REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ DES MATHEMATIQUES ET D'INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Réseaux et Télécommunication
Par :

SIMERABET ABDESALEM
HOUARI ABDELHAMID

Sur le thème

Amélioration des Systèmes de Détection d'Intrusion à l'aide de techniques de Deep Learning et les Mécanismes d'Attention

Soutenu publiquement le 15/06/2025 à Tiaret devant le jury composé de :

Mr Aid Lahcen	MCA	Université de Tiaret	Président
Mr Douad Mohamed Amine	MCB	Université de Tiaret	Encadrant
Mr Kouadria Abderrahmane	MCA	Université de Tiaret	Examinateur

2024-2025

Remerciements

Nous rendons grâce à ***Dieu****, le Tout-Puissant, de nous avoir accordé la santé et la force nécessaire pour mener à bien ce mémoire.*

Nos premiers remerciements vont à ***Mr.Dr.Mohamed Amine DAOUD*** *, notre directeur de mémoire, dont l'encadrement exceptionnel, la rigueur, la patience et la disponibilité ont été déterminants pour la réussite de ce travail. Sans son soutien précieux, cette recherche n'aurait pu aboutir.*

Nous remercions également les membres du jury de notre soutenance de mémoire, ***Mr.Dr Kouadria Abderrahmane*** *et* ***Mr.Dr Aid lahcen****, pour avoir accepté de faire partie du jury.*

Nous exprimons également notre profonde gratitude à l'ensemble de nos professeurs pour leur générosité, leurs enseignements et leur patience, malgré leurs lourdes charges académiques et professionnelles. Leurs conseils ont grandement enrichi notre parcours.

Enfin, nous pensons avec reconnaissance à nos familles et amis, dont les encouragements et la bienveillance nous ont portés tout au long de cette aventure.

Merci à tous ceux qui, de près ou de loin, ont contribué à ce travail.

Dédicaces

C'est avec l'aide et la grâce d'Allah que nous avons pu mener à bien ce modeste travail.

Je le dédie, avec reconnaissance, à ceux qui m'ont soutenu dans les épreuves :
À mes chers parents, pour leur amour et leurs sacrifices,
À ma grand-mère, pour sa tendresse et son soutien,
À mes frères et sœurs, pour leur présence à mes côtés,
À mon binôme HOUARI et sa famille,
À toute ma promotion, pour l'entraide et les souvenirs partagés,

Simerabet Abdesslem

À mes très chers parents, Aucun mot, aucune dédicace ne peut exprimer tout le respect, la considération et l'amour éternel que je ressens pour vous, en raison des sacrifices que vous avez consentis pour mon instruction et mon bien-être,
À toute ma chère famille,
À mon binôme SIMERABET et à sa famille,
À mes professeurs,
À mes chers amis,
À mes chers collègues,
À tous ceux qui m'ont aidé de près ou de loin.

Houari Abdelhamid

À la mémoire éternelle de nos martyrs, et spécialement aux martyrs de la Palestine bien-aimée, qui ont sacrifié leur vie et leur peuple pour leur terre bande du gaza, Palestine

Free Gaza

Résume

La montée des cybermenaces et la sophistication croissante des attaques rendent nécessaire la mise en œuvre de systèmes de détection d'intrusions (IDS) à même de s'adapter à une grande variété de données réseau. Les techniques de détection classique, bien que performantes, révèlent cependant leurs limites face à des attaques complexes et en constante évolution. Ce projet de Master se propose d'intégrer une approche innovante notamment en recourant aux mécanismes d'attention, tirés des recherches en intelligence artificielle et traitement du langage naturel, afin de contribuer à l'amélioration des dispositifs IDS concernant leur capacité de détection des comportements anormaux du réseau. L'approche visée s'appuie sur des mécanismes d'attention pour focaliser les ressources de calcul sur les segments critiques du flux de données réseau, favorisant ainsi la détection d'anomalies dans les réseaux, à la fois rapides et efficaces. Elle sera conjointement mise en œuvre avec des techniques de deep learning comme les réseaux neuronaux récurrents (RNN) ou les Transformers, en vue de mieux modéliser les interactions complexes/combinaison entre les paquets de données afin de mieux détecter de modèles malveillants. Le système proposé sera évalué sur différents jeux de données standard d'intrusions, permettant d'en tester l'efficacité par rapport aux approches classiques.

Mots-clés : *Systèmes de détection d'intrusion (IDS), mécanismes d'attention, deep learning, Cybermenaces, anomalies réseau.*

Abstract

The rise in cyberthreats and the increasing sophistication of attacks have made it necessary to implement intrusion detection systems (IDS) capable of adapting to a wide variety of network data. Conventional detection techniques, though effective, have their limits when faced with complex and constantly evolving attacks. The aim of this Master's project is to integrate an innovative approach, in particular using attention mechanisms drawn from research in artificial intelligence and natural language processing, to help improve the ability of IDS devices to detect abnormal network behavior. The approach relies on attention mechanisms to focus computational resources on critical segments of the network data flow, enabling fast and efficient detection of network anomalies. It will be jointly implemented with deep learning techniques such as Recurrent Neural Networks (RNN) or Transformers, to better model complex interactions/combination between data packets in order to better detect malicious patterns. The proposed system will be evaluated on various standard intrusion datasets, enabling us to test its effectiveness against conventional approaches.

Keywords: *Intrusion detection system , attention mechanisms, deep learning, Cybersecurity, network anomalies.*

ملخص

إن تزايد التهديدات السيبرانية وتزايد تعقيد الهجمات يعني أن أنظمة كشف التسلل بحاجة إلى أن تكون قادرة على التكيف مع مجموعة واسعة من بيانات الشبكة. تقنيات الكشف التقليدية، على الرغم من فعاليتها، إلا أن لها حدودها عند مواجهة الهجمات المعقدة والمتطورة باستمرار. الهدف من مشروع الماجستير هذا هو دمج نهج مبتكر، لا سيما من خلال استخدام آليات الانتباه المستمدة من الأبحاث في مجال الذكاء الاصطناعي ومعالجة اللغة الطبيعية، للمساعدة في تحسين قدرة أنظمة على اكتشاف السلوك غير الطبيعي للشبكة. ويعتمد النهج المستهدف على آليات الانتباه لتركيز موارد الحوسبة على الأجزاء الحرجة من تدفق بيانات الشبكة، وبالتالي تسهيل اكتشاف الحالات الشاذة في الشبكات بسرعة وكفاءة. وسيتم تنفيذه بالاشتراك مع تقنيات التعلّم العميق مثل الشبكات العصبية المتكررة أو المحولات، بهدف نمذجة التفاعلات المعقدة/التركيبات المعقدة بين حزم البيانات بشكل أفضل من أجل اكتشاف الأنماط الضارة بشكل أفضل. سيتم تقييم النظام المقترح على مختلف مجموعات بيانات التطفل القياسية، مما يتيح اختبار فعاليته في مقابل الأساليب التقليدية.

الكلمات المفتاحية: *نظام الكشف عن التسلل، آليات الانتباه والتعلّم العميق، الأمن السيبراني، شذوذات الشبكة.*

Table des matières

Liste des Figures

Liste des Tableaux

List des Abréviations

ANN Artificiel Neural Network. iv, 24

AS-IDS Agent-Based and Signature-Based Intrusion Detection System. 40

CIC Institut canadien de cybersécurité. 13, 14, 43

CNN Convolutional Neural Network. iv, 26, 27

CPU Central Processing Unit. 50

CST Centre de la sécurité des télécommunications. 14

DataFrame Des tableaux avec plusieurs colonnes, comme dans Excel. 50

Dataset Ensemble des données. iii, 13, 14, 18, 35, 43

DDOS Déni de service distribué. 14, 35, 43

DL Deep Learning. v, 1, 16, 18, 22, 23, 25, 41, 43, 45, 50, 51

DNN Deep Neural Network. 22, 36

DOS Déni de service. 4, 13, 14, 36, 43

FFNN FeedForwards Neural Network. iv, 24, 25

FTP File Transfer Protocol. 43

GPU Graphics Processing Unit. 50

GRU Gated recurrent units . iv, v, 28, 29

HIDS Host-based Inrusion Detection System. v, vii, 10, 11, 16

HTTP Hypertext Transfer Protocol. 43

HTTPS Hypertext Transfer Protocol Secure. 43

IA Intelligence Artificielle. v, 18, 22, 43, 48

IDES Intrusion Détection Expert System. 5

IDS système de détection d'intrusion. iii, v, 1, 3–10, 13, 14, 16, 18, 35, 36, 43, 51

IoT Internet of things. 35, 38, 39

IPS Intrusion Prevention System. 40

LSTM Long short-term memory . iv, v, 28

ML Machine Learning. v, 18–20, 22, 23, 35, 36, 41, 43, 48, 50, 51

NIDS Network-based Intrusion Detection System. v, vii, 5, 10, 11, 16

R2L Remote to Local. 13

RNN Recurrent Neural Network. iv, v, 27, 28, 33

Serie une seule colonne de données. 50

SI système informatique. 3, 4, 18

Sklearn Scikit-learn. 51

SQL Structured Query Language. 5

SSH Secure Shell. 43

TPU Tensor Processing Unit. 50

U2L User to Local. 13

INTRODUCTION GÉNÉRALE

Aujourd'hui, avec Internet la sécurité des réseaux est un vrai casse-tête. Les systèmes qui détectent les intrusions (IDS) sont importants parce qu'ils essaient de devancer les attaques plutôt que juste réagir. Le principe, c'est qu'ils analysent le trafic réseau avec des algorithmes pour repérer ce qui semble bizarre. Sauf que les vieilles méthodes ont un gros défaut : elles regardent les données vite fait, sans vraiment comprendre ce qui se passe en profondeur. Du coup, elles laissent passer pas mal de menaces.Le problème, c'est que les attaques se multiplient et évoluent sans arrêt. Soit ce sont des variantes d'anciennes attaques, soit des techniques toutes nouvelles. Résultat, à peu près tout ce qui est connecté peut devenir une cible. Si un serveur important se fait pirater, ça peut coûter très cher à une boîte, autant en argent qu'en image. Pourtant, les IDS classiques sont souvent dépassés : ils voient pas les attaques récentes (comme les zero-day) et sonnent l'alarme pour rien trop souvent. Clairement, on a besoin de mieux. C'est là que le deep learning change la donne. Contrairement aux vieux systèmes, il peut digérer des masses de données en vrac et y trouver des motifs complexes. En réglant bien ses paramètres, il arrive à focaliser sur ce qui compte vraiment pour détecter les vraies menaces. Mais même lui a ses limites : parfois il comprend pas tout seul où chercher, donc il rate des attaques ou fait des erreurs. Pour améliorer ça, nous avons eu l'idée d'ajouter un 'mécanisme d'attention' au DL. En gros, ça aide le système à concentrer ses efforts sur les parties vraiment suspectes du trafic, comme quand on fait le tri dans une masse d'infos. Résultat : moins de faux positifs et plus de vraies menaces détectées.(Sun, 2021).
Notre mémoire est structuré comme suit :
Le premier chapitre se consacre à la présentation d'IDS, à ses principes de fonctionnement, ainsi qu'aux différents concepts liés à la détection des intrusions.
Le deuxième chapitre comprend DL et les Mécanismes d'Attention pour la détection d'intrusion.
Le troisième Chapitre présente une comparaison entre des travaux connexe avec notre mémoire et les modèles que nous proposons dans le domaine de DL et des mécanismes d'attention.

Chapitre 1

SYSTEME DE DETECTION D'INTRUSION

1.1 Introduction

Avec l'intérêt et les progrès récents dans le développement de l'internet et des technologies de communication au cours de la dernière décennie, la sécurité des réseaux est devenue un domaine de recherche essentiel (Ahmad, 2021). La sécurité des réseaux s'est imposée comme un domaine de recherche essentiel. Elle fait appel à des outils tels que les pares-feux, les logiciels antivirus et les IDS pour garantir la sécurité du réseau et de tous les actifs associés au sein d'un cyberespace. L'idée de l'IDS a été proposée pour la première fois par Jim Anderson en 1980. Depuis lors, de nombreux produits IDS ont été développés et ont mûri pour satisfaire les besoins en matière de sécurité des réseaux. Cependant, l'immense évolution des technologies au cours de la dernière décennie a entraîné une forte expansion du marché des IDS.
En conséquence, une énorme quantité de données importantes est générée et partagée entre les différents nœuds du réseau(Tarter, 2017).

1.2 La Sécurité Informatique

1.2.1 Définition

La sécurité informatique englobe toutes les mesures mises en place pour réduire la vulnérabilité d'un système face aux menaces, qu'elles soient accidentelles ou intentionnelles. Il est essentiel de définir les exigences fondamentales en matière de SI, car elles représentent les attentes des utilisateurs de SI's en termes de protection et de fiabilité (Salah, 2022).

1.2.2 Les Objectifs de la sécurité

La SI vise généralement les cinq principaux objectifs suivants:
Confidentialité: C'est la garantie que seules les personnes autorisées pourront accéder aux informations qui leur sont destinées.
Intégrité: C'est la garantie que les informations restent telles qu'elles sont et ne subissent aucune modification non autorisée.
Disponibilité: La disponibilité garantit que les systèmes chargés de fournir, stocker et traiter les informations sont accessibles aux utilisateurs autorisés lorsqu'ils en ont besoin.
Authenticité: Cela fait référence aux caractéristiques d'une communication, d'un document ou de toute donnée, garantissant que ces informations sont authentiques et non modifiées ou falsifiées. Son rôle principal est de s'assurer que l'utilisateur est bien celui qu'il prétend être et que les messages sont authentiques.
Non répudiation : C'est la garantie qu'un utilisateur ne peut pas nier l'authenticité de sa signature lorsqu'il effectue une tâche ou envoie un message.

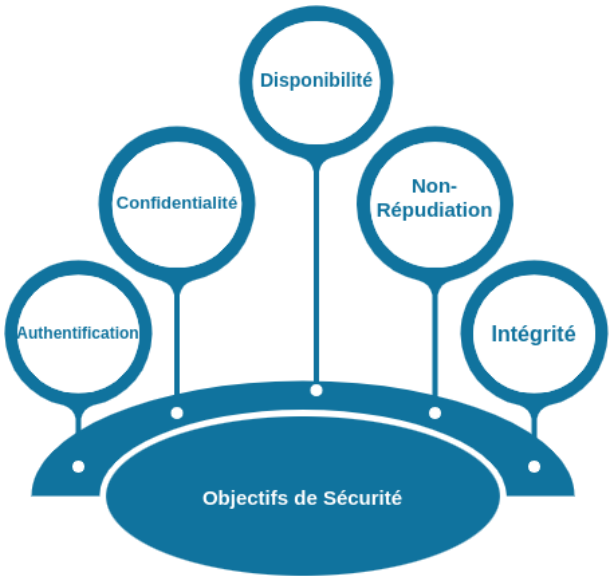


Figure 1.1: Objectifs de la sécurité

1.2.3 Les Vulnérabilités

Une vulnérabilité est une faiblesse dans la conception, la configuration ou l'implémentation d'un réseau ou d'un système, ce qui le rend sensible à des menaces. La plupart des vulnérabilités proviennent généralement de l'une des trois causes suivantes: une mauvaise conception, mise en œuvre incorrecte ou gestion insuffisante.(Majorczyk et al., 2005)

1.2.4 Les Virus

un virus est tout programme d'ordinateur capable d'infecter un autre programme d'ordinateur en le modifiant de façon a ce qu'il puisse son tour se reproduire.

1.2.5 Les Attaque

Les attaques de réseau sont des actions malveillantes visant à perturber, dégrader ou détruire les informations et services des réseaux informatiques. Elles se produisent via le flux de données et cherchent à compromettre l'intégrité, la confidentialité ou la disponibilité des systèmes de réseau. Ces attaques peuvent inclure l'envoi de courriels indésirables, l'intrusion dans des données sensibles, l'utilisation non autorisée de systèmes, et les attaques par DOS, qui exploitent des failles ou des bogues logiciels pour altérer les données du système(Ghorbani, 2010).

1.3 IDS

1.3.1 Intrusion

On définit une intrusion comme une tentative de compromettre la confidentialité, l'intégrité ou la disponibilité des données, ou de contourner les mécanismes de sécurité d'un ordinateur ou d'un réseau. Ces intrusions peuvent être causées par des attaquants accédant

aux systèmes depuis Internet, par des utilisateurs autorisés qui cherchent à obtenir des privilèges supplémentaires non autorisés, ou encore par des utilisateurs autorisés qui abusent des privilèges qui leur ont été accordés(G., 2018).

1.3.2 La Détection d'intrusion

La détection d'intrusion consiste à surveiller les événements se produisant dans un SI ou un réseau et à les analyser à la recherche de signes d'intrusion(G. D. Karatas, 2018).

1.3.3 IDS

Les IDS sont des outils de sécurité logiciels ou matériels très importants qui permettent d'éliminer les menaces qui pèsent sur le transport des informations, empêcher les accès non autorisés ou les abus, et signaler les attaques aux responsables de la sécurité . La détection des attaques a été introduite pour la première fois dans l'étude **Computer security threat monitoring and surveillance** publiée en 1980. Les raisons de la nécessité des IDS:

1. Ils détectent les attaques qui ne peuvent être empêchées par d'autres mécanismes de sécurité.
2. Il répond à la phase d'analyse avant que l'attaque ne se produise.
3. Il permet d'analyser l'attaque, de réparer le système et de corriger les facteurs d'attaque(Rashid, 2020).

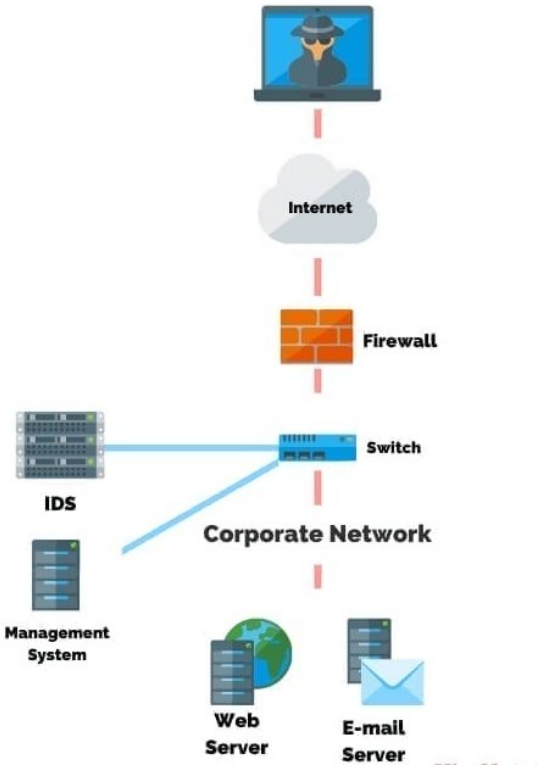


Figure 1.2: IDS

1.3.4 Fonctionnement des IDS

Un fonctionnement d'un IDS surveille le trafic du réseau et de l'hôte pour détecter des menaces malveillantes, soit manuellement, soit automatiquement. Entre 1984 et 1986, Dorothy Denning et Peter Neumann ont développé un modèle initial d'IDS en temps réel, appelé IDES (figure 1.2).
Vers le début des années 2000, les IDS dans les réseaux (NIDS) ont émergé comme la norme en matière de sécurité des réseaux, supplantant les pare-feu qui étaient confrontés à de multiples menaces liées au trafic réseau, car il permettait de contrer les menaces telles que les scripts interdites et les requêtes SQL malveillantes.(Hamouda, 2020)

1.3.5 Les composants d'un IDS

Divers schémas ont été proposés pour illustrer les composants d'un IDS. Parmi eux, nous avons choisi celui élaboré par le groupe de travail sur le format d'échange de détection d'intrusion de l'Internet Engineering Task Force.
Ce standard repose sur le modèle d'architecture présenté dans (la figure 1.3).

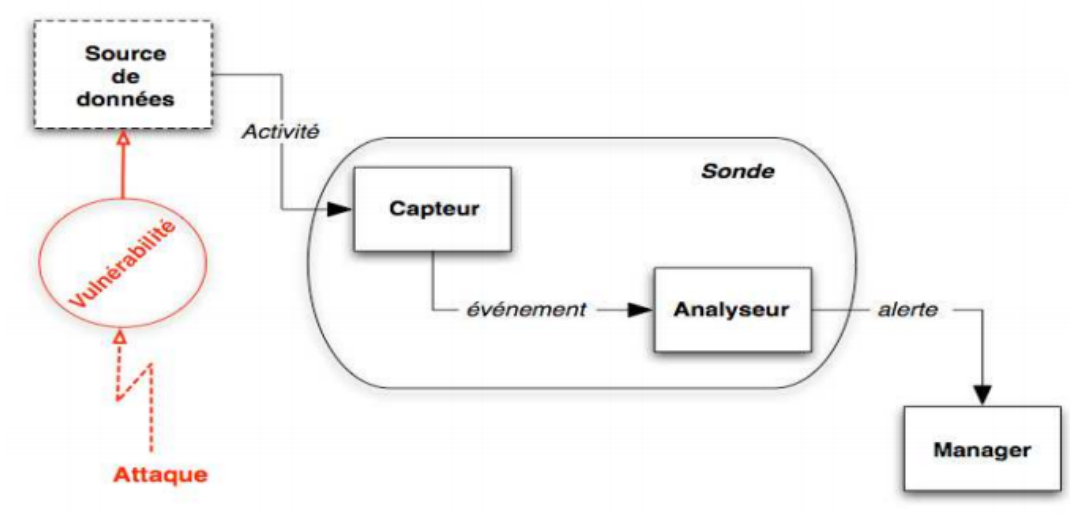


Figure 1.3: Architecture d'un IDS (Debar, 2000)

Source de donnée: Les IDS exploitent diverses sources de données, telles que les réseaux, les systèmes, les applications et les alertes. Ils ne se limitent pas à ces sources spécifiques, mais utilisent des capteurs adaptés pour analyser les informations provenant de ces différentes origines, dans le but de détecter toute activité frauduleuse ou indésirable.

Capteur: Le capteur de données surveille l'activité du système et produit des événements qui indiquent l'évolution de l'état du système en filtrant les informations provenant d'une source de données.

Analyseur: Les analyseurs prennent en entrée les données provenant de divers collecteurs ou d'autres analyseurs ayant déjà effectué un premier traitement. Ils génèrent des alertes lorsque le flux d'événements fourni par les capteurs contient des signes d'activités malveillantes.

Manager: Le manager collecte les alertes produites par l'analyseur, et les transmit ensuite à l'opérateur sous forme de notifications afin de lui permettre de gérer les alertes reçues et prendre des décisions.

1.3.6 Classification des IDS

On distingue trois classification d'IDS : selon la la technique de détection , l'emplacement et le comportement de réaction.

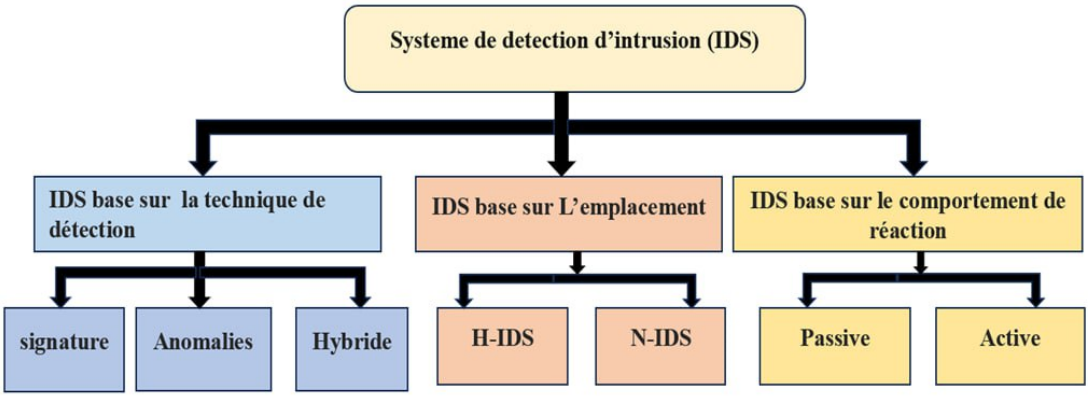


Figure 1.4: Classification des IDS

1.3.6.1 Classification selon la technique de détection

Il existe principalement deux techniques de détection:

1. la détection des abus (basée sur la signature)
2. la détection des anomalies (basée sur le comportement)

Figure 1.5: Les techniques de détection d'intrusions

A) La détection des abus (basée sur la signature)

La détection d'intrusion basée sur les signatures repose sur la comparaison des paquets de trafic réseau avec des modèles d'attaques préalablement enregistrés dans une base de données. Lorsqu'un paquet correspond à l'un de ces modèles, il est considéré comme suspect. Cependant, cette méthode ne peut détecter que les attaques déjà connues, Exemples: Suricata est un IDS basé sur la signature (Pharate, 2015).

Les avantage

- Lorsque les signatures d'attaques sont bien définies, le taux de faux positifs est réduit.
- Son utilisation est simple et intuitive.

Les inconvénients

- **Dépendance à la connaissance spécifique:**
  Les systèmes d'identification basés sur les signatures requièrent une collection précise de modèles d'attaque. En l'absence de ce savoir-faire préalable, la détection peut être restreinte et les nouvelles attaques peuvent être négligées.
- **Difficulté à détecter les attaques inconnues:**
  Les IDS basés sur les signatures sont efficaces pour détecter les attaques connues, mais ils ont du mal à identifier les attaques inédites ou les variantes légèrement modifiées.
- **Alertes fréquentes:**
  Ces systèmes génèrent des alertes pour chaque correspondance avec une signature, même si l'attaque échoue. Par exemple, un ver Windows tentant d'attaquer un système Linux déclencherait de multiples alertes, même si l'attaque n'a pas réussi.
- **Dépendance au contexte:**
  La pertinence des signatures dépend de l'environnement spécifique. Ce qui est considéré comme une menace dans un contexte peut ne pas l'être dans un autre (Kumar, 2012).

B) la détection des anomalies (basée sur le comportement)

Les IDS basés sur les anomalies analysent le comportement du réseau. Ce comportement peut être défini par l'administrateur ou appris à partir des données collectées lors de la phase d'apprentissage du système. Des règles sont ensuite établies pour identifier les comportements normaux et anormaux Par exemple: Snort et Bro-IDS sont des IDS basés sur les anomalies (Jyothsna, 2011).

Les avantage

- Il est capable de détecter des attaques inconnues.

Les inconvénients

- La création d'un ensemble de règles pour la détection des intrusions est une tâche complexe.
- L'efficacité du système repose sur la pertinence des règles et leur validation à l'aide d'ensembles de données de test.

C) La détection hybride

L'intégration des IDS basés sur les anomalies et ceux basés sur les signatures constitue un modèle hybride. Ces modèles hybrides permettent de trouver un équilibre optimal entre les coûts de stockage et de calcul, tout en réduisant le nombre de fausses alertes positives. De nos jours, la majorité des systèmes adoptent des IDS hybrides en raison de leur efficacité accrue en matière de détection et de leur simplicité d'utilisation (Smys, 2020).

1.3.6.2 La comparaison entre détection par(signature et anomalies)

Table 1.1: La comparaison entre la détection des abus et la détection des anomalies (H. Liu, 2019).

	Détection des abus(signature)	**Détection des anomalies**
Détection des performances	Faible taux de fausses alarmes, mais taux élevé de fausses alarmes manquées.	Faible fréquence de fausses alertes, Haute fréquence de fausses alertes.
Performance de détection	Élevée, diminue avec la taille de la base de données de signatures.	Dépend de la complexité du modèle.
Dépendance à l'égard de la du domaine	La plupart des détections reposent sur la connaissance du domaine.	Limitée, seule la conception des caractéristiques repose sur la connaissance du domaine.
Domaine interprétation	Conception basée sur la connaissance du domaine, avec une forte capacité d'interprétation.	Ne fournit que des résultats de détection, avec une capacité d'interprétation limitée.
Détection des attaques inconnues	Ne reconnaît que les attaques déjà identifiées.	Détecte les attaques connues et Inconnues

1.3.6.3 Le comportement de réaction

IDS Passive

Un IDS à réponse passive détecte une éventuelle faille de sécurité sans intervenir directement. Il enregistre les données et envoie une alerte, souvent sous forme de courriel, à un destinataire désigné. Contrairement à une réponse active, il ne tente pas de stopper l'intrusion mais se contente de signaler l'incident. Certains IDS passifs peuvent être connectés à une console d'administration centrale via des plug-ins, permettant ainsi une intégration dans un système de réponse active distribué. Dans ce cas, l'IDS passif envoie des rapports à une console centrale qui peut ensuite gérer les appareils et systèmes du réseau concernés. Les notifications d'intrusion peuvent être envoyées par divers moyens, tels que téléavertisseur, téléphone portable, courrier électronique ou directement sur le PC de l'administrateur. Pour éviter toute interception ou manipulation des alertes par un attaquant, il est crucial que ces communications soient sécurisées.

IDS Active

Un IDS actif, c'est un système qui va plus loin qu'une simple alerte : il agit pour stopper les attaques en cours. En plus de détecter les intrusions, il met en place des mesures automatiques pour réagir, comme bloquer l'adresse IP de l'attaquant. Contrairement à un IDS passif qui se contente de surveiller, l'IDS actif intervient directement pour protéger le réseau. Il peut aussi ajuster sa configuration pour empêcher l'attaquant de continuer, par exemple en fermant des ports ou en redémarrant des services. En résumé, il combine détection et action pour une protection plus proactive.

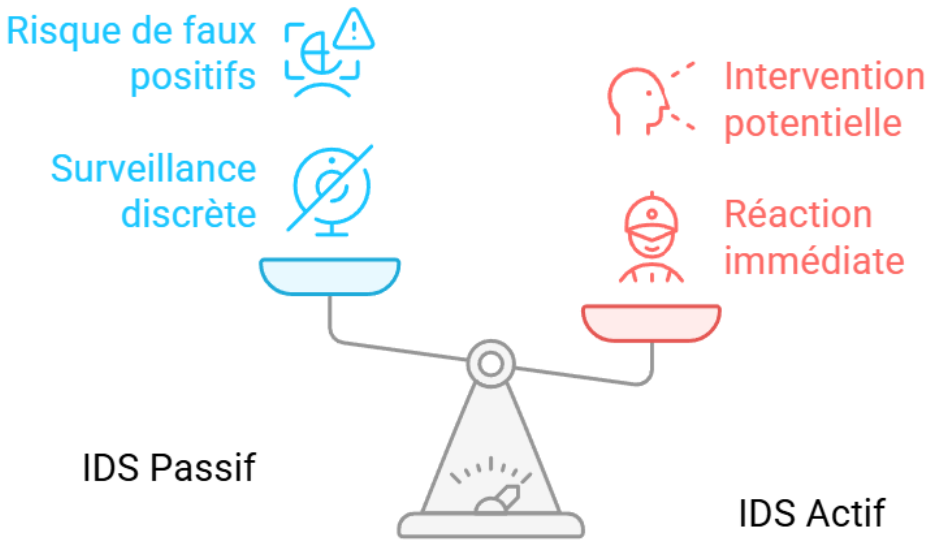


Figure 1.6: Schémas présente le Comportement de réaction

1.3.6.4 L'emplacement d'IDS

Les IDS ne fonctionnent pas tous de la même manière. Certains, appelés NIDS, surveillent la sécurité au niveau du réseau en analysant les paquets de données qui y circulent. D'autres, comme les HIDS, se focalisent sur la sécurité des machines individuelles en examinant les activités du système d'exploitation ou des applications installées localement. Enfin, il existe des IDS hybrides qui combinent les deux approches pour offrir des alertes plus précises et adaptées.

HIDS (Basé sur l'hôte)

Il surveille et collecte les caractéristiques des hôtes qui contiennent des informations sensibles, des serveurs qui exécutent des services publics, ainsi que des activités suspectes.

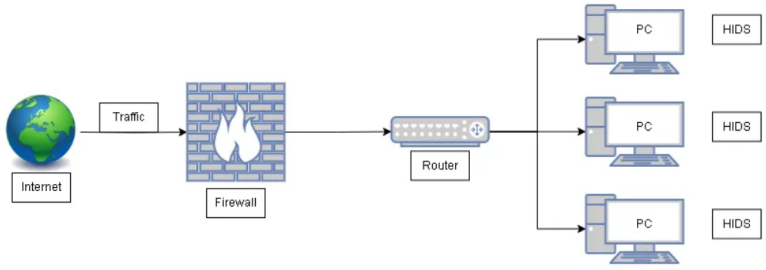


Figure 1.7: un schéma de HIDS

NIDS (Basé sur le réseau)

IDS basée sur le réseau est un logiciel de sécurité informatique surveillant constamment le trafic sur un réseau dans le but de repérer, ou de bloquer, les manifestations suspectes. Le NIDS, qui est déployé au sein du réseau, inspecte les paquets de données, bloquant ainsi les attaques et produisant des rapports pour avertir le personnel de sécurité du réseau. Même principe que celui des barrières infranchissables avec un gardien qui surveille et intercepte l'intrus (Rashid, 2020).

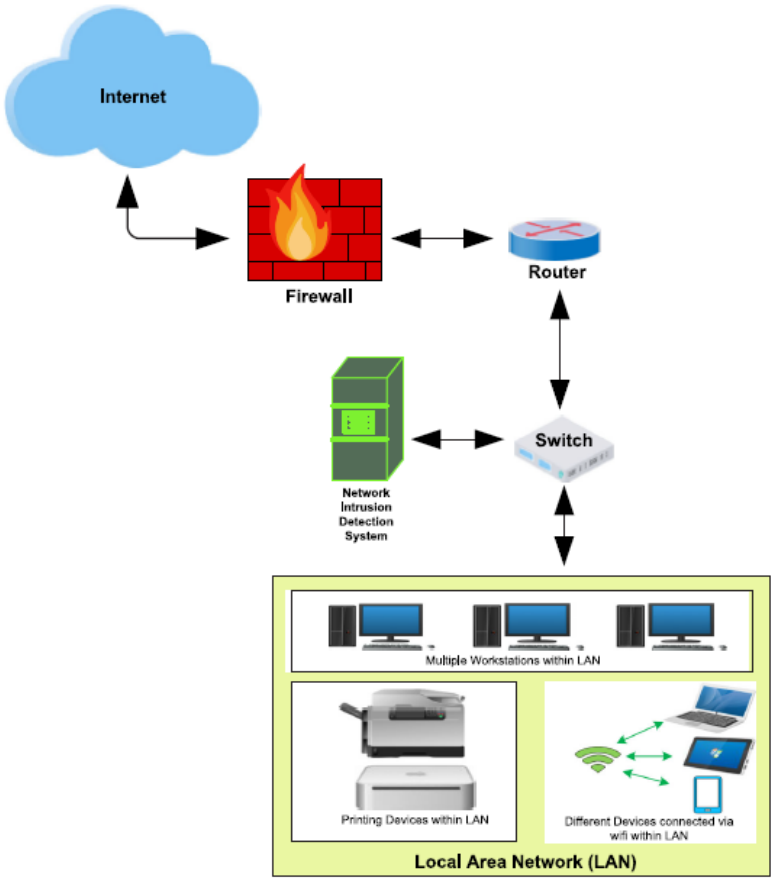


Figure 1.8: un schéma de NIDS. (Ahmad, 2021)

1.3.6.5 La comparaison entre HIDS et NIDS

Table 1.2: La comparaison entre NIDS et HIDS (Yadav, 2013).

	NIDS	**HIDS**
Résidence	L'ordinateur ou l'application reliée à un segment du réseau d'une organisation.	Un ordinateur ou un serveur spécifique , désigné comme hôte.
Outils (Exemples)	Snort, Cisco NIDS et Netprowler.	eTripwire,Cisco HIDS, et Symantec ESM.
Dispositif de travail	Le NIDS emploie un port de surveillance lorsqu'il est positionné à proximité d'un dispositif réseau tel qu'un concentrateur ou un commutateur. Ce port permet de visualiser tout le trafic transitant par l'appareil.	Capable de surveiller les bases de données de configuration du système, y compris les registres de Windows et les fichiers de configuration tels que .ini, .cfg et .dat .
Principe de fonctionnement	Il repose sur la correspondance des signatures, comparant les schémas d'attaque aux signatures connues dans sa base de données.	Fonctionnant sur le principe de la gestion de la configuration et des changements, une alerte est déclenchée lorsque les attributs d'un fichier sont modifiés, que de nouveaux fichiers sont créés ou que des fichiers existants sont supprimés.
Protection marche/arrêt(LAN)	Il assure votre protection sur le réseau local, mais pas au-delà de celui-ci.	Il vous protège aussi bien sur le terrain qu'en dehors.
Polyvalence	Moins.	Plus d'informations.
Prix	Coûteux.	Les HIDS sont plus abordables systèmes.
Formation requise	Plus d'informations.	Inférieur à NIDS.
Désactiver le facteur de risque	Le taux d'échec est beaucoup plus élevé.	Moins.

1.3.7 Dataset

Les Datasets de détection d'intrusion, générés à partir de traces de trafic réseau réel, sont largement utilisés par les chercheurs pour évaluer les performances des IDS afin de collecter des informations provenant de diverses sources, telles que les flux de trafic réseau. Ces flux contiennent des données sur les hôtes c'est une collection de données, le comportement des utilisateurs et les configurations système. Ces informations sont cruciales pour analyser les schémas d'attaque et détecter les activités anormales liées aux différentes attaques réseau. L'activité réseau est généralement collectée via un routeur ou un commutateur de réseau. Une fois le trafic réseau entrant et sortant capturé, une analyse des

flux est réalisée pour comprendre le comportement du réseau. Cette analyse consiste à examiner des éléments tels que l'adresse IP source, l'adresse IP de destination, le numéro de port source, le numéro de port de destination et le type de services réseau. Parmi les plus connue Dataset (Thakkar, 2020):

- **DARPA (1998):** Créé par le Lincoln Laboratory du MIT dans le cadre d'un projet financé par la DARPA.(Brugger & Chow, 2007)
- **KDD Cup'99:** Le jeu de données KDD Cup'99 est l'un des Datasets les plus renommés et couramment utilisés pour les IDS. Il contient environ cinq millions d'enregistrements pour l'entraînement et deux millions pour les tests. Chaque enregistrement est composé de 41 caractéristiques ou attributs distincts et est étiqueté comme normal ou comme une attaque. Les attaques sont classées en quatre catégories : DOS, sondes, attaques de type R2L et attaques de type U2L.(Bay, 1999)
- **Kyoto 2006+:** Le jeu de données Kyoto 2006+ a été élaboré à partir des enregistrements de trafic réseau obtenus grâce au déploiement de honeypots, de capteurs darknet, de serveurs de messagerie, de robots d'exploration web et d'autres mesures de sécurité réseau mises en place par l'Université de Kyoto. Le jeu de données le plus récent inclut les enregistrements de trafic de 2006 à 2015. Chaque enregistrement comporte 24 caractéristiques statistiques, dont 14 sont dérivées du jeu de données KDD Cup'99, tandis que les 10 restantes sont des caractéristiques supplémentaires.(Song et al., 2011)
- **NSL-KDD:** Il s'agit de la version révisée et améliorée de Dataset de la KDD Cup'99, avec plusieurs aspects supprimés. Cet Dataset comporte 41 caractéristiques, les attaques étant classées en quatre catégories, comme spécifié dans la KDD Cup'99.(Tavallaee et al., 2009)
- **UNSW-NB15:** Cet Dataset, élaboré par l'Australian Center for Cyber Security, comprend des millions d'enregistrements et un total de 49 caractéristiques. Ces caractéristiques sont extraites à l'aide des outils Bro-IDS, Argus, ainsi que de certains algorithmes nouvellement développés. Les types d'attaques inclus dans cet ensemble de données sont les suivants : vers, shellcode, reconnaissance, scans de ports, génériques, portes dérobées, DOS, exploits et fuzzers.(Moustafa & Slay, 2015)
- **CIC-IDS2017:** Le Dataset CICIDS2017, élaboré par CIC en 2017, intègre des flux de trafic réseau normaux ainsi que des attaques réelles actualisées. Le trafic est analysé par CICFlowMeter, qui utilise des informations basées sur les horodatages, les adresses IP source et destination, les protocoles et les types d'attaques. Cet Dataset inclut des scénarios d'attaque courants tels que les attaques par force brute, HeartBleed, botnet, DOS, DDOS, attaques par Internet et infiltrations.(Sharafaldin et al., 2018)
- **CSE-CIC-IDS2018:** Le Dataset CSE-CIC-IDS2018, élaboré conjointement par le CST et CIC en 2018, crée des profils d'utilisateurs représentant abstraitement divers événements. Pour générer cet Dataset, ces profils sont combinés avec un ensemble unique de caractéristiques. Il inclut sept scénarios d'attaque distincts : force brute, Heartbleed, botnet, DOS,DDOS, attaques Web et infiltration du réseau de l'intérieur.(G. Karatas et al., 2020)

Table 1.3: Résumé des ensembles de données de référence publics (Ahmad, 2021).

Dataset	**Année**	**Types d'attaques**	**Attaques**
KDD Cup'99	1998	4	DoS, Probe, R2L, U2R
Kyoto 2006+	2006	2	Attaques connues, inconnues
NSL-KDD	2009	4	DoS, Probe, R2L, U2R
UNSW-NB15	2015	9	Backdoors, DoS, Exploits, Fuzzers, génériques, balayages de ports, reconnaissance, Shellcode, vers
CIC-IDS2017	2017	14	Force brute, HeartBleed, Botnet, DoS, DDoS, Web, Infiltration
CSE-CICIDS2018	2018	15	HeartBleed, DoS, Botnet, DDoS, BruteForce, Infiltration, Web.

1.3.8 Les mesures de performances de IDS

Cette sous-section décrit les principaux paramètres d'évaluation utilisés pour mesurer l'efficacité des IDS. Toutes les mesures d'évaluation reposent sur divers attributs de la matrice de confusion, qui fournit des informations sur les classes réelles et prédites (X. L. Deng, 2016).

Table 1.4: Matrice de confusion (X. L. Deng, 2016)

		Prediction de la classe	
		Classe positive (Attaque)	**Classe négative (Normal)**
Class Actual	**Classe positive (Attaque)**	Vrai positive (TP)	Faux positive (FP)
	Classe négative (Normal)	Faux négative (FN)	Vrai négative (TN)

- **Vrai positif (TP):** nombre d'intrusions correctement détectées.
- **Vrai négatif (TN):** nombre de non-intrusions correctement détectées.
- **Faux négatif (FN):** nombre d'intrusions mal détectées.
- **Faux positif (FP):** nombre de non-intrusions mal détectées.

La diagonale de la matrice de confusion représente les prédictions correctes, tandis que les éléments hors diagonale indiquent les erreurs de prédiction d'un classificateur donné. Le tableau 1.4 illustre ces attributs de la matrice de confusion. De plus, les différentes mesures d'évaluation utilisées dans les études récentes incluent:

- **Précision:** C'est le rapport entre les attaques correctement identifiées et l'ensemble des échantillons prédits comme des attaques.

$$\text{Précision} = \frac{TP}{TP + FP}$$

- **Recall:** Le taux de détection, également connu sous le nom de taux de détection des intrusions, est le rapport entre le nombre d'échantillons correctement identifiés comme des attaques et le nombre total d'échantillons qui sont effectivement des attaques.

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

- **Taux de fausses alarmes:** Le taux de faux positifs, également connu sous ce nom, est calculé en divisant le nombre d'échantillons d'attaque incorrectement identifiés par le nombre total d'échantillons normaux.

$$\text{Taux De Fausses Alarmes} = \frac{FP}{FP + TN}$$

- **Taux de vrais négatifs:** est le rapport entre le nombre d'échantillons normaux correctement identifiés et le nombre total d'échantillons normaux.

$$\text{Taux De Vrais Négatifs} = \frac{TN}{TN + FP}$$

- **Taux de fausses négatifs:** est le rapport entre le nombre d'échantillons malveillente correctement identifiés et le nombre total d'échantillons malveillente.

$$\text{Taux De fausses Négatifs} = \frac{FN}{FN + FP}$$

- **Taux d'Exactitude:** La précision de détection, qui est le rapport entre le nombre d'instances correctement classées et le nombre total d'instances, est une mesure de performance pertinente uniquement lorsque le jeu de données est équilibré.

$$\text{Taux d'Exactitude} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **F-Mesure:** Le score F1 est calculé en prenant la moyenne harmonique de la précision et du rappel. Cela signifie qu'il évalue la performance d'un système en intégrant à la fois la précision et le rappel.

$$F - Mesure = 2 * \frac{\text{Précision} + Recall}{\text{Précision} * Recall}$$

- **ROC (Receiver Operating Characteristic) :** Le ROC est un outil d'évaluation de la performance d'un modèle de classification, généralisé aux problèmes multi-classes. Plutôt que de mesurer la discrimination entre deux classes (comme en binaire), il mesure la capacité du modèle à distinguer chacune des classes des autres.

1.4 Conclusion

Dans ce chapitre, nous avons présenté les bases des systèmes de détection d'intrusion (IDS) en expliquant leur rôle dans la sécurité du réseau. Nous avons défini les intrusions, les types d'attaques et les propriétés de la sécurité, les catégories différentes d'IDS, y compris HIDS, NIDS et hybrides, ont également été fournies, accompagnées de leurs méthodes de détection spécifiques, notamment les signatures et l'analyse comportementale. Enfin, nous avons discuté des réactions réactives et actives des systèmes IDS aux menaces, et de leurs performances à cet égard. Ces concepts préparent à l'examen plus détaillé des IDS dans les chapitres à suivre.(Singh, 2014)

Chapitre 2

DEEP LEARNING & LES MÉCANISMES D'ATTENTION

2.1 Introduction

Ces dernières années, l'IA a pris une place centrale dans les débats publics et médiatiques. Que ce soit dans les journaux, les émissions de télévision ou les discussions en ligne, on parle beaucoup ML, DL. Dans ce contexte, l'IA est également devenue un outil précieux pour résoudre des problèmes complexes, comme la détection d'intrusions dans les SI. Les algorithmes d'IA, et en particulier ceux basés sur le DL, sont de plus en plus utilisés pour identifier des comportements anormaux ou malveillants dans les réseaux (Chollet, 2021). Ce chapitre vise à fournir ce cadre théorique, en expliquant les principes de DL. Nous aborderons également les mécanismes d'attention, qui jouent un rôle de plus en plus important dans les IDS's modernes.

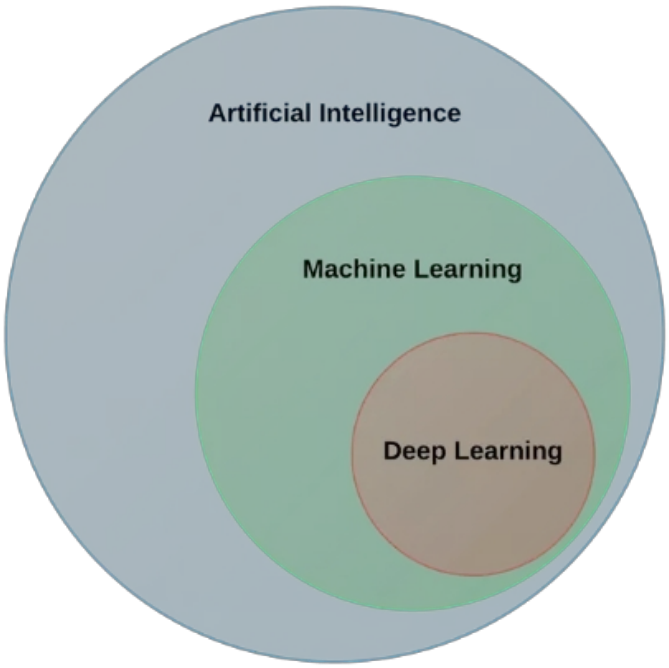


Figure 2.1: IA, ML et DL (Labed, 2018)

2.2 Machine Learning

Le ML repose sur une question clé : un ordinateur peut-il apprendre à accomplir une tâche spécifique sans instructions explicites, simplement en analysant des données ? Plutôt que de programmer manuellement des règles, peut-on permettre à l'ordinateur de les découvrir par lui-même ? ML, qui a vraiment pris son essor dans les années 1990, est devenu le sous-domaine le plus populaire de l'IA, grâce à des avancées en matériel informatique et à l'accès à de vastes Dataset's. Contrairement aux statistiques traditionnelles, ML traite souvent des Dataset's complexes et volumineux, pour lesquels les méthodes classiques ne sont pas adaptées. (Wagh, 2013).

2.2.1 Types de Machine Learning

Il existe plusieurs types de ML:

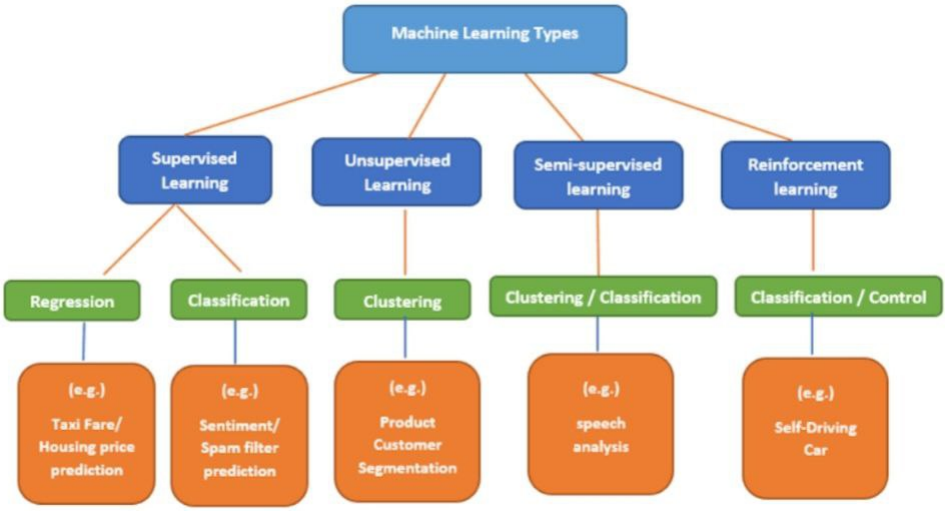


Figure 2.2: Types de ML (Soula, 2020)

2.2.1.1 Apprentissage Supervisé

L'apprentissage supervisé consiste à utiliser des données étiquetées pour apprendre à associer des entrées spécifiques à des sorties souhaitées. L'objectif est de développer une règle générale permettant de faire correspondre les entrées aux sorties. voici quelques-uns des algorithmes d'apprentissage supervisé les plus importants:

- k plus proches voisins (KNN)
- Régression linéaire
- Régression logistique
- Machines à vecteurs de support (SVMs)
- Arbres de décision et forêts aléatoires
- Réseaux neuronaux (NN)

L'apprentissage supervisé est généralement utilisé pour de la régression ou de la classification(DE MATTEIS et al., 2022) :

1. **Régression**
   La régression est une méthode utilisée pour prédire des valeurs continues, c'est-à-dire des nombres réels. Par exemple, cela peut servir à estimer la consommation électrique d'une installation ou à prévoir l'évolution du cours d'une action en bourse.

2. **Classification**
   La classification, quant à elle, consiste à attribuer une entrée à une catégorie spécifique parmi plusieurs options possibles. Par exemple, cela peut être utilisé pour reconnaître un chiffre écrit à la main sur une image ou pour déterminer si une tumeur est bénigne ou maligne.

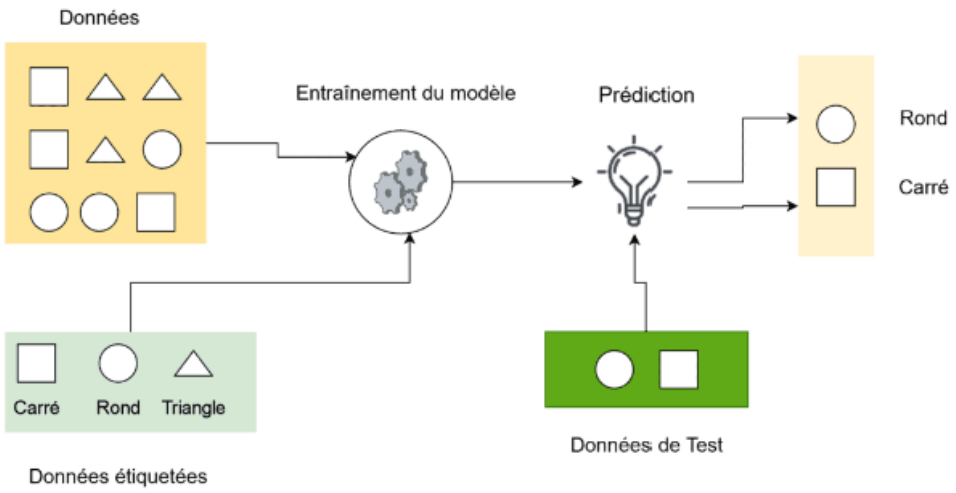


Figure 2.3: Un ensemble d'apprentissage étiqueté pour l'apprentissage supervisé

2.2.1.2 Apprentissage Non Supervisé

En apprentissage non supervisé, les données d'entraînement ne sont pas étiquetées. Le système tente d'apprendre de manière autonome, sans guide ni professeur, voici quelques-uns des principaux algorithmes d'apprentissage non supervisé(DE MATTEIS et al., 2022):

- **Regroupement**
  Le regroupement ('clustering') est une technique d'apprentissage non-supervisé qui consiste à organiser les données en groupes distincts. Chaque groupe rassemble des éléments ayant des caractéristiques similaires entre eux, tout en étant différents de ceux des autres groupes. L'objectif est de structurer les données en familles cohérentes sans utiliser d'étiquettes préalables. Par exemple, cela permet de classer des patients selon leurs profils pour prédire leurs réactions à des traitements.

- **Réduction de la dimensionnalité**
  La réduction de dimensionnalité est une technique qui simplifie les données en diminuant le nombre de caractéristiques (ou dimensions) à analyser. Bien qu'il soit souvent utile d'avoir beaucoup de données pour l'ML, un trop grand nombre de dimensions peut rendre les données complexes à visualiser ou à interpréter. Cette méthode permet de conserver les informations essentielles tout en éliminant les caractéristiques inutiles ou redondantes. Pour y parvenir, des algorithmes comme l'analyse en composantes principales (ACP) ou la décomposition en valeurs singulières (SVD) sont utilisés, garantissant que les données restent significatives malgré la réduction.

- **Apprentissage de règles d'association**
  les règles d'association sont des techniques qui permettent de trouver des liens entre les éléments dans des données. Elles fonctionnent sur le principe des règles "si-alors", révélant des associations fréquentes, comme des produits souvent achetés ensemble ou des symptômes liés à des maladies. Ces règles sont très utiles pour des applications comme les recommandations en ligne (ex : "Ceux qui ont acheté ce produit ont aussi acheté...") ou pour aider à l'analyse médicale. Des outils comme Apriori, Eclat ou FP-Growth sont souvent utilisés pour découvrir ces associations.

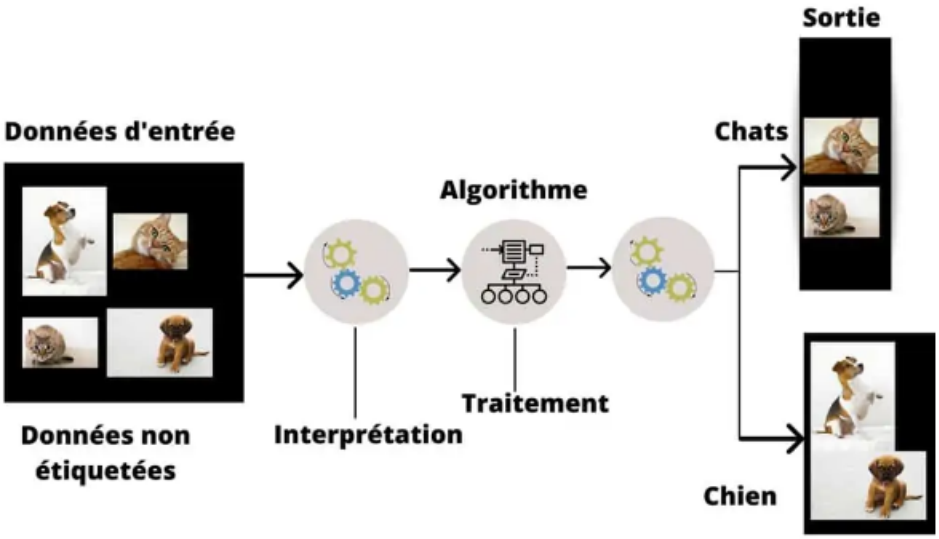


Figure 2.4: Un ensemble d'apprentissage non étiqueté pour l'apprentissage non supervisé

2.2.1.3 Apprentissage Par Renforcement:

L'apprentissage par renforcement se distingue nettement des autres méthodes d'apprentissage. Dans ce contexte, l'agent, qui est le système d'apprentissage, observe son environnement, choisit et exécute des actions, puis reçoit des récompenses ou des pénalités (récompenses négatives) en retour. L'objectif de l'agent est d'apprendre de manière autonome la meilleure stratégie, appelée politique, pour maximiser la récompense totale au fil du temps. Une politique définit l'action que l'agent doit entreprendre dans une situation donnée.

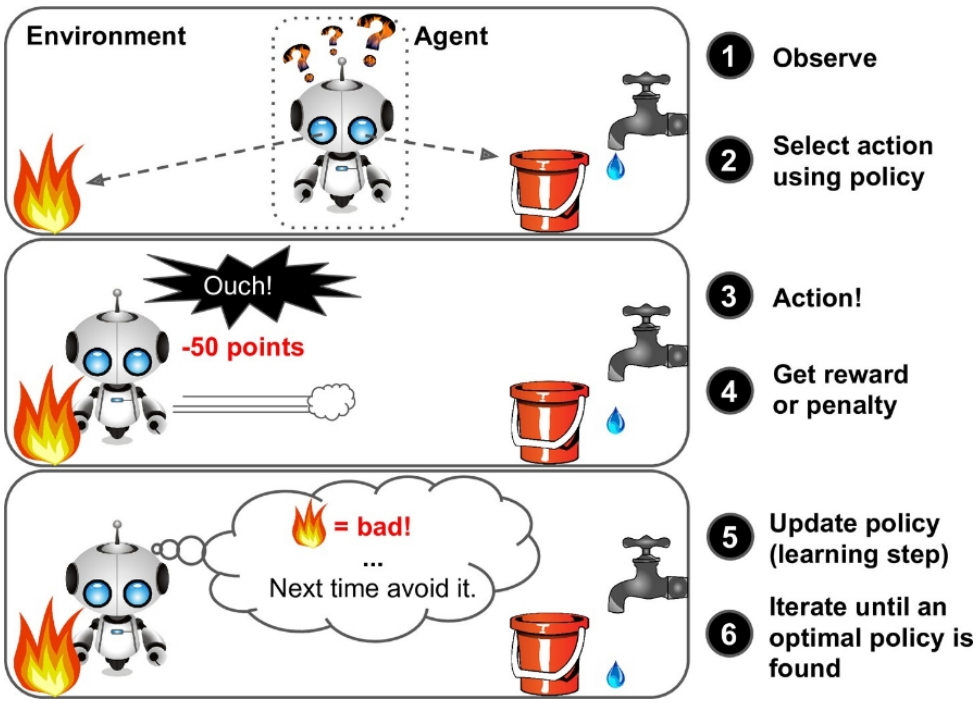


Figure 2.5: Apprentissage par renforcement

2.3 Deep Learning

2.3.1 Définition de Deep Learning

L'apprentissage profond, également connu sous le nom de DL, est une sous-catégorie des techniques de ML. Il a gagné en popularité grâce à ses performances remarquables dans diverses tâches d'IA, surpassant souvent les algorithmes de ML traditionnels. Les modèles de DL sont caractérisés par des architectures récentes qui exploitent plusieurs étapes de traitement non linéaire de l'information. Ces étapes permettent de traiter les données en couches hiérarchiques, où chaque couche reçoit et interprète les informations de la couche précédente, facilitant ainsi l'apprentissage des représentations de données (Chassagnon, 2020).

2.3.2 Principe de Fonctionnement

L'architecture des DNN est généralement organisée en plusieurs couches : une couche d'entrée, une ou plusieurs couches cachées, et une couche de sortie. Chaque paire de couches adjacentes est connectée par des poids. Les neurones d'une même couche, souvent appelés "nœuds", ne sont pas interconnectés entre eux. Le DL repose sur un grand nombre de neurones non linéaires disposés en plusieurs couches de traitement. Ces neurones extraient et transforment les valeurs des variables d'entrée pour créer plusieurs niveaux d'abstraction, permettant ainsi de représenter les données de manière plus complexe. Le DNN consiste à optimiser les paramètres de poids et de biais entre les couches adjacentes. Ce processus évalue la précision du modèle et permet de l'ajuster pour mieux répondre aux besoins des données d'apprentissage. Lorsque le modèle atteint une précision maximale avec des paramètres optimaux, il peut être généralisé pour les données réelles. La quantité et la qualité des données d'entraînement déterminent le degré d'apprentissage et donc la précision des modèles obtenus (L. Deng, 2014).

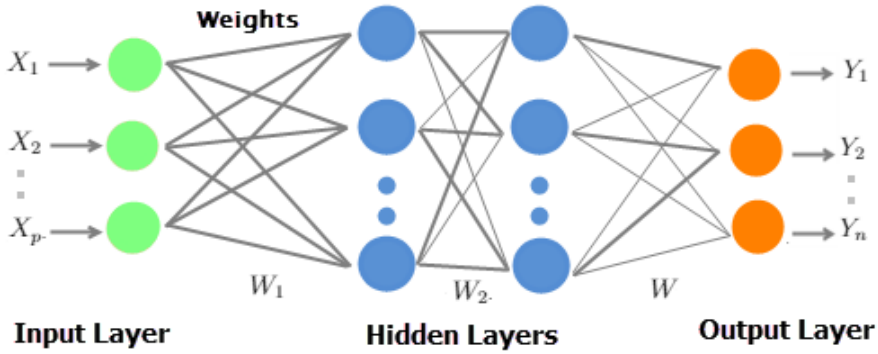


Figure 2.6: Architecture d'un model DL (W. Liu, 2017)

2.3.3 Importance de Deep Learning

Le ML traditionnel montre des limites avec les données volumineuses et complexes, notamment dans les domaines du traitement du langage et de la vision par ordinateur. En revanche, le DL révolutionne l'analyse en proposant une extraction automatique des caractéristiques.

Contrairement aux modèles classiques qui nécessitent une intervention manuelle fastidieuse, les algorithmes de DL identifient naturellement les traits pertinents avec une intervention minimale du programmeur (Figure 2.7). Cette capacité leur permet d'atteindre des niveaux de précision exceptionnels, notamment dans les tâches de vision par ordinateur (DataScientest, 2020).

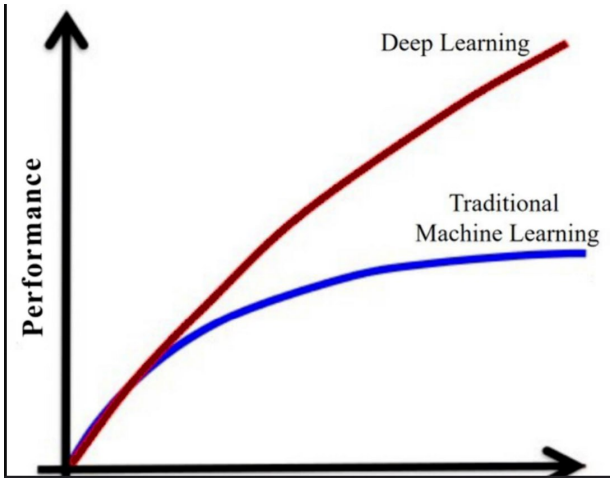


Figure 2.7: Performance de DL et de ML illustrée.

2.3.4 Réseaux de neurones Artificiel (ANN)

Les ANN reproduisent le mécanisme d'apprentissage des organismes biologiques. Le système nerveux humain est composé de cellules appelées neurones. Ces neurones sont interconnectés par des axones et des dendrites, et les zones de connexion entre les axones et les dendrites sont appelées synapses. Ces connexions sont illustrées dans (Figure 2.8) . Les dendrites reçoivent des signaux d'entrée provenant d'autres neurones via les

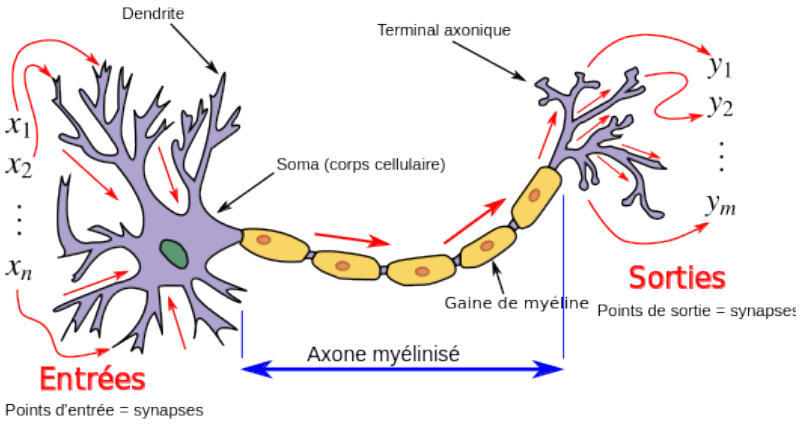


Figure 2.8: Structure du neurone biologique

synapses. Ces signaux sont ensuite intégrés dans le corps cellulaire, et si leur somme dépasse un certain seuil, le neurone déclenche un potentiel d'action le long de son axone. Ce signal électrique parcourt l'axone jusqu'à ses terminaisons, où il est transmis à d'autres neurones du système nerveux (Pfeiffer & Pfeil, 2018).

2.3.5 Réseaux de neurones FeedForwards (FFNNs)

Le FFNN est un type d'ANN qui traite l'information de manière unidirectionnelle, de l'entrée vers la sortie. Il est composé d'une couche d'entrée, d'une ou plusieurs couches cachées, et d'une couche de sortie. Pendant l'entraînement, les poids et les biais sont ajustés pour minimiser la différence entre la sortie prédite et la sortie réelle.(Choutri et al., 2023)

2.3.5.1 Architecture d'un FFNNs

Pour construire un réseau de neurones à une seule couche, les neurones sont empilés les uns sur les autres au sein d'une même couche, comme illustré dans (Figure 2.9) Dans cette architecture, chaque valeur d'entrée de la couche d'entrée est transmise à tous les neurones de la couche cachée. Le calcul de la somme des entrées pondérées et du biais, ainsi que l'application de la fonction d'activation, sont effectués indépendamment pour chaque neurone de la couche cachée.

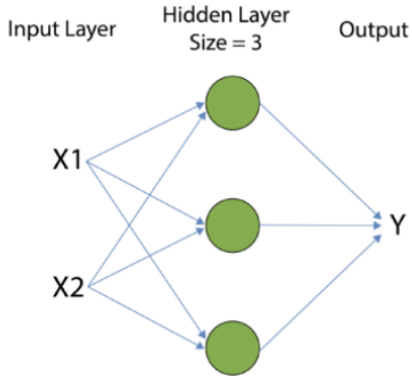


Figure 2.9: Réseau de neuron avec un seul couche cachée

Les réseaux neuronaux multicouches peuvent également être construits en empilant plusieurs couches de nœuds de traitement en série. (Figure 2.10) illustre un réseau neuronal à deux couches avec une entrée bidimensionnelle.

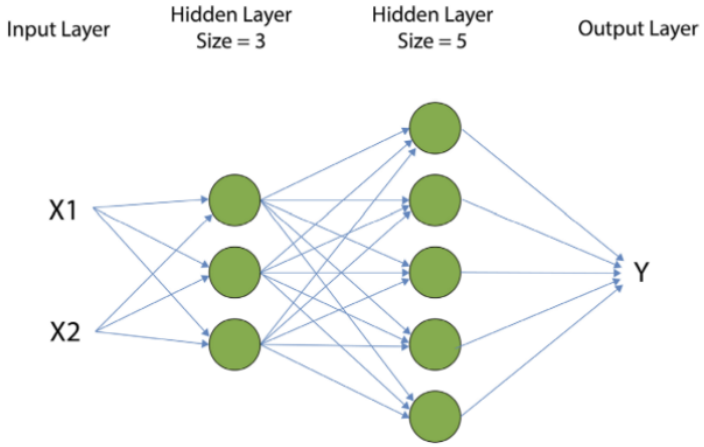


Figure 2.10: Réseau de neuron avec deux couches cachée

En DL, le nombre de neurones dans la couche d'entrée correspond au nombre de caractéristiques des données d'entrée, tandis que le nombre de neurones dans la couche de sortie correspond aux dimensions des données de sortie. En revanche, la taille des couches cachées, c'est-à-dire le nombre de neurones qu'elles contiennent, doit être déterminée par le développeur. Une couche cachée plus grande confère au modèle une plus grande flexibilité et une meilleure capacité à apprendre des fonctions complexes. Cependant, cette flexibilité accrue nécessite davantage de données pour l'entraînement ainsi qu'une puissance de calcul plus élevée. Les paramètres que le développeur doit choisir sont appelés hyperparamètres. Ils incluent, entre autres, le nombre de couches, le nombre de neurones par couche, le nombre d'époques d'entraînement et la fonction de perte utilisée. Les FFNNs peuvent rencontrer des difficultés à prédire correctement les sorties, en particulier pour des problèmes complexes. Dans ces cas, il est nécessaire d'ajuster les poids et les biais du réseau afin de minimiser l'écart entre les prédictions et les valeurs réelles. C'est là qu'intervient l'algorithme de rétropropagation (backpropagation), qui permet de modifier ces paramètres de manière itérative (Figure 2.11).

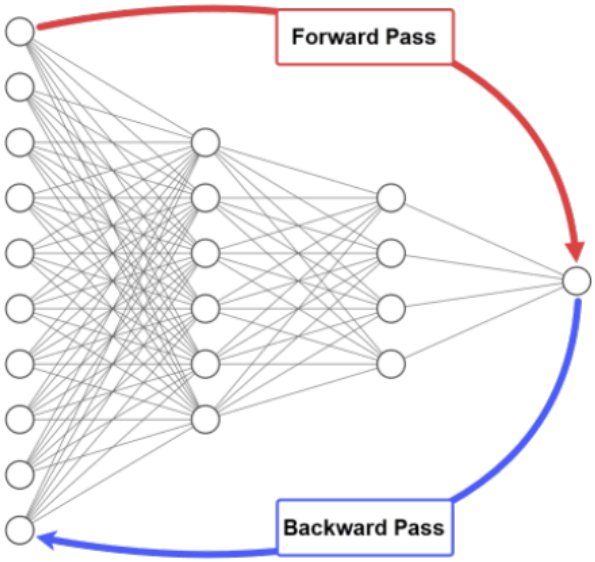


Figure 2.11: Propagation avant et rétropropagation

2.3.5.2 Rétropropagation

La rétropropagation est un algorithme d'apprentissage largement utilisé qui permet au réseau d'ajuster ses poids et ses biais lors de l'entraînement en propageant l'erreur de la couche de sortie vers la couche d'entrée (Figure 2.12). Ce processus repose sur le calcul des dérivées partielles de l'erreur par rapport aux poids et aux biais de chaque neurone, en appliquant la règle de dérivation en chaîne.Durant la phase d'entraînement, la rétropropagation est appliquée de manière répétée sur un ensemble de paires entrée-sortie. Les poids et biais du réseau sont ajustés en fonction des dérivées calculées, dans le but de minimiser l'écart entre la sortie prédite et la sortie réelle .Ainsi la rétropropagation permet au réseau d'apprendre et d'améliorer sa capacité à prédire avec précision la sortie pour une entrée donnée (Bouguerne et al., 2022).

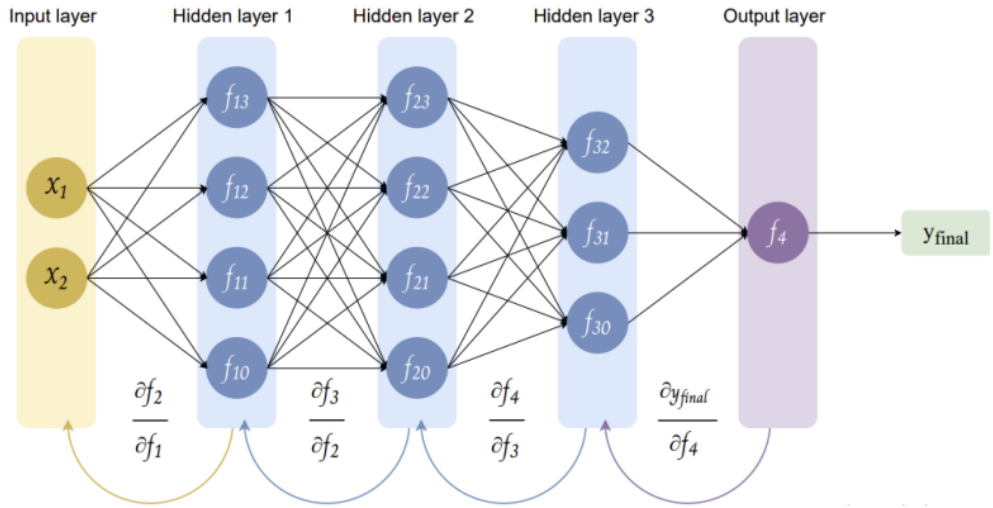


Figure 2.12: Rétropropagation

2.3.6 Réseaux de neurones Convolutionnels (CNN)

Les CNN sont des réseaux multicouches inspirés du cortex visuel des animaux. Leur architecture est particulièrement adaptée au traitement et à la reconnaissance d'images. À l'origine, les CNN étaient utilisés pour la lecture de caractères manuscrits, comme les codes postaux. Les premières couches d'un réseau profond identifient des caractéristiques simples, telles que les bords, tandis que les couches ultérieures réassemblent ces informations en caractéristiques de plus haut niveau.
L'architecture des CNN est multicouche et se concentre sur l'extraction des caractéristiques d'une entrée plutôt que sur sa classification directe. Par exemple, une couche convolutive reçoit des champs réceptifs de l'image et extrait des informations pertinentes. Ensuite, une étape de pooling réduit la dimensionnalité des caractéristiques extraites tout en conservant les données les plus cruciales, souvent par pooling max. Après plusieurs étapes de convolution et de pooling, une perception multicouche entièrement connectée est alimentée. La couche de sortie finale du réseau est constituée de nœuds qui identifient les éléments de l'image. Le réseau est entraîné par rétropropagation et a été appliqué avec succès à la reconnaissance vidéo, au traitement du langage naturel et à diverses tâches de classification.
Dans le domaine de la détection d'intrusions, les CNN peuvent extraire des modèles spatiaux des données de trafic réseau, comme les en-têtes de paquets ou les informations de charge utile (Chattopadhyay, 2022).

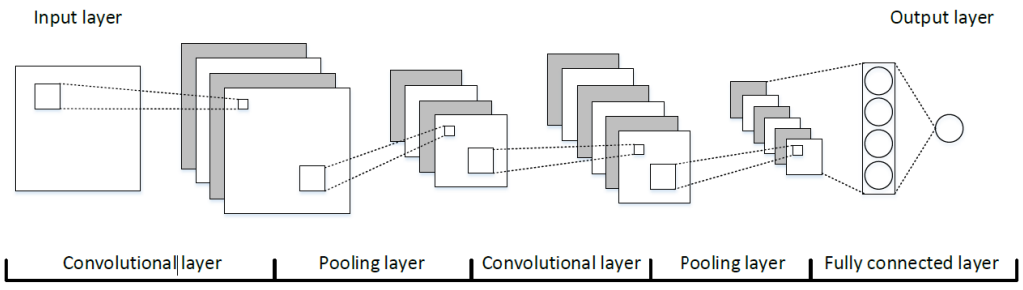


Figure 2.13: la structure d'un CNN (Li, 2019)

2.3.7 Réseaux de neurones récurrents (RNN)

Contrairement aux réseaux neuronaux traditionnels qui traitent chaque information indépendamment, les RNNs fonctionnent comme notre mémoire. Ils ont une "mémoire interne" qui leur permet de se souvenir et d'utiliser des informations précédentes pour comprendre et traiter la donnée actuelle. Ces réseaux intègrent des boucles dans leur architecture, ce qui leur permet de conserver et de réutiliser des informations. C'est un peu comme si chaque neurone gardait un petit carnet de notes où il inscrit ce qu'il a appris, et s'en sert pour mieux comprendre la prochaine information.
Ainsi, la décision finale d'un RNN résulte non seulement de l'information immédiate, mais aussi de tout ce qu'il a "mémorisé" avant - exactement comme nous, humains, qui raisonnons en puisant dans notre expérience. Les réseaux récurrents sont parfaits pour les situations où l'ordre des éléments est aussi important que leur présence. Ils sont donc idéaux pour traiter des données séquentielles, comme les textes ou les données temporelles. Leur capacité à mémoriser et utiliser des informations passées les rend particulièrement efficaces pour ces tâches.(Zitouni, 2024)

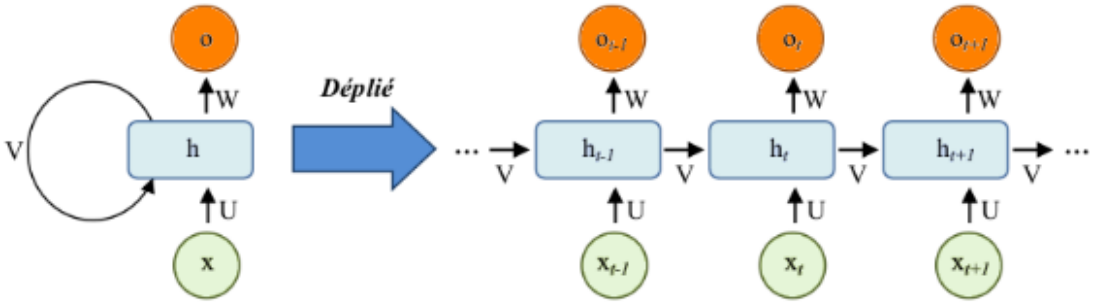


Figure 2.14: Architecture d'un model RNN

2.3.8 Unités de mémoire à court terme (LSTM)

Les LSTMs sont des réseaux neuronaux récurrents conçus pour résoudre les problèmes de fiabilité à long terme des RNN classiques. Ils utilisent des unités de mémoire à la place des neurones de la couche cachée, avec des portes d'entrée, d'oubli et de sortie pour gérer les informations à chaque pas de temps. Grâce à leur capacité à apprendre les corrélations temporelles, les LSTM sont utilisés dans des domaines comme la traduction linguistique et la reconnaissance vocale. Ils sont également efficaces pour prévoir la consommation électrique en analysant les modèles de consommation et en stockant les états en mémoire pour faire des prévisions précises.
(Figure 2.15) montre que la porte d'entrée agit comme un filtre, bloquant toute entrée non pertinente pour l'unité. La porte d'oubli permet à l'unité d'oublier les informations précédemment stockées dans sa mémoire, ce qui lui permet de se concentrer sur les nouvelles informations reçues. La porte de sortie décide si le contenu de la cellule de mémoire doit être révélé ou non à la sortie de l'unité LSTM. Cette porte, dotée d'une fonction d'activation sigmoïde, ne peut sortir qu'une valeur comprise entre 0 et 1, limitant ainsi la sortie de l'unité LSTM (Abumohsen, 2023)

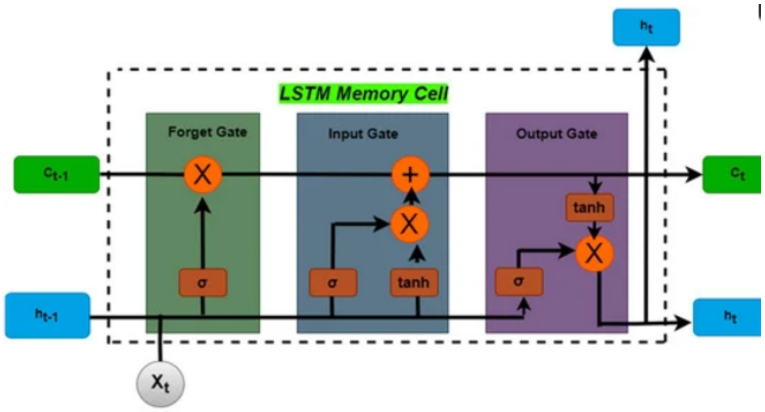


Figure 2.15: L'architecture d'un modèle LSTM

2.3.9 Modèle de l'unité récurrente de la porte d'entrée (GRU)

Les GRU sont une méthode de gating dans les réseaux neuronaux récurrents, introduite en 2014. Similaires aux LSTM avec une porte d'oubli, les GRU ont moins de paramètres car elles n'ont pas de porte de sortie. Les GRU ont surpassé les LSTM dans des tâches spécifiques comme la modélisation de la musique polyphonique, le traitement du signal de la parole et le traitement du langage naturel. Elles sont particulièrement performantes sur des ensembles de données plus petits et moins fréquents. Les GRU sont composées de trois portes : une porte de mise à jour, une porte de réinitialisation et une sortie temporaire. Voici les symboles associés :

1. La variable x_t représente l'entrée du réseau à l'instant t.
2. Les variables h_t et h_{t-1} sont des vecteurs d'information reflétant respectivement la sortie temporaire et la sortie de la couche cachée à l'instant t et $t-1$.
3. Les variables z_t et r_t sont des vecteurs de porte représentant la sortie de la porte de mise à jour et de la porte de réinitialisation à l'instant tt.
4. Les fonctions d'activation sigmoïde et tanh sont représentées par $\sigma(x)$ et $tanh(x)$ respectivement.

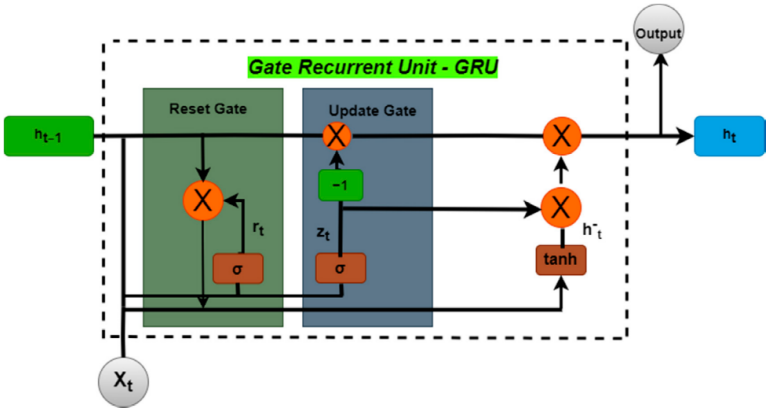


Figure 2.16: L'architecture d'un modèle GRU

2.4 Les Mécanismes d'Attention

Le mécanisme de l'attention s'inspire du fonctionnement humain : les gens se concentrent souvent sur des zones spécifiques d'une image ou sur certains mots d'une phrase pour mieux comprendre l'information. Ce principe permet d'optimiser les ressources limitées. L'attention fonctionne en reliant des requêtes à des paires clé-valeur pour produire une sortie. Les requêtes, les clés et les valeurs sont toutes représentées sous forme de vecteurs. Il s'agit d'auto-attention ou d'attention croisée, ces éléments sont générés à partir de différentes entrées. En pratique, l'attention permet de reconstruire les requêtes en utilisant les informations des valeurs, en se basant sur la similarité entre les requêtes et les clés. Cela permet de déterminer dynamiquement quelles parties des données sont les plus importantes pour la tâche en cours. Ce mécanisme n'est pas une méthode unique, mais un concept basé sur deux aspects essentiels : l'adressage et le calcul. Dans un modèle d'attention, une entrée peut être représentée par X=[x1,x2,...,Xn] où correspond soit aux étapes temporelles pour des données en trois dimensions, soit au nombre de caractéristiques dans un vecteur en une dimension. L'adressage, également appelé fonction de score d'alignement, aide à identifier les parties pertinentes de ces données (C. L. Liu, 2020).

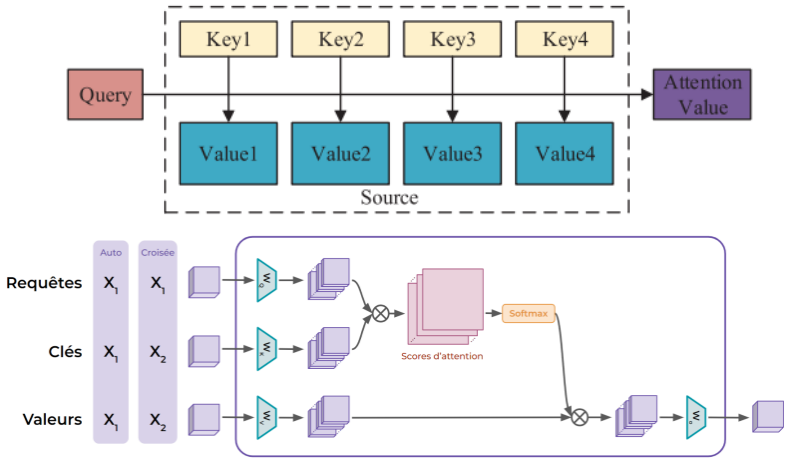


Figure 2.17: Des schémas de fonctionnement du mécanisme d'attention.

2.4.1 Types de mécanismes d'attention

2.4.1.1 L'Auto-Attention (self-attention)

L'auto-attention est un mécanisme qui permet de comprendre les relations entre les différents éléments d'une même séquence. Concrètement, il prend une séquence de vecteurs (par exemple, x1, x2, ..., xn) et analyse comment chaque élément (xi) interagit avec les autres. En sortie, il génère une nouvelle séquence de vecteurs (z1, z2, ..., zn) qui intègre ces relations, enrichissant ainsi chaque élément avec des informations contextuelles.
Utilise les vecteurs Query (Q), Key (K), et Value (V) pour pondérer l'information.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

2.4.1.2 Multi-Head Attention

Il est démontré qu'appliquer plusieurs fois un mécanisme d'attention sur des requêtes, des clés et des valeurs projetées linéairement est plus efficace qu'utiliser un seul mécanisme d'attention. Comme le montre (Figure 2.18) l'attention multi-têtes repose sur la projection linéaire des requêtes, des clés et des valeurs à l'aide de h projections linéaires distinctes, chacune apprise indépendamment. Ainsi, les requêtes, clés et valeurs de dimension d_{model} sont transformées en des espaces vectoriels de dimensions d_k et d_v, respectivement. Le mécanisme d'attention est ensuite appliqué en parallèle sur chacune de ces projections. Les résultats obtenus sont enfin concaténés et projetés pour former la sortie finale. Cette méthode, appelée attention multi-têtes, permet de capturer des informations plus variées et plus riches. En résume plusieurs mécanismes d'attention en parallèle pour capturer des relations variées dans les données (Chaudhari, 2021).

$$\text{Multi-Head Attention}(Q, K, V) = \text{Concat}(head_1, head_2, \ldots, head_h)W^O$$

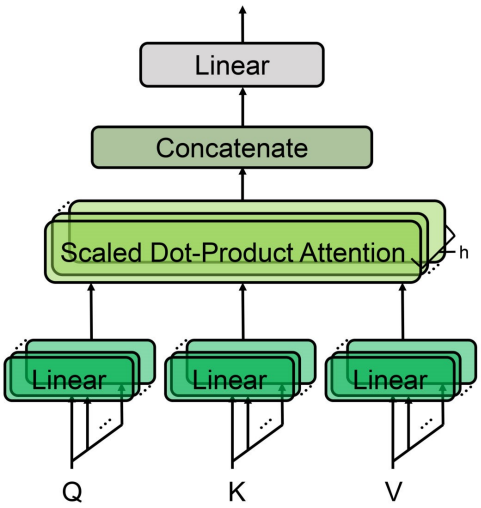


Figure 2.18: Schéma de l'attention multi-tête

Table 2.1: Comparison entre Self-attention et Multi-head Attention.

Critère	Self-Attention	Multi-Head Attention
Définition	Mécanisme qui permet à chaque mot d'une séquence de prêter attention aux autres mots de cette même séquence.	Extension du self-attention qui applique plusieurs mécanismes de self-attention en parallèle.

Critère	Self-Attention	Multi-Head Attention
Mode de fonctionnement	Un seul mécanisme d'attention est appliqué sur l'ensemble des données.	Plusieurs "têtes" d'attention sont utilisées simultanément, chacune capturant différentes relations entre les mots.
Avantages	Permet de capturer les relations entre les mots efficacement.	Offre une compréhension plus riche du contexte grâce à la diversité des représentations..
Inconvénients	Peut être limité dans la capture des différents types de relations.	Plus coûteux en termes de calcul et de mémoire, car plusieurs têtes sont exécutées en parallèle.
Utilisation	Utile pour des tâches où une seule perspective contextuelle est suffisante.	Recommandé pour des modèles avancés comme les Transformers, qui nécessitent une compréhension plus fine du contexte.

2.4.1.3 L'attention par produit scalaire

Une autre technique qui se base sur les produits de points pour mesurer les similitudes entre les états du générateur et des entrées. Ces méthodes permettent d'orienter l'attention vers les éléments les plus pertinents lors du traitement des données.

$$\alpha_t = \text{softmax}(W_a h_t)$$

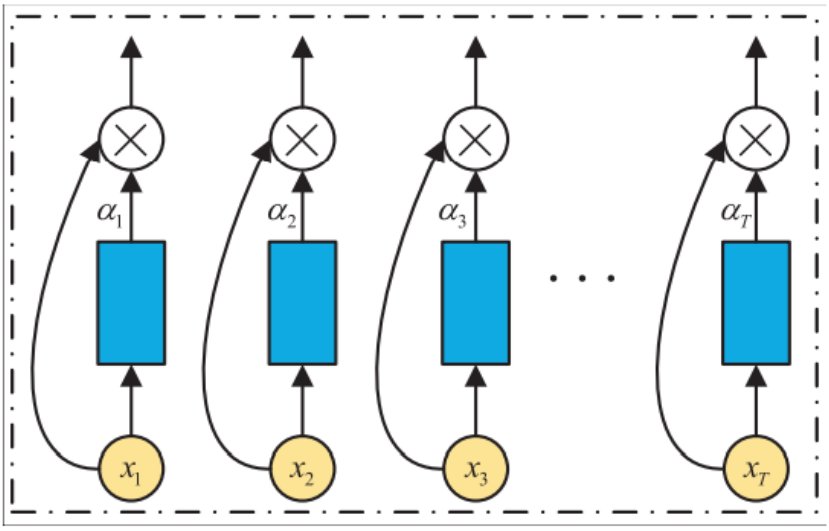


Figure 2.19: Comment l'attention se concentre sur une zone précise.

Dans ce processus, W_a représente la matrice des poids et h_t l'état caché actuel. L'attention par produit scalaire repose sur trois éléments principaux : une matrice de clés K, une matrice de valeurs V, et un vecteur de requête q. Le vecteur d'attention est obtenu en combinant ces trois composants. Ce mécanisme est illustré visuellement à la (Figure 2.20).

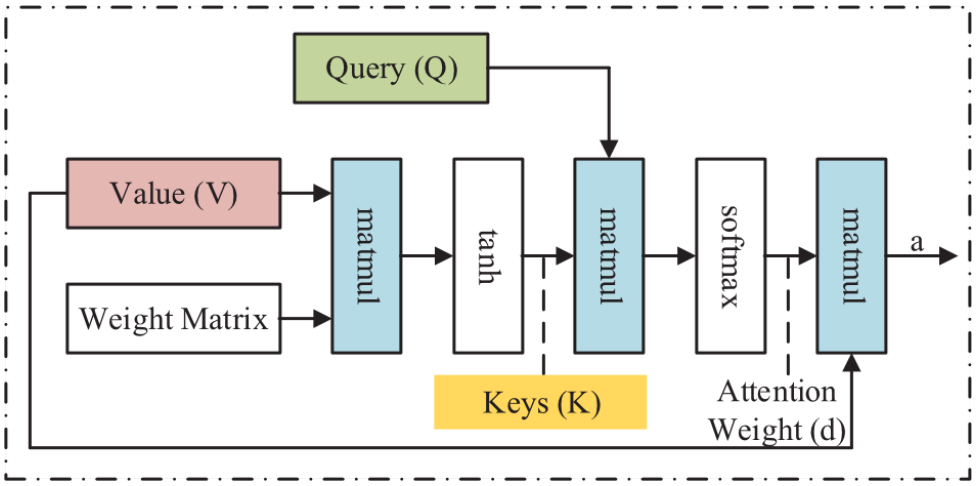


Figure 2.20: Explication visuelle de l'attention basée sur le produit scalaire.

On obtient tout d'abord la matrice des clés:

$$K = \tan(VW^a)$$

W^a est une matrice de poids initialisée aléatoirement. Ensuite, pour obtenir la matrice clé actuelle, chaque vecteur de requête est comparé aux vecteurs de la matrice clé. Cela permet de mesurer leur similarité et de générer un vecteur de probabilités normalisé, appelé vecteur de poids d.

$$d = \text{softmax}(qK^T)$$

Pour finir, on peut calculer le vecteur d'attention en procédant ainsi:

$$a = dV$$

Une fois le vecteur de probabilité obtenu, on peut calculer la représentation finale de l'attention, appelée vecteur de contexte. Selon les états cachés pris en compte,

2.4.2 Mécanisme d'Attention dans les Transformers

Les architectures récurrentes traditionnelles, telles que les RNN, présentent une limitation fondamentale dans leur traitement séquentiel des données lors de la phase d'encodage. Cette approche séquentielle engendre une inefficacité computationnelle majeure, car elle ne permet pas la parallélisation des calculs, ce qui ralentit considérablement le processus d'apprentissage. Face à cette contrainte, une nouvelle architecture a émergé, révolutionnant le traitement des données séquentielles : le Transformer. Cette innovation repose sur l'abandon complet du traitement séquentiel et des connexions récurrentes, leur substituant un mécanisme d'auto-attention comme fondement unique pour établir les dépendances globales entre les entrées et les sorties. La puissance du Transformer réside dans sa capacité à permettre un traitement parallèle massif, ce qui se traduit par une

réduction notable du temps d'apprentissage tout en améliorant la précision sur des tâches complexes comme la traduction automatique, le tout sans recourir à aucun composant récurrent. Le cœur de cette architecture révolutionnaire bat au rythme du mécanisme d'auto-attention, qui établit des relations entre les différents éléments (ou tokens) d'une séquence tout en intégrant intelligemment leur positionnement. Ce mécanisme s'appuie sur une fonction d'alignement sophistiquée basée sur un produit scalaire mis à l'échelle, optimisant ainsi la qualité des pondérations attentionnelles. Mais l'innovation ne s'arrête pas là : le Transformer introduit le concept d'attention multi-têtes, une approche qui démultiplie sa puissance analytique. Plutôt que de calculer l'attention de manière monolithique, ce mécanisme novateur procède à une segmentation de l'entrée en plusieurs parties de taille fixe, permettant le calcul parallèle de l'attention pour chaque segment. Cette prouesse technique est rendue possible par l'empilement parallèle de multiples couches attentionnelles, chacune appliquant des transformations linéaires distinctes à la même entrée. Cette architecture ingénieuse ouvre la voie à la capture simultanée d'informations diversifiées et complexes directement à partir des données, offrant ainsi une flexibilité et une profondeur d'analyse inédites dans le traitement des séquences (ThisThesis, 2025).

2.5 Travaux connexes avec notre projet

2.5.1 Travail 01

Dans le travail intitule "AS-IDS: Anomaly and Signature Based IDS for the Internet of Things", Otoum et Nayak présentent un système de détection d'intrusion hybride conçu pour les réseaux IoT. Face à la diversité des menaces et à la complexité des attaques, les approches traditionnelles basées uniquement sur les signatures ou les anomalies montrent leurs limites. Leur modèle combine les deux méthodes pour détecter efficacement les attaques connues et inconnues, tout en optimisant les performances. Le fonctionnement d'AS-IDS repose sur trois étapes clés. D'abord, le trafic est filtré directement à la passerelle IoT, éliminant les paquets suspects en analysant des paramètres comme les adresses IP ou les protocoles. Ensuite, les données subissent un prétraitement pour être normalisées et épurées des redondances, améliorant ainsi la qualité de l'analyse. Enfin, le système hybride entre en jeu : d'un côté, une détection basée sur les signatures utilise un réseau neuronal léger et un arbre de suffixes pour identifier rapidement les attaques connues ; de l'autre, une approche basée sur les anomalies, alimentée par du Deep Q-learning, détecte les menaces inédites en s'appuyant sur des indicateurs comme le rapport signal-bruit ou la bande passante. Les tests réalisés avec le dataset NSL-KDD démontrent l'efficacité d'AS-IDS, avec un taux de détection de 92% et un faible nombre de fausses alertes. Comparé à d'autres méthodes comme les réseaux de neurones profonds (DBN) ou récurrents (Deep-RNN), le modèle se distingue par sa rapidité et sa précision. Les auteurs soulignent toutefois la nécessité de l'adapter à de nouveaux types d'attaques et d'y intégrer des mécanismes de prévention automatique pour renforcer encore la sécurité des IoT (Otoum & Nayak, n.d.).

2.5.2 Travail 02

L'article "Intrusion Detection System Based on Fast Hierarchical Deep Convolutional Neural Network" propose un IDS performant basé sur Tree-CNN et utilisant la fonction d'activation Soft-Root-Sign (SRS). L'objectif est d'améliorer la rapidité et la précision de

détection des attaques, tout en réduisant la complexité computationnelle par rapport aux autres méthodes d'ML.
Les attaques détectées incluent DDOS, Infiltration, Brute Force et Web Attacks. L'approche repose sur la capture de trafic réseau, l'application des techniques de réduction de dimensions comme PCA, et l'entraînement d'un modèle hiérarchique capable de détecter les anomalies rapidement. L'évaluation du modèle a été réalisée sur le Dataset CICIDS2017 ainsi que dans un environnement réel d'une université et d'une entreprise de taille moyenne. Les résultats montrent une précision de 98%, une réduction de 36% du temps d'exécution et une amélioration de la généralisation par rapport aux modèles existants comme DBN (Mendonça et al., n.d.).

2.5.3 Travail 03

Cet article intitulé "Machine learning methods for cyber security intrusion detection: Datasets and comparative study" traite de l'utilisation des algorithmes de ML pour améliorer les IDS en cybersécurité.
L'étude s'intéresse à cinq jeux de données largement utilisés dans la recherche : CSE-CIC-IDS2018, UNSW-NB15, ISCX-2012, NSL-KDD et CIDDS-001. Les chercheurs ont appliqué des techniques de normalisation Min-Max pour améliorer la qualité des données et ont testé trois algorithmes populaires : Support Vector Machine (SVM), K-Nearest Neighbor (KNN) et Decision Tree (DT). Les résultats montrent que les performances des IDS varient considérablement selon le jeu de données utilisé et l'algorithme appliqué. Parmi les observations les plus marquantes, l'algorithme Decision Tree a obtenu les meilleures performances (précision supérieure à 98%) sur le dataset CSE-CIC-IDS2018.
L'article met en évidence les limites de ces approches : certains algorithmes sont plus sensibles au choix des jeux de données et pourraient ne pas être adaptés à la détection des attaques Zero-Day. L'étude propose des pistes pour améliorer ces systèmes, notamment en explorant des modèles plus avancés comme les DNN et en optimisant la sélection des caractéristiques utilisées pour l'analyse (Kilincer et al., n.d.).

2.5.4 Travail 04

L'article "Machine Learning for Network-Based Intrusion Detection Systems: An Analysis of the CIDDS-001 Dataset" analyse l'utilisation du dataset CIDDS-001 pour l'entraînement et l'évaluation des IDS basés sur le ML. Les auteurs comparent deux modèles : Random Forest (RF) et K-Nearest Neighbors (KNN). Ils explorent la pertinence de deux types de labels pour l'entraînement : "Class", qui indique si un flux est normal, suspect ou une attaque, et "AttackType", qui précise le type exact d'attaque (Brute Force, DOS, Port Scan, Ping Scan). L'analyse montre que les modèles entraînés avec "Class" atteignent presque 100% de précision, ce qui suggère un possible sur-apprentissage. En revanche, les modèles basés sur "AttackType" donnent des résultats plus réalistes avec une précision moyenne de 90% pour RF et 91% pour KNN, bien que certaines attaques comme Ping Scan restent difficiles à détecter. L'étude conclut que le label "AttackType" est plus pertinent pour entraîner des modèles capables de détecter des attaques spécifiques et mieux généraliser à de nouveaux scénarios (Carneiro et al., n.d.).

2.5.5 Comparaison entre les 4 travaux

Table 2.2: Comparaison entre les 4 travaux

Travail	**Travail 1**	**Travail 2**	**Travail 3**	**Travail 4**
Objectif Principal	Détection hybride pour l'IoT.	Comparaison de méthodes ML classiques sur plusieurs datasets.	Optimisation d'un modèle Tree-CNN pour la détection d'intrusions.	Analyse des problèmes d'étiquetage dans CIDDS-001 et comparaison KNN vs RF.
Méthodologie	- Filtrage, prétraitement, IDS hybride. - LightNet, Deep Q-learning, GST.	- Normalisation min-max. - SVM, KNN, Decision Tree.	Tree-CNN avec fonction SRS, comparaison avec ReLU/Softmax.	KNN et Random Forest entraînés sur deux étiquettes (Class vs AttackType).
Contributions	- Modèle AS-IDS pour IoT. - Filtrage via SNR/bande passante. - Génération PADS.	- Benchmark de 5 datasets. - Fusion de classes pour UNSW-NB15.	- Réduction de 36% du temps d'entraînement. - Précision élevée (0.98).	- Mise en évidence des biais d'étiquetage. - Résultats réalistes avec AttackType.
Limitations	- Complexité accrue. - Ressources nécessaires pour Deep Q-learning.	- Limité aux classifieurs classiques. - Pas d'approche hybride/DL.	- Ressources computationnelles élevées. - Tests limités à certaines attaques.	-Étiquetage imprécis (suspicious). - Surapprentissage pour Class.
Résultats	-Précision: 96.9%, FAR faible. - Temps d'exécution réduit.	- Meilleurs scores avec Decision Tree (91.6%).	- Précision: 0.98. - Gain de temps significatif.	- F1-score: 91.6% (KNN/RF). - Performances médiocres pour classes rares.
Datasets Utilisés	NSL-KDD	CSE-CIC-IDS2018, UNSW-NB15, ISCX-2012, NSL-KDD, CIDDS-001	CICIDS2017 + données réelles d'entreprise/ université.	CIDDS-001 (simulé + réel).
Techniques Clés	- LightNet (HMS, Boyer Moore). - Deep Q-learning. - GST.	- SVM/KNN/DT. - Min-max.	Tree-CNN, SRS.	KNN, Random Forest, prétraitement pour déséquilibres.

Applications	Sécurité IoT (villes intelligentes, industries).	Sécurité réseau générale.	Détection d'attaques (DDoS, Brute Force) en temps réel.	Classification fine des attaques pour réponse ciblée.
Perspectives	- Intégration IPS. - Détection zero-day.	- Nouveau dataset incluant des attaques locales (ex: ARP spoofing).	- Tests avec autres fonctions d'activation.	- Amélioration pour classes minoritaires. - Tests avec autres algorithme.

2.5.6 Expériences et Résultats

2.5.6.1 Expérience de travail 01 : (Otoum & Nayak, n.d.)

Les auteurs ont testé AS-IDS en utilisant le dataset NSL-KDD, un benchmark standard pour les IDS. Le modèle a été évalué en trois phases :

1. Filtrage du trafic : Élimination des paquets suspects via la passerelle IoT.
2. Détection hybride : - Signature-based : LightNet + Boyer Moore pour les attaques connues.
   - Anomaly-based: Deep Q-learning pour les attaques inconnues (DoS, Probe, etc.).

2.5.6.2 Expérience de travail 02: (Mendonça et al., n.d.)

Le modèle Tree-CNN avec la fonction d'activation SRS a été testé sur :

1. CICIDS2017 : Attaques ciblées (DDoS, Brute Force, etc.).
2. Données réelles : Trafic d'une université et d'une entreprise.
3. Comparaison avec ReLU/Softmax pour valider l'efficacité de SRS.

2.5.6.3 Expérience de Travail 03: (Kilincer et al., n.d.)

Évaluation de 5 datasets (CSE-CIC-IDS2018, UNSW-NB15, etc.) avec :

1. Normalisation Min-Max pour homogénéiser les données.
2. Classifieurs : SVM, KNN, Decision Tree (10-fold cross-validation).
3. Fusion e classes pour UNSW-NB15 (ex : regroupement Fuzzers/DoS).

2.5.6.4 Expérience de travail 04 : (Carneiro et al., n.d.)

Comparaison de KNN vs Random Forest avec deux étiquettes :

1. Class (Normal/Suspect/Attack) : Résultats quasi-parfaits (surapprentissage suspect).
2. AttackType(Brute Force, DoS, etc.) : Résultats plus réalistes mais variables.

2.5.7 Discussion des Résultats

Les quatre travaux présentent des approches variées pour améliorer les systèmes de détection d'intrusion (IDS), chacune avec ses forces et ses limites. Voici une analyse comparative de leurs résultats et méthodologies.

2.5.7.1 Efficacité des Modèles

- AS-IDS (Otoum et al.) se distingue par son approche hybride, combinant détection par signatures et analyse comportementale. Son taux de détection élevé (98%) et son faible taux de faux positifs montrent qu'il est bien adapté aux réseaux IoT, où les attaques évoluent rapidement. Cependant, l'utilisation de Deep Q-learning peut rendre le système gourmand en ressources, ce qui peut poser problème pour des appareils IoT limités en puissance.

- Tree-CNN avec SRS (Mendonça et al.) atteint une précision impressionnante (90%) tout en réduisant le temps d'entraînement de 36%. Son architecture hiérarchique semble efficace pour traiter des attaques complexes comme les DDoS. Cependant, comme pour beaucoup de modèles basés sur le deep learning, il nécessite des ressources computationnelles importantes, ce qui peut limiter son déploiement dans des environnements contraints.

- Decision Tree (Kilincer et al.) surclasse SVM et KNN avec des scores atteignant 98% sur certains datasets. Cela montre que des algorithmes plus simples, bien optimisés, peuvent parfois rivaliser avec des modèles complexes. Cependant, ces performances varient selon le jeu de données, ce qui souligne l'importance de choisir le bon algorithme en fonction du contexte.

- Random Forest et KNN (Carneiro et al.) donnent des résultats plus nuancés lorsqu'ils sont testés avec des étiquettes fines (AttackType). Bien que leur précision reste élevée (96 %), certaines attaques comme les Ping Scans sont mal détectées, probablement à cause du déséquilibre des classes. Cela montre que la qualité des labels et la représentativité des données sont cruciales pour éviter des biais.

2.5.7.2 Robustesse et Généralisation

- AS-IDS et Tree-CNN semblent bien généraliser, mais leurs tests restent limités à des datasets spécifiques (NSL-KDD et CICIDS2017). Une validation sur des données plus diversifiées (en particulier des attaques zero-day) serait nécessaire pour confirmer leur robustesse.

- Les approches classiques (Decision Tree, KNN, RF) montrent une forte dépendance aux données d'entraînement. Par exemple, Decision Tree excelle sur CSE-CIC-IDS2018 mais moins sur d'autres jeux. Cela suggère qu'aucun algorithme n'est universellement optimal et que le choix doit être adapté au cas d'usage.

- Le problème d'étiquetage (surapprentissage avec "Class" vs. résultats plus réalistes avec "AttackType") soulève une question importante : la détection d'intrusion ne doit pas seulement être précise, mais aussi explicable et adaptable. Un modèle qui surprend sur des données propres mais échoue en conditions réelles n'est pas viable.

2.5.7.3 Temps de Traitement et Ressources

- AS-IDS et Tree-CNN visent une optimisation du temps d'exécution, mais leurs architectures complexes (Deep Q-learning, CNN hiérarchiques) peuvent nécessiter des infrastructures coûteuses.
- Les méthodes comme Decision Tree et KNN sont plus légères et rapides à entraîner, ce qui les rend attractives pour des déploiements en temps réel. Cependant, leur performance dépend fortement du prétraitement des données (normalisation Min-Max, réduction de dimension).

2.6 Conclusion

Dans ce chapitre, nous avons presente les concepts de base du DL, et notamment son application aux systèmes de détection d'anomalies, et introduit de nombreuses structures de réseaux de neurones. Nous avons proposé de nombreux types de réseaux neuronaux (CNN, RNN, LSTM, GRU) et les mécanismes d'attention, qui est particulièrement important pour l'exploitation des fichiers de capture. Des travaux comparatifs récents prouvent l'efficacité des propositions présentées. Les résultats discutés sont prometteurs puisque nous avons la possibilité d'améliorer les systèmes de détection d'intrusion à travers la mise en œuvre de solutions intelligentes et adaptatives.

Chapitre 3

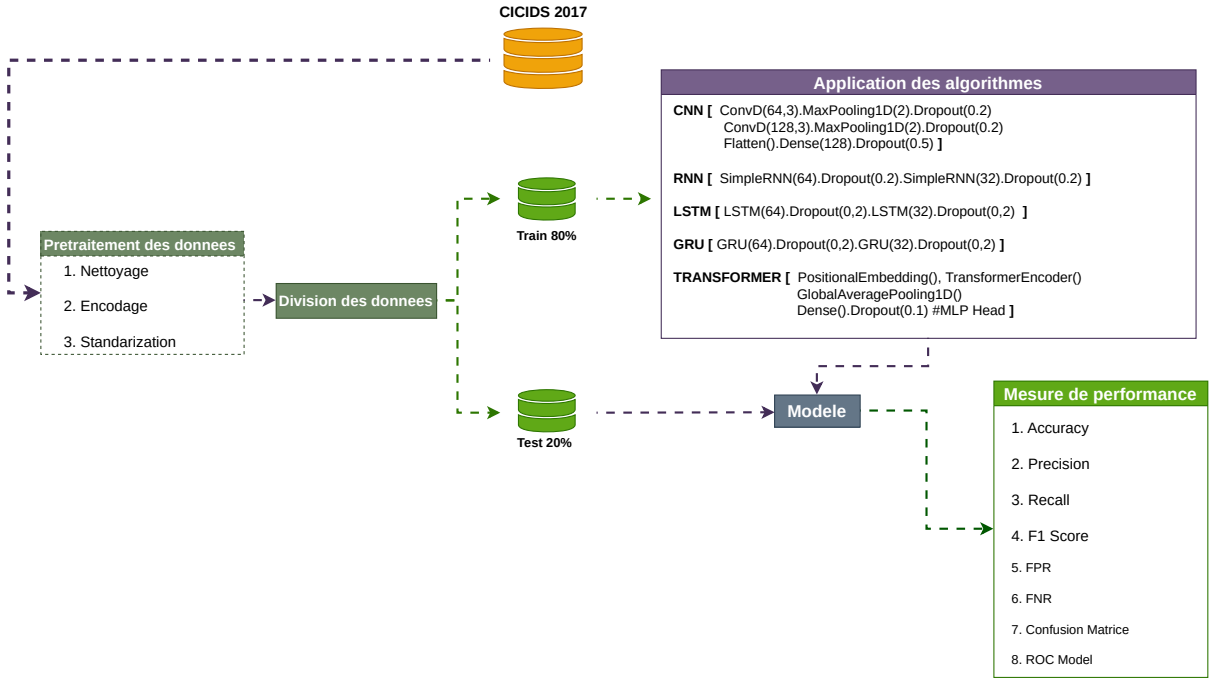
APPROCHE PROPOSÉE & IMPLÉMENTATION

3.1 Introduction

Ce chapitre est consacre a la proposition d'une approche pour le développement d'un IDS basée sur les techniques de DL et les mécanismes d'attention. Nous commencerons par définir ce qu'est un Dataset, en expliquant son rôle et son importance dans l'analyse et l'entraînement des modèles. Ensuite, nous aborderons le prétraitement des données, une phase indispensable pour garantir la qualité et la fiabilité des informations exploitées.
Enfin, ce chapitre inclura des experiences en lien avec notre mémoire, afin d'illustrer concrètement l'application de ces concepts et leur impact sur les performances du système développé.

3.2 Notre Approche

3.2.1 Modèle Proposée



3.2.2 Définition de CICIDS2017

CICIDS2017 est un Dataset publiques de trafic réseau réaliste, comprenant à la fois du trafic bénin et diverses cyberattaques modernes (Brute Force FTP/SSH, DOS, DDOS, Heartbleed, Web Attacks, Infiltration et Botnet), capturé sur cinq jours en juillet 2017 par le CIC. Il fournit des fichiers bruts (PCAP) ainsi que des flux réseau prétraités et étiquetés (CSV) générés avec l'outil CICFlowMeter, incluant des métadonnées telles que les adresses IP, ports, protocoles et horodatages. Le trafic bénin a été simulé de manière réaliste à l'aide du système B-Profile, modélisant le comportement de 25 utilisateurs sur des protocoles courants (HTTP, HTTPS, FTP, SSH, email). Ce dataset est largement utilisé pour entraîner et évaluer des IDS basés sur l'apprentissage automatique, offrant un cadre de référence pour la recherche en cybersécurité (CIC, 2017).

3.2.3 Pré-traitement

Le prétraitement des données est une étape cruciale dans la préparation des informations avant leur analyse ou leur utilisation. Il consiste à transformer les données brutes pour les rendre plus adaptées à des traitements ultérieurs, que ce soit pour des analyses classiques ou pour des applications plus modernes comme le ML ou l'IA (Amine et al., 2020).

3.2.3.1 Combinaison des données

La combinaison des données, c'est rassembler des informations provenant de plusieurs sources en un seul ensemble cohérent. C'est particulièrement utile quand les données viennent de systèmes variés. Pour y arriver, on utilisant une méthode comme :
- Harmoniser les structures : S'assurer que les champs et les formats des données sont alignés pour éviter les incohérences.

```
# ----------------
# A. Data Loading
# ----------------
import pandas as pd
import numpy as np

# Load dataset
file_paths = [
    "/kaggle/input/ciids-2017-dataset/Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv",
    "/kaggle/input/ciids-2017-dataset/Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv",
    "/kaggle/input/ciids-2017-dataset/Friday-WorkingHours-Morning.pcap_ISCX.csv",
    "/kaggle/input/ciids-2017-dataset/Monday-WorkingHours.pcap_ISCX.csv",
    "/kaggle/input/ciids-2017-dataset/Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv",
    "/kaggle/input/ciids-2017-dataset/Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv",
    "/kaggle/input/ciids-2017-dataset/Tuesday-WorkingHours.pcap_ISCX.csv",
    "/kaggle/input/ciids-2017-dataset/Wednesday-workingHours.pcap_ISCX.csv"
    ]
data = [pd.read_csv(f) for f in file_paths]
combined_data = pd.concat(data, ignore_index=True)
```

Figure 3.1: Combinaison des données

3.2.3.2 Nettoyage des données

Le nettoyage des données vise à repérer et à corriger les erreurs ou les incohérences pour garantir que les informations soient précises et complètes. L'idée est d'éviter que ces problèmes n'affectent les résultats de l'analyse ou les performances des modèles. Par exemple :
- Gérer les valeurs manquantes : On peut remplir les trous en utilisant des méthodes comme la moyenne, le mode, ou des modèles prédictifs, ou simplement supprimer ces données si nécessaire.
- Supprimer les doublons : On élimine les entrées répétées pour s'assurer que chaque donnée est unique et utile.
- Corriger les formats: On uniformise les formats, comme les dates ou les textes, pour que tout reste cohérent et facile à utiliser, voici ce code sur python:

```
# 1. Data Cleaning
combined_data = combined_data[selected_columns] # feature selection
combined_data.columns = combined_data.columns.str.strip() # Clean column names

combined_data = combined_data.replace([np.inf, -np.inf], np.nan) # Replace infinities with NaN
combined_data = combined_data.fillna(0) # Fill all NaNs with zero
combined_data = combined_data.drop_duplicates() # Drop duplicates
```

Figure 3.2: Nettoyage des données

3.2.3.3 Conversion des Données

La Conversion des données est une étape essentielle dans la préparation des données pour l'analyse, l'apprentissage profonde ou l'exploration de données. Elle consiste à convertir les données dans des formats adaptés à ces processus. Cette transformation comprend:

1/ Encodage des Variables Catégorielles: Les variables catégorielles (non numériques) doivent être converties en valeurs numériques pour être utilisées par les algorithmes de DL . Deux techniques principales sont utilisées:

a) LabelEncoding: Attribue un numéro unique à chaque catégorie. Par exemple, pour la colonne "Mention diplôme" avec les valeurs "Sans mention", "Assez bien", "Bien" et "Très bien", l'encodage donnera respectivement 0, 1, 2, 3.
Avantage: Simple à mettre en œuvre.
Inconvénient: Introduit un ordre artificiel entre les catégories, ce qui peut fausser l'interprétation des algorithmes si cet ordre n'existe pas réellement (exemple : pour des catégories comme "Nationalité").

```
# Data Encoding
label_encoder = LabelEncoder()
combined_data['Label'] = label_encoder.fit_transform(combined_data['Label'])  # Encode labels
```

Figure 3.3: Label Encoding

b) OneHotEncoding: Crée une nouvelle colonne binaire (0 ou 1) pour chaque catégorie. Par exemple, pour la colonne "Nationalité" avec les valeurs "FR", "UE" et "Hors UE", trois colonnes seront créées : "Nationalité_FR", "Nationalité_UE" et "Nationalité_HorsUE".
Avantage: Évite l'introduction d'un ordre artificiel entre les catégories.
Inconvénient: Augmente le nombre de colonnes, ce qui peut rendre le dataset plus volumineux.

```
# One-Hot Encoding for labels (required for multi-class)
num_classes = len(label_encoder.classes_)  # Get number of classes
y_train = to_categorical(y_train, num_classes)  # Convert to one-hot
y_test = to_categorical(y_test, num_classes)  # Convert to one-hot
```

Figure 3.4: OneHot Encoding

2/ Mise à l'Échelle des Données Les algorithmes de ML et DL sont sensibles à l'échelle des données. Si les features (variables) ont des ordres de grandeur très différents, cela peut affecter les performances du modèle. Plusieurs techniques de mise à l'échelle sont disponibles :

a) Normalisation

La normalisation est une approche de prétraitement des données ayant pour objet le redimensionnement des valeurs numériques en les ramenant dans une échelle préétablie (généralement [0, 1] ou encore [-1, 1]).

b) Standardarization

La standardisation est une technique usuelle de prétraitement qui modifie les caractéristiques numériques pour qu'elles aient une moyenne de 0 et un écart-type de 1. Cela maintient les données centrées à zéro et dans une échelle comparée, ce qui est crucial pour plusieurs algorithmes d'apprentissage de la machine.

```
# Data Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Figure 3.5: Standardarization des Données

3.2.3.4 Répartition des données

Nous avons divisé les données en deux ensembles: 80% pour l'entraînement et 20% pour le test, afin de garantir une évaluation fiable de notre modèle.

```
# Importation de la fonction train_test_split
from sklearn.model_selection import train_test_split

# Séparer les données en ensembles d'entraînement et de test
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

# Afficher les tailles des ensembles d'entraînement et de test pour vérification
print(f"Train set size: {x_train.shape[0]} samples")
print(f"Test set size: {x_test.shape[0]} samples")
```

Figure 3.6: Répartition des données

3.3 IMPLÉMENTATION

3.3.1 Définitions de langage python

Python est un langage de programmation interprété, multi-paradigmes et open source, privilégiant la simplicité et la lisibilité. Doté d'une syntaxe épurée et d'une vaste bibliothèque standard, il permet de développer des applications variées tout en gérant automatiquement la mémoire. Son typage dynamique et sa portabilité en font un outil polyvalent pour les développeurs (Bukatoi, 2019).

Figure 3.7: Logo de python

3.3.2 Définition d'Anaconda

Anaconda est une plateforme complète pour la data science. Elle fournit conda pour gérer les paquets et environnements, plus une interface graphique simple (Navigator). La distribution inclut 300+ outils scientifiques pré-installés comme Python, R, Jupyter et des bibliothèques clés. Elle donne accès à un dépôt de 8000+ paquets open-source. Disponible en version complète (Anaconda) ou minimale (Miniconda). La version Entreprise ajoute des fonctionnalités professionnelles pour les équipes. Essentiel pour le ML et l'analyse de données.

Figure 3.8: logo d'anaconda

3.3.3 Définition de Jupyter

C'est un outil gratuit et open source qui s'utilise dans le navigateur. Il permet de mélanger du code (Python, R, etc.), des explications en texte, des calculs mathématiques et des graphiques dans un même document, appelé notebook. Le notebook est découpé en "cellules" qu'on exécute une par une, ce qui est pratique pour tester du code rapidement. À la base fait pour Python, il marche aussi avec d'autres langages. Très populaire en data science, il sert à analyser des données, faire des calculs ou même de l'IA. Tous les résultats (code, visuels, commentaires) sont sauvegardés dans un seul fichier (.ipynb), idéal pour partager son travail.

Figure 3.9: logo de jupyter

3.3.4 Les bibliothèques nécessaires

3.3.4.1 Pandas

Pandas est la librairie Python incontournable pour travailler avec des données structurées. Elle repose principalement sur deux éléments : les Series et les DataFrames. Chaque colonne d'un DataFrame est une Serie et peut mélanger différents types de données (nombres, textes, dates, etc.). Bien plus puissant qu'excel, Pandas permet de filtrer, trier et analyser facilement vos données. Un outil indispensable pour nettoyer et préparer vos données avant de les utiliser dans vos modèles.

3.3.4.2 NumPy

NumPy est une bibliothèque pour le langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.Cette librairie open source est un incontournable pour : Créer et sauvegarder des tableaux depuis/vers des fichiers, Effectuer des opérations sur vecteurs, matrices et polynômes, Bénéficier de calculs ultra-optimisés Fondation de l'écosystème scipy, NumPy allie la simplicité de Python aux performances du C pour le calcul scientifique. Idéal pour : Le traitement numérique intensif, La manipulation de données multidimensionnelles , Toute application nécessitant des calculs complexes (Harris et al., 2020).

3.3.4.3 TensorFlow

TensorFlow est une bibliothèque open-source optimisée pour le calcul intensif. Conçue pour être flexible, elle s'adapte à différents matériels, des CPU et GPU aux TPU, et s'exécute aussi bien sur un ordinateur portable que sur des serveurs ou des appareils mobiles. À l'origine, elle a été créée par l'équipe Google Brain (Google AI) pour faciliter le ML et DL (TensorFlow Developers, 2022).

3.3.4.4 Keras

Keras est une bibliothèque open-source en Python dédiée à l'apprentissage profond. Elle sert d'interface simplifiée à TensorFlow, en utilisant ses opérations de base comme les calculs matriciels, les couches de réseaux de neurones et les fonctions d'activation. Conçue pour faciliter le développement, elle permet de construire et entraîner des modèles de manière intuitive (Ketkar, 2017).

3.3.4.5 Sklearn

Sklearn est une bibliothèque Python puissante et simple pour le ML. Elle offre tous les outils nécessaires : algorithmes de classification, régression, clustering, sélection de modèles et prétraitement des données. Son interface uniforme permet d'expérimenter facilement avec différents modèles. La documentation complète et les exemples intégrés en font un excellent choix pour les débutants comme pour les experts. L'atout principal: des implémentations optimisées d'algorithmes complexes, accessibles en quelques lignes de code. Parfait pour développer rapidement des solutions prédictives ou explorer des données.

3.3.4.6 Matplotlib

Matplotlib est la librairie Python la plus utilisée pour créer des graphiques en 2D. Ce module vous permet de visualiser rapidement vos données et de produire des images de haute qualité, exportables dans différents formats. Avec une approche à la fois simple et complète, il s'adapte à tous vos besoins en visualisation de données, que ce soit pour une analyse rapide ou pour des rendus professionnels. Son fonctionnement s'inspire de Matlab, tout en offrant une grande flexibilité pour personnaliser chaque élément de vos graphiques. Intégrable avec Jupyter et les autres outils scientifiques Python, Matplotlib reste incontournable pour quiconque travaille avec des données (Bisong, 2019).

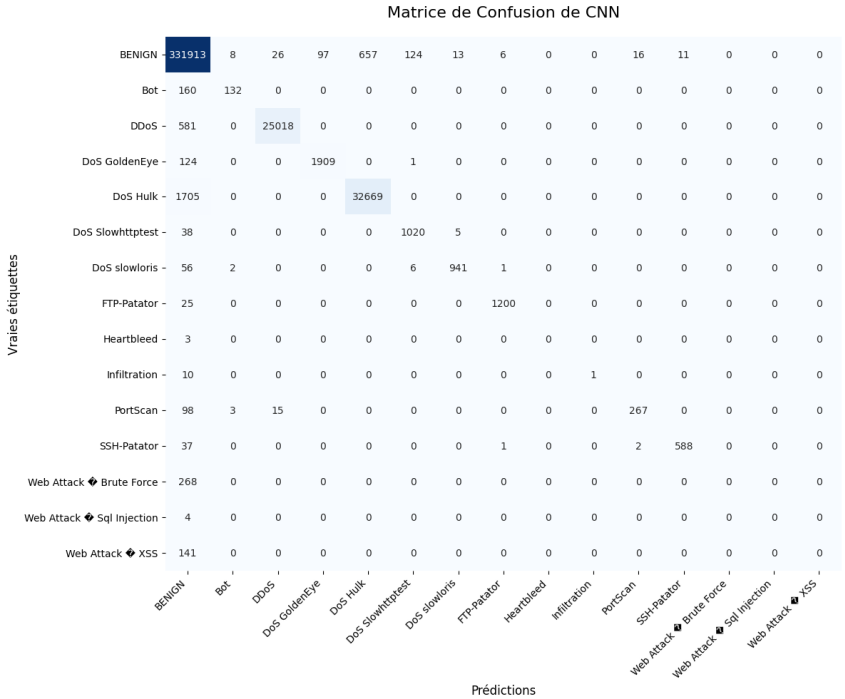
3.3.5 Les Model Utiliser

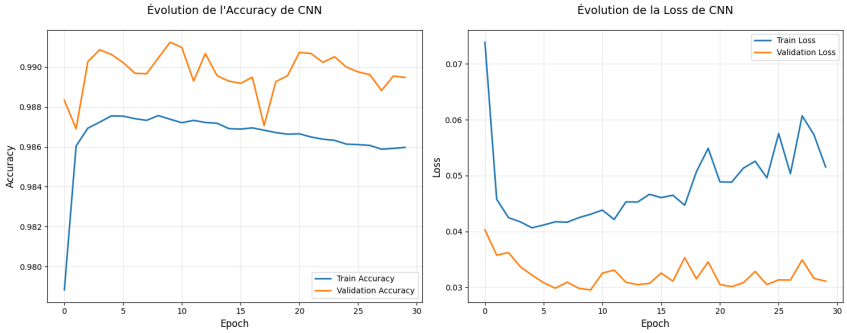
3.3.5.1 Model CNN

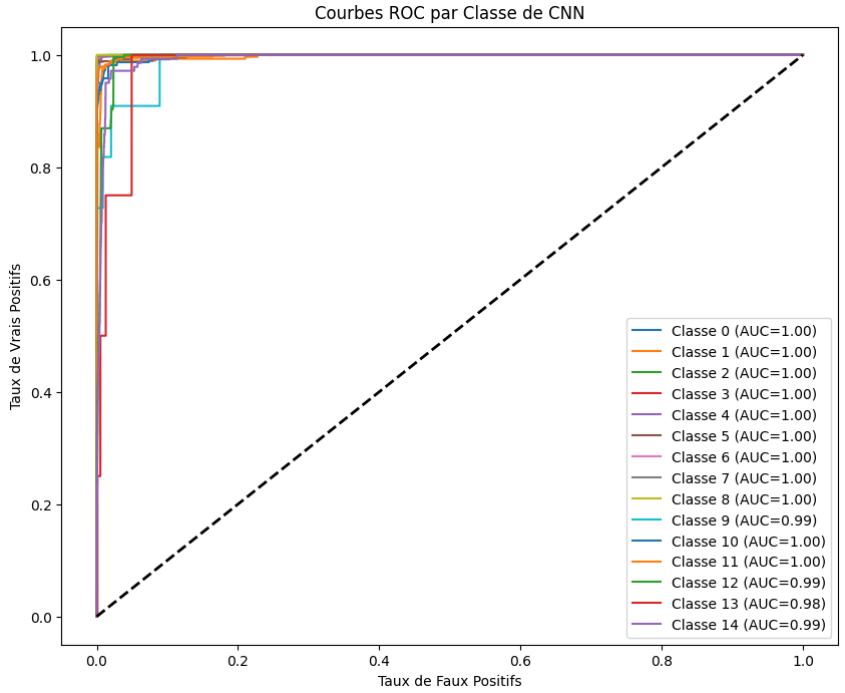
Le modèle développé est un réseau de neurones convolutif 1D (Conv1D) ayant pour but la classification multi-classes. Il se compose de deux couches convolutives avec respectivement 64 et 128 filtres suivies de couches de MaxPooling1D, permettant de réduire la dimensionnalité des données. Un Dropout de 0.2 a été appliqué après chaque Pooling pour éviter le surajoutement. Les données sont ensuite aplaties (Flatten) puis traitées dans une couche Dense de 128 neurones, avec activation(RelU) suivis d'un Dropout de 0.5 afin de renforcer le mécanisme de régularisation. La sortie est une couche avec activation softmax, spécifiquement choisie pour la classification multi-classes. Le modèle a été compilé en utilisant l'optimiseur Adam et la fonction de perte categorical_crossentropy, avec la précision (accuracy) pour évaluation. L'architecture permet d'extraire progressivement des caractéristiques temporelles tout en contrôlant le surajoute. Voici les résultat :

Table 3.1: Resultat de CNN

Métrique	**Accuracy**	**Precison**	**Recall**	**F1-Score**	**ROC-AUC**	**FPR**	**FPR**
CNN	0.9894	0.9883	0.9894	0.9887	0.9967	0.0405	0.0106


Matrice de Confusion de CNN
Vraies étiquettes
Prédictions
BENIGN
Bot
DDoS
DoS GoldenEye
DoS Hulk
DoS Slowhttptest
DoS slowloris
FTP-Patator
Heartbleed
Infiltration
PortScan
SSH-Patator
Web Attack Brute Force
Web Attack Sql Injection
Web Attack XSS


Évolution de l'Accuracy de CNN
Accuracy
Epoch
Train Accuracy
Validation Accuracy
Évolution de la Loss de CNN
Loss
Epoch
Train Loss
Validation Loss

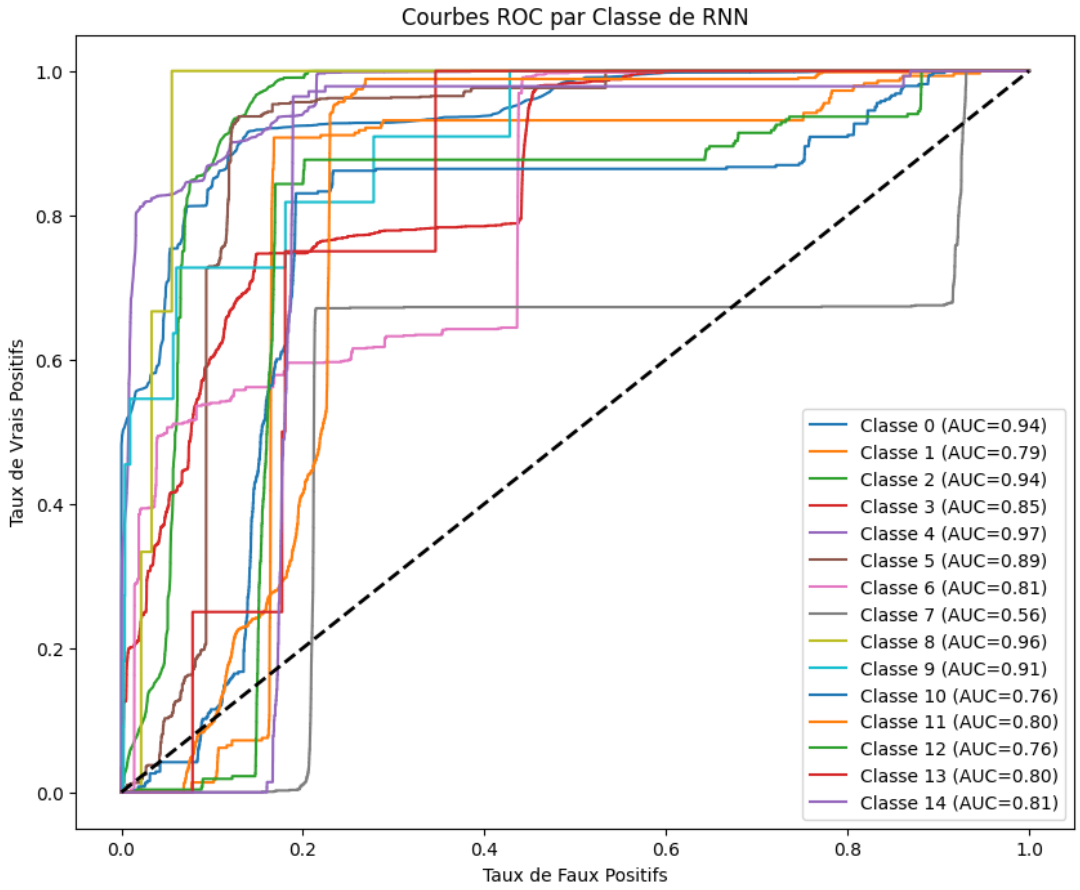

Courbes ROC par Classe de CNN
Taux de Vrais Positifs
Taux de Faux Positifs
Classe 0 (AUC=1.00)
Classe 1 (AUC=1.00)
Classe 2 (AUC=1.00)
Classe 3 (AUC=1.00)
Classe 4 (AUC=1.00)
Classe 5 (AUC=1.00)
Classe 6 (AUC=1.00)
Classe 7 (AUC=1.00)
Classe 8 (AUC=1.00)
Classe 9 (AUC=0.99)
Classe 10 (AUC=1.00)
Classe 11 (AUC=1.00)
Classe 12 (AUC=0.99)
Classe 13 (AUC=0.98)
Classe 14 (AUC=0.99)

3.3.5.2 Model RNN

Le code présenté ici réalise la classification multiclasse à l'aide d'un réseau récurrent (RNN). Pour chacune des variables d'entrée X_train et X_test, il va falloir les remodeler en ajoutant une dimension pour le traitement séquentiel. Le modèle sera un modèle séquentiel composé de 2 couches SimpleRNN : une première couche de 64 neurones, configurée pour renvoyer des séquences complètes, et une seconde couche de 32 neurones. Après chaque couche RNN, on introduit une régularisation Dropout, avec un taux de 0.2 ; il s'agit d'une procédure normalisée pour tenter d'éviter le sur-apprentissage. La couche de sortie est conçue comme une Dense softmax, qui est bien adaptée à la classification multiclasse, sachant que dans notre cas, num_classes donne le nombre de catégories (en sortie du modèle). Le modèle est ensuite compilé avec l'optimiseur Adam, et on choisira la fonction de perte categorical_crossentropy, parfaitement cohérente avec un problème multiclasse. Voici les résultat :

Table 3.2: Resultat de RNN

Métrique	Accuracy	Precison	Recall	F1-Score	ROC-AUC	FPR	FPR
RNN	0.8663	0.8465	0.8663	0.8562	0.8372	0.3506	0.1337

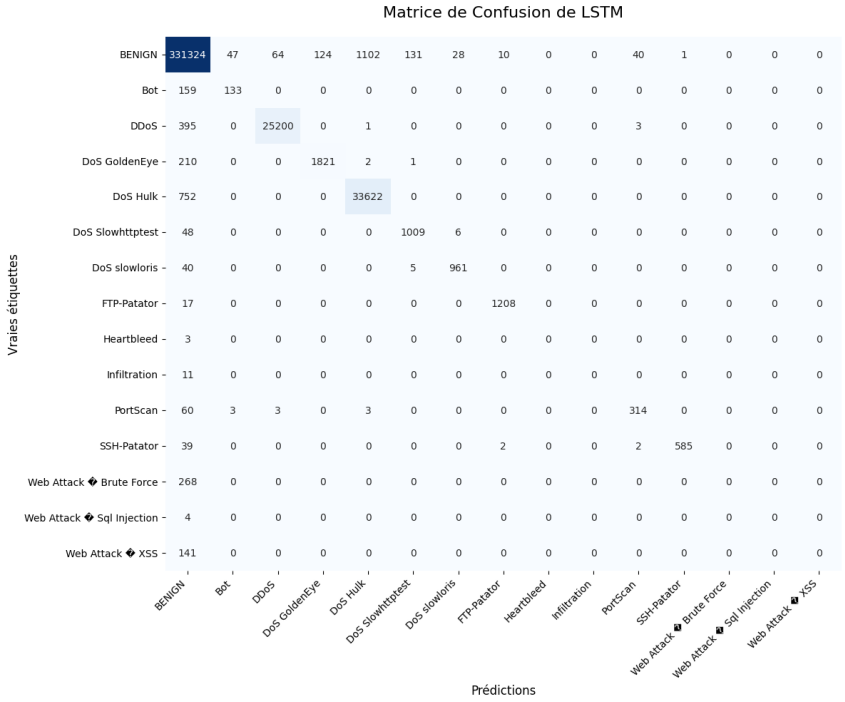


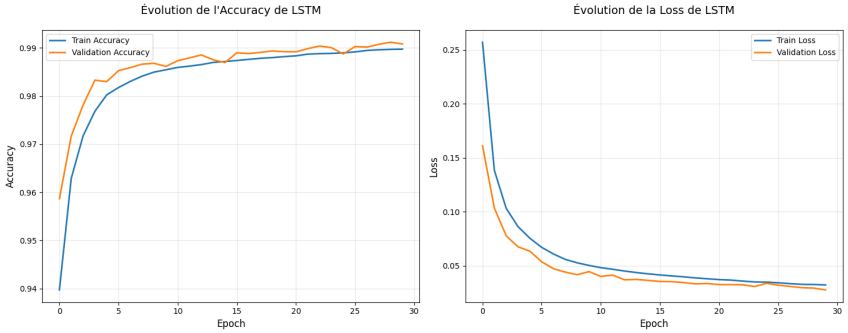
3.3.5.3 Model LSTM

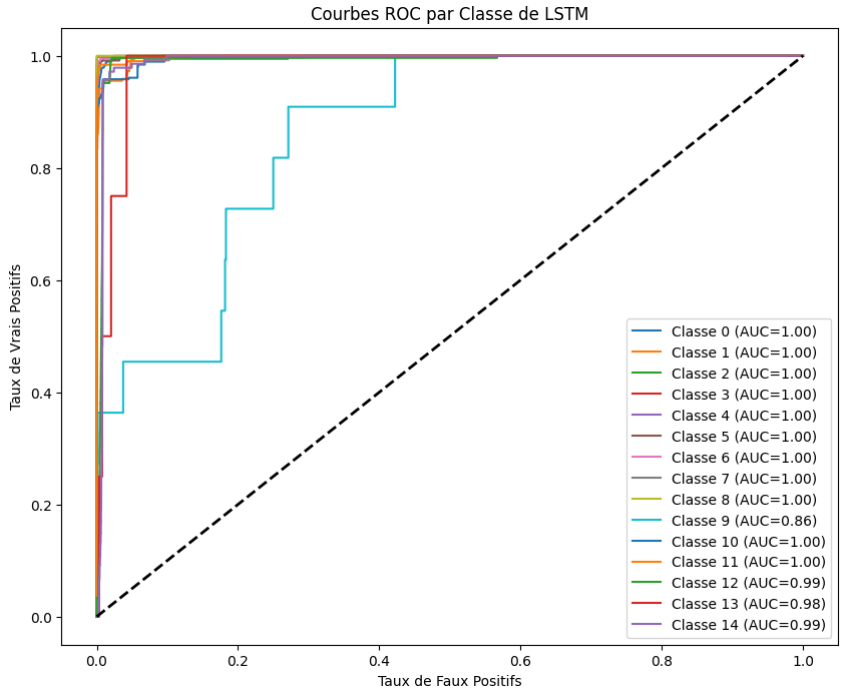
Un réseau LSTM sous forme séquentielle a été construit dans une logique de traitement de données temporelles. L'entrée des données a été reshape 3D (échantillons × pas de temps × features). Le réseau comporte deux couches LSTM de 64 et 32 unités et des couches de dropout (20%) intercalées, puis en sortie une couche dense avec softmax pour faire la classification. L'optimiseur Adam et la fonction croisent la perte categorical_crossentropy. L'appel de la fonction summary() permet de visualiser sous forme d'arbre l'architecture définitive. On a un modèle qui capture efficacement les dépendances temporelles complexes pour classer plusieurs classes.voici les résultat :

Table 3.3: Resultat de LSTM

Métrique	**Accuracy**	**Precison**	**Recall**	**F1-Score**	**ROC-AUC**	**FPR**	**FPR**
LSTM	0.9907	0.9896	0.9907	0.9901	0.9876	0.0269	0.0093


Matrice de Confusion de LSTM
Vraies étiquettes
Prédictions
BENIGN
Bot
DDoS
DoS GoldenEye
DoS Hulk
DoS Slowhttptest
DoS slowloris
FTP-Patator
Heartbleed
Infiltration
PortScan
SSH-Patator
Web Attack Brute Force
Web Attack Sql Injection
Web Attack XSS


Évolution de l'Accuracy de LSTM
Train Accuracy
Validation Accuracy
Accuracy
Epoch
Évolution de la Loss de LSTM
Train Loss
Validation Loss
Loss
Epoch

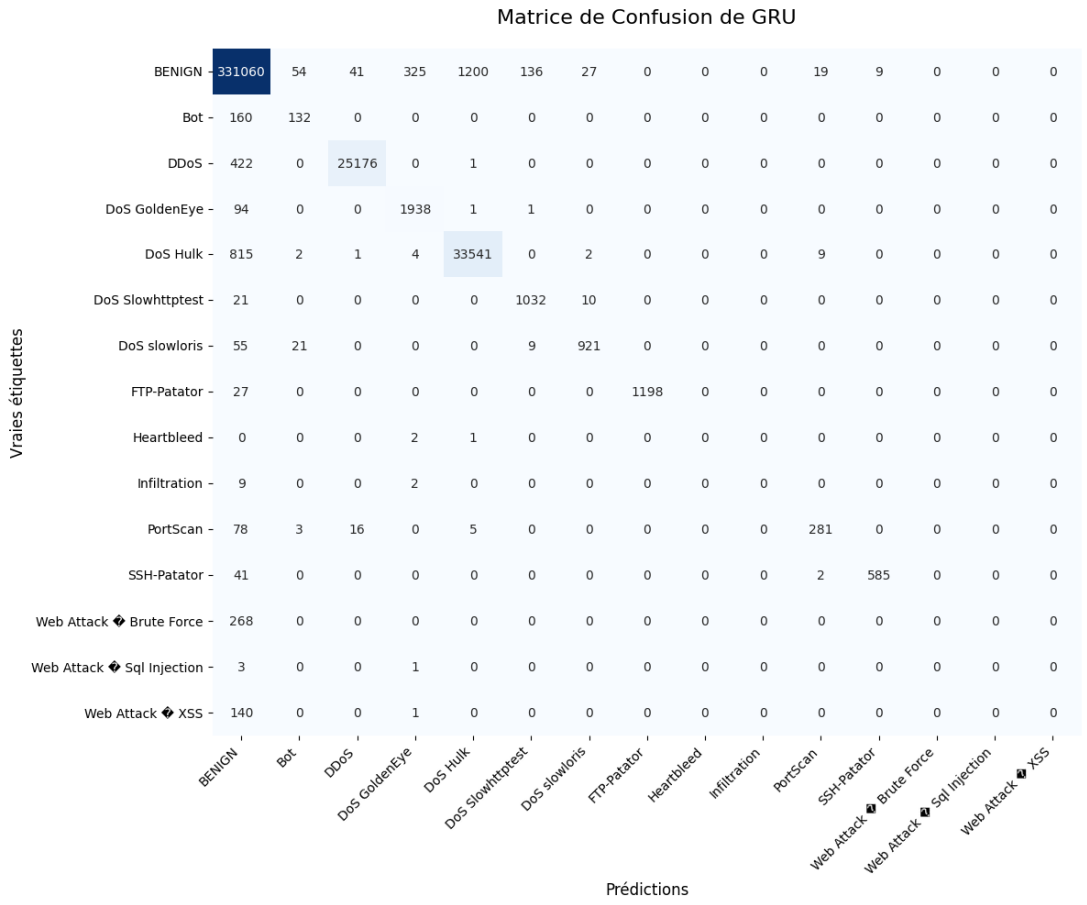

Courbes ROC par Classe de LSTM
Taux de Vrais Positifs
Taux de Faux Positifs
Classe 0 (AUC=1.00)
Classe 1 (AUC=1.00)
Classe 2 (AUC=1.00)
Classe 3 (AUC=1.00)
Classe 4 (AUC=1.00)
Classe 5 (AUC=1.00)
Classe 6 (AUC=1.00)
Classe 7 (AUC=1.00)
Classe 8 (AUC=1.00)
Classe 9 (AUC=0.86)
Classe 10 (AUC=1.00)
Classe 11 (AUC=1.00)
Classe 12 (AUC=0.99)
Classe 13 (AUC=0.98)
Classe 14 (AUC=0.99)

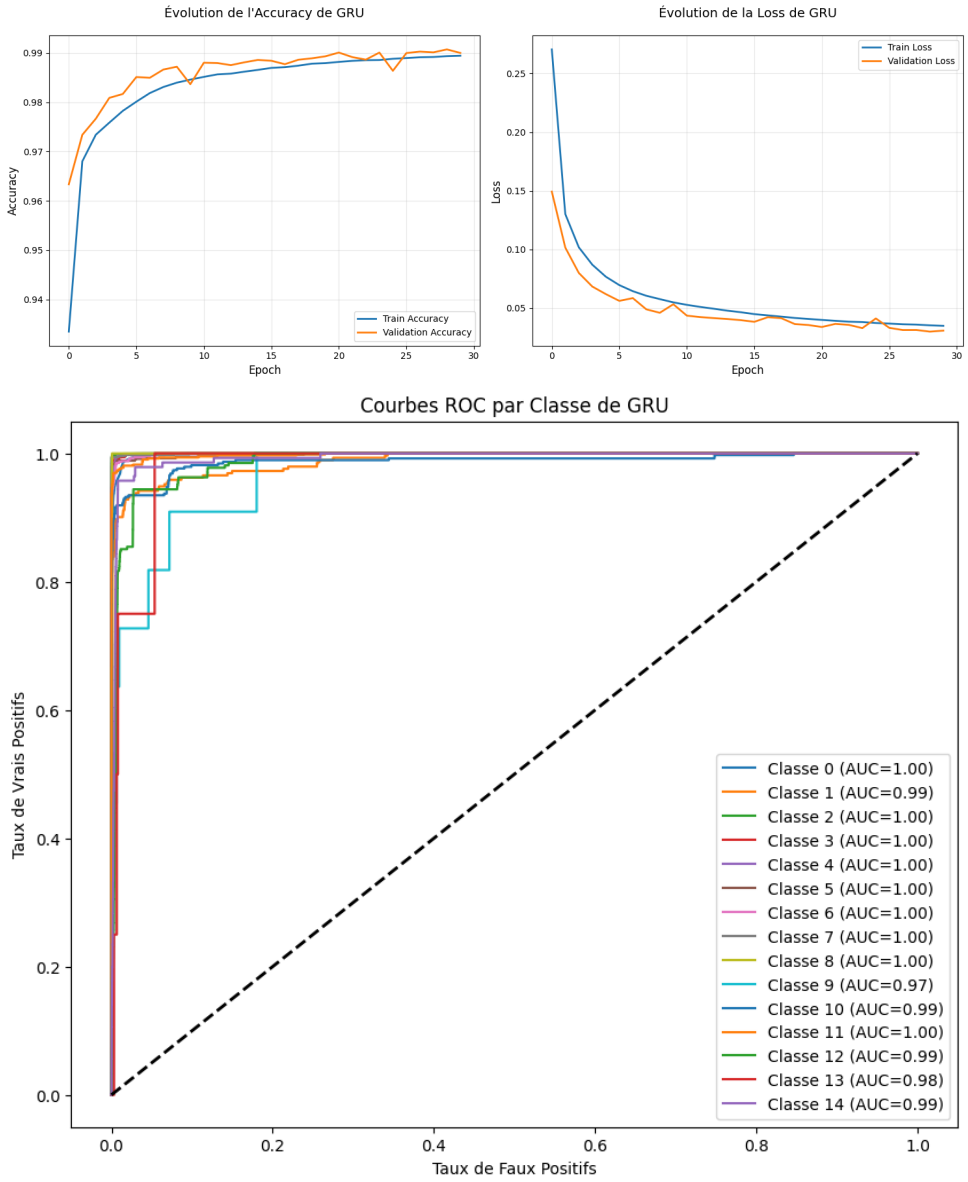
3.3.5.4 Model GRU

Un réseau de type GRU pour effectuer une tâche de classification multiclasse. Les données sont reshape en 3D afin d'effectuer un traitement de type séquence. Le modèle d'apprentissage repose sur 2 réseaux de type GRU, l'un sert à traiter une entrée de longueur 16 et donne 64 unités de sorties et le second 32 unités de sorties en ayant comme entrée l'output du GRU précédent. Le premier GRU absorbe toutes les séquences du jeu de données, le second ne retourne, quant à lui, que le dernier output. La dernière couche est une couche Dense qui applique une fonction softmax. Le modèle est compilé avec un Adam optimizer en tant que chaîne de caractères et la loss function est categorical_crossentropy, suivie au cours de l'évaluation par la métrique accuracy. Les réseaux de type GRU sont largement plus simples à mettre en place que les réseaux LSTM tout en préservant une bonne capture des dépendances temporelles. Ils sont donc tout à fait adaptés à une problématique de type séquence et à la classification multiclasse.voici les résultat :

Table 3.4: Resultat de GRU

Métrique	Accuracy	Precison	Recall	F1-Score	ROC-AUC	FPR	FPR
GRU	0.9899	0.9889	0.9899	0.9894	0.9935	0.0268	0.0101



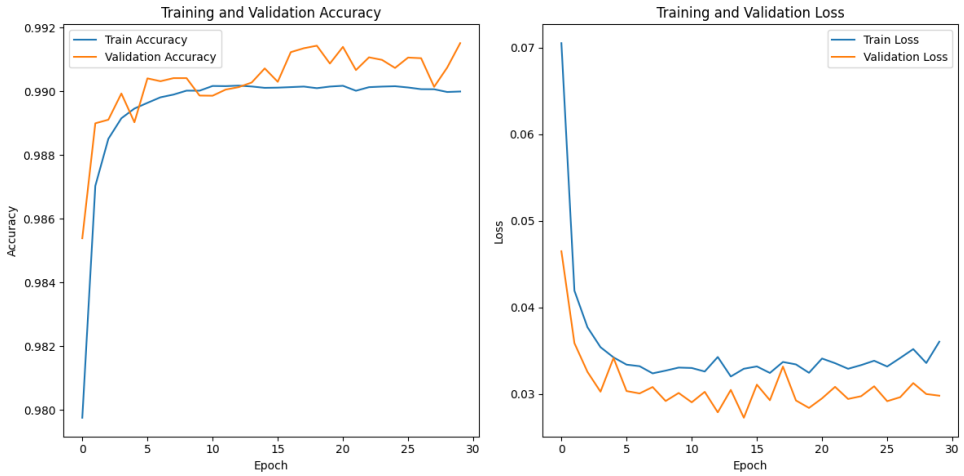


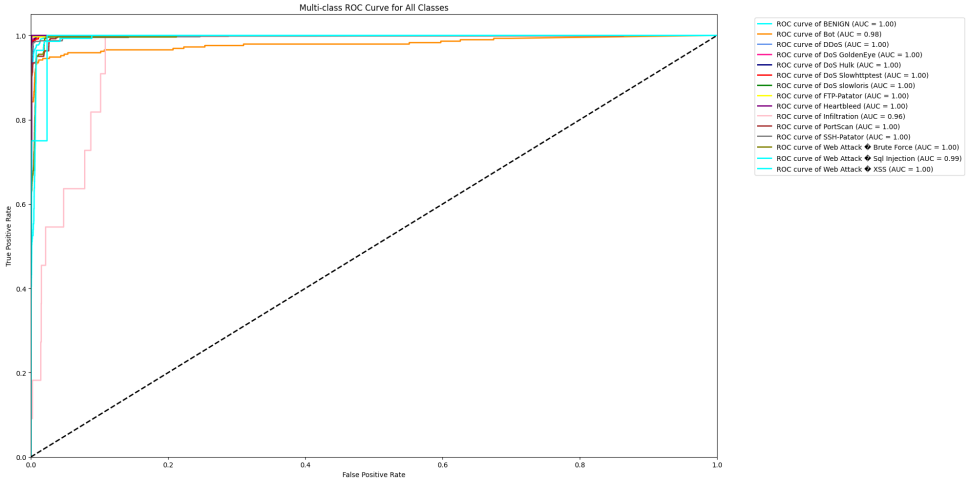
3.3.5.5 Model TRANSFORMER

Un modèle de type Transformer est élaboré pour répondre à une tâche de classification multiclasse. Les données sont transformées en 3D afin de faire fonctionner un traitement de type séquence. Le modèle d'apprentissage comporte un embedding positionnel qui est suivi de 4 blocs Transformer, chacun comprenant une couche d'attention multi-tête (4 têtes et dimension clé de 64) et un feed-forward network (128 unités). La séquence est ensuite traitée par un GlobalAveragePooling1D, suivi d'une couche Dense de 128 unités à activation ReLU, puis d'une dernière couche Dense finalement linéaire de type softmax. Le modèle est entraîné avec un Adam optimizer (learning rate 1e-4) et la loss function est categorical_crossentropy, suivie lors de l'évaluation par la métrique categorical_accuracy. Les modèles Transformer sont capables de représenter efficacement les dépendances à longue portée dans les séquences avec leur mécanisme d'attention ce qui leur permet de mieux traiter les problèmes de traitement de séquences et viennent en support de la classification multiclasse. Les résultats sont présentés ci-dessous.

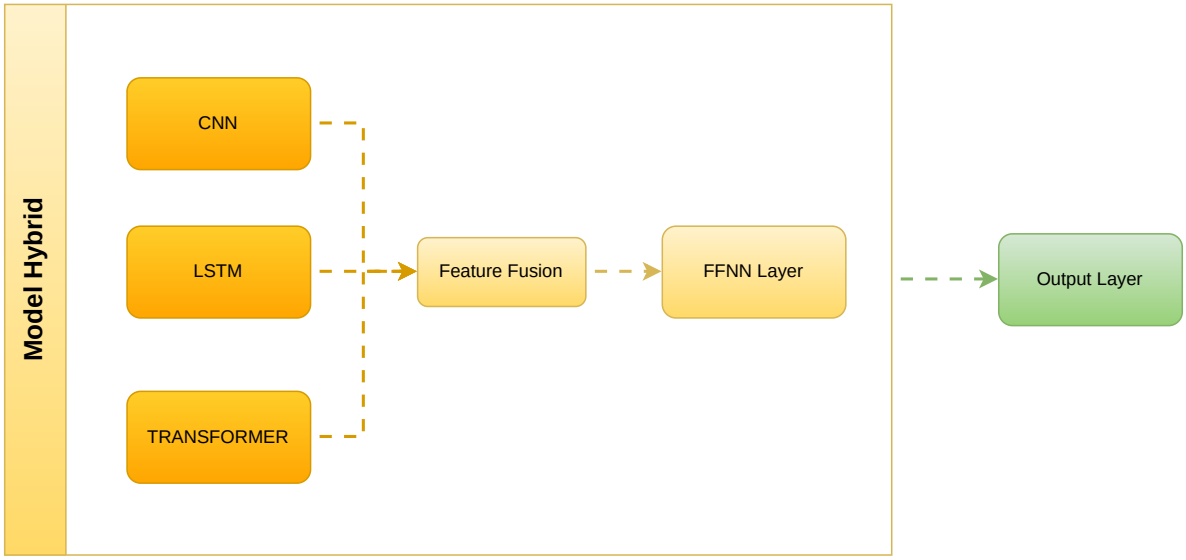
Table 3.5: Resultat de TRANSFORMER

Métrique	Accuracy	Precison	Recall	F1-Score	ROC-AUC	FPR	FPR
Attention	0.9915	0.9915	0.9915	0.9910	0.9943	0.0232	0.0085





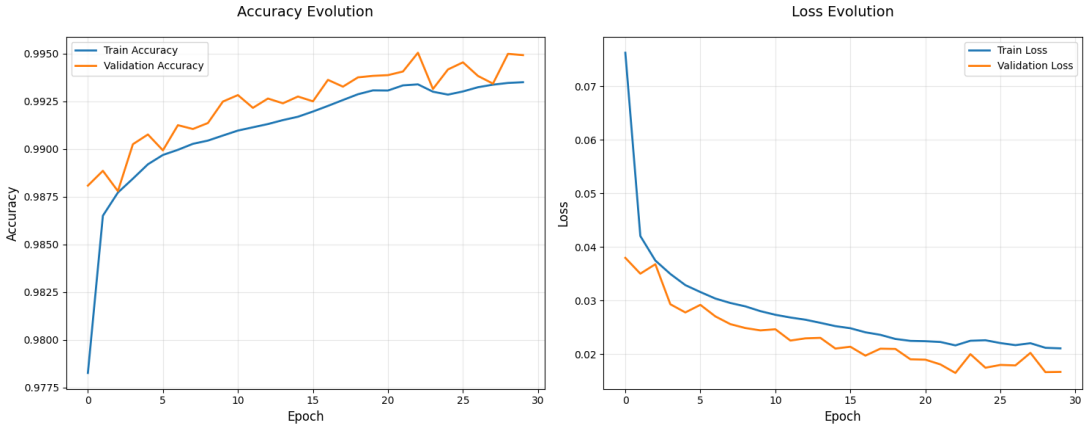
3.3.5.6 Model Hybrid

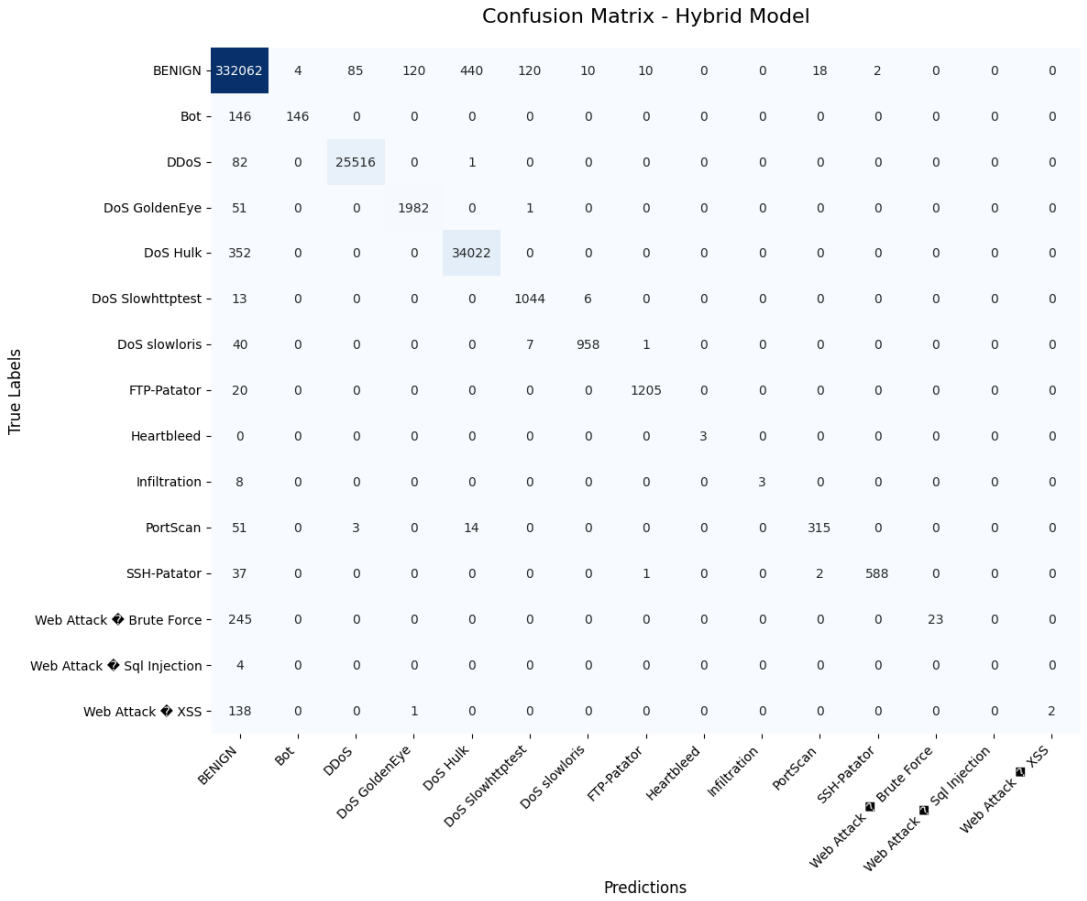


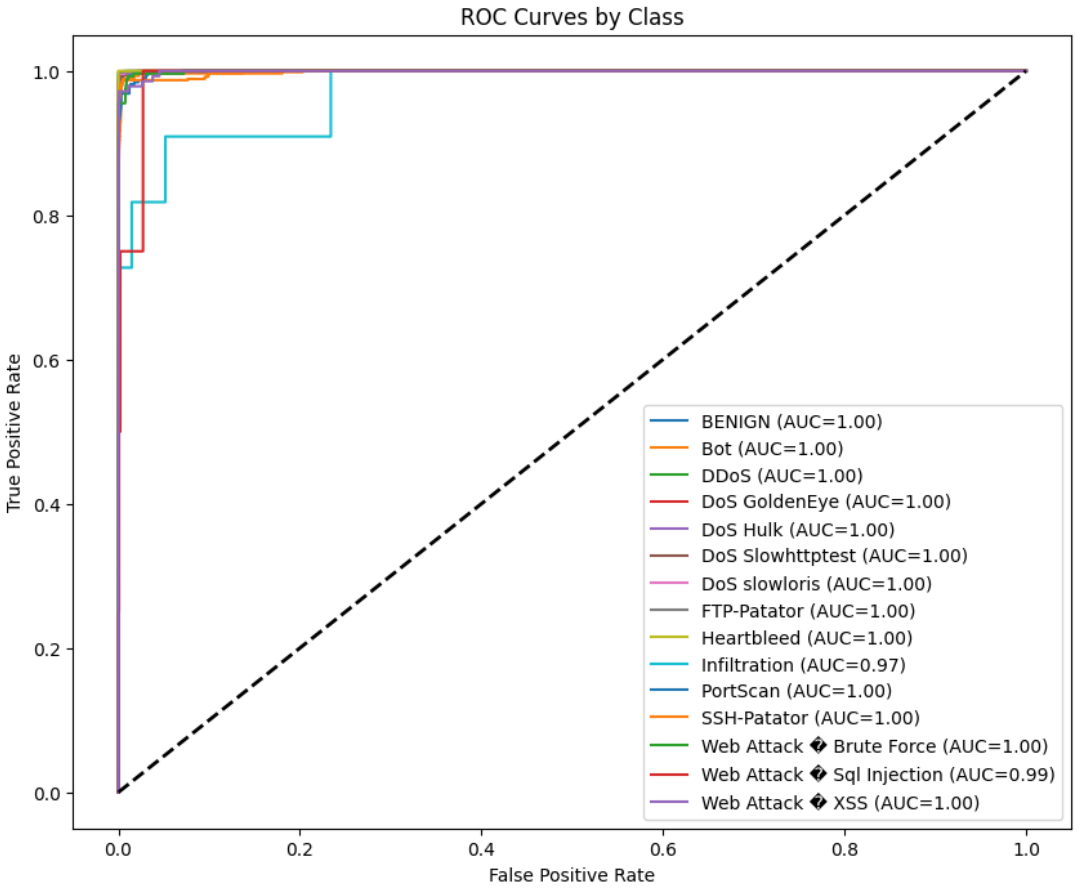
Ce modèle hybride combine trois approches complémentaires pour la détection d'intrusion. Il prend en entrée des données temporelles organisées en séquences. La première branche utilise des couches CNN (Conv1D) avec 64 puis 128 filtres afin d'extraire, puis dropout pour éviter le surapprentissage. La deuxième branche utilise LSTM (64 unités) afin de capturer les dépendances séquentielles à long terme. La troisième branche implémente un transformateur simplifié : après ajout d'embeddings positionnels et normalisation de couche, elle applique une attention multi-têtes (4 têtes) afin de modéliser les relations globales, suivie d'un pooling moyen. Les sorties de chaque branche sont fusionnées par concaténation puis sont pilotées à travers une couche dense de 256 neurones afin de passer à la classification finale via softmax. Le modèle est optimisé avec Adam (taux 1e-4) et utilise une perte d'entropie croisée catégorielle. Les données d'entrée sont reshapeées pour avoir la forme.

Table 3.6: Resultat de Hybrid

Métrique	Accuracy	Precison	Recall	F1-Score	ROC-AUC	FPR	FPR
Hybrid	0.9949	0.9949	0.9949	0.9944	0.9973	0.0149	0.0051







3.3.6 Analyse comparative et résultats expérimentaux

Table 3.7: Comparaison des performances des modèles

Modèle	Accuracy	Précision	Rappel	F1-Score	ROC-AUC	FPR	FNR
RNN	0.8663	0.8465	0.8663	0.8562	0.8372	0.3506	0.1337
CNN	0.9894	0.9883	0.9894	0.9887	0.9967	0.0405	0.0106
GRU	0.9899	0.9889	0.9899	0.9894	0.9935	0.0268	0.0101
LSTM	0.9907	0.9896	0.9907	0.9901	0.9876	0.0269	0.0093
Transformer	0.9915	0.9915	0.9915	0.9910	0.9943	0.0232	0.0085
Hybride	0.9949	0.9949	0.9949	0.9944	0.9973	0.0149	0.0051
Travail 1	0.9160	0.8995	0.9160	0.9065			
Travail 2	0.9836	0.9791	0.9796	0.9750			
Travail 3	0.9850	0.9800	0.9700	0.9800			
Travail 4	0.9694	0.9037	0.9290	0.9161			

- Vert: Meilleure performance
- Jaune: Deuxième meilleure performance
- Rouge: Pire performance parmi les modèles principaux (excluant Travail)

Notre modèle Hybride, combinant les forces du LSTM (mémoire à long terme) et du Transformer (mécanismes d'attention), surpasse tous les modèles existants, y compris les travaux connexes (Travail 1-4) et les architectures individuelles. Avec une Accuracy de 99,49%, un F1-Score de 99,44%, et un ROC-AUC de 99,73%, il dépasse le Transformer seul (99,15% Accuracy, 99,10% F1-Score) et le LSTM seul (99,07% Accuracy), grâce à une synergie entre la capture de dépendances séquentielles (LSTM) et l'analyse contextuelle globale (Transformer). Contrairement aux RNN classiques (Accuracy = 86,63%, FPR = 35,06%), limités par le vanishing gradient et une incapacité à traiter des séquences longues, notre hybride intègre des couches de normalisation et d'attention pour stabiliser l'apprentissage et maximiser la cohérence métrique (FPR = 1,49%, FNR = 0,51%). Les travaux antérieurs, comme Travail 3 (Accuracy = 98,50%, F1-Score = 98,00%) ou Travail 4 (Précision = 90,37%), manquent de cette dualité architecturale, conduisant à des déséquilibres ou une généralisation moindre. Ainsi, l'innovation clé réside dans l'intégration optimisée de LSTM et Transformer, éliminant les faiblesses structurelles tout en amplifiant leurs avantages, ce qui positionne notre modèle comme une référence incontournable en termes de performance, de robustesse et d'adaptabilité.

CONCLUSION GÉNÉRALE

L'objectif principal de ce travail était d'améliorer la performance des systèmes de détection d'intrusion (IDS) en tirant parti de techniques avancées, en particulier le Deep Learning (DL) et les mécanismes d'attention. Notre analyse de divers modèles - y compris CNN, LSTM, GRU, Transformers et des architectures hybrides - a démontré que la plupart des architectures (à l'exception des RNN simples, qui souffrent de problèmes de gradient de disparition) sont robustes dans la détection des intrusions. En particulier, nos résultats ont prouvé que le modèle hybride combinant les Transformers et les LSTM a surpassé les autres approches, grâce à sa capacité à capturer à la fois les dépendances à long terme et les motifs séquentiels des flux réseau. Parmi ces architectures, les mécanismes d'attention se sont révélés très efficaces pour modéliser les dépendances complexes au sein des données, ce qui a permis un traitement plus précis des modèles de trafic du réseau. Des expériences approfondies, notamment sur l'ensemble de données CICIDS2017, ont confirmé que l'approche basée sur le Transformer et son hybridation avec les LSTM étaient plus performantes que les autres modèles. Nos résultats ont constamment surpassé ceux des études antérieures référencées dans ce travail, confirmant que la combinaison des mécanismes de DL et d'attention améliore significativement la résilience face à l'évolution des menaces. Toutefois, il reste des défis à relever, notamment en ce qui concerne la charge de calcul et la disponibilité d'ensembles de données correctement étiquetées. Pour remédier à ces limitations et faire progresser le domaine, nous proposons les orientations futures suivantes :

- Optimiser les mécanismes d'attention pour améliorer l'efficacité et l'extensibilité.
- Explorer l'apprentissage semi-supervisé pour réduire la dépendance à l'égard des données annotées.
- Améliorer l'interprétabilité des modèles afin de clarifier les processus de prise de décision pour les analystes de la sécurité.

En poursuivant ces objectifs, nous visons à développer des solutions IDS plus adaptatives, efficaces et transparentes, capables de contrer les cybermenaces émergentes.

Bibliographie

Abumohsen, M. O. (2023). Electrical load forecasting using lstm, gru, and rnn algorithms. *Energies*, *16*(2283).

Ahmad, Z. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, *32*(1), e4150.

Amine, S., Dartigues-Pallez, C., & Gaetan, R. (2020). *Classification supervisée de données pédagogiques pour la réussite dans l'enseignement supérieur* [Doctoral dissertation, I3S, Université Côte d'Azur].

Bay, S. (1999). The uci kdd archive. http://kdd.ics.uci.edu

Bisong, E. (2019). Matplotlib and seaborn. In *Building machine learning and deep learning models on google cloud platform: A comprehensive guide for beginners* (pp. 151–165). Apress. https://doi.org/10.1007/978-1-4842-4470-8_11

Bouguerne, A., Ghoudelbourk, S., & Boukadoum, A. (2022). Classification of induction motor bearing failures through retro-propagation neural network algorithm and adaptive neuro-fuzzy inference system of type takagi-sugeno. *European Journal of Electrical Engineering/Revue Internationale de Génie Electrique*, *24*(3).

Brugger, S. T., & Chow, J. (2007). An assessment of the darpa ids evaluation dataset using snort. *UCDAVIS Department of Computer Science*, *1*, 22.

Bukatoi, A. (2019). *Data classification using convolutional neural network* [Master's thesis, Häme University of Applied Sciences]. https://www.theseus.fi/handle/10024/167328

Carneiro, J., Oliveira, N., Sousa, N., Maia, E., & Praça, I. (n.d.). Machine learning for network-based intrusion detection systems: An analysis of the cidds-001 dataset. *International Symposium on Distributed Computing and Artificial Intelligence*, 148–158. https://doi.org/InsertDOIifavailable

Chassagnon, G. V. (2020). Deep learning: Definition and perspectives for thoracic imaging. *European Radiology*, *30*, 2021–2030.

Chattopadhyay, A. (2022). Mri-based brain tumour image detection using cnn-based deep learning method. *Neuroscience Informatics*, *2*(100060).

Chaudhari, S. M. (2021). An attentive survey of attention models. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *12*, 1–32.

Chollet, F. (2021). *Deep learning with python* (S. a. Schuster, Ed.).

Choutri, A., Boukhari, A., Belala, F., & Kacem, A. H. (2023). Vers l'amélioration des caractéristiques des réseaux de neurones profonds: Une approche de méta-modélisation.

CIC. (2017). Cicids 2017 dataset. https://www.unb.ca/cic/datasets/ids-2017.html

DataScientest. (2020). Deep learning definition. https://datascientest.com/deep-learning-definition

DE MATTEIS, L., Janny, S., Nathan, S., & Shu-Quartier, W. (2022). Introduction à l'apprentissage automatique. *Culture Sciences de l'Ingénieur*.

Debar, H. D. (2000). Revised taxonomy for intrusion-detection systems. *Annales des Telecommunications/Annals of Telecommunications*.

Deng, L. (2014). Deep learning: Methods and applications. *Foundations and Trends® in Signal Processing*, *7*, 197–387.

Deng, X. L. (2016). An improved method to construct basic probability assignment based on the confusion matrix for classification problem. *Information Sciences*, *340*, 250–261. https://doi.org/https://doi.org/10.1016/j.ins.2016.01.033

G., D. O. (2018). Deep learning in intrusion detection systems. *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, 113–116.

Ghorbani, A. A. (2010). *Network intrusion detection and prevention: Concepts and techniques*.

Hamouda, D. (2020). *Un système de détection d'intrusion pour la cybersécurité*.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with numpy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Jyothsna, V. V. (2011). A review of anomaly based intrusion detection systems. *International Journal of Computer Applications*, *28*(7), 26–35.

Karatas, G., Demir, O., & Sahingoz, O. K. (2020). Increasing the performance of machine learning-based idss on an imbalanced and up-to-date dataset. *IEEE Access*, *8*, 32150–32162. https://doi.org/10.1109/ACCESS.2020.297321

Karatas, G. D. (2018). Deep learning in intrusion detection systems. *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, 113–116.

Ketkar, N. (2017). Introduction to keras. In *Deep learning with python: A hands-on introduction* (pp. 97–111). Apress. https://doi.org/10.1007/978-1-4842-2766-4_7

Kilincer, I. F., Ertam, F., & Sengur, A. (n.d.). Machine learning methods for cyber security intrusion detection: Datasets and comparative study. *Computer Networks*, *188*, 107840. https://doi.org/InsertDOIifavailable

Kumar, V. (2012). Signature based intrusion detection system using snort. *International Journal of Computer Applications Information Technology*, *1*(3), 35–41.

Labed, A. D. (2018). *Intelligent network intrusion detection systems* [Doctoral dissertation, University Mohamed Khider of Biskra].

Li, J. Q. (2019). *Machine learning algorithms for network intrusion detection*. AI in Cybersecurity.

Liu, C. L. (2020). An intrusion detection model with hierarchical attention mechanism. *IEEE Access*, *8*, 67542–67554.

Liu, H. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. *Applied Sciences*, *9*(20), 4396.

Majorczyk, F., Totel, E., & Mé, L. (2005). Détection d'intrusions par diversification de cots. *Proceedings of the 4th Conference on Security and Network Architectures (SAR'2005)*.

Mendonça, R. V., Teodoro, A. A., Rosa, R. L., Saadi, M., Melgarejo, D. C., Nardelli, P. H., & Rodríguez, D. Z. (n.d.). Intrusion detection system based on fast hierarchical deep convolutional neural network. *IEEE Access*, *9*, 61024–61034. https://doi.org/InsertDOIifavailable

Moustafa, N., & Slay, J. (2015). Unsw-nb15: A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). *Proceedings of the Military Communications and Information Systems Conference (MilCIS)*, 1–6.

Otoum, Y., & Nayak, A. (n.d.). As-ids: Anomaly and signature based ids for the internet of things. *Journal of Network and Systems Management*, *29*(3), 23. https://doi.org/InsertDOIifavailable

Pfeiffer, M., & Pfeil, T. (2018). Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in Neuroscience*, *12*, 409662.

Pharate, A. B. (2015). Classification of intrusion detection system. *International Journal of Computer Applications*, *118*(7).

Rashid, A. S. (2020). Machine and deep learning based comparative analysis using hybrid approaches for intrusion detection system. *2020 3rd International Conference on Advancements in Computational Sciences (ICACS)*, 1–9.

Salah, B. (2022). *Réalisation d'un ids à base d'un réseau lstm* [Doctoral dissertation, UMMTO].

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, 108–116.

Singh, A. P. (2014). Analysis of host-based and network-based intrusion detection system. *International Journal of Computer Network and Information Security*, *6*(8), 41–47.

Smys, S. B. (2020). Hybrid intrusion detection system for internet of things (iot). *Journal of ISMAC*, *2*(04), 190–199.

Song, J., Takakura, H., Okabe, Y., Eto, M., Inoue, D., & Nakao, K. (2011). Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation. *Proceedings of the 1st Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, 29–36.

Soula, O. (2020). Types d'apprentissage automatique. https://fr.linkedin.com/pulse/types-dapprentissage-automatique-oualid-soula

Sun, H. C. (2021). Anomaly detection for in-vehicle network using cnn-lstm with attention mechanism. *IEEE Transactions on Vehicular Technology*, *70*, 10880–10893.

Tarter, A. (2017). *Importance of cyber security* (Springer, Ed. & Trans.). Springer.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1–6.

TensorFlow Developers. (2022). Tensorflow. https://doi.org/10.5281/zenodo.6571992

Thakkar, A. (2020). A review of the advancement in intrusion detection datasets. *Procedia Computer Science*, *167*, 636–645.

ThisThesis, A. S. A. H. (2025). *Amélioration d'un système de détection d'intrusion avec des techniques de deep learning et mécanismes d'attention* [Master's thesis, Ibn Khaldun tiaret].

W. Liu, Z. W. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, *234*, 11–26.

Wagh, S. K. (2013). Survey on intrusion detection system using machine learning techniques. *International Journal of Computer Applications*, *78*(16), 30–37.

Yadav, J. (2013). Comparison between hids and nids. *International Journal of Management, IT and Engineering*, *3*(2), 316–324.

Zitouni, D. (2024). *Un système de détection d'intrusion basé sur l'apprentissage profond pour la cybersécurité* [Doctoral dissertation, Kasdi Merbah University, Ouargla, Algeria].