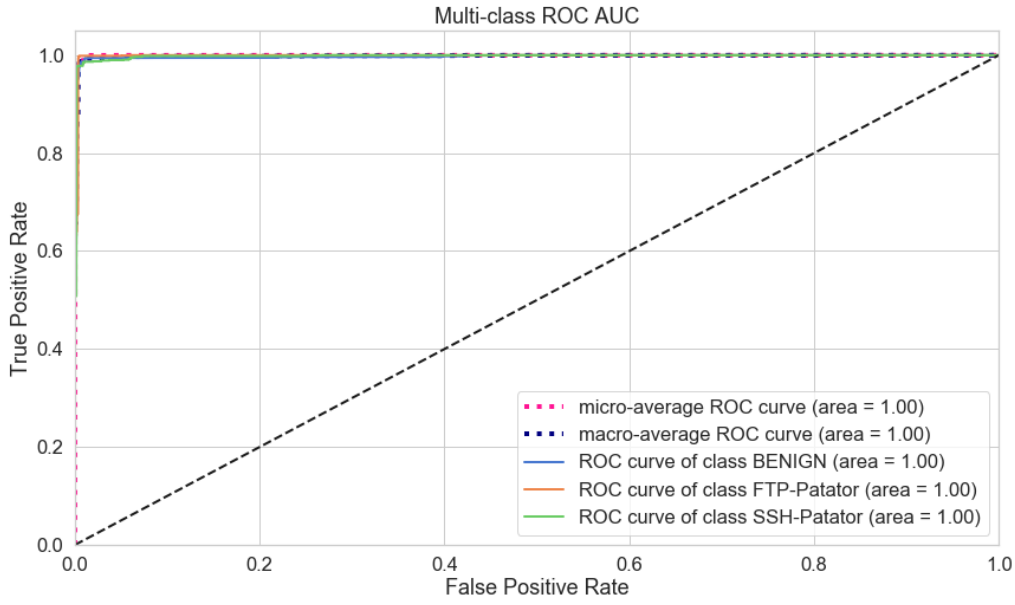



Multi-class ROC AUC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 1.00)
macro-average ROC curve (area = nan)
ROC curve of class BENIGN (area = nan)
ROC curve of class * SUSPECT * (area = nan)

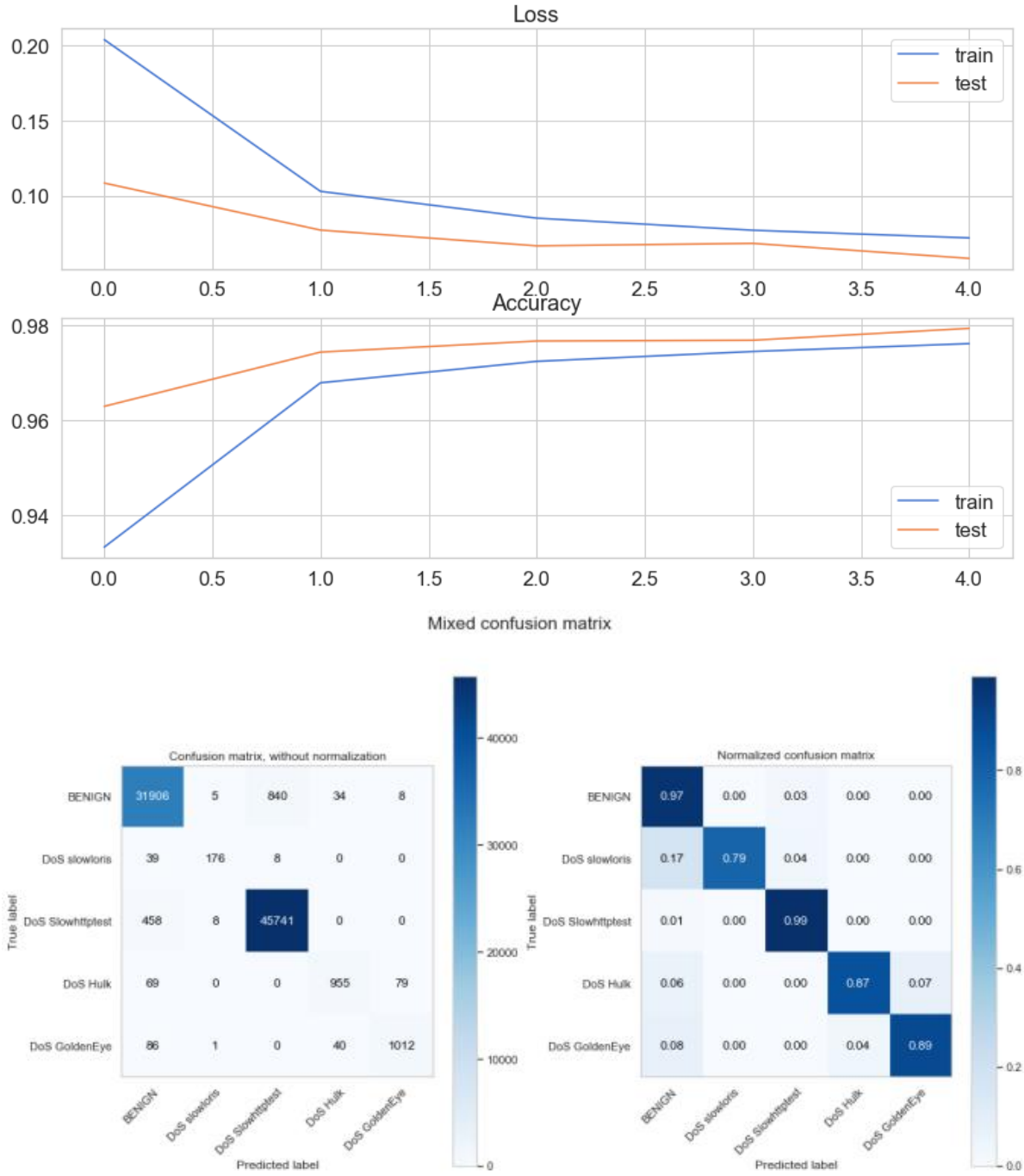
Lundi

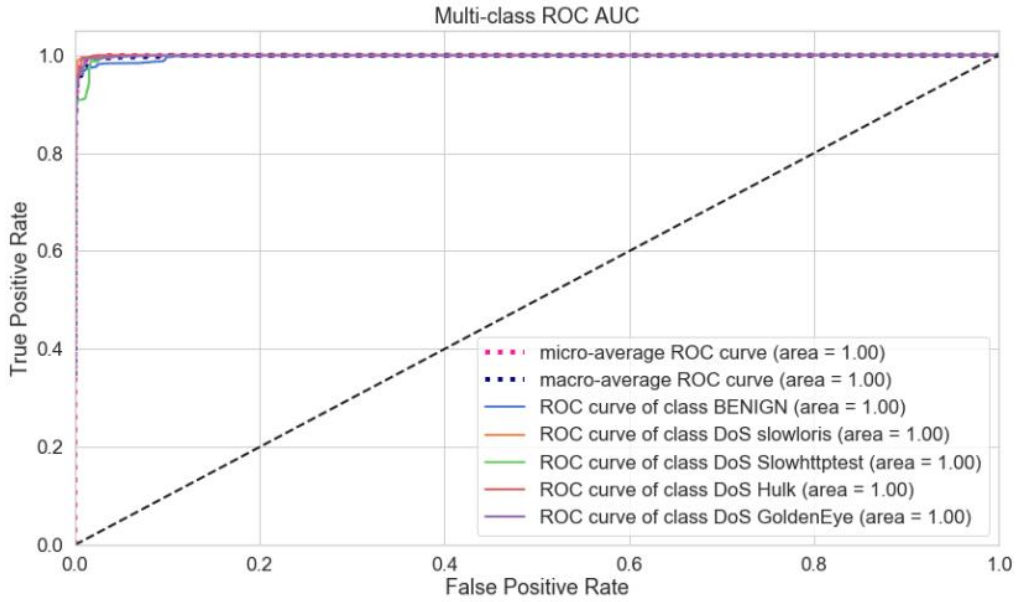

Loss
train
test
0.10
0.08
0.06
0.04
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Accuracy
train
test
0.985
0.980
0.975
0.970


Mixed confusion matrix
Confusion matrix, without normalization
BENIGN
FTP-Patator
SSH-Patator
86291
120
0
581
993
0
589
0
608
True label
Predicted label
Normalized confusion matrix
1.00
0.00
0.00
0.37
0.63
0.00
0.49
0.00
0.51

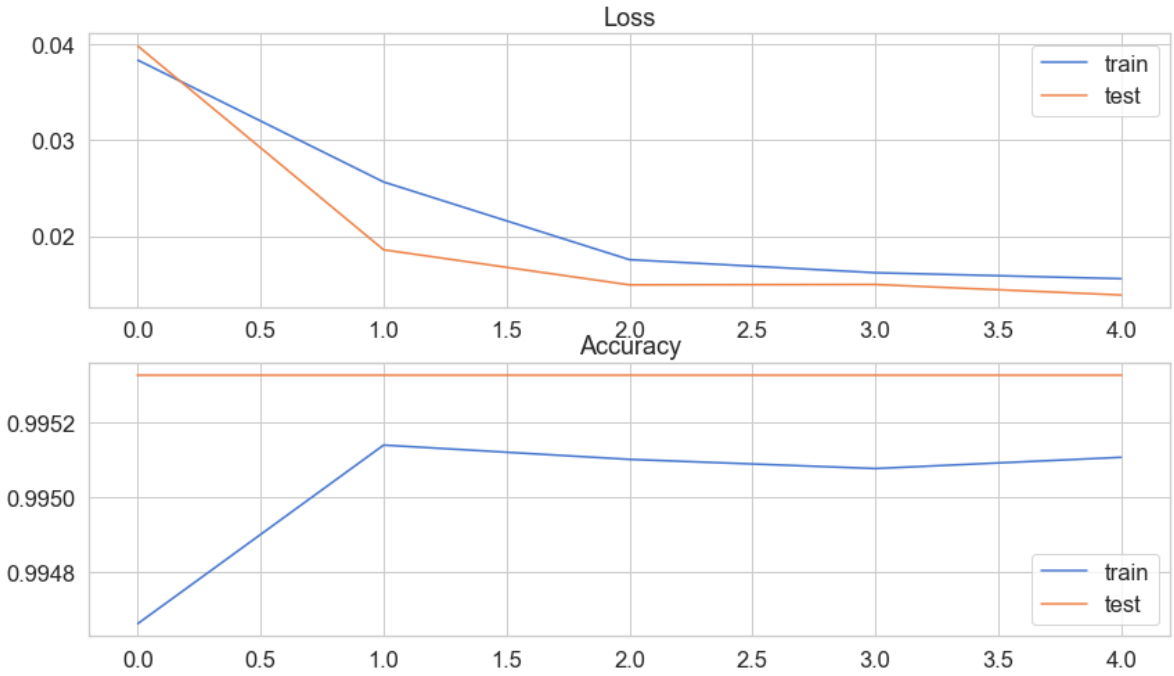

Multi-class ROC AUC
True Positive Rate
False Positive Rate
micro-average ROC curve (area = 1.00)
macro-average ROC curve (area = 1.00)
ROC curve of class BENIGN (area = 1.00)
ROC curve of class FTP-Patator (area = 1.00)
ROC curve of class SSH-Patator (area = 1.00)

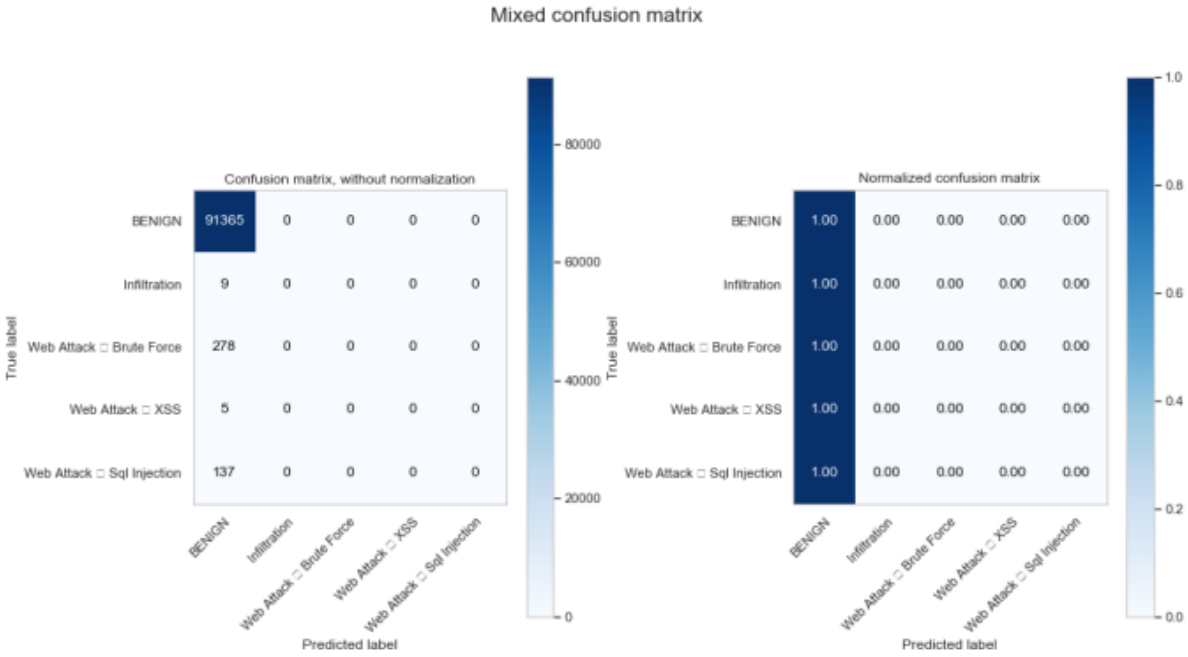
Mardi

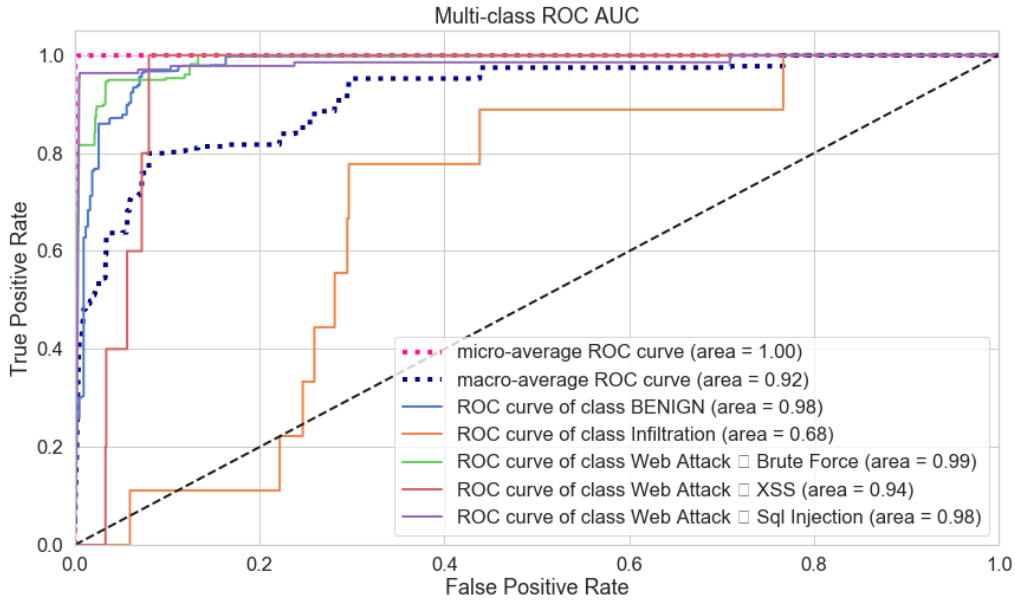

Loss
train
test
Accuracy
train
test
Mixed confusion matrix
Confusion matrix, without normalization
BENIGN
DoS slowloris
DoS Slowhttptest
DoS Hulk
DoS GoldenEye
True label
Predicted label
Normalized confusion matrix


Multi-class ROC AUC
True Positive Rate
False Positive Rate
micro-average ROC curve (area = 1.00)
macro-average ROC curve (area = 1.00)
ROC curve of class BENIGN (area = 1.00)
ROC curve of class DoS slowloris (area = 1.00)
ROC curve of class DoS Slowhttptest (area = 1.00)
ROC curve of class DoS Hulk (area = 1.00)
ROC curve of class DoS GoldenEye (area = 1.00)

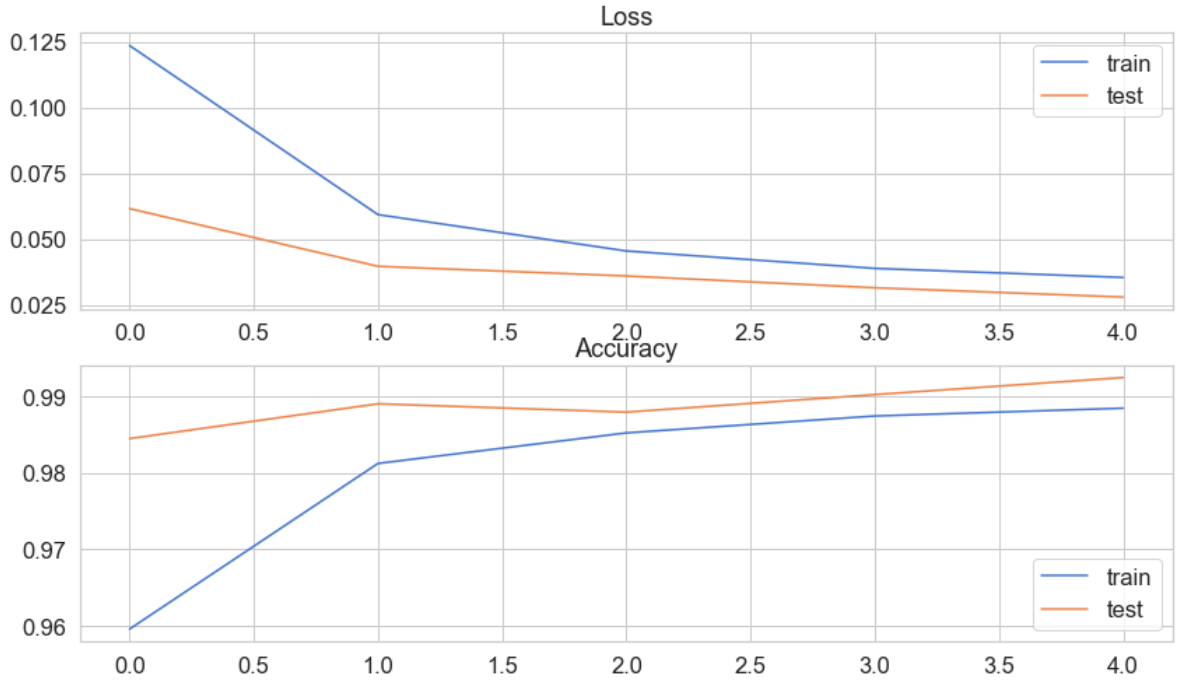
Mercredi

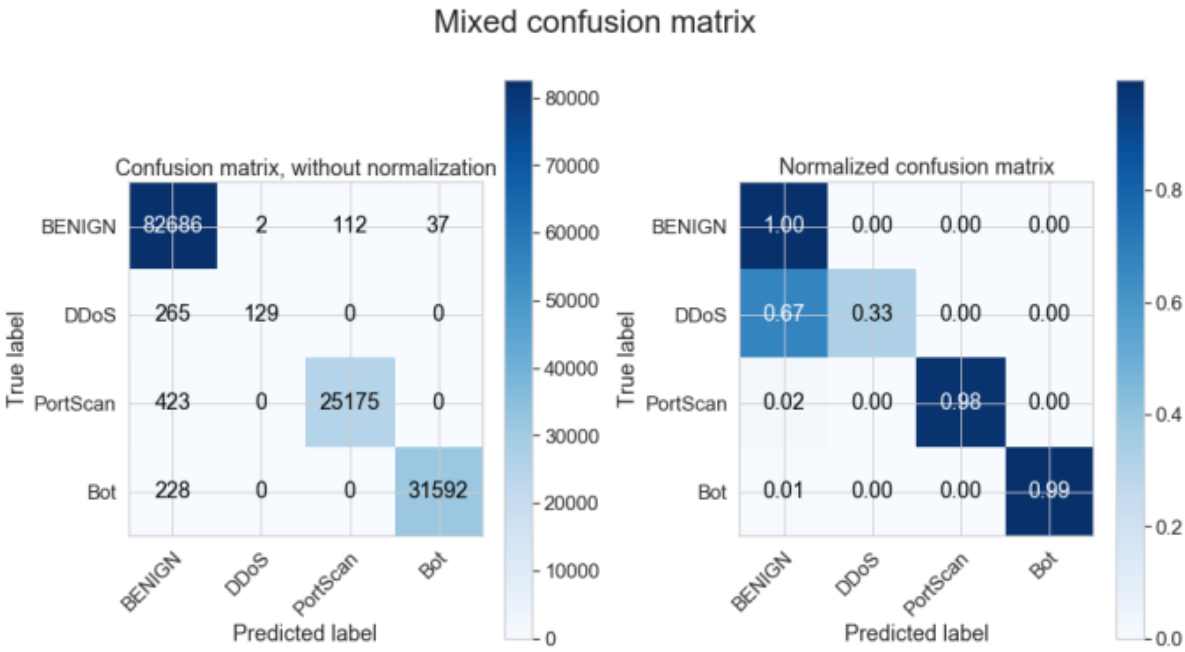

Loss
train
test
Accuracy
train
test

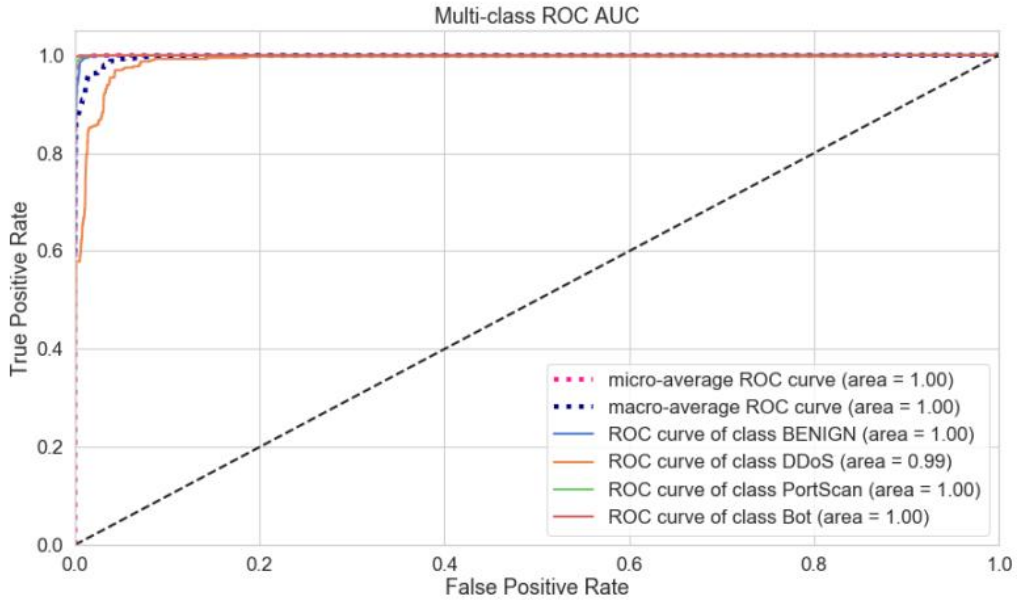

Mixed confusion matrix
Confusion matrix, without normalization
BENIGN
Infiltration
Web Attack □ Brute Force
Web Attack □ XSS
Web Attack □ Sql Injection
True label
Predicted label
91365
9
278
5
137
Normalized confusion matrix
1.00
0.00


Multi-class ROC AUC
True Positive Rate
False Positive Rate
micro-average ROC curve (area = 1.00)
macro-average ROC curve (area = 0.92)
ROC curve of class BENIGN (area = 0.98)
ROC curve of class Infiltration (area = 0.68)
ROC curve of class Web Attack □ Brute Force (area = 0.99)
ROC curve of class Web Attack □ XSS (area = 0.94)
ROC curve of class Web Attack □ Sql Injection (area = 0.98)

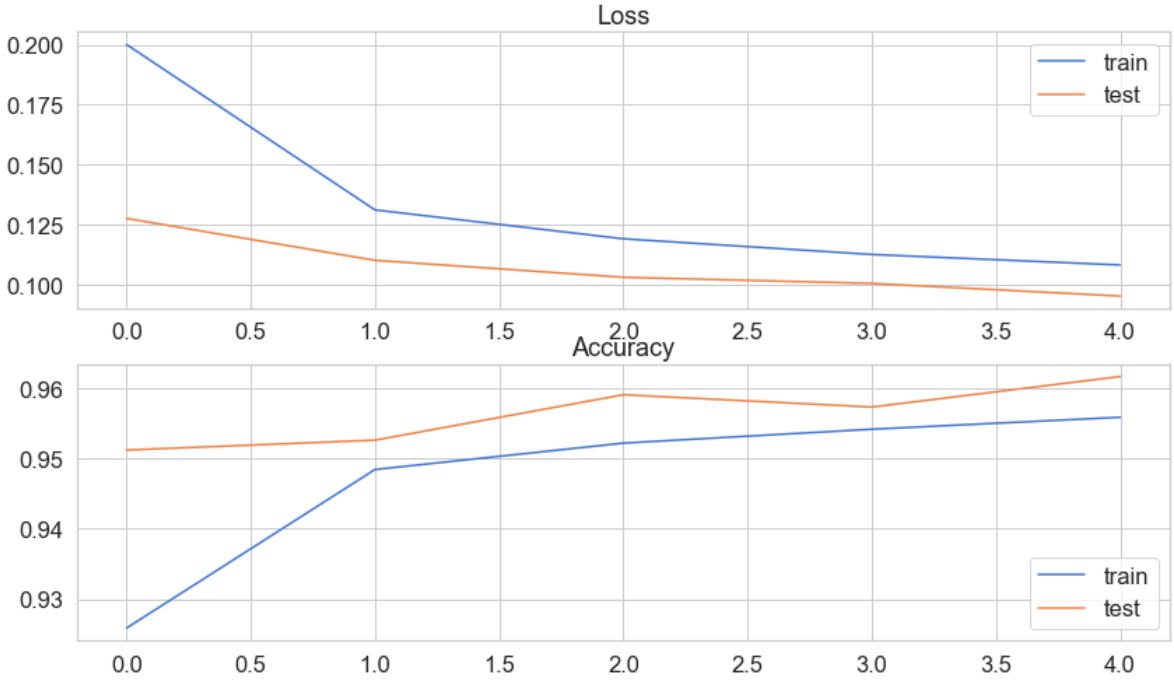
Jeudi

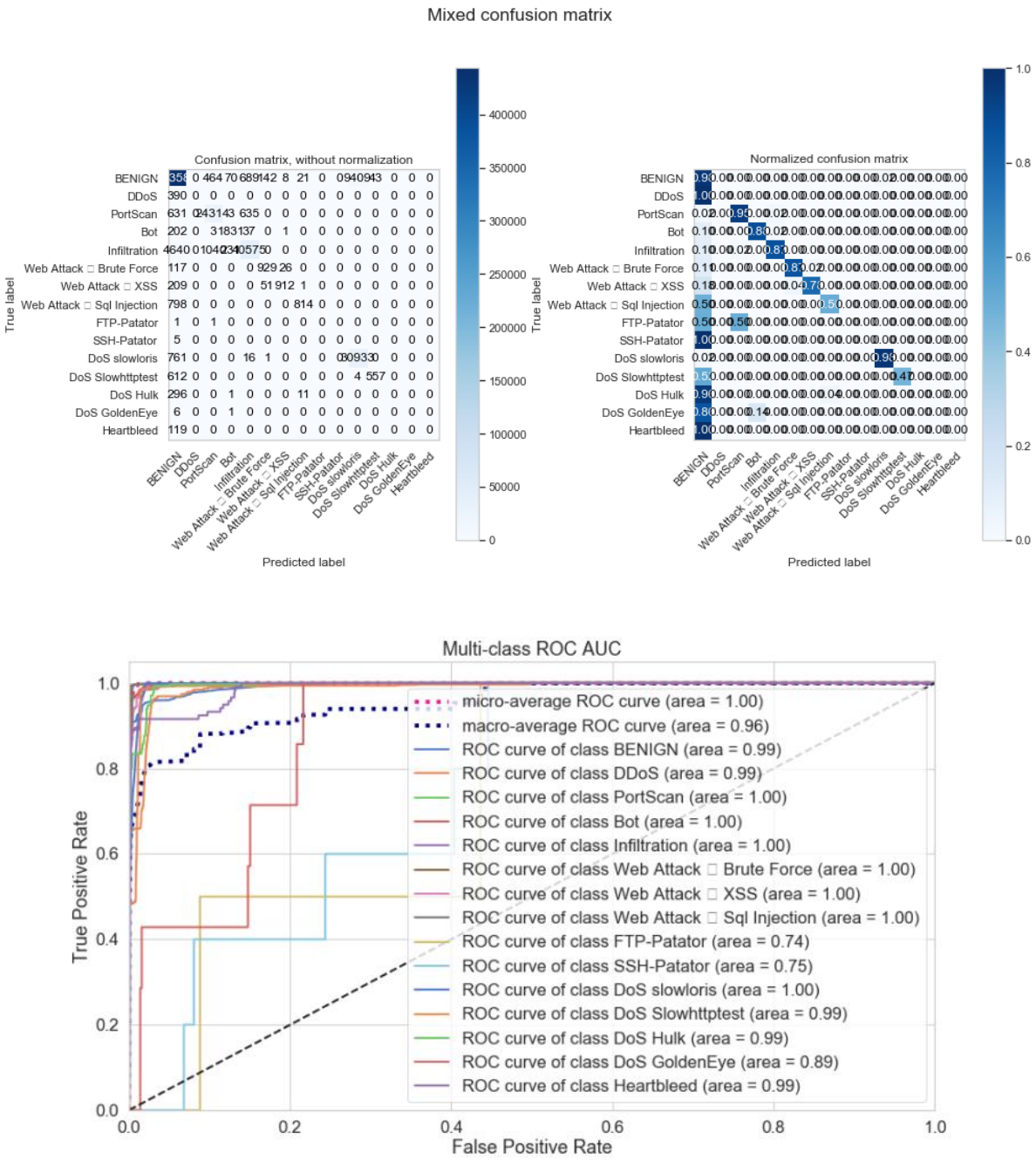

Loss
train
test
Accuracy
train
test


Mixed confusion matrix
Confusion matrix, without normalization
Normalized confusion matrix
BENIGN
DDoS
PortScan
Bot
True label
Predicted label
82686
2
112
37
265
129
0
0
423
0
25175
0
228
0
0
31592
1.00
0.00
0.00
0.00
0.67
0.33
0.00
0.00
0.02
0.00
0.98
0.00
0.01
0.00
0.00
0.99


Multi-class ROC AUC
True Positive Rate
False Positive Rate
micro-average ROC curve (area = 1.00)
macro-average ROC curve (area = 1.00)
ROC curve of class BENIGN (area = 1.00)
ROC curve of class DDoS (area = 0.99)
ROC curve of class PortScan (area = 1.00)
ROC curve of class Bot (area = 1.00)

Vendredi


Loss
train
test
Accuracy
train
test



Global

Figure V.2. *Graphes de la fonction loss et accuracy et matrices de confusion combinées du CNN avec les différents fichiers de tests*

En étudiant les matrices de confusion simples et normalisée des différentes journées de test, en remarque bien que, globalement, les classes ont été prédites convenablement, sauf pour les données de la journée du jeudi. Cela peut être expliqué par le fait que l'ensemble des données de cette journée n'est pas équilibré (*unbalanced dataset*)

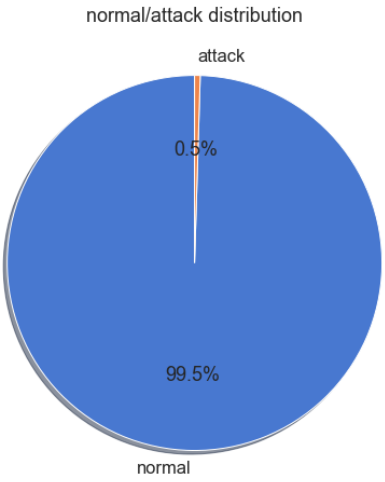


Figure V.3. *Distribution des données du jeudi*

V.6.4. Réglage fin des hyper-paramètres du réseau CNN

Afin de trouver les meilleurs paramètres de notre réseau de neurones convolutionnel dédié à la classification multi-classes, nous avons développé un petit script capable de lancer le réseau CNN en question, plusieurs fois, en variant les hyper-paramètres du modèle. Les paramètres ciblés par l'évaluation sont les valeurs des cartes de filtres (*filter maps*) et du *kernel*.

Dans le premier cas, nous avons évalué notre modèle avec une batterie de cinq répétitions, dans chacune d'elles nous l'avons testé avec les valeurs du filtre suivantes : 8, 16, 32, 64 et 3 époques et 32 instances par lot. Nous avons pris les valeurs: maximale, minimale et médiane pour chaque test. La meilleure valeur du nombre de cartes de filtres a été constatée pour **64**. Un diagramme en boîte (*box and whisker plot*) des résultats est également créé, ce qui permet de comparer la distribution des résultats avec chaque valeur des filtres.

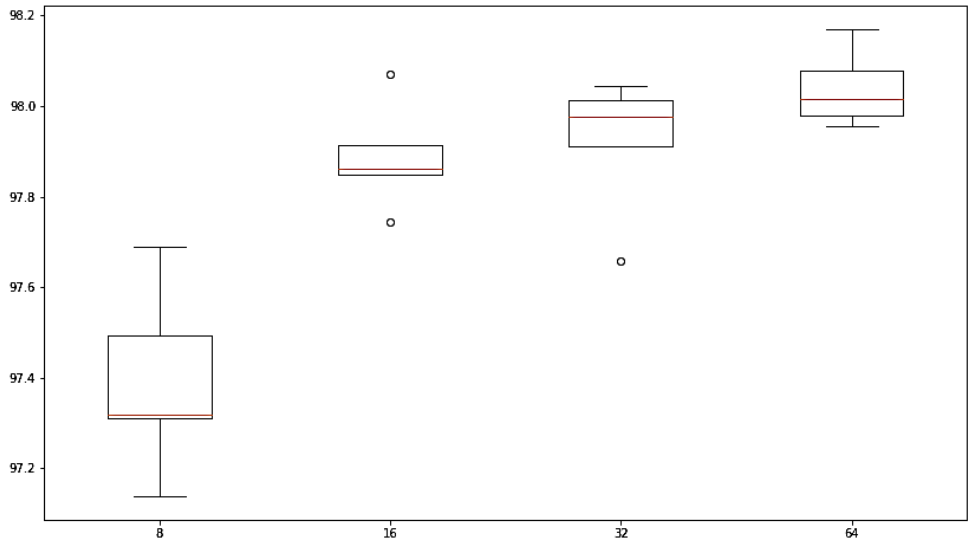


Figure V.4. *Réponse du modèle en fonction du nombre des cartes de filtres*

Pour le deuxième cas de l'évaluation c'est-à-dire la taille du noyau ou *kernel*. La taille du noyau contrôle le nombre de pas de temps pris en compte dans chaque "lecture" de la séquence d'entrée, qui est ensuite projetée sur la carte de fonctions (*feature map*) via le processus de convolution. Une grande taille de noyau signifie une lecture moins rigoureuse des données, mais peut aboutir à une vue plus généralisée de l'entrée.

Nous avons évalué notre modèle avec une batterie de cinq répétitions, dans chacune d'elles nous l'avons testé avec un taille du *kernel* parmi les valeurs suivantes : 2, 3, 5, 7, 11 et 3 époques et 32 instances par lot. Nous avons pris, comme précédemment, les valeurs: maximale, minimale et médiane pour chaque test. La meilleure valeur trouvée pour la taille du noyau du modèle a été constatée pour le nombre **5**.

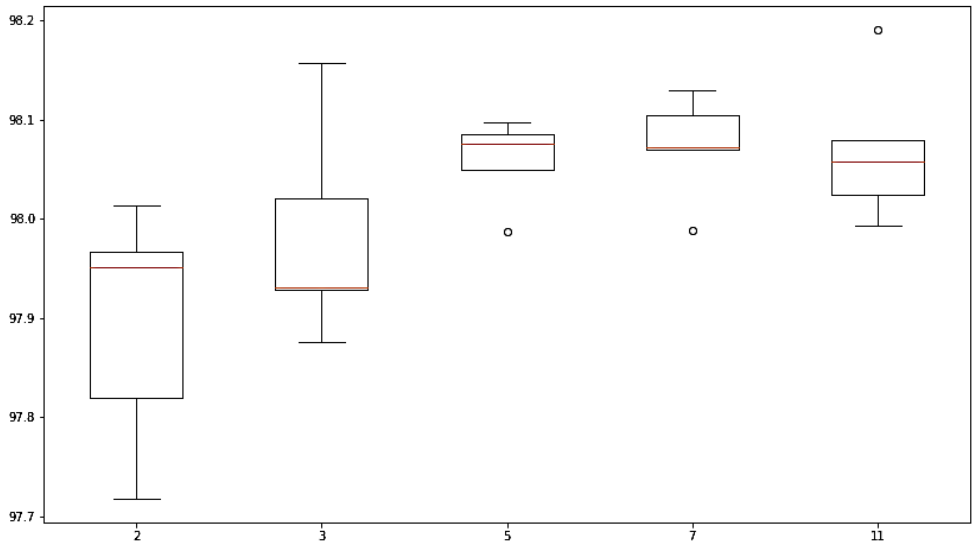


Figure V.5. *Réponse du modèle en fonction de la taille du kernel*

V.7. Comparaison du modèle avec les modèles des recherches antérieures

Afin d'évaluer notre modèle, nous avons comparé nos résultats avec ceux obtenus par l'étude de recherche utilisant la forêt aléatoire (*random forest*) pour l'élimination récursive des attributs (*features*) et l'apprentissage profond pour la classification (en utilisant *Deep Multilayer Perceptron DMLP*) [24]. Dans cette étude, seulement les données du premier fichier de la journée du jeudi ont été utilisées. En comparant les résultats obtenus par notre modèle avec l'étude précédente nous remarquons bien que notre modèle est plus précis (voir le tableau suivant).

Tableau V.7. *Comparaison de l'accuracy/loss de la prédiction de la classification avec les études antérieures utilisées avec CICIDS2017*

Etude	Classifieur	Accuracy %	Loss %
Ustebay, et al. [24]	DMLP	89.00	18
Notre modèle	**CNN**	**99.02**	**2.1**

Dans le tableau suivant, nous résumons la précision des différentes méthodes de classification utilisées avec la base CICIDS2017 dans des études antérieures. Nous pouvons conclure que notre classifieur CNN a obtenu la troisième meilleure précision pour la classification binaire avec CICIDS2017.

Tableau V.8. *Comparaison de la précision de la prédiction de la classification avec les études antérieures utilisées avec CICIDS2017*

Etude	Classifieur	Précision %
Fakre, et al. [5]	Deep neural network	97.73
	Random forest	92.72
	Gradient Boosted Tree	**99.97**
Iman , et al. [23]	K-Nearest Neighbors	96.00
	Random Forest	98.00
	ID	98.00
	Adaboost	77.00
	Multilayer Perceptron	77.00
	Naive Bayes	88.00
	Quadratic Discriminant Analysis	97.00
Vijayanand, et al. [25]	SVM+ Genetic	**99.85**
Resende and Drummond. [21]	Genetic + Profiling	92.85
Notre modèle	**CNN**	**99.20**

V.8. Conclusion

Ce dernier chapitre décrit, d'une manière détaillée, les outils utilisés pour réaliser le travail expérimental et fournit les résultats obtenus avec une analyse et une comparaison des résultats de prédiction des deux méthodes utilisées.

Chapitre VI : **CONCLUSION**

“Cattle die, Kinsmen die,
Even you yourself leave this world,
But one thing I know that never dies,
Judgement on a dead man.”

Hávamál

Nous rappelons que l'objectif de notre travail est l'étude et la réalisation d'une approche basée sur l'apprentissage profond des systèmes de détection des intrusions, de ce fait nous avons proposé dans ce mémoire une solution basée, comme demandé, sur l'apprentissage profond comprenant deux modules principaux.

Dans la première partie de notre travail, nous avons proposé une méthode pour la réduction de la dimensionnalité de bases de données volumineuses. La réduction est faite en employant un réseau de neurones profond ou empilé de type auto-encodeur empilé (*SAE/DAE*), avec lequel nous avons réussi à réduire la dimensionnalité des différentes bases de données avec un taux de perte très réduit.

Le deuxième objectif visé est la classification des nouvelles instances avec beaucoup moins d'attributs, de même que la réduction, la prédiction des classes a été faite avec un réseau de neurones convolutionnel (*CNN*) pour prédire les classes des nouvelles instances issues de la phase de réduction de la dimensionnalité.

Ces méthodes ont été testées sur la nouvelle base de données *CICIDS2017* qui est, considérablement, volumineuse, mais plus complète et riche que ses précédentes (*KDD99*, *NSL-KDD*, *DARPA*, etc.). Elles ont donné des résultats prometteurs et de très bonne qualité, donc nous pouvons qualifier la fiabilité et l'efficacité de notre approche comme très satisfaisante.

Le travail sur notre projet nous a permis d'approfondir notre connaissance dans le domaine de l'apprentissage profond, il nous a permis, aussi, de connaître et expérimenter de nouvelles bibliothèques python dédiées à ce domaine d'avant-garde et de savoir le mode de fonctionnement de plusieurs algorithmes de l'apprentissage automatique tels que les différentes architectures du *deep learning*.

Bien que les objectifs visés, au préalable, ont été atteints, mais il reste toujours des perspectives et des améliorations possibles qui peuvent encore être réalisés dans le future, telles que :

- L'amélioration de ce travail en utilisant d'autres méthodes du monde du *deep learning* et faire une étude comparative entre ces dernières au niveau des résultats, performance et rapidité.
- Penser à la création d'un modèle capable de capturer le trafic réseau en temps réel sans avoir besoin d'utiliser l'ensemble de données CICIDS2017 ou autre.

REFERENCES BIBLIOGRAPHIQUES

[1] M. Amand, M. Nsiri, "*Etude d'un système de détection d'intrusion comportemental pour l'analyse du trafic aéroportuaire,*" Rapport de projet tutoré, Jan 2011.

[2] I. Benmessahel, K. Xie and M. Chellal "*New Improved Training for Deep Neural Networks Based on Intrusion Detection System,*" IOP Conf. Ser.: Mater. Sci. Eng. 435 012034, 2018.

[3] G. Bourkache, "*Un IDS réparti basé sur une société d'agents intelligents,*" Mémoire de magistère, Université M'Hamed BOUGARA, Boumerdès, Algérie, 2007.

[4] L. Deng, D. Yu., "*Deep Learning: Methods and Applications,*" Foundations and Trends in Signal Processing, vol. 7, nos. 3–4, pp. 197–387, 2013.

[5] O. Fakre, E. Dogdu "*Intrusion Detection Using Big Data and Deep Learning Techniques,*" DOI 10.1145/3299815.3314439, 2019

[6] T. Farth, "*Deep Learning: A Comprehensive Guide for Beginners,*" Independently published, Jan 2019.

[7] C. Fung, R. Boutaba "*Intrusion Detection Networks: A Key to Collaborative Security,*" Florida, USA, CRC Press-Taylor & Francis Group, pp 19, 2014.

[8] Google Developers, "*ML Practicum: Image Classification,*" ML Practicum: Image Classification | Machine Learning | Google Developers. [En ligne]. Disponible: https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks. [Consulté le 12 avril 2019].

[9] S. Gunadiz, "*Algorithme d'intelligence artificielle pour la classification d'attaque réseau à partir de données TCP,*" Mémoire de magistère, Université M'Hamed BOUGARA, Boumerdès, Algérie, 2011.

[10] G. E. Hinton, *Deep belief networks,* Scholarpedia, 4(5), 5947.

[11] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis, and R. Atkinson. "*Shallow and deep networks intrusion detection system: A taxonomy and survey,*" arXiv preprint arXiv:1701.02145, 2017.

[12] K. Kim, M. E. Aminanto, H. C. Tanuwidjaja, "*Network Intrusion Detection using Deep Learning: A Feature Learning Approach,*" Singapore, Springer, pp 1, Sep 2018.

[13] C. A. Leke, T. Marwala "*Deep Learning and Missing Data in Engineering Systems,*" Suisse, Springer, pp 21-22, Déc 2018.

[14] C. Llorens, L. Levier, D. Valois, B. Morin, "*Tableaux de bord de la sécurité réseau,*" Paris, France, Editions Eyrolles, 2010.

[15] S. Martin, "*Anti-IDS Tools and Tactics,*" SANS Technology Institute, Août 2001.

[16] C. Michel "*Langage de description d'attaques pour la détection d'intrusions par corrélation d'événements ou d'alertes en environnement réseau hétérogène,*" Thèse de doctorat, Université de Rennes 1, Rennes, France, Déc 2003.

[17] L. Mohammadpour, T. Chaw Ling, C. Sun Liew, C. Yong Chong, "*A Convolutional Neural Network for Network Intrusion Detection System,*" Proceedings of the APAN – Research Workshop 2018, pp. 50–55, 2018.

[18] A. Osseiran, F. Boccardi, V. Braun, K. Kusume, P. Marsch, M. Maternia, O. Queseth, M. Schellmann, H. Schotten, H. Taoka, H. Tullberg, M. A. Uusitalo, B. Timus, and M. Fallgren, "*Scenarios for 5G mobile and wireless communications: The vision of the metis project,*" *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 26–35, Mai 2014.

[19] S. Park, H. Park, "*ANN Based Intrusion Detection Model,*" Suisse, Springer, pp. 433–437, 2019.

[20] B. Ramsundar, R. Bosagh Zadeh, "*TensorFlow pour le Deep learning - De la régression linéaire à l'apprentissage par renforcement,*" Paris, France, Editions First-O'Reilly, pp. 9, Oct 2018.

[21] P.A.A. Resende, A.C. Drummond, "*Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling. Security and Privacy,*" John Wiley & Sons, Ltd 2018.

[22] S. Sengupta, S. Basak, P. Saikia, S. Paul, V. Tsalavoutis, F. D. Atiah, V. Ravi, R. A. Peters II, "*A Review of Deep Learning with Special Emphasis on Architectures, Applications and Recent Trends,*" Preprints 2019, 2019020233 (doi: 10.20944/preprints201902.0233.v1).

[23] I. Sharafaldin, A. Lashkari, and A. Ghorbani, "*Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization,"* In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018), pp 108-116. DOI: 10.5220/0006639801080116

[24] S. Ustebay, Z. Turgut and M. A. Aydin, "Intrusion Detection System with Recursive Feature Elimination by Using Random Forest and Deep Learning Classifier," 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT), ANKARA, Turkey, 2018, pp. 71-76.

[25] R. Vijayanand, D. Devaraj, B. Kannapiran, "*Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection,*" Computers & Security 2018, doi: 10.1016/j.cose.2018.04.010.

[26] Wikipédia, "*Machine de Boltzmann restreinte*," Machine de Boltzmann restreinte — Wikipédia. [En ligne]. Disponible: https://fr.wikipedia.org/wiki/Machine_de_Boltzmann_restreinte. [Consulté le 9 avril 2019].