République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A/Mira de Béjaia
Faculté des Sciences Exactes
Département d'Informatique

MEMOIRE DE MASTER RECHERCHE

En
Informatique

Option

Intelligence Artificielle
et
Administration et Sécurité des Réseaux Informatique

Thème

Détection d'intrusion en exploitant les méthodes d'apprentissage profond

Présenté par : Mr RAHMANI Omar
Mlle SI-MOUSSA Yamina

Sous la direction de : Dr AMROUN Kamal

Soutenu le septembre 2022 devant le jury composé de :

Présidente	**Dr OUYAHIA Samira**
Examinatrice	**Dr ADEL Karima**
Examinatrice	**Mlle BENLALA Wissam**

Béjaia, septembre 2022.

⁕ *Remerciements* ⁕

On remercie dieu le tout puissant de nous avoir donné la santé et la volonté d'entamer et de terminer ce mémoire.

Tout d'abord, ce travail ne serait pas aussi riche et n'aurait pas pu avoir le jour sans l'aide et l'encadrement de **Mr Kamel Amroun**, on le remercie pour la qualité de son encadrement exceptionnel, pour sa patience, sa rigueur et sa disponibilité durant notre préparation de ce mémoire.

Nos remerciement s'adresse également à tout nos professeurs pour leurs générosités et la grande patience dont ils ont su faire preuve malgré leurs charges académiques et professionnelles.

Nous tenons à remercier également tous ceux qui nous ont aidés de prés et de loin pour l'élaboration de ce mémoire.

A tous ceux dont le soutien nous a été utile et nécessaire, nous disons :

Merci

⁕ *Dédicaces* ⁕

l'expression de ma reconnaissance, je dédie ce modeste travail à ceux qui, quels que soient les termes embrassés, je n'arriverais jamais à leur exprimer mon amour sincère

À l'homme, mon précieux offre du dieu, qui doit ma vie, ma réussite et tout mon respect : mon cher père **Lounès** paix a son âme.

À la femme qui a souffert sans me laisser souffrir, qui n'a jamais dit non à mes exigences et qui n'a épargné aucun effort pour me rendre heureuse : mon adorable mère **Nadia**.

À mes chères sœurs **Sihem**, **Lydia** ainsi que leurs maris **Didine** et **Abdou**, sans oublier mon grand frère **Sifax** qui n'ont pas cessées de me conseiller, encourager et soutenir tout au long de mes études. Que Dieu les protèges et leurs offres, la chance et le bonheur.

À mes adorables neveux **Anas**, **Ishak** et **Rayan** qui reflètent la joie et le bonheur pour toute la famille.

À ma grand-mère **Zahra**, qui m'a soutenu et encouragé durant ces années d'études, paix a son âme.

À mes copines **Lydia**, **Lilia** et **Nassima** avec qui j'ai partagé tous mes bons et mauvais moments.

À mes oncles et mes tantes, tous les cousins, et les amis que j'ai connus jusqu'à maintenant.

Merci pour leurs amours et leurs encouragements.

*M. **Si-moussa Yamina***

⁕ *Dédicaces* ⁕

Après dieu, je tiens à remercier chaleureusement tous ceux qui m'ont soutenu durant mes études et préparation de ce travail, que je dédié à :

Mes très chers parents **Mokhtar** et **Ayouni Khadidja**, qui ont été toujours là pour moi.

Mon petit frère **Billel** et mes trois adorables sœurs **Meriem**, **Zahra** et **Nadjet** paix a son âme

Pour toute ma famille maternelle et cousins

Notre chat de maison aimé **Smokey**

Mon cher pote **Mathouk**

Et tous mes amis et collègues avec qui j'ai partagé des moments d'émotion de près ou de loin.

Merci pour votre support.

*M. **Rahmani Omar***

RÉSUMÉ

Dans l'intérêt d'aider les administrateurs des systèmes à détecter et prévenir toute violation de la sécurité d'information dans les organisations, Beaucoup de recherches ont fait des systèmes de détection d'intrusions (IDS) comme objet. Cette étude est menée dans le but de modéliser un tel système qui joue un rôle primordial dans ce domaine. Pour cela, nous avons étudié les performances des méthodes d'apprentissage profond (DL) pour la détection d'intrusions dans les réseaux. Nous avons implémenté 8 modèles pour la multi-classification supervisée, qui se basent sur 8 différentes techniques de DL, un réseau neuronal profond (DNN), un réseau neuronal convolutif (CNN), deux variantes de réseau neuronal récurrent RNN : un réseau de longue mémoire à court terme (LSTM) et un réseau récurrent à portes (GRU). Et des modèles hybrides : (CNN-LSTM), (CNN-GRU), CNN et bidirectionnel LSTM (CNN-BiLSTM), CNN en combinaison avec bidirectionnel GRU (CNN-BiGRU). Nous avons évalué ces modèles avec le jeu de donnée NSK-KDD, plus particulièrement KDDTrain+ pour la phase de formation et KDDTest20+ pour le test et validation, deux sous-ensembles de données avec d'échantillons les plus difficiles a détectés. Afin de présenté une étude comparative de performance entre ces différents modèles, des différentes mesures ont été appliquée : taux de réussite (Accuracy), Précision, rappel (Recall), moyenne harmonique (F1-Score). Et deux autres indicateurs importants : taux de vraie détection d'intrusions (TPR) et taux de fausse détection de non-intrusion (FPR). Les exprimentations ont donné de très bons résultats, vu la complexité de dataset. DNN marque le meilleur score en taux de réussite totale de 80.72%, surpassant CNN-BiLSTM et CNN-BiGRU qui ont montré une amélioration par rapport aux travaux antérieurs, avec l'avantage d'une excellente précision de 96.36% et FPR=1.51% pour ce dernier ce qui montrent la bonne efficacité des modèles proposés. Notons que nous avons également entrainé et testé les modèles proposés avec un KDDTest moins complexe et les résultats étaient excellents, ce que confirme d'efficacité de nos modèles.

Mots clés : Système de détections d'intrusions (IDS), Apprentissage profond (DL), NSL-KDD (KDDTrain+, KDDTest20+) dataset.

ABSTRACT

In the interest of helping system administrators to detect and prevent breaches of information security in organizations, many researches have made Intrusion Detection Systems (IDS) as object. . This study is conducted with the aim of modeling such a system, which plays a key role in this field. For this, we studied the performance of deep learning (DL) methods for the detection of intrusions in networks. We have implemented 8 models for supervised multi-classification, which are based on 8 different DL techniques, a deep neural network (DNN), a convolutional neural network (CNN), two variants of recurrent neural network RNN : a long-short-term memory network (LSTM) and a gated recurrent network (GRU). In addition, hybrid models : (CNN-LSTM), (CNN-GRU), CNN and bidirectional LSTM (CNN-BiLSTM), CNN in combination with bidirectional GRU (CNN-BiGRU). We evaluated these models with the NSK-KDD dataset, more specifically KDDTrain+ for the training phase and KDDTest20+ for the test and validation, two subsets of data with the most difficult samples to detect. In order to present a comparative study of performance between these different models, different measures were applied : success rate (Accuracy), Precision, Recall, harmonic mean (F1-Score). And two other important indicators : true intrusion detection rate (TPR) and false non-intrusion detection rate (FPR). The experiments gave very good results, given the complexity of the dataset. DNN scores the highest score in total success rate of 80.72%, surpassing CNN-BiLSTM and CNN-BiGRU which showed improvement over previous work, with the advantage of excellent precision of 96.36% and FPR=1.51% for the last one, which show the good efficiency of the proposed models. Note that we also trained and tested the proposed models with a less complex KDDTest and the results were excellent, which confirms the effectiveness of our models.

Key words : Intrusion Detection System (IDS), Deep Learning (DL), NSL-KDD (KDDTrain+, KDDTest20+) dataset.

Table des matières

Table des figures

Liste des tableaux

Liste des acronymes

APIDS : Application Protocol Intrusion Detection System
CNN : Convolutional Neural Network
CSRF : Cross-Site Request Forgery
DBN : Deep Belief Network
DDOS : Distributed Denial of Service
DIDS : Distributed and Collaborative Intrusion Detection System
DL : Deep Learning
DOS : Denial of service
DNN : Deep Neural Network
DNS : Domain Name Service
FN : False Negative
FP : false Positive
GAN : Generative Adversarial Network
HIDS : Host Based Intrusion Detection System
IA : Artificial Intelligence
IDS : Intrusion Detection System
IDWG : Intrusion Detection Working Group
IETF : Internet Engineering Task Force
LSTM : Long Short-Term Memory
MIDS : Mixed Intrusion Detection System
ML : Machine Learning
MLP : Multi-Layer Perceptrons
NBA : Network Based Analysis
NIDS : Network Intrusion Detection System
PIDS : Protocol Intrusion Detection System
RBM : Restricted Boltzmann machine
ReLU : Rectified Linear Unit
RNN : Recurrent Neural Network
RTP : Real-Time Transport Protocol
SQL : Structured Query Language
TCP : Transmission Control Protocol
TN : True Negative
TP : True Positive

V-IDS : Virtual Intrusion Detection System

VMI-IDS : Virtual Machine Introspection Intrusion Detection System

WIDS : Wireless Intrusion Detection System

Introduction générale

Nous vivons maintenant dans un monde sans frontières où rien n'est hors de portée. La croissance profonde et rapide de la technologie a donné lieu à de nouvelles vulnérabilités et menaces à l'ère de l'informatisation. En outre, la dépendance à l'égard des équipements réseau et Internet pour répondre au besoin croissant de services en ligne a certainement augmenté le taux de cybercriminalité. Les menaces et les attaques sont de plus en plus fréquentes et doivent être gérées de manière plus efficace et compétent.
De nos jours, nous voyons quotidiennement de nouvelles techniques d'attaque, par conséquent, il doit y avoir un mécanisme pour surveiller et contrôler ces activités. Il existe un besoin certain d'un nouveau type de protection contre ce nouveau danger. Les systèmes de détection d'intrusion fournissent une deuxième couche de défense avant les techniques de sécurité conventionnelles telles que l'authentification et le contrôle d'accès. En raison de l'importance du maintien de la confidentialité, de la disponibilité et de l'intégrité de nos actifs les plus précieux, à savoir l'information, IDS est devenu une nécessité.
Depuis les années quatre-vingt, les IDS portent une nouvelle voie. Peu à peu les modèles mis en place dans ce contexte, évoluent, potentiellement avec l'évolution des réseaux. L'apparition de l'intelligence artificielle a été un nouvel axe d'intérêts.
Dans notre travail de master nous nous somme focalisés sur l'implémentation et comparaison des systèmes de détection d'intrusion dans les réseaux en exploitant certains algorithmes d'apprentissage profond.

1) Contexte :

Il n'y aura jamais assez ou trop de sécurité mise en œuvre, en particulier avec tous les services en ligne mis à disposition via Internet. Cependant, la sécurité des réseaux est devenue de plus en plus importante. La prévention des attaques est très difficile en utilisant les solutions de sécurité passives, pare-feu (firewall) ou autres mécanismes. Les IDS représentent une technologie efficace aide à la protection contre les différentes attaques.
Le but de notre travail est d'implémenter des systèmes de détection et préventions d'attaques qui se basent sur des travaux existants dans ce domaine. Cse modèles vise à améliorer les taux de détection des intrusions en utilisant les algorithmes d'apprentissage profond (DL) avec le dataset NSL-KDD, plus précisément KDDTrain+ et KDDTest20+ qui marquent les moins taux d'exactitudes avec les

travaux menés avec.

2) Problématique et objectifs :

Aujourd'hui les mécanismes de sécurité ont devenus une nécessité pour se protéger contre toute malveillance sur les réseaux. Les systèmes de détection d'intrusion réseaux (NIDS) sont l'un des mécanismes le plus utilisé aujourd'hui pour détecter les intrusions. Le but des NIDS est de protéger les réseaux contre des attaques qui ne peuvent pas être identifiés par des firewalls. L'un des problèmes majeurs est le taux des faux positifs c'est à dire la mauvaise classification de certaines actions.
Notre objectif dans ce travail est d'implémenter plusieurs modèles qui se basent sur de différentes méthodes de l'apprentissage profond pour la multi-classification des 5 classes de dataset NSL-KDD, dans le but de comparer et distinguer les méthodes les plus appropriés pour ce type de jeux de données qui marque les moins résultats en précision et exactitude, en comparaison a d'autres datasets.

3) Structure du mémoire :

Ce mémoire est décomposé en quatres chapitres, incluant cette introduction
Dans le chapitre 1, nous allons présenter introduction générale sur les systèmes de détection d'intrusion en étudiant ses différents types, classifications et mesures d'évaluation, et présentant les principales menaces liées aux NIDS afin d'avoir une meilleure idée du problème.
Dans le chapitre 2, nous allons présenter une vue globale sur l'apprentissage profond DL, ses fonctionnalités et quelques algorithmes de DL qui seront présentés brièvement et que nous jugeons intéressants pour notre travail.
Concernant le chapitre 3, nous allons donner un aperçu sur les travaux antérieurs qui se basent sur différentes techniques de DL avec différents datasets.

Au final, le chapitre 4 représente une partie application dont laquelle nous expérimentons et obtenons les résultats d'implémentation.

Chapitre 1

Généralité sur le Système de détection d'intrusion

1.1 Introduction

Avec le développement continu des nouvelles technologies liées à l'informatique, il y a un nombre croissant de problèmes de piratage et d'intrusion en cours. Ce piratage et les intrusions peuvent être trouvés dans un large éventail de destinations serveurs DNS, ordinateurs de bureau, les réseaux et l'internet. Network et les failles de sécurité du système d'exploitation peut être exploré et utilisé par un utilisateur malveillant qui tente de prendre le contrôle d'un hôte. Les systèmes de détection d'intrusion (IDS) sont conçus pour tester et étudier un système et réseau pour la susceptibilité au violations [6].
Il est très important que les mécanismes de sécurité d'un système soient conçus de manière à empêcher tout accès non autorisé aux ressources et aux données du système. Cependant, empêcher complètement les failles de sécurité apparaît, à l'heure actuelle, irréaliste. On peut cependant tenter de détecter ces tentatives d'intrusion afin d'intervenir ultérieurement pour réparer les dégâts. Ce domaine de recherche s'appelle la détection d'intrusion [89].
Anderson[60], tout en introduisant le concept de détection d'intrusion en 1980 [36], a défini une tentative d'intrusion ou une menace comme étant la possibilité potentielle d'une tentative délibérée non autorisée de

- accéder aux informations.
- manipuler des informations.
- rendre un système non fiable ou inutilisable.

Depuis, plusieurs techniques de détection d'intrusions ont été étudiées.
Dans ce chapitre, nous allons expliquer plusieurs termes et définitions relatifs à notre thème que nous avons jugé nécessaires à connaitre pour une bonne compréhension du sujet, où nous avons présenté une introduction détaillée au système de détection d'intrusion (IDS) avec son modèle de base, ses catégories, ainsi que ses types, les mesures de son évaluation, et présentant les attaques réseaux qui les menacent.

1.2 Définitions

1.2.1 Intrusion

Une intrusion peut être considérée comme l'ensemble d'actions qui pour le but de compromettre l'intégrité, la confidentialité ou la disponibilité d'une ressource. Ces actions de franchissement d'un accès non-autorisé ou de manipulation interdite d'une ressource peuvent être menées par un individu externe n'ayant aucuns privilèges sur les ressources d'un système, ou par un individu interne qui outrepasse ses privilèges.

1.2.2 Détection d'intrusion

La détection d'intrusion est un ensemble de technique et de méthode employée dans l'analyse des informations collectées par les mécanismes d'audit de sécurité pour détecter toute activité suspecte au niveau du réseau et ses hôtes.

1.2.3 Systèmes détection d'intrusions

Les systèmes de détection d'intrusion (IDS) sont des outils qui parcourent les systèmes ou réseaux informatiques à la recherche de caractéristiques indicatrices d'une violation des politiques de sécurité et les signalent aux analystes de cyber sécurité [7].
Il existe deux types principaux d'IDS chacun utilisant des données différentes :

- Les IDSs réseaux analysent les informations qui sont généralement encapsulées dans des paquets, parcourant un réseau à n'importe quel moment donné. Ces paquets contiennent entre autres les adresses IP expéditrices et récipiendaires liées aux informations en question.
- Les IDSs hôtes recueillent les données des ordinateurs individuels (les appels système, les dynamiques de frappe de clavier ou de la souris, la journalisation, la base de registre, etc.).

car les employés malintentionnés sont déjà présents sur le réseau de l'entreprise et peuvent, par exemple, utiliser les systèmes informatiques pour accéder de façon non autorisée aux fichiers confidentiels et les copier sur un support amovible. Malgré leur popularité, les IDS hôtes sont insuffisants, car plus de la moitié des cas d'exfiltration des données se produit sur les réseaux. En effet, les acteurs internes utilisent principalement leur adresse courriel professionnelle pour envoyer les données confidentielles de l'entreprise à l'extérieur du réseau.

1.3 Modèle de base d'un système de détection d'intrusion

Pour détecter des intrusions, deux approches principales ont été proposées. La première, dite approche comportementale, consiste à modéliser dans une phase initiale le comportement normal d'entités du système pour rechercher ensuite des attitudes déviantes dans le comportement courant de ces entités. La seconde, dite approche par scénarios, consiste à rechercher dans les activités des entités des traces de scénarios d'attaque connus, exploitant les nombreuses vulnérabilités présentes dans les systèmes utilisés aujourd'hui.

L'IDWG (Intrusion Detection Working Group) de l'IETF a défini dans un modèle générique de la détection d'intrusions qui représente les fonctionnalités communes à tous les IDS (qu'ils utilisent l'approche comportementale ou l'approche par scénarios).

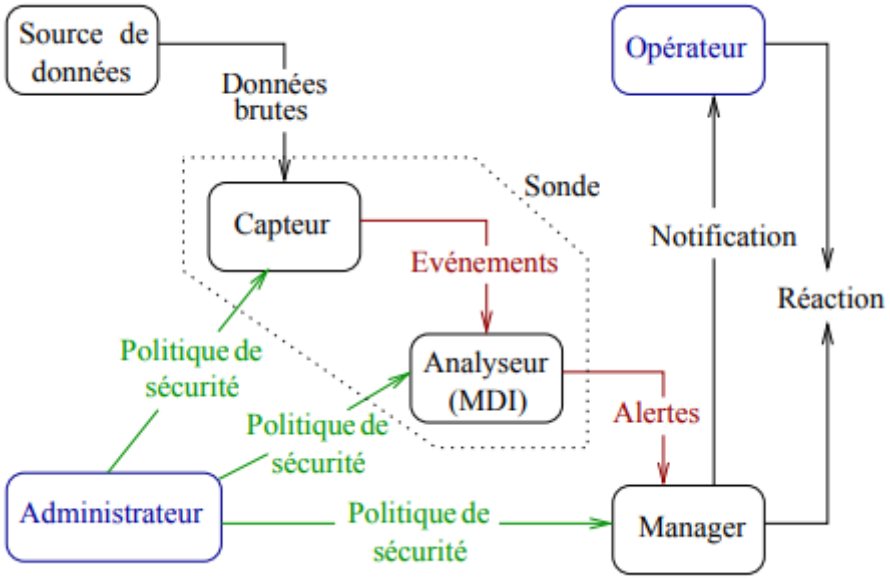


FIGURE 1.1 – Modèle générique de la détection d'intrusions proposé par l'IDWG (Wood and Erlinger, 2012 [8, 73])

Dans la figure 1, on peut voir le processus complet de la détection avec le cheminement des données au sein du système.

L'administrateur configure les différents composants (capteur(s), analyseur(s), manager(s)). Les capteurs accèdent aux données brutes, les filtrent et les formatent pour ne renvoyer que les événements intéressants à un analyseur. Les analyseurs utilisent ces événements pour décider de la présence ou non d'une intrusion et envoient le cas échéant une alerte au manager (qui notifie l'opérateur humain). Une réaction éventuelle peut être menée automatiquement par le manager ou manuellement par l'opérateur[73].
Un IDS particulier n'est pas forcément organisé comme dans la figure 1. Il peut regrouper plusieurs modules logiques en un seul module logiciel ou bien comporter plusieurs instances d'un type de module (ex : plusieurs capteurs ou plusieurs analyseurs).
On note que ce modèle est récursif par nature. En effet, on ne précise pas sur le schéma d'où proviennent les données brutes (qui peuvent donc être des données de plus haut-niveau telles que les alertes). Dans ce cas précis, un manager peut contenir un module permettant de faire de la corrélation à partir de plusieurs alertes (considérées alors comme des événements de base), et à son tour émettre de nouvelles alertes.

Nous pouvons classifier les IDS (Intrusion detection system) selon deux facteurs, son emplacement et son comportement dans une cible donnée.

1.4 Catégories de système de détection d'intrusion

On peut distinguer 4 principales variantes du system de détection d'intrusions, en se basant sur le comportement lors d'inspection d'environnement et données traitées.

1.4.1 Systèmes de détection basé sur la signature

Système de détection basé sur la signature ((Signature based detection system) également appelé basé sur l'utilisation abusive), ce type de détection est très efficace contre les attaques connues, et il dépend de la réception de mises à jour régulières des modèles et sera incapable de détecter les menaces précédentes inconnues ou les nouvelles versions [37].

1.4.2 Système de détection basé sur les anomalies

Anomaly based detection system, ce type de détection dépend de la classification du réseau entre le normal et l'anormal, car cette classification est basée sur des règles ou des heuristiques plutôt que sur des modèles ou des signatures et la mise en œuvre de ce système nécessite d'abord de connaître le comportement normal du réseau. Système de détection basé sur les anomalies contrairement au système de détection basé sur les abus, car il peut détecter les menaces inconnues précédentes, mais les faux positifs augmentent plus probablement.[71]

1.4.3 Système de détection basé sur les spécifications

Ce type de système de détection est chargé de surveiller les processus et de faire correspondre les données réelles avec le programme. En cas de comportement anormal, une alerte sera émise et doit être maintenu et mis à jour chaque fois qu'un changement a été apporté aux programmes de surveillance afin d'être capable de détecter les attaques précédentes l'inconnu et le nombre de faux positifs ce qui peut être inférieur à l'approche du système de détection d'anomalies [91].
La première méthode incapable de détecter des nouvelles intrusions qu'ils n'existent pas dans la base de données, ainsi que La deuxième méthode a un taux de faux positifs élevé (fausse alarme), la dernière méthode est incapable de détecter les attaques qui se ressemblent à des utilisations bénins de protocole.

1.4.4 Système de détection Hybride

Combine plusieurs méthodologies pour fournir une détection plus étendue et précise.

1.5 Types de système de détection d'intrusion

Les IDS sont également classés selon leur emplacement dans un environnement ciblé. On cite 3 types d'IDS en selon ce facteur.

1.5.1 IDS basé sur l'hôte (Host based IDS (HIDS))

HIDS a été le premier type développé de détection d'intrusion. HIDS surveille et analyse le système informatique interne ou les activités au niveau du système d'un seul hôte, telles que : la configuration du système, l'activité des applications, le trafic réseau sans fil (uniquement pour cet hôte) ou l'interface réseau, les journaux système ou le journal d'audit, l'exécution des processus utilisateur ou d'application, les fichiers accès et modifications.
Les capacités de HIDS incluent la vérification de l'intégrité, la corrélation des événements, l'analyse des journaux, l'application des politiques, la détection des rootkits, l'utilisation du processeur, de la mémoire, du disque dur et de la batterie, et l'alerte [47].
HIDS a tendance à être plus précis et moins de faux positifs que l'IDS basé sur le réseau car il analyse les fichiers journaux et, par conséquent, il peut déterminer si une attaque a réussi ou non.
Le système de détection d'intrusion basé sur l'hôte nécessite l'installation de programmes (ou d'agents) sur le système pour générer des rapports indiquant si une activité malveillante s'est produite. Le problème avec les systèmes basés sur l'hôte est qu'ils ont tendance à être gourmands en ressources car ils utilisent les mêmes ressources informatiques installées dessus et n'ont pas de système d'exploitation indépendant comme les autres types d'IDS . Il existe de nombreux systèmes existants qui introduisent le type de système host-ID, par exemple OSSEC et Tripwire [40, 63].
HIDS analyse les fichiers journaux d'audit pour identifier et détecter tout processus système malveillant. Cependant, analyser une grande quantité de données pour faire la distinction entre processus normal et processus malveillants, nécessite un long temps de calcul et beaucoup de ressources.
De nombreuses recherches ont introduit des méthodes pour résoudre ce problème pour les instances : Marteau a introduit une nouvelle mesure de similarité dans les données séquentielles symboliques pour détecter une attaque inconnue. Dans la méthode proposée, l'auteur s'est concentré sur les séquences d'appels système en utilisant l'algorithme de couverture de séquence pour la détection d'intrusion (SC4ID). Algorithme SC4ID basé sur le recouvrement optimal d'une séquence par une série de sous-séquences extraites d'un ensemble prédéfini de séquences. L'algorithme SC4ID a été évalué sur des ensembles de données d'appels système bien connus UNM et ADFA-LD. Subba et al [80] Ont introduit un cadre pour améliorer l'efficacité du calcul dans HIDS. Le cadre proposé a transformé l'appel système en vecteur n-gramme, puis a réduit la taille des vecteurs de caractéristiques d'entrée par un processus de réduction de dimensionnalité. Les vecteurs de caractéristiques sont finalement analysés par divers classificateurs d'apprentissage automatique nommés (Naive Bayes, MLP, C4.5 DecisionTree et SVM) pour identifier les processus intrusifs. Pour évaluer le modèle proposé, le benchmark ADFA-LD a été utilisé. Deshpande et al[80] à proposé pour analyser uniquement les traces d'appel système sélectives pour détecter toute activité malveillante dans le système, puis alerter l'utilisateur du cloud en cas de processus malveillant trouvé.

1.5.2 IDS basé sur le réseau (Network based IDS (NIDS))

NIDS est utilisé pour surveiller et analyser le trafic réseau sur un segment de réseau spécifique pour la détection d'activités suspectes.
NIDS utilisé dans l'analyse au niveau des paquets pour tous les systèmes du segment de réseau en vérifiant les activités au niveau de l'IP, du réseau de transport et du protocole d'application et les entêtes de paquet pour détecter de nombreuses attaques DOS (Denial Of Service) basées sur IP comme l'attaque TCP SYN, l'attaque par paquet fragmenté [75].
NIDS se concentre davantage sur l'abus de vulnérabilités tandis que HIDS se concentre sur l'abus de privilège [63]. NIDS coûte moins cher et plus rapidement en réponse que HIDS car il n'est pas nécessaire de maintenir la programmation du capteur au niveau de l'hôte, et il surveille le trafic en temps réel ou en temps réel proche. Par conséquent, NIDS peut détecter les attaques au fur et à mesure qu'elles se produisent. Cependant, NIDS n'indique pas si de telles attaques réussissent ou non car il n'analyse pas le système de journalisation. Le problème avec NIDS est qu'il a une visibilité restreinte à l'intérieur de la machine hôte et qu'il n'existe aucun moyen efficace d'analyser le trafic réseau crypté pour détecter une attaque [75]. Par conséquent, jusqu'à présent, de nombreuses recherches ont progressé pour développer des moyens efficaces permettant aux NIDS de détecter les attaques. Plusieurs produits de détection d'intrusion réseau existent, tels que Snort et NetSTAT, qui est un outil destiné au NIDS en temps réel.
Jusqu'à présent, de nombreuses recherches ont introduit des méthodes pour les SNID telles que : Sklavounos et al, ont proposé une nouvelle méthode de NIDS pour la détection des attaques DOS basée sur le graphique de la somme cumulative tabulaire (CUSUM) et le graphique de la moyenne mobile pondérée exponentielle (EWMA) sur les octets sources UDP et ICMP de l'ensemble de données expérimental NSL-KDD. Suad Othman et al, ont proposé un modèle de détection d'intrusion dans un environnement de méga données utilisant un algorithme d'apprentissage automatique nommé SVM pour la classification et un sélecteur Chi pour la sélection de fonctionnalités afin de réduire la dimensionnalité du trafic réseau. Pour tester le modèle proposé, l'ensemble de données KDD a été utilisé. Parvat et al, ont proposé des NIDS utilisant l'apprentissage en profondeur.
Dans la méthode proposée a été utilisé un ensemble de classificateurs binaires multiples qui modélisent l'apprentissage en profondeur avec une stratégie de division pour mieux régner. Pour évaluer le système, le jeu de données NSL-KDD a été utilisé.

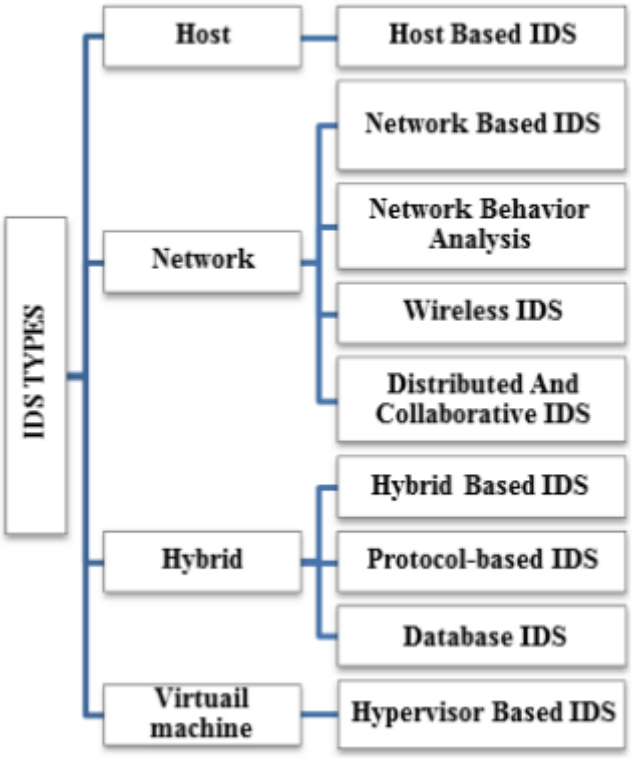


FIGURE 1.2 – Classification des types d'IDS[80]

1.5.3 IDS hybride ou IDS mixte (Hybrid based IDS or mixed IDS (MIDS))

MIDS Combine deux types d'IDS ou plus pour obtenir les avantages de l'IDS et effectuer une détection précise [68] telle que Double Guard qui utilise les identifiants d'hôte et l'ID de réseau. Cependant, MIDS prend beaucoup de temps à analyser les données.
Comme le montre la FIGURE [80], il existe d'autres sous-type d'IDS tel que :

a) **IDS basé sur protocole (Protocol-based Intrusion Detection System (PIDS)) :**

PIDS surveille et vérifie le comportement spécifique du protocole et son état comme le protocole de transfert hypertexte (HTTP) [99] . PIDS peut être spécialisé pour surveiller le protocole d'application, appelé APIDS [99] . Il se concentre sur les actions qui se produisent dans une application particulière en surveillant et en analysant les fichiers journaux de l'application ou en mesurant leurs performances [81] .

b) **Analyse du comportement du réseau (Network Behavior Analysis (NBA)) :**

NBA surveille et vérifie le trafic réseau pour connaître les menaces qui produisent des flux de trafic inhabituels, telles que les attaques DDOS, les logiciels malveillants et les violations de politique [75, 86] . Le système NBA étudie le trafic réseau pour identifier les attaques avec des flux de trafic inattendus [67] . Sindhu [59] A décrit la NBA comme une méthode qui surveille passivement le trafic de mouvement dans un réseau pendant un temps spécifique et forme une

norme pour le trafic normal. De plus, le comportement est comparé à une norme pour trouver toute activité inhabituelle dans un réseau. Les systèmes NBA sont le plus souvent déployés sur les réseaux internes d'une organisation et sont, parfois aussi, déployés là où ils peuvent surveiller les flux entre le réseau d'une organisation et des réseaux externes [78] .

L'avantage de NBA est qu'il se concentre sur le comportement global des appareils sur le réseau ; il est donc permis de répondre à des menaces inconnues ou spécifiques pour lesquelles aucune signature n'est disponible et à une attaque zero-day.

c) **IDS sans fil (Wireless IDS (WIDS)) :**

WIDS surveille et analyse le trafic sans fil pour détecter toute attaque. Le trafic sans fil est un réseau ad hoc, un réseau maillé sans fil et un réseau de capteurs sans fil [67] . Il existe de nombreux types d'attaques dans les réseaux sans fil, telles que l'attaque Sinkhole, l'attaque de routage altérée par usurpation, l'attaque par inondation et l'attaque Sybil [90] .
Les réseaux sans fil ont de nombreuses caractéristiques telles que l'existence dans l'environnement ouvert, la puissance de calcul limitée des capteurs, la durée de vie de la batterie et la capacité de mémoire, par conséquent, les IDS produits pour les réseaux câblés ne peuvent pas être appliqués complètement aux réseaux sans fil. Les réseaux sans fil sont plus vulnérables aux attaques que les réseaux filaires car leurs infrastructures sont dynamiques par nature [42] .

d) **IDS distribué et collaboratif (Distributed and Collaborative IDS (DIDS)) :**

se compose de plusieurs IDS sur un réseau, qui communiquent tous entre eux ou avec un serveur central qui permet la surveillance du réseau [**?**].
DIDS est conçu pour fonctionner dans un environnement non homogène, ce qui signifie que DIDS offre la possibilité d'agréger des informations provenant de différentes sources pour détecter les attaques contre un système réseau telles que les attaques par poignée de porte et les attaques DDoS. Il existe trois composants dans le cadre de DIDS, qui sont l'agent IDS, le composant de communication et le serveur d'analyse central. Le DIDS présente plusieurs avantages par rapport à l'IDS centralisé [51, 64].

e) **IDS de base de données (Database IDS) :**

IDS de base de données surveille et vérifie les attaques contre la base de données [87] . Il existe plusieurs types d'attaques de base de données telles que l'attaque par injection SQL, Direct DB Attack . Plusieurs recherches ont porté sur l'attaque par injection SQL, par exemple : Liu A et al. [69] a proposé un bloqueur basé sur un proxy SQL. Dans la méthode proposée par SQLProb, les algorithmes génétiques ont été exploités pour détecter et extraire dynamiquement les entrées des utilisateurs pour les SQL indésirables, et utilisé un proxy intégré à l'environnement offrant une protection aux serveurs Web frontaux et aux bases de données principales.

f) **IDS basé sur l'hyperviseur (IDS basé sur l'introspection de machine virtuelle (VMI-IDS)**

IDS virtuel (V-IDS) :

Le concept de VMI a été introduit pour la première fois par Garfinkel et al. [50] en tant qu'IDS au niveau de l'hyperviseur qui offrait une isolation pour l'IDS, tout en offrant une visibilité sur l'état de l'hôte. L'IDS basé sur VM est avancé sur la base de trois capacités de VM : isolation, inspection et interposition [70].
L'hyperviseur est une plate-forme qui exécute des IDS basés sur l'hyperviseur de machines virtuelles qui fonctionnent au niveau de la couche hyperviseur. Il permet aux utilisateurs de surveiller et d'analyser les connexions entre les VM ou entre l'hyperviseur et la VM et au sein du réseau virtuel basé sur l'hyperviseur [74, 75] . Il peut conserver et appliquer diverses stratégies de sécurité pour chaque machine virtuelle en fonction de leurs besoins [74].

1.6 Les attaques réseaux

Dans cette section, nous intéressons aux attaques que l'on souhaite détecter dans notre travail. Ghorbani et al, Définissent les attaques réseaux comme l'activité malicieuse visant l'interruption, la dégradation, la perturbation de services accessibles aux traves d'un réseau. L'objectif de ces attaques est de porter atteinte à l'intégrité, la disponibilité ou la confidentialité de ces services. Les attaques sont diverses et variées et peuvent cibler une machine ou un système complet.
Quant à Hansman et al [54], ils définissent les attaques réseaux comme des attaques visant un réseau ou des utilisateurs en manipulant les protocoles réseaux de la couche physique à la couche application. Dans ce mémoire, nos IDS's traitent une attaque réseau au niveau de réseau de communication en tant que NIDS. Nous présentons dans la suite de cette section des attaques réseaux.

1.6.1 Le buffer overflow (dépassement de tampon)

Les attaques par débordement de tampon (en anglais Buffer overflow, parfois également appelées dépassement de tampon) ont pour principe l'exécution de code arbitraire par un programme en lui envoyant plus de données qu'il n'est censé en recevoir[11].

En effet, les programmes acceptant des données en entrée, passées en paramètre, les stockent temporairement dans une zone de la mémoire appelée tampon (en anglais buffer). Or, certaines fonctions de lecture, telles que les fonctions strcpy() du langage C, ne gèrent pas ce type de débordement et provoquent un plantage de l'application pouvant aboutir à l'exécution du code arbitraire et ainsi donner un accès au système.

La mise en œuvre de ce type d'attaque est très compliquée car elle demande une connaissance fine de l'architecture des programmes et des processeurs. Néanmoins, il existe de nombreux exploits capable d'automatiser ce type d'attaque et la rendant à la portée de quasi-néophytes.

1.6.2 Les virus, les vers et les chevaux de troie

Sont des programmes dont l'objectif est d'infecter une machine hôte. Les actions réalisées par ces programmes sont variées : il peut s'agir d'envoi de courriels indésirables, d'écoute passive ou de recherche d'informations confidentielles, ou encore de manipulation de hôte pour réaliser des malveillances.
Les vecteurs de propagation de ces attaques sont également très divers, mais la plupart de ces menaces se répliquent en utilisant le réseau. Qu'il s'agisse d'une pièce jointe dans un courriel, ou bien de recherche active de victimes potentielles, la phase de réplication peut être détecté en cherchant des motifs particuliers dans les communications portées par le réseau [12].

1.6.3 Les bots nets

Le bot net(ou réseau de machines zombies) est un hybride des menaces précédentes intégré à un système de commande et de contrôle et des centaines de millions d'ordinateurs connectés à Internet sont infectés [66]. à l'insu de leurs propriétaires respectifs, ont été configurés de manière à transmettre des informations (notamment des spams ou des virus) à d'autres ordinateurs reliés à Internet[13].
Bot net on traction de l'anglais (robot net : réseau de robots), un réseau de bots informatiques, des programmes connectés à Internet qui communiquent avec d'autres programmes similaires pour l'exécution de certaines tâches.

1.6.4 Le déni de service (Denial of Service en anglais)

Est une attaque dont l'objectif principal est de dégrader les performances d'un système. Ces attaques peuvent cibler différent composants du système, tels que la mémoire, le processeur ou encore la carte réseau. La popularité de ces attaques est grandissante ces dernières années, à tel point que l'on parle maintenant de Distibuted Denial of Service (DDoS) où l'attaque est menée non par un seul système, mais par un grand nombre d'hôtes attaquants. Le vecteur de propagation par excellence de cette attaque est le réseau.

1.7 Les mesures d'évaluation d'IDS

Les mesures qui nous permettent d'évaluer l'efficacité globale des systèmes de détection d'intrusion selon (Debar et al. [20]) sont :

- **La précision :** le système IDS est précis lorsqu'il détecte les attaques sans faire des fausses alarmes. La non-précision survient lorsqu'il déclare comme anormale ou instructive une action légitime dans l'environnement.
- **La performance de traitement :** est mesurée par la vitesse dans laquelle les événements sont traités. Lorsque le système IDS est plus performant alors la détection en temps réel sera possible.
- **La complétude :** c'est la capacité d'un IDS de détecter toutes les attaques.

- **La tolérance aux pannes :** la plupart des systèmes de détection d'intrusion s'exécutent dans des systèmes d'exploitation ou des matériels qui sont connus pour être vulnérables aux attaques. Donc un IDS devrait être résistant à ces attaques en particulier les attaques de déni de service.
- **La rapidité :** L'IDS doit être plus rapide dans l'analyse et l'exécution pour minimiser le temps de réagir, et aussi pour empêcher l'attaquant d'altérer la source de vérification ou interrompre le fonctionnement du système[44].

Généralement, un taux de détection élevé est nécessaire pour un système IDS afin de prévenir les attaques avant de causer tout type de violations de sécurité pour les systèmes IDSs basés sur la machine learning une précision de détection élevée avec un faible taux de fausses alarmes est essentiel pour l'éfficacité des Systèmes. Les principaux aspects à considérer lors de la mesure de détection et la précision de classification des attaques sont :

- **True Positive (TP) :** nombre d'intrusions correctement détectées.
- **True Negative (TN) :** nombre de non-intrusions correctement détectées.
- **False Positive (FP) :** nombre de non-intrusions mal détectées.
- **False Negative (FN) :** nombre d'intrusions mal détectées

Il existe plusieurs types d'erreurs venant d'un détecteur, influençant plus ou moins sa puissance. Les vrais positifs sont les cas où une alarme se déclenche quand il y a une violation des politiques de sécurité. Les vrais négatifs sont les cas où aucune alarme ne se déclenche et rien d'anormal ne se produit. Les faux positifs sont les cas où une alarme se déclenche alors qu'il ne se produit rien d'anormal. Les faux négatifs sont les cas où une alarme ne se déclenche pas alors qu'il se produit une chose anormale. À la première vue, on pourrait supposer qu'un faux positif est moins dangereux qu'un faux négatif.

1.8 Conclusion

Dans ce chapitre, nous avons donné un aperçu général sur les IDS définissant le modèle de base de ces systèmes, et décrivant les types des IDS existants. En outre nous avons montré certains types d'attaques qui peuvent menacer les systèmes informatiques à partir des réseaux, ainsi les différentes mesures d'évaluation des systèmes IDSs. Dans le chapitre suivant, nous allons parler sur l'apprentissage profond.

Chapitre 2

Apprentissage profond

2.1 Introduction

L'apprentissage automatique (Machine Learning) est une forme d'intelligence artificielle qui permet à un système d'identifier et prendre des décisions avec un minimum d'intervention humaine à partir de bases de données d'échantillons, donc d'apprendre à partir des données structurées et non à l'aide d'une programmation. Au fur et à mesure que les algorithmes ingèrent les données de formation, il devient possible de créer des modèles plus précis basés sur ces données.
L'apprentissage profond (deep Learning DL) un type d'apprentissage automatique qui imite la façon dont les humains acquièrent certains types de connaissances plus précisément tentent d'imiter le traitement d'information et de communication observés dans le système nerveux biologique, comme le codage neuronal qui tente de définir et de décrire les interrelations existant entre plusieurs stimuli et les réponses neuronales associées dans le cerveau [65]. Ce système a révolutionné les industries technologiques. La traduction automatique moderne, les moteurs de recherche et les assistants informatiques sont tous alimentés par un apprentissage profond. Dans ce chapitre on va s'intéresser au système d'apprentissage profond et ces méthodes de détection [38].

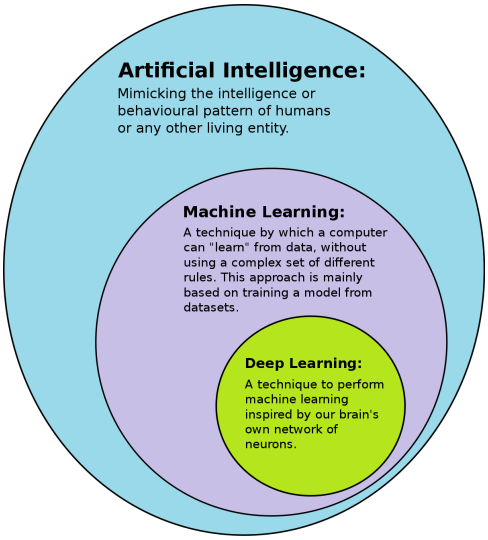


FIGURE 2.1 – La relation entre IA et ML et DL[9]

2.2 Définitions

2.2.1 Définition de l'apprentissage automatique

L'apprentissage automatique peut être présenté comme la capacité d'une machine à apprendre sans être explicitement programmée. Dans ce contexte, l'apprentissage automatique est à l'origine de nombreuses organisations publiques et privées et de la société moderne, de la recherche sur le web, le filtrage du contenu sur les réseaux sociaux et bien d'autre encore.
L'apprentissage automatique est essentiellement une forme de statistiques appliquées qui met davantage l'accent sur l'utilisation d'ordinateurs pour estimer statistiquement des fonctions compliquées et moins sur la preuve d'intervalles de confiance autour de ces fonctions. Ainsi, l'apprentissage automatique peut être utilisé pour identifier des objets dans des images, de transcrire la parole en texte, de faire correspondre de nouveaux éléments, de sélectionner des résultats de recherche pertinents, etc[76].

2.2.2 Définition de l'apprentissage profond

L'apprentissage profond (DL) fait référence à une classe de techniques d'apprentissage automatique (ML) qui utilise des algorithmes pour imiter le cerveau humain le plus fidèlement possible. Les architectures des modèles profondes sont relativement récentes où de nombreuses étapes de traitement non linéaire de l'information sont exploitées, dans lesquelles les informations sont traitées en couches hiérarchiques, chacune recevant et interprétant les informations de la couche précédente pour l'apprentissage des représentations de données[[48].
L'apprentissage des représentations de données pourrait se faire par le biais d'approches semi-supervisées, supervisées ou non supervisées. En pratique, tous les algorithmes d'apprentissage profond sont des réseaux de neurones (Neural networks), qui partagent certaines propriétés de base communes. Ils sont tous constitués de neurones interconnectés organisés en couches. Ce qui les différencie, c'est l'architecture du réseau (ou la manière dont les neurones sont organisés dans le réseau) et parfois la manière dont ils sont formés [65].

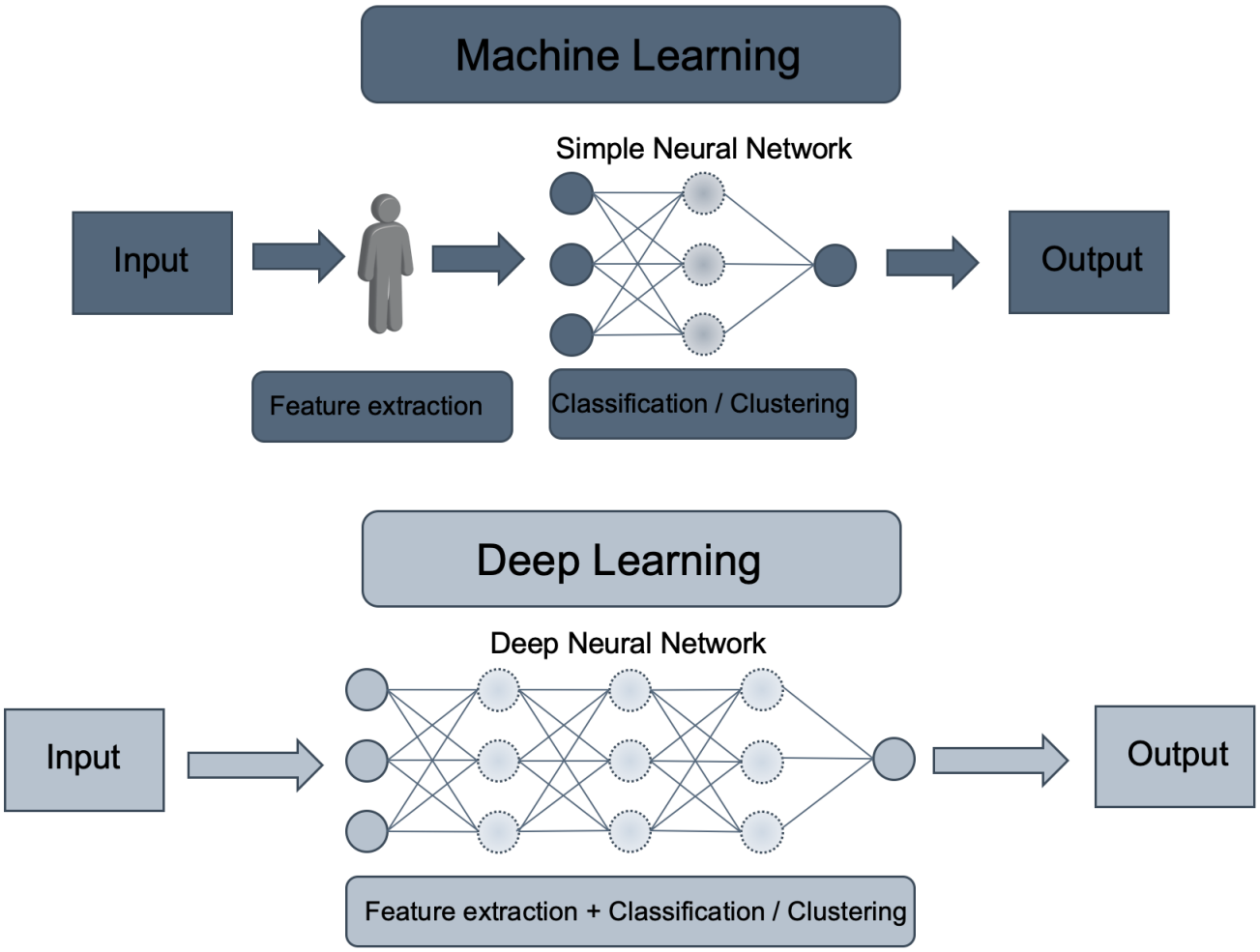


FIGURE 2.2 – Machine learing vs Deep learing

2.3 Le principe et le fonctionnement de l'apprentissage profond

L'apprentissage profond se présente comme un système de calcul avancé, il est constitué d'une variété de techniques issues du domaine de l'apprentissage automatique qui utilisent un déluge de neurones (nœuds) non linéaires disposés en plusieurs couches de traitement qui extraient et convertissent des valeurs de variables d'entité à partir du vecteur d'entrée pour créer plusieurs niveaux d'abstraction afin de représenter les données. L'apprentissage du DNN c'est l'optimisation des paramètres de poids et le paramètre de biais entre deux couches voisines, Il évalue la justesse du modèle et permet de mieux l'adapter aux données d'apprentissages. Lorsque le modèle arrive à une précision maximale avec des paramètres optimaux, il sera 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[46].

L'apprentissage profond est une branche des algorithmes d'apprentissage automatique qui :

- Utilise plusieurs couches de nœuds de traitement non linéaires pour l'extraction et la transformation d'entités. Les couches successives utilisent les sorties des couches précédentes en entrée.

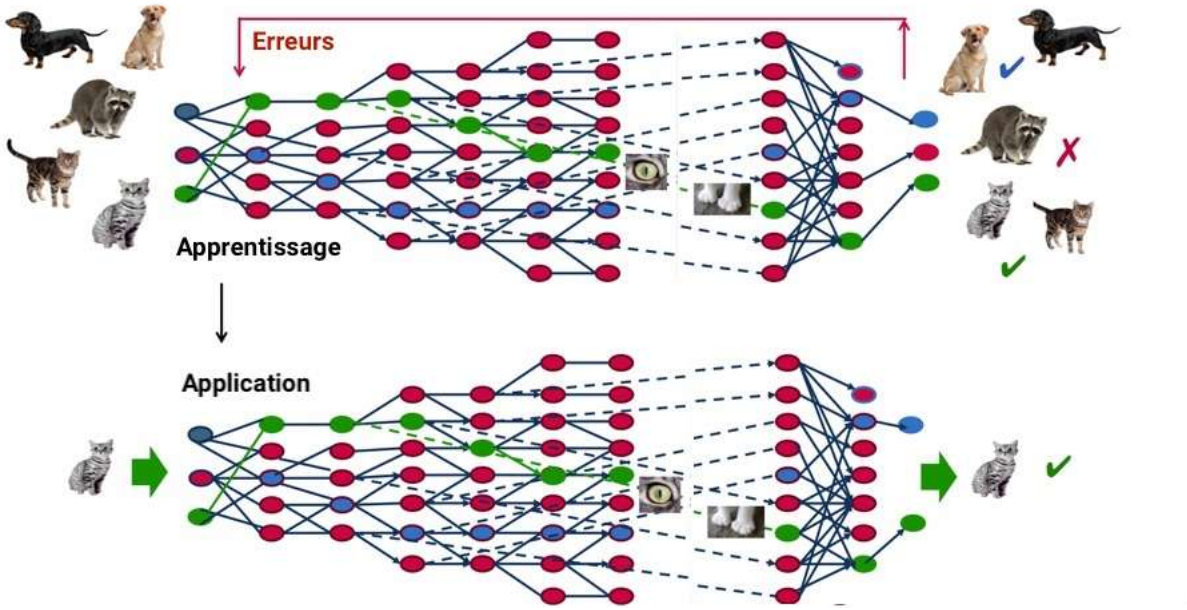


FIGURE 2.3 – Fonctionnement de deep learning[10]

- Apprends de manière non supervisée (analyse de modèle, par exemple) et / ou supervisée (classification, par exemple).
- Apprends plusieurs niveaux de représentations liés à différents niveaux d'abstraction. Ces niveaux représentent une hiérarchie de concepts. [38]

2.4 Classification de l'apprentissage profond

L'apprentissage profond peut être classé en trois modèles principales en fonction de la manière dont elles sont formées et destinées à être utiliser. [46]

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

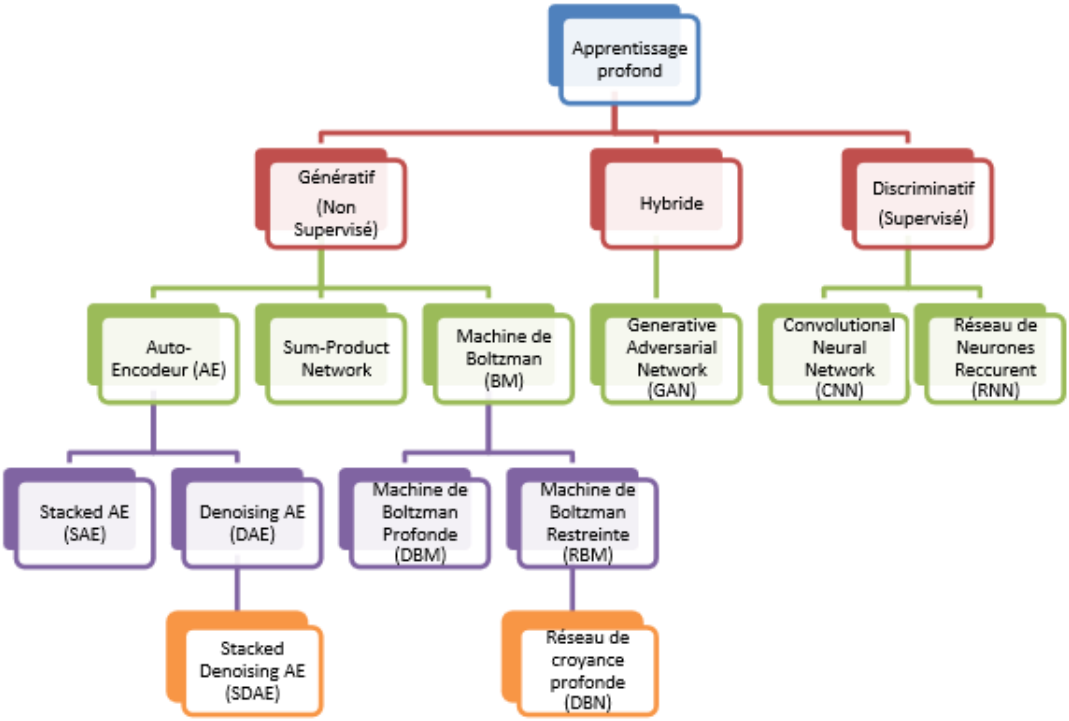


FIGURE 2.4 – Classification des méthodes d'apprentissage profond [38]

2.5 Quelques méthodes d'apprentissage profond

Tous les algorithmes de l'apprentissage profond sont des réseaux de neurones, qui partagent certaine propriétés de base communes. Nous allons présenter ci-dessus les principales méthodes d'apprentissage profond qui contient les algorithmes les plus utilisés de nos jours

2.5.1 Machines de Boltzmann Restreintes (RBM)

Machine de Boltzmann restreinte est un réseau neuronal artificiel stochastique à deux couches avec des capacités génératives. Ils ont la capacité d'apprendre une distribution de probabilité sur son ensemble d'entrées.

Les RBM sont une classe spéciale de machines Boltzmann et elles sont limitées en termes de connexions entre les unités visibles et cachées. Cela facilite leur mise en œuvre par rapport aux machines Boltzmann. Comme indiqué précédemment, il s'agit d'un réseau de neurones à deux couches (l'une étant la couche visible et l'autre étant la couche cachée) et ces deux couches sont connectées par un graphe entièrement bipartite. Cela signifie que chaque nœud de la couche visible est connecté à chaque nœud de la couche cachée, mais que deux nœuds du même groupe ne sont pas connectés entre eux. Cette restriction permet des algorithmes d'apprentissage plus efficaces. Il est possible d'empiler plusieurs couches de machines de Boltzmann restreintes pour créer des réseaux profonds qui sont plus performants [23]. Une machine Boltzmann, est un réseau d'unités avec une " énergie " définie pour l'ensemble du réseau. La fonction d'activation pour une Machine de Boltzmann Restreinte est définie

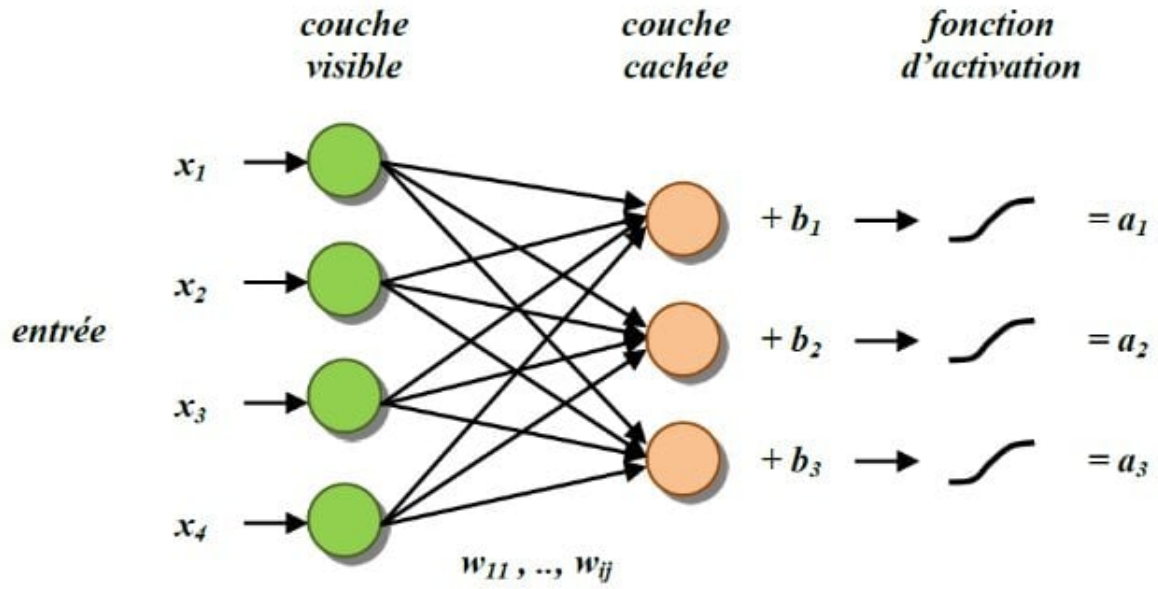


FIGURE 2.5 – Machines de Boltzmann Restreintes[23]

comme suit :
E=-[$\sum_{ij}(w_{ij}x_ih_j)$+$\sum_i(b_ix_i)$+$\sum_j(c_jh_j)$] [48]
Avec :

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

2.5.2 Perceptrons multicouches (MLP)

Le perceptron multicouche (MLP) est un type de réseau neuronal artificiel à anticipation avec plusieurs couches de perceptrons dotés de fonctions d'activation. Il se compose de plusieurs couches au sein desquelles une information circule de la couche d'entrée vers la couche de sortie uniquement. Ils ont le même nombre de couches d'entrée et de sortie mais peuvent avoir plusieurs couches cachées. Chaque couche est constituée d'un nombre variable de neurones, les neurones de la dernière couche étant les sorties du système global[17].

Les couches de perceptron sont organisées en trois parties :

- La couche d'entrée (input layer) = un ensemble de neurones qui portent le signal d'entrée.
- La couche cachée (hidden layer) ou plus souvent LES couches cachées (couche cachée 1, couche cachée 2,...). Il s'agit du cœur de notre perceptron, là où les relations entre les variables vont être mises en exergue.
- La couche de sortie (output layer) : cette couche représente le résultat final de notre réseau, sa prédiction.

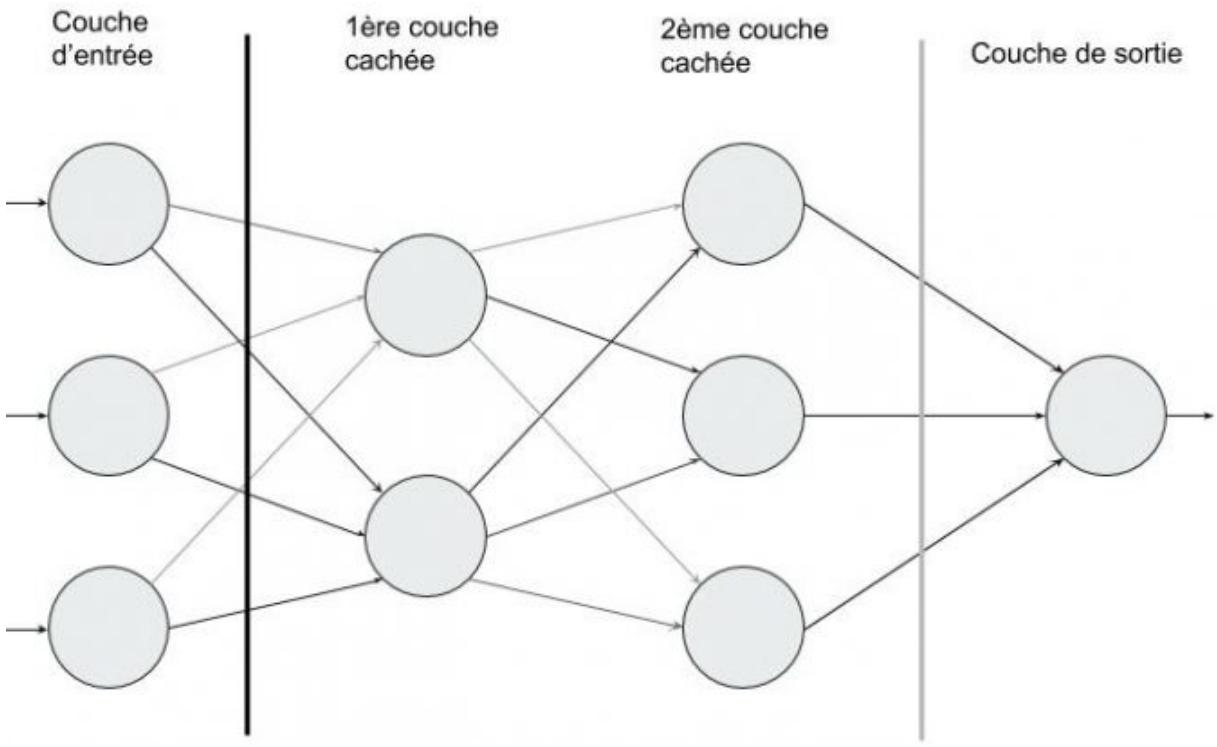


FIGURE 2.6 – Perceptrons multicouches[22]

2.5.3 Réseaux de croyance profonde (DBN)

Réseau DBN est un type sophistiqué de réseau neuronal profonds qui est essentiellement un modèle génératif comprenant plusieurs couches de variables cachées. Ces réseaux ont à la fois des bords dirigés et non dirigés[23]. Ce type de réseau illustre une partie du travail qui a été accompli récemment en utilisant des données relativement sans étiquette pour créer des modèles non supervisés.

Réseau DBN est un réseau de croyances multicouches dans lequel chaque couche est un RBM empilé les uns sur les autres pour former le réseau DBN en suivant les étapes suivante :

- La première étape de l'apprentissage d'un DBN consiste à apprendre une couche de caractéristiques des unités visibles.
- L'étape suivante consiste à traiter les activations des caractéristiques précédemment formées comme des unités visibles, puis à en apprendre davantage à partir des caractéristiques des couches cachées suivantes.
- La dernière étape consiste à former l'ensemble du réseau DBN de manière supervisée, ce qui permet d'affiner les paramètres du réseau.

Comme indiqué précédemment, les RBM peuvent être empilés et formés de manière gourmande pour former ces DBN qui sont des modèles graphiques permettant d'apprendre à extraire des représentations hiérarchiques profondes des données d'entraînement en entrée.

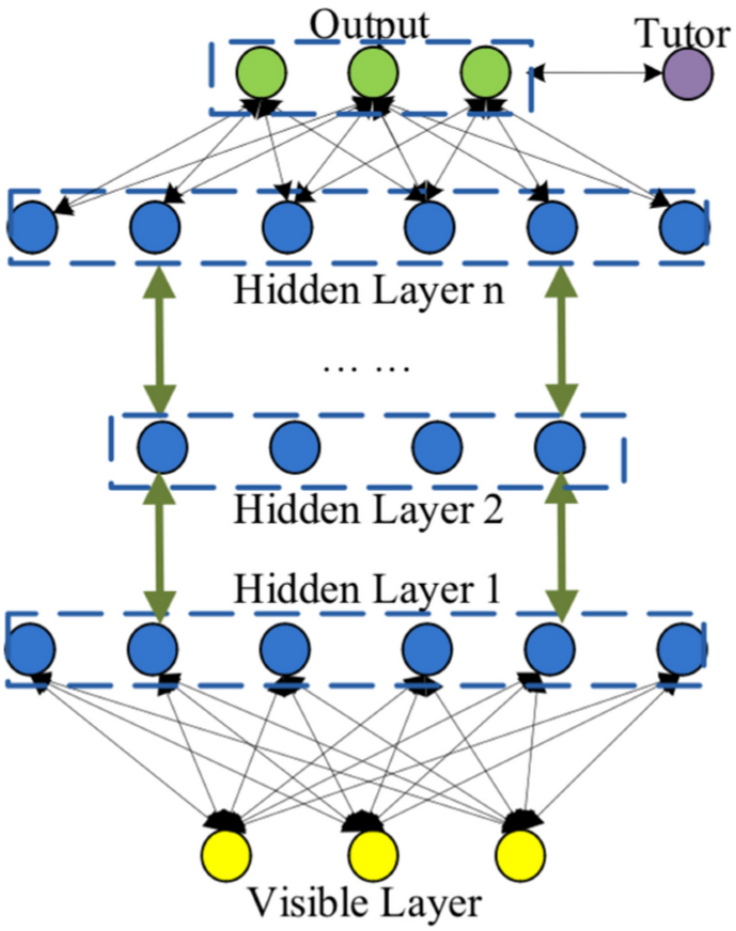


FIGURE 2.7 – Réseaux de croyance profonde[14]

2.5.4 Réseaux neuronaux convolutifs (CNN)

Un CNN est un réseau de neurones artificiels acycliques à propagation directe (feed-forward) avec plusieurs types de couches spéciales. Par exemple, les couches convolutives appliquent un filtre à l'image d'entrée (ou au son) en faisant glisser ce filtre sur tout le signal entrant, afin de produire une carte d'activation (activation map) à n dimensions. Il existe certaines preuves que les neurones dans les CNN sont organisés de la même manière que les cellules biologiques organisées dans le cortex visuel du cerveau. Aujourd'hui, les CNN surpassent tous les autres algorithmes ML pour un grand nombre de tâches dans lesquelles des motifs répétitifs peuvent être trouvés comme le traitement de vision par ordinateur (la reconnaissance d'image et vidéo) et le traitement du langage naturel. Les réseaux de convolution sont particulièrement efficaces pour capturer les modèles locaux de données en raison de la structure des couches de convolution [18].

Les CNN ont plusieurs couches qui traitent et extraient les caractéristiques des données :

- **Couche de convolution :** Le CNN possède une couche de convolution qui comporte plusieurs filtres pour effectuer l'opération de convolution.
- **Unité linéaire rectifiée (ReLU) :** Les CNN ont une couche ReLU pour effectuer des opérations sur les éléments. La sortie est une carte de caractéristiques rectifiée.
- **Couche de mise en commun :** La carte de caractéristiques rectifiée alimente ensuite une

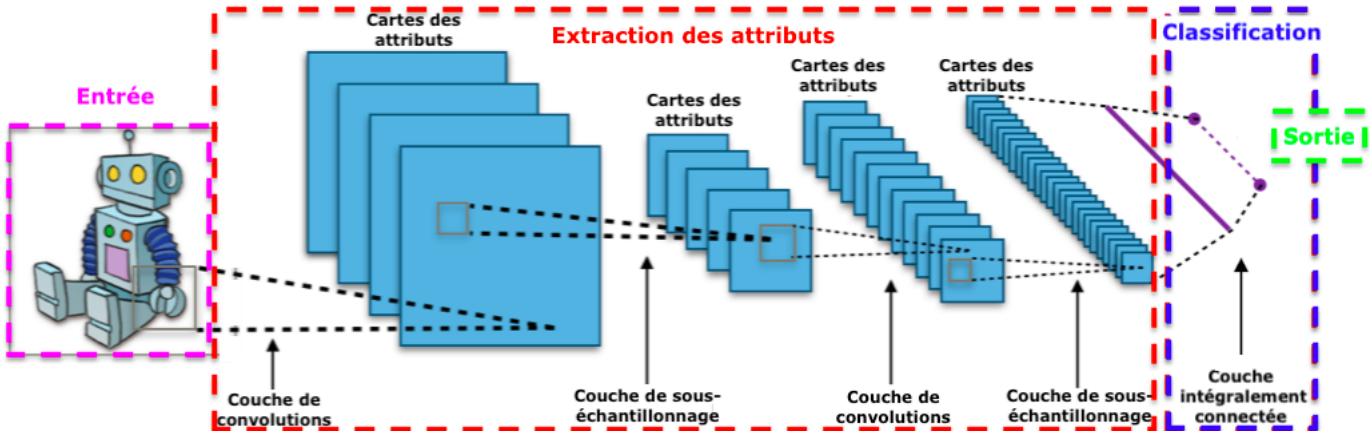


FIGURE 2.8 – Schéma d'un réseau de neurones convolutifs[24]

couche de mise en commun. La mise en commun est une opération de sous-échantillonnage qui réduit les dimensions de la carte des caractéristiques.

- **La couche de regroupement :** Convertit les tableaux bidimensionnels résultants de la carte d'entités regroupées en un seul vecteur linéaire, long et continu, en les aplatissant.
- **Couche entièrement connectée :** Une couche entièrement connectée se forme lorsque la matrice aplatie de la couche de regroupement est utilisée comme entrée, ce qui permet de classer et d'identifier les images.

2.5.5 Réseaux de neuronaux récurrents (RNN)

Les réseaux de neurones récurrents RNN, sont un type de réseau de neurones largement utilisé dans le domaine de l'apprentissage en profondeur (Deep Learning) . Les RNN ont des connexions qui forment des cycles dirigés, ce qui permet aux sorties du LSTM d'être utilisées comme entrées dans la phase actuelle et sont parfaitement adaptés au traitement de données séquentielles. La sortie du LSTM devient une entrée pour la phase actuelle et peut mémoriser les entrées précédentes grâce à sa mémoire interne. Généralement [19], elles se présentent sous la forme suivante :

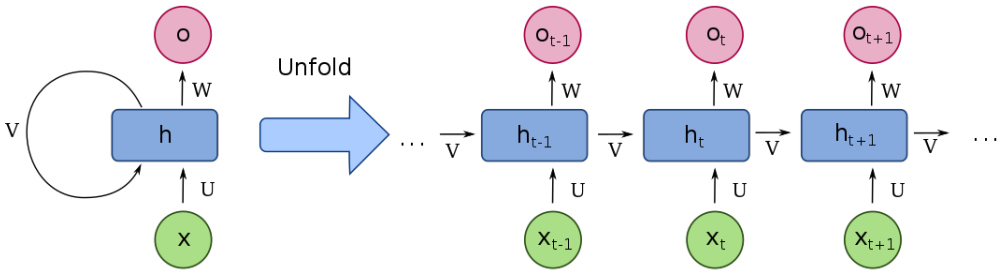


FIGURE 2.9 – Schéma d'un réseau de neurones récurrents à une unité reliant l'entrée et la sortie du réseau[25]

Les RNN sont couramment utilisés pour le sous-titrage d'images, l'analyse de séries temporelles, le traitement du langage naturel, la reconnaissance de l'écriture manuscrite et la traduction automatique.

2.5.6 Mémoire longue à court terme (LSTM)

Les unités ou blocs de mémoire à court terme long font partie d'une structure de réseau neuronal récurrent. est une unité complexe avec divers composants tels que des entrées pondérées, des fonctions d'activation, des entrées de blocs précédents et des sorties éventuelles.
L'unité est appelée un bloc de mémoire à long terme car le programme utilise une structure fondée sur des processus de mémoire à court terme pour créer une mémoire à plus long terme. Ces systèmes sont souvent utilisés, par exemple, dans le traitement du langage naturel. Le réseau neuronal récurrent utilise les blocs de mémoire à court terme pour prendre un mot ou un phonème particulier et l'évaluer dans le contexte des autres dans une chaîne, où la mémoire peut être utile pour trier et catégoriser ces types d'entrées. En général, LSTM est un concept accepté et commun dans les réseaux de neurones récurrents pionniers [65]. Voici une figure qui illustre Schéma d'un réseau LSTM.

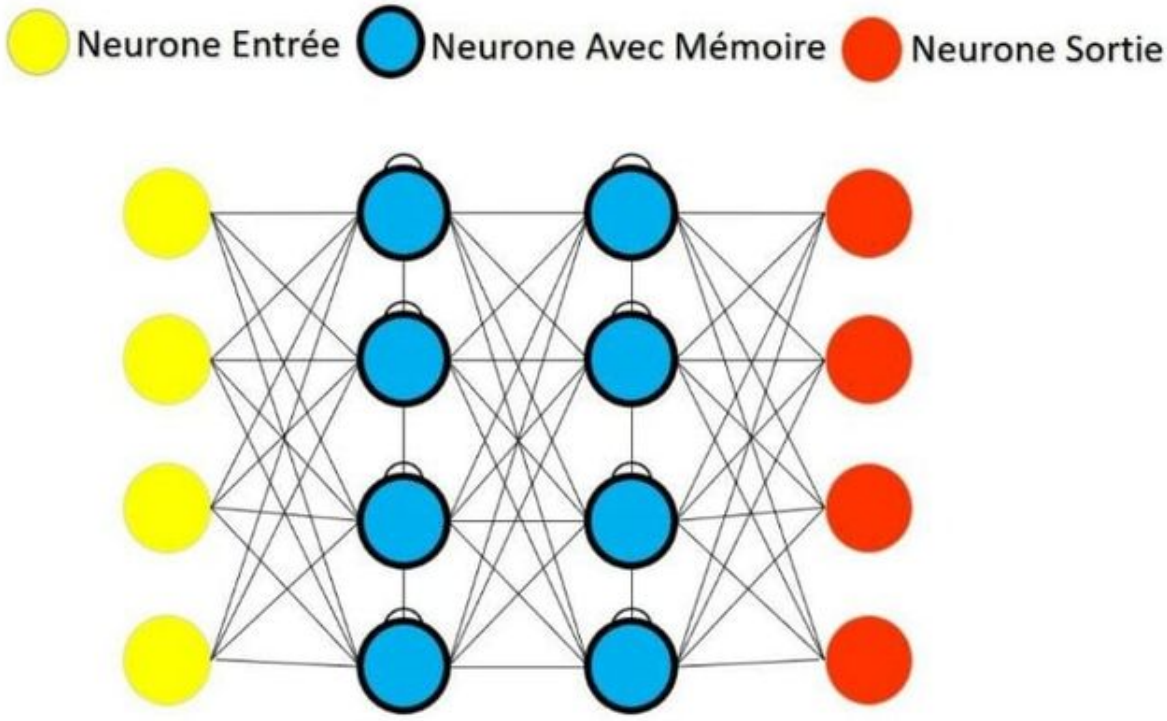


FIGURE 2.10 – Schéma d'un réseau LSTM

2.5.7 Auto-encodeurs

Les auto-encodeurs sont des algorithmes d'apprentissage non supervisé à base de réseaux de neurones artificiels, qui permettent de construire une nouvelle représentation d'un jeu de données. L'architecture d'un auto-encodeur est constituée de deux parties : l'encodeur et le décodeur qui

servent a minimiser l'erreur de reconstruction entre les données codées/décodées et les données initiales.
L'encodeur est constitué par un ensemble de couches de neurones, qui traitent les données afin de construire de nouvelles représentations dites encodées. À leur tour, les couches de neurones du décodeur, reçoivent ces représentations et les traitent afin d'essayer de reconstruire les données de départ. Les différences entre les données reconstruites et les données initiales permettent de mesurer l'erreur commise par l'auto-encodeur. L'entraînement consiste à modifier les paramètres de l'auto-encodeur afin de réduire l'erreur de reconstruction mesurée sur les différents exemples du jeu de données [20]. La plupart du temps, on ne s'intéresse pas à la dernière couche du décodeur, qui contient uniquement la reconstruction des données initiales, mais plutôt à la nouvelle représentation créée par l'encodeur. Ci-dessous On vous présente une architecture de l'auto-encadreur.

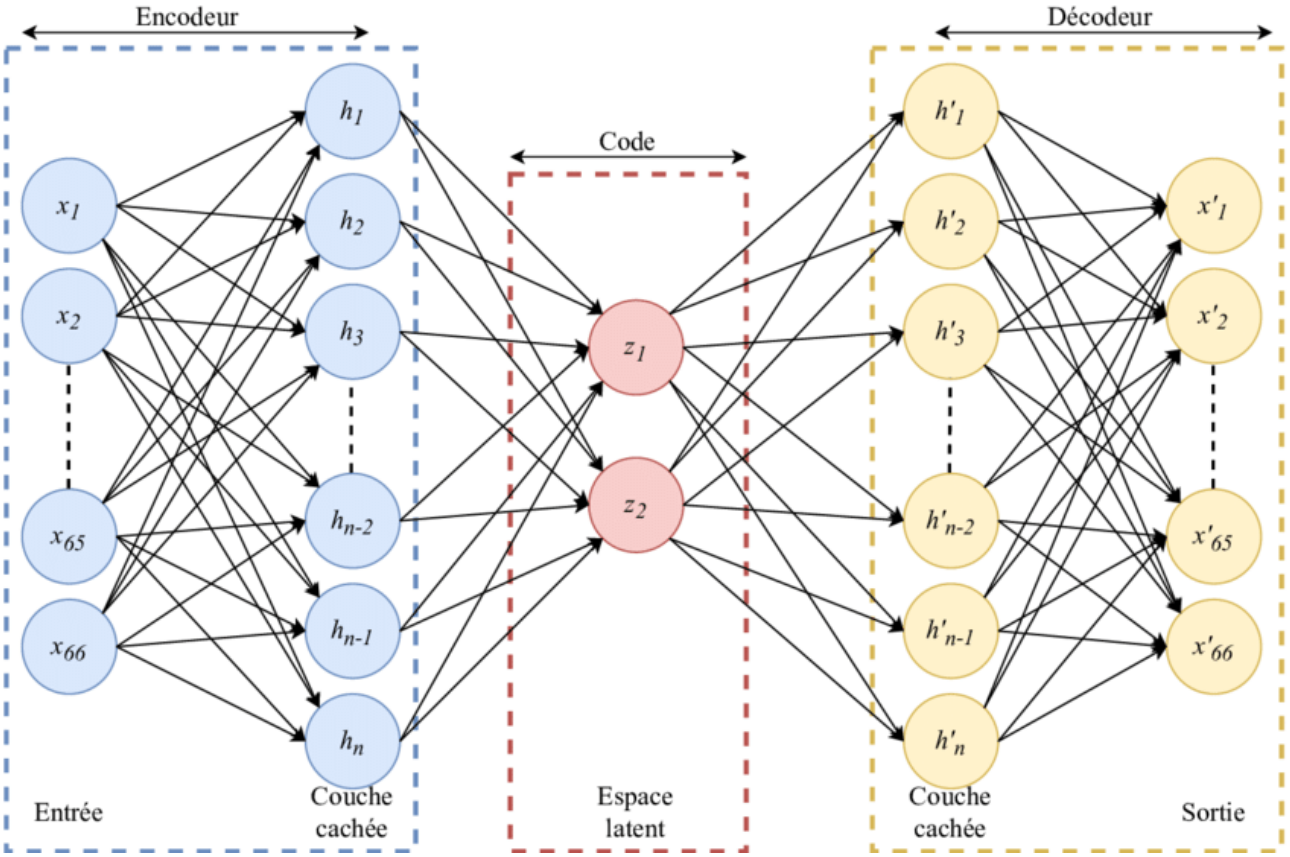


FIGURE 2.11 – Structure schématique d'un auto-encodeur[15]

2.5.8 Réseaux adversariaux génératifs (GAN)

En intelligence artificielle, les réseaux adverses génératifs sont une classe d'algorithmes d'apprentissage non supervisé. Qui créent de nouvelles instances de données qui ressemblent aux données d'apprentissage. Les GAN ont deux composants : un générateur, il génère un échantillon (apprend à générer des données fausses), et un discriminateur, détecté si un échantillon est réel ou bien s'il est le résultat du générateur (apprend à partir de ces fausses informations).

L'utilisation des GANs a augmenté au fil du temps. Ils peuvent être utilisés pour améliorer les images astronomiques et simuler les lentilles gravitationnelles pour la recherche sur la matière noire. Les développeurs de jeux vidéo utilisent les GAN pour améliorer les textures 2D à faible résolution des anciens jeux vidéo en les recréant en 4K ou à des résolutions plus élevées via l'apprentissage d'images. Voici Ci-dessous schéma de réseau génératif [21].

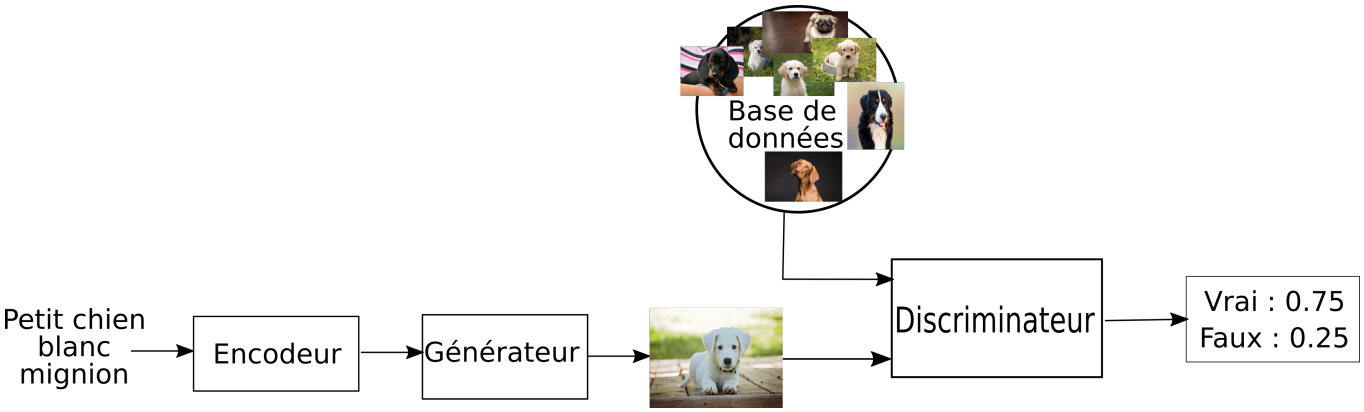


FIGURE 2.12 – Schéma d'un réseau adversarial génératif[26]

2.6 Conclusion

Les Réseaux de neurones sont un domaine de recherche très actif, qui ne cesse de progresser afin d'améliorer les performances des résultats.
L'apprentissage profond (DL) a pu s'imposer et révolutionner plusieurs domaines technologiques. Traduction automatique moderne, moteurs de recherche, assistants informatiques et plusieurs applications de notre vie quotidienne sont tous alimentés par un apprentissage profond. Les modèles d'apprentissage profond tentent d'imiter au maximum les modes de traitement de l'information et de communication observés dans le système nerveux biologique.
Ce chapitre a présenté les architectures des réseaux de neurones les plus connues et les plus utilisées ainsi que les nouvelles architectures qui semblent avoir un avenir prometteur dans différents domaines d'application de la technologie contemporaine.

Chapitre 3

Détection d'intrusion avec les méthodes d'apprentissage profond

3.1 Introduction

Récemment, plusieurs approches d'apprentissage en profondeur ont été envisagées pour la détection d'intrusion. Les systèmes de détection basés sur les signatures sont des intrusions détecté en comparant le comportement surveillé aux modèles d'attaque prédéfinis, tandis que les systèmes basés sur les anomalies se concentrent sur la connaissance du comportement normal afin d'identifier toute déviation et toutes activités suspectes. Par conséquent, les techniques d'apprentissage en profondeur peuvent être utilisées pour les deux types de détection, pouvant extraire des relations non linéaires de niveaux plus élevés entre les données, pour identifier les écarts par rapport à une activité bénigne. Cependant, ces méthodes nécessitent de grandes quantités de données dans le but de détecter et identifier les motifs de différentes classes.
Dans ce chapitre, Nous présenterons quelques approches de DL qui ont été appliqué aux domaines de la détection des intrusions, dont nous avons étudié et analysé ses divers mécanismes et travaux basés sur l'apprentissage profond. Le résumé contient les jeux de données utilisés, l'architecture des réseaux et les mesures d'évaluation avec les résultats obtenus.

3.2 Travaux anterieurs

Selon les travaux antérieurs, les méthodes d'apprentissage profond surpassent les autres algorithmes d'apprentissage machine, tels que la machine à vecteur de support (SVM) et les réseaux artificiels neuronales traditionnels (ANN) dans la détection des anomalies [61].

La détection d'intrusion basé sur l'apprentissage profond a été commencé en 2011, lorsque Salama et al. 2011 [85] ont présenté une approche hybride combinant le réseau de croyances profondes (DBN) et SVM afin de classifier les intrusions du réseau en deux catégories : normale ou attaque. Le réseau DBN est composé de multiples couches de Machines Boltzman Restreinte (RBM) est utilisé comme une méthode de réduction de dimension pour obtenir de meilleures caractéristiques d'apprentissage suivies par un classificateur SVM. Les performances de l'approche

DBN-SVM proposées sont testées sur NSL-KDD data-set. Le nombre de caractéristiques a été réduit de 41 à 5 grâce au réseau DBN qui a été ensuite transmises à un classificateur SVM pour effectuer une classification binaire (normale/attaque). La méthode proposée atteint une exactitude supérieure à 90%. De plus, les résultats ont montré que le réseau DBN en tant qu'une méthode de réduction des caractéristiques est plus performant par rapport à d'autres méthodes d'analyse de données.

Kang et al. [61] ont proposé un système de détection d'intrusion qui se base sur le réseau neuronal profond (DNN) dans le but de sécuriser les réseaux de véhicules. Le scénario d'attaque a été réalisé sur des paquets de données malveillants. Ces derniers sont injectés dans un bus de réseau de contrôleurs de véhicules. Le système proposé introduit le vecteur de caractéristiques dans les nœuds d'entrée afin de classer les paquets en deux classes (un paquet normal et un paquet d'attaque). Les auteurs utilisent le réseau de croyances profondes (DBN) pour former efficacement les paramètres du poids caché (Hidden Weights) initialisant le réseau DNN. Ensuite, les couches cachées suivantes sont liées à ces sorties qui sont calculées sur la base de la fonction d'activation (ReLU). Le système proposé atteint un taux de faux positive (TPR) inférieure à 1 ou 2%, et un taux de détection (DR) de 99%.

Alom et al. [35]ont utilisé le réseau de croyances profondes (Deep beleif network) pour la détection des intrusions. Le système proposé est capable de détecter les attaques, et aussi identifier et classer la précision des activités du réseau en cinq groupes sur la base de sources de données limitées, incomplètes et non linéaires. Par rapport au système existant, la précision de détection du système a atteint 97,5%, et n'a nécessité que 50 itérations.

Wu et al. [98] ont proposé un modèle IDS en utilisant des réseaux neuronaux convolutifs (CNN). Le CNN est utilisé pour sélectionner automatiquement les caractéristiques du trafic à partir d'un ensemble de données brutes. Les auteurs ont utilisé l'ensemble de données NSL-KDD pour l'évaluation, et aussi pour résoudre le problème de déséquilibre des ensembles de données (the imbalanced Dataset problem), ils fixent le coefficient de pondération de la fonction de coût de chaque classe en fonction de ses nombres. Pour réduire le coût du calcul, ils convertissent le format vectoriel de trafic brut en format d'image. Le modèle proposé améliore la précision de la classe avec de petits nombres et réduit également le taux de fausses alarmes (FAR).

Un système de détection d'intrusion de réseau (NIDS) basé sur le deep learning a été proposé par Javaid et al.[58], Les auteurs ont utilisé l'apprentissage en autodidacte (Selftaught Learning STL). Cette méthode se représente en deux étapes pour la classification, la première étape appelée l'apprentissage de caractéristiques (non-supervisées). S'agit d'une bonne représentation des caractéristiques obtenue à partir d'une massive collection de données non étiquetées. Dans la deuxième étape, cette représentation apprise sera appliquée aux données étiquetées et utilisée pour la tâche de classification. L'approche STL a été testée sur le benchmark dataset NSL-KDD pour la détection des anomalies de réseau. Il s'agit d'une classification multi-classes des records du trafic normal et autres types d'attaques réseaux. Les meilleurs résultats atteint une précision de 85.44% et un rappel (recall) de 95.95% pour une classification binaire (trafic normal / trafic malveillant) lorsqu'ils ont

appliqué des données des testes malgré que ces résultats a un taux de fausse positive considérable, cette approche du SLT donne de bons résultats comparent une simple soft-max régression (machine learning algorithme pour multi- classification) et d'autres travaux précédents.

Kim et al. [62] ont proposés IDSs basé sur RNN, les paquets TCP/IP du trafic réseau ont été représenté sous forme de séquence. Les auteurs ont utilisé la méthode Hessian-Free Optimisation pour surmonter le problème explosion/vanishing gradiant du RNN, cette technique aide l'algorithme de gradient de trouver le minimum globale pendant l'entraînement et donc améliorer la précision. L'ensemble de données DARPA a été utilisé pour l'évaluation de modèle proposé composé de 41 caractéristiques et 22 différentes attaques. Le taux de détection était de 95,37% et le taux de fausses alertes de 2,1%. La précision de la classification de chaque classe d'attaque était bonne, et les auteurs concluent que l'utilisation de RNN avec optimisation sans hélium pour la détection des intrusions est une assez bonne approche.

Torres et al. [95] fournissent une analyse de la viabilité des réseaux neuronaux récurrents (RNN) pour détecter le comportement du trafic réseau en le modélisant comme une séquence d'états variant dans le temps. La méthode de détection LSTM est proposée et évaluée sur deux ensembles de données différents CTU13-42 et CTU13-46 du trafic réseau, chacune contient deux classes, les connexions botnet et les connexions normales. Un botnet peut être considéré comme un groupe d'ordinateurs compromis qui peuvent être contrôlés à distance pour mener des attaques coordonnées ou commettre des actes frauduleux. Les auteurs on analysé également le nombre d'états par connexion dans le modèle comportemental, et des expériences démontrent que le RNN est capable de classer le trafic avec un taux de détection d'attaque (ADR) de 0,970% et un taux de fausses alarmes (FAR) très faible 0,018%. L'impact des techniques d'échantillonnage (suréchantillonnage/sous-échantillonnage) utilisées pour gérer les classes déséquilibrées sur les performances du LSTM est également analysé. Les résultats montrent aucune différence significative n'a été observée entre les deux techniques d'échantillonnage, et les valeurs de l'ADR (Attack detection Rate) ne présentent pas une déférence confédérale sur les trois cas évalué. Cependant, les expérimentations ont montré que les modèles de détection RNN ont des problèmes pour traiter les comportements du trafic qui ne sont pas facilement différenciés ainsi que certains cas particuliers de trafic de réseau déséquilibré.

Vinayakumar et al. [1] on analysé l'efficacité du réseau de neurones à convolution pour la détection des intrusions en modélisant les événements de trafic réseau sous forme des séries chronologiques de paquets TCP/IP (transmission control protocol / internet protocol) sur des périodes prédéfinies. Les auteurs ont suivi la méthode CNN appliquée au traitement du langage naturel avec la couche Convolution 1D. Sur cette base, les auteurs modélisent les événements de trafic réseau comme une série temporelle de données, où au lieu de prendre des données d'image 2D en entrée, le CNN traite la série de données comme 1D, dans laquelle les données sont disposées dans un intervalle de temps. Les auteurs ont proposé différentes architectures du CNN contient une couche d'entrée, une couche cachée et une couche de sortie, ou la couche cachée contient une ou plusieurs couches CNN suivies de FFN ou RNN/LSTM/GRU afin de déterminer l'architecture optimal. Les performances

de chaque modèle sont évaluées sur l'ensemble de données de KDDCup 99. Les résultats montrent que CNN-LSTM a bien fonctionné par rapport aux autres structures du réseau CNN et a atteint une précision de 99%.

D'après Gurung et al. [53] Un DNN pour NIDS composé d'un encodeur automatique épars a été utilisé pour l'apprentissage non supervisé des caractéristiques. Un classificateur logistique est ensuite utilisé pour la classification sur le jeu de données NSL-KDD. Le système prend 115 fonctions en entrée, le sparse auto-codeur a été utilisé pour former et apprendre de nouvelles fonctions, donc ces caractéristiques ont été réduites à 50, puis 10, puis affectées à la logistique de classification des classificateurs de régression. La performance du système a été mesurée en termes d'accuracy, de précision et de rappel. La précision globale du modèle était de 87,2%.

Tang et al. [57] ont appliqué le réseau neuronal profond (DNN) pour implémenter un IDS dans un contrôleur SDN (Software Defined Network) qui peut surveiller tous les flux des commutateurs OpenFlow. Ils ont formé le modele avec avec l'ensemble de données NSL-KDD pour une classification binaire (normale / anomalie), où l'ensemble de données se composait de quatre catégories à savoir : attaques DoS, attaques R2L, attaques U2R et attaques probe. Puis ils ont utilisé que 6 fonctionnalités principales qui ont ensuite été extraites de 41 fonctionnalités Toutes les mesures d'évaluation ont tendance à augmenter à mesure qu'elles diminuent le taux d'apprentissage de 0,1 à 0,001, et elles finissent par donner de meilleurs résultats lorsqu'elles utilisent 0,001 comme taux d'apprentissage. Le modèle obtient des performances avec une précision de 75,75%.

Rezvy et al. [83] ont choisi l'approche AutoEncoder qui fonctionne comme un constructeur de caractéristiques, le modèle a été d'abord préformé par l'AutoEncoder pour ré-configurer les caractéristiques , puis un classificateur DNN a été utilisé pour classer le trafic L'auteur a évalué l'algorithme avec l'ensemble de données de référence NSL-KDD'99 à cinq classes. Encodeur automatique a reconfiguré les 122 fonctionnalités, qui ont ensuite été intégrées dans un DNN à trois couches. La précision des cinq classes varie de 89,2% à 99,9%.

Gao et al. [49] ont adopté un modèle de détection d'intrusion basé sur Deep Belief Networks est proposé pour s'appliquer dans le domaine de la reconnaissance d'intrusion. L'architecture de ce modèle hiérarchique profond est un classificateur de réseau neuronal profond composé d'une combinaison de réseaux d'apprentissage non supervisés multicouches, appelés machine Boltzmann restreinte, et d'un réseau d'apprentissage supervisé, appelé réseau de rétro-propagation. Les résultats expérimentaux sur l'ensemble de données KDD CUP 99 démontrent que les performances du modèle DBN sont meilleures que celles du SVM et de l'ANN.

Le tableau suivant présente une sélection d'études portant sur la détection des intrusions basés sur des modèles DL, les caractéristiques utilisés, le secteur d'intérêt, les ensembles de données et les mesures de performance.

Auteurs	Années	Méthodes	Datasets	Mesures de performances
Salama et al, [85]	2011	DBN-SVM	NSL-KDD	ACC = 90%
Kang et al, [61]	2016	DNN-DBN	Vehicular Network communication	DR = 99% - 99.46% FPR = 1-2%
Alom et al, [35]	2015	DBN	NSL-KDD	ACC = 97.5%
Wu et al, [98]	2018	CNN	NSL-KDD	ACC = 97.88% - 99.46% DR = 68.66% FPR = 27.9%
Javaid et al, [58]	2016	STL	KDD'99	ACC = 79.10% - 88.39% PR = 85.44% RC = 95.95% F1 = 90.39%
Kim et al, [62]	2015	RNN	DARPA	DR = 95.37% FPR = 2.1%
Torres et al, [95]	2016	RNN-LSTM	CTU-13	DR = 97.96% FPR = 2.27%
Vinayakumar et al, [1]	2017	T CNN-RNN	KDD'99	ACC = 99.99% PR = 99.99% RC = 99.99% F1 = 99.99%
Gurung et al, [53]	2018	Deep auto-encoder	NSL-KDD	ACC = 99.99% PR = 84.6% RC = 92.8% specificity =80.7%
Tang et al, [**?**]	2016	DNN	NSL-KDD	ACC = 75.75% PR = 83% RC = 76% F1 = 75% ROC_AUC = 86%
Rezvy et al, [83]	2018	Deep auto-encoder	NSL-KDD	ACC = 99.3%
Gao et al, [49]	2014	DBN	KDD'99	ACC = 74.22% - 93.49% DR = 75.6% - 92.33% FAR = 3.15% - 0.76%

TABLEAU 3.1 – Travaux antérieurs connexes pour la détection d'intrusion basé sur le deep learning

3.3 Conclusion

Diverses approches et architectures d'apprentissage en profondeur ont été appliquées pour détection d'intrusion. Ces algorithmes de deep learning proposés ont des performances varient en fonction des jeux de données sélectionnés et des caractéristiques entrées. Cependant, en utilisant les mêmes méthodes et techniques d'apprentissage ne garantit pas toujours le même résultat, pour différentes classes les attaques peuvent varier.

Chapitre 4

Modélisation et Implémentation

4.1 Introduction

Beaucoup de projets et recherches ont été réalisés afin d'obtenir des meilleurs systèmes de détection d'intrusion au terme d'efficacité contre tout type d'attaque. Dans le chapitre précèdent, nous avons cité les différentes méthodes appliquées avec succès pour différents jeux de données dédiés aux attaques réseaux. La qualité, la performance et la précision du système de détection dépendent fortement de jeux de données utilisés. Dans notre travail nous avons testé et évalué nos modèles sur le dataset NSL-KDD fournit par kaggle, permettant aux ids d'analyser le trafic réseau afin d'identifier les paquets suspects, le NSL-KDD est mis à jour quotidiennement par ses fournisseur, ce qui fait son point fort, le nôtre a été introduit à la plateforme récemment (le 13 aout 2022), la version de NSL-KDD utilisée dans ce projet est considérée la plus difficile à un ids de classifier et ses résultats avec les techniques de deep learning sont faibles par rapport au autres version NLS-KDD ou même par rapport aux autres jeux de données vu aux nombre et types d'attaques qu'il contient. Après beaucoup de recherches nous avons proposés des modèles et nous les avons testés sur 8 méthodes de deep learning (DNN, CNN, RNN-LSTM, RNN-GRU, CNN-LSTM, CNN-GRU, CNN-BiLSTM et CNN-BiGRU). Nous avons introduit dans ce chapitre les modèles dont lesquels nous avons atteint nos meilleurs résultat pour chacune des méthodes cités. Les performances des approches de détection proposées ont été évalué en prenant en compte les différentes mesures d'évaluation des algorithmes de l'apprentissage profond savoir, la précision, le rappelle, le score F1, le taux de détection et le taux de fausse alarmes. Toutes les méthodes ont été implémentées et évaluer avec une multi classification de 5 classes (normal, DoS, Probe, R2L, U2R).

4.2 Environnement de l'exécution

Afin d'implémenter et tester nos modèles, On a commencé par la configuration d'un environnement local de développement Python (Jupyter) utilisant la plate-forme Anaconda. Mais vu que le deep learning est un domaine avec des exigences de la disponibilité des ressources matériel (surtout en GPU) capable de faire des calculs intenses, on avait besoin d'aller vers le Cloud, ce dernier fournit des ressources de calculs et de mémoires importantes qui dépassent notre ordinateurs. Selon le besoin le Cloud peut fourni l'accès à un processeur graphique GPU totalement gratuit. Parmi les outils

Cloud les plus utilisés dans le domaine de la machine learning c'est Google Colab.

Google Colaboratory : est un service Cloud basé sur **Jupyter Notebooks**, permet de développer des applications en Deep Learning en Python, il offre un processeur GPU gratuit, 12 Go de RAM et plus de 100 Go de stockage. Pour l'accès dans ce service il nous suffit simplement d'avoir qu'un compte Google[27].

Python : est le langage de programmation qu'on a choisi, un langage de développement interprété, multi-paradigme et multi-plateformes, il est aussi un langage plus commun et plus populaire pour l'apprentissage automatique et l'intelligence artificielle grâce à sa flexibilité et aussi parce qu'il a un nombre important de bibliothèques logicielles open source disponible. Permet ses bibliothèques utilisées dans notre projet : Pandas, Numpy, Scikit-Learn .etc[28].
Pandas et **Numpy** sont utilisés pour la manipulation des données (le chargement, la réorganisation et le traitement des données), Scikit-Learn nous permet d'expérimenter différentes techniques et algorithmes d'apprentissage automatique et d'analyse de données prédéfinis rapidement et facile a utilisé[29, 30].
Les frameworks TensorFlow et Keras ont été choisit pour l'implémentions des méthodes deep learning proposés.
TensorFlow est une bibliothèque de deep Learning open source développée par Google utilisée pour effectuer des opérations numériques complexes et plusieurs autres tâches pour modéliser les architectures de Deep Learning. Il peut déployer facilement des calculs sur plusieurs plates-formes comme les CPU, les GPU[31].
Keras est une API de haut niveau qui vise à créer et à entraîner des modèles de Deep Learning basé sur python. Elle a été développée dans le but de permettre des expérimentations rapides[32]. Parmi ces avantages :

- Était capable d'aller de l'idée au résultat avec le plus faible délai possible. Et ça c'est la clé d'une recherche efficace.
- Supporte à la fois les réseaux convolutifs (CNN) et les réseaux récurrents (RNN), ainsi que la combinaison des deux items, pas de fichiers séparés de configuration des modèles, tout est déclaré dans le code.
- fonctionne sur CPU et GPU.

Afin de bénéficier les avantages des deux nous avons utilisé TensorFlow comme back-end avec keras.

4.3 Dataset

4.3.1 Le choix de Dataset

Le moyen le plus efficace d'évaluer les performances d'un IDS consiste à le tester à l'aide d'un jeu de données public standardise [96], l'objectif est de faire une comparaison de différents systèmes. Il existe peu de jeux de données publics utilise dans la détection d'intrusions, les ensembles

de données couramment utilisés incluent KDD Cup99 [82], NSL-KDD [39], l'ensemble de données UNSW-NB15 [77] et le dataset CIC-IDS [94].
Nous avons utilisé dans nos expérimentations le NSL-KDD, qui est une version améliorée de KDD99 qui résout le problème de redondance des données et est l'un des jeux de données de référence les plus utilisés pour évaluer les performances de l'IDS. Et comme nous voulions rendre la tâche plus difficile possible aux nos IDS afin de mieux les évalués avec des cas le plus réels possibles, nous avons choisi la version de NSL-KDD qui marque le moins de récurrence et précision avec les méthodes de deep learning comme montré dans la Figure4.1. (de 70% 83% pour les meilleurs modèles), avec 125 973 échantillons de trafic déséquilibrés comme données d'entrainement et 22 544 de test, ce qui est encore plus complexe à détecter et prouvera l'efficacité de nos modèles.

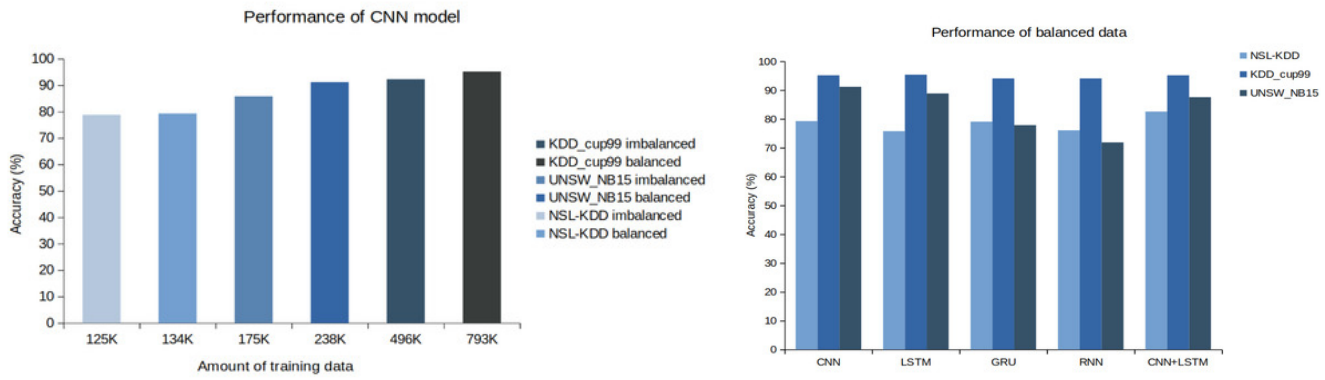


FIGURE 4.1 – Performance et difficulté de différents datasets par rapport aux méthodes de DL, nombre et déséquilibrement des données [72].

4.3.1.1 Description de la base NSL-KDD

La base NSL-KDD (Network Security Layer-Knowledge Discovery in Databases) a été fondé sur l'ensemble de données KDD99, Cette dernière est une base de données qui contient des connexions TCP /IP extraites de l'ensemble de données d'évaluation des systèmes de détection d'intrusions. KDD99 été réalisées en 1998 par l'agence de L'armé américain DARPA (Défense Advanced Research Projects Agency) et AFRL (Laboratoire de recherche de l'armée de l'air), ensuite MIT Lincoln Labs8 a collecté et distribué les ensembles de données pour l'évaluation du système de détection d'intrusions de réseau informatique [33].

La base NSL-KDD est un ensemble de données qui représente une version réduite de l'originale KDD 99, proposé en 2010 par les chercheurs dans le domaine de détection d'intrusions réseaux afin de résoudre certains problèmes qui ont apparu dans la base KDD 99. NSL-KDD considérée comme un ensemble de données de référence pour aider les chercheurs à comparer les différentes méthodes de détection d'intrusions. Le NSL-KDD présente les différences suivantes par rapport à l'originale KDD99 :

- Il n'inclut pas les enregistrements redondants dans les données d'apprentissage, ce qui améliore la performance de classification.
- Il n'y a pas d'enregistrements en double dans les ensembles de test proposés, ce qui aide à l'obtention de meilleurs taux de détection.
- Le nombre d'enregistrements dans les données d'apprentissage et les ensembles d'essais sont raisonnables, ce qui il est abordable d'exécuter les expériences sur l'ensemble complet sans la nécessité de choisissez au hasard une petite portion. Par conséquent, les résultats d'évaluation des travaux de recherche seront cohérents et comparables.

4.3.2 Distribution de NSL-KDD

Le dataset que nous avons utilisé se compose d'un ensemble d'apprentissage (KDDTrain+), contenant 125 973 échantillons de trafic, et d'un ensemble de test (KDDTest), contenant 22 544 échantillons de trafic. Afin de restaurer davantage la situation complexe du réseau dans la réalité, il n'y a que 23 types d'attaques dans l'ensemble d'apprentissage, et les 15 autres types d'attaques n'existent que dans l'ensemble de test comme le montre la Figure 4.2.

```
len(train['class'].cat.categories), len(test['class'].cat.categorie
s)
```

```
(23, 38)
```

FIGURE 4.2 – Types d'attaques dans Train et Test data.

4.3.2.1 Distribution des attaques

Nous avons établis des modèles de détection d'intrusions on se basant sur différentes techniques de deep learning, et lorsque notre classification est multi-classe où les classes sont 0, 1, 2, 3, 4. La couche de sortie comporte 5 neurones, et prennent 5 valeurs possibles : 0 :Dos, 1 :normal, 2 :Probe, 3 :R2L, 4 :U2L.
Le Tableau4.1 montre la distribution des connexions réseau dans les deux ensembles KDDTrain et KDDTest respectivement.

Classes	KDDTrain+		KDDTest20+	
Normal	67 343	53.45%	9 889	43.86%
Dos	45 927	36.45%	7 460	43.86%
Probe	11 656	9.25%	2421	10.74%
R2L	995	0.78%	2707	12.01%
U2L	52	0.04%	67	0.29%
Total des enregistrements	125 973		22 544	

TABLEAU 4.1 – Distribution des connexions réseau de NSL KDDTrain+ et KDDTest20+.

Selon les caractéristiques des attaques de jeux de données, il est divisé en quatre types d'attaques : DoS (Denial of Service attaques), R2L (attaques racine vers local), U2R (attaques utilisateur vers racine), et Probe (attaques de sondage). Il existe certains types spécifiques d'attaques qui ne sont pas dans l'ensemble de formation, ce qui fournit un plus base théorique réaliste pour la détection d'intrusion.

4.3.2.2 Attribution détaillée de Dataset utilisé

Le TABLEAU 4.2 montre l'attribution des enregistrements de trafic dans le jeu de données que nous avons utilisé pour nos expérimentations. NSL-KDD contient 41 caractéristiques attributs et 1 étiquette de classe (class) et un attribut (difficulty_level) que nous avons enlevé car on l'aura pas besoin dans ce projet. Le dataset contient trois grands types de fonctionnalités, les attributs 1 à 10 sont les caractéristiques de base du réseau connexion, les attributs 11 à 22 sont les caractéristiques de contenu de la connexion réseau et les attributs 23 à 41 sont le trafic les caractéristiques.

N°	Nom de l'attribut	Type	Description
1	Duration	Numérique	La durée de connexion
2	Protocol_type	Nominal	Protocole utilisé dans la connexion (tcp, udp, icmp)
3	Service	Nominal	Service réseau de destination,(http,telnet, ftp_data, etc.)
4	Flag	Nominal	Statut de la connexion –Normal ou Erreur (SF, REJ, S0, S1, etc.)
5	Src_bytes	Numérique	Nombre d'octets de donnée transférés de la source à la destination (491, etc.)
6	Dst_bytes	Numérique	Nombre d'octets de données transférés de destination à la source (0, etc.)
7	Land	Numérique	Si l'adresse IP de source et destination et le nombre de port sont les mêmes alors, land=1 sinon land=0
8	Wrong_fragment	Numérique	Nombre total de fragments erronés dans cette connexion
9	Urgent	Numérique	Nombre de paquets urgents
10	Hot	Numérique	Nombre d'indicateurs « Hot »
11	Num_failed_logins	Numérique	Nombre de tentatives de connexion échouées
12	Logged_in	Numérique	Si connecté avec succès alors logged_in=1 sinon logged_in=0
13	Num_compromised	Numérique	Nombre de conditions compromises
14	Root_shell	Numérique	1 si le root shell est obtenu, 0autrement
15	Su_attempted	Numérique	1 si la commande "su root" a été tentée ou utilisée, sinon 0
16	Num_root	Numérique	Nombre d'accès " root " ou nombre d'opérations effectuées comme racine dans la connexion
17	Num_file_creations	Numérique	Nombre d'opérations de création de fichiers
18	Num_shells	Numérique	Nombre d'invites du shell
19	Num_access_files	Numérique	Nombre d'opérations sur les fichiers de contrôle d'accès
20	Num_outbound_cmds	Numérique	Nombre de commandes sortantes dans une session FTP
21	Is_host_login	Numérique	1 si la connexion appartient à la liste du « hot » (root ou admin) ;sinon 0
22	Is_guest_login	Numérique	1 si le login est un login «guest » ; sinon 0
23	Count	Numérique	Nombre de connexions vers le même hôte de destination que la connexion en cours dans les deux dernières secondes
24	Srv_count	Numérique	Nombre de connexions vers le même service (N° Port) que la connexion en cours dans les deux dernières secondes
25	serror_rate	Binaire	Le pourcentage de connexions qui ont activé le flag s0, s1, s2 ou s3, parmi les connexions agrégées dans count
26	Srv_serror_rate	Binaire	Le pourcentage de connexions qui ont activé le flag s0, s1, s2 ou s3, parmi les connexions agrégées dans srv_count
27	Rerror_rate	Binaire	Le pourcentage de connexions qui ont activé le flag REJ, parmi les connexions agrégées dans count
28	Srv_rerror_rate	Binaire	Le pourcentage de connexions qui ont activé le flag REJ, parmi les connexions agrégées dans srv_count
29	Same_srv_rate	Binaire	Le pourcentage de connexions qui sont au même service, parmi les connexions agrégées dans count
30	Diff_srv_rate	Binaire	Le pourcentage de connexions qui sont aux différents services, parmi les connexions agrégées dans count
31	Srv_diff_host_rate	Binaire	Le pourcentage de connexions qui sont à différentes machines de destination, parmi les connexions agrégées dans srv_count
32	Dst_host_count	Numérique	Nombre de connexions ayant la même adresse IP de l'hôte de destination
33	Dst_host_srv_count	Numérique	Nombre de connexions ayant le même numéro de port
34	Dst_host_same_srv_rate	Binaire	Le pourcentage de connexions qui sont au même service, parmi les connexions agrégées dans dst_host_count
35	Dst_host_diff_srv_rate	Binaire	Le pourcentage de connexions qui sont aux différents services, parmi les connexions agrégées dans dst_host_count
36	Dst_host_same_src_port_rate	Binaire	Le pourcentage de connexions qui sont au même port de source, parmi les connexions agrégées dansdst_host_srv_count
37	Dst_host_srv_diff_host_rate	Binaire	Le pourcentage de connexions qui sont à différentes machines de destination, parmi les connexions agrégées dans dst_host_srv_count
38	Dst_host_serror_rate	Binaire	Le pourcentage de connexions qui ont activé le flag s0, s1, s2 ou s3, parmi les connexions agrégées dans dst_host_count
39	Dst_host_srv_serror_rate	Binaire	Le pourcentage de connexions quiont activé le flag s0, s1, s2 ou s3, parmi les connexions agrégées dans dst_host_srv_count
40	Dst_host_rerror_rate	Binaire	Le pourcentage de connexions qui ont activé le flag REJ, parmi les connexions agrégées dans dst_host_count
41	Dst_host_srv_rerror_rate	Binaire	Le pourcentage de connexions qui ont activé le flag REJ, parmi les connexions agrégées dans dst_host_srv_count
42	Class	Nominal	Label de classification
43	Difficulty_level	Numérique	Niveau de difficulté

TABLEAU 4.2 – Les détails des attributs de NSL-KDD utilisé [45].

4.3.3 Prétraitement de Dataset

NSL-KDD contient trois types de données : numérique, Nominale et binaire. Les attributs 2, 3 et 4 sont nominales, 7, 12, 14, 15, 21 et 22 sont binaires, et le reste des attributs sont de type numérique comme le montre le Tableau 4.2, avant de passer au travail expérimental, l'ensemble de données NSL-KDD est d'abord passé par une opération de prétraitement des données et la conversion du type des attributs en suivant les étapes décrites dans ce qui suit.

4.3.3.1 One-Hot Encoding (Numérisation)

Puisqu'il existe trois types non numériques de valeurs de caractéristiques et que le modèle ne peut accepter que des types numériques, One-Hot Encoding a été adopté pour convertir les trois caractéristiques non numériques en caractéristiques numériques. Par exemple, les valeurs de protocol_type sont TCP, UDP et ICMP, et après le codage, les valeurs deviennent respectivement [1,0,0], [0,1,0] et [0,0,1] comme le montre Tableau 4.3. Enfin, le jeu de données contient 122 données dimensionnelles après encodage.

Avant la numérisation	Après la numérisation
ICMP	0
TCP	1
UDP	2

TABLEAU 4.3 – Numérisation de l'attribut 'protocol$_type'$[93].

4.3.3.2 Normalisation

Un grand écart entre les différentes données d'entités dimensionnelles au sein de l'ensemble de données peut entraîner des problèmes tels qu'un apprentissage lent du modèle et une amélioration insignifiante de la précision ; par conséquent, afin de résoudre ce problème, le MinMaxScaler a été adopté pour mapper les données dans la plage de (0,1) comme suit :
X' = (XXmin) / (XmaxXmin)
Tel que Xmax est la valeur maximale et Xmin est la valeur minimale.

4.3.3.3 Sélection des attributs

Dans le cadre de notre projet, nous nous sommes contentés l'élimination de dernier attribut 'difficulty_level', et la sélection des attributs constitue une tache non obligatoire mais très importante pour extraire des sous-ensembles des attributs en préservant les plus significatives et pertinents et en excluant les attributs considérés comme source de bruits dont le but est de faire une classification

des enregistrements de le jeu de données NSL-KDD et de réduire le coût et le temps nécessaire pour l'opération d'apprentissage, car le travail sur tous les attributs pour générer le modèle de classification sera très fastidieux et peut affecter considérablement les performances de l'algorithme d'apprentissage en matières de temps d'exécution et consommation des ressources systèmes, en plus les attributs de la base de données ne seront pas tous utilisés dans la classification des connexion TCP/IP effectuée par l'IDS pour détecter les attaques, certains étant plus pertinents que d'autres.

4.4 Conception et Mesures d'évaluation

Après beaucoup d'expérimentations et tests sur de multiple algorithmes et techniques de classification qui se basent sur l'apprentissage en profondeur avec NSL-KDD. Nous avons pu implémenter 6 modèles discriminatoires basés sur 6 méthodes de DL multi classification de 5 classes, les modèles que nous allions présentés dans ce mémoire de fin de cycle master, sont lesquelles dont nous avons atteint un taux de prédiction plus élevé, et nous jugeant plus qualitatifs.
Nous avons commencé par l'implémentation d'un modèle simple avec un minimum de couche et paramètres pour chacune des techniques DNN (Deep Neural Network), CNN (Convolutional Neural Network) et deux variantes de RNN (Recurrent Neural Network) qui résout le problème de disparition du gradient (Vanishing Gradient Problem)[55] :

- LSTM (Long Short-Term Memory) : sont un type de réseau neuronal récurrent capable d'apprendre la dépendance d'ordre dans les problèmes de prédiction de séquence. Il s'agit d'un comportement requis dans des domaines problématiques complexes tels que la traduction automatique, la reconnaissance vocale, etc. Les LSTM sont un domain complexe d'apprentissage en profondeur [41].
- GRU (Gated Recurrent Unit) [5] : sont aussi une version améliorée du réseau neuronal récurrent standard. Le GRU peut également être considéré comme une variante moins complexe du LSTM car les deux sont conçus de manière similaire avec moins de paramètres pour GRU, et dans certains cas, produisent des résultats tout aussi excellents.

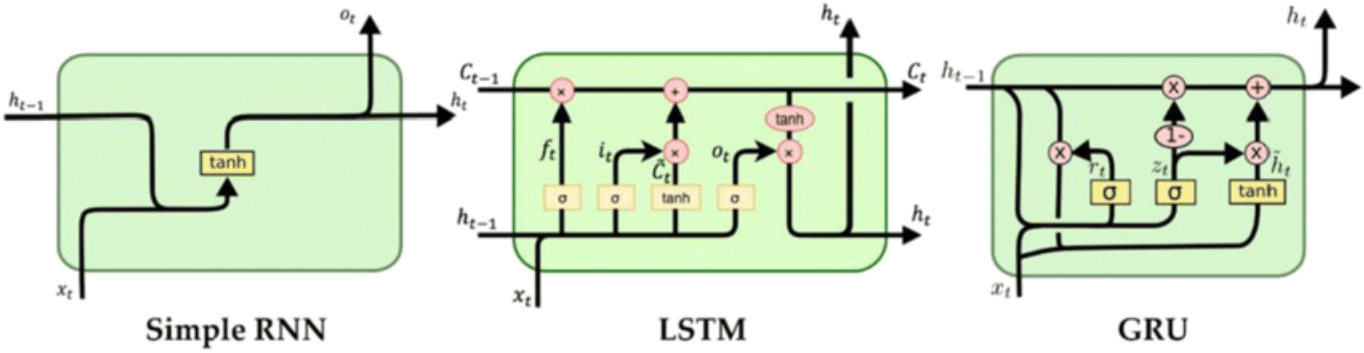


FIGURE 4.3 – Les structures internes des différentes cellules du réseau neuronal récurrent (RNN).

Ensuite nous avons combiné les méthodes CNN et les variantes de RNN, afin d'avoir deux modèles hybrides méthodes CNN-LSTM et CNN-GRU.

Nous avons également essayés d'implémenter deux autres variantes encore plus avancées de RNN (BiLSTM et BiGRU), Mais malheureusement les résultats de nos modèles pour ses deux types avec notre IDS n'étaient pas à notre satisfaction, pour cella nous avons contentés de les utiliser en hybrides méthodes avec le CNN en obtenant CNN-BiLSTM et CNN-BiGRU.

- BiLSTM (Bidirectional Long Short-Term Memory) [88] : est un ajout aux LSTM réguliers qui est utilisé pour améliorer les performances du modèle sur les problèmes de classification de séquences. Les BiLSTM utilisent deux LSTM pour s'entraîner sur l'entrée séquentielle. Le premier LSTM est utilisé tel quel sur la séquence d'entrée. Le deuxième LSTM est utilisé sur une représentation inversée de la séquence d'entrée comme le montre la Figure 4.4. Cela aide à compléter le contexte supplémentaire et rend les modèles rapides.
- BiGRU (Bidirectional Gated Recurrent Unit) [43] : de meme que les BiLSTM, les BiGRU fonctionnent à base de GRU

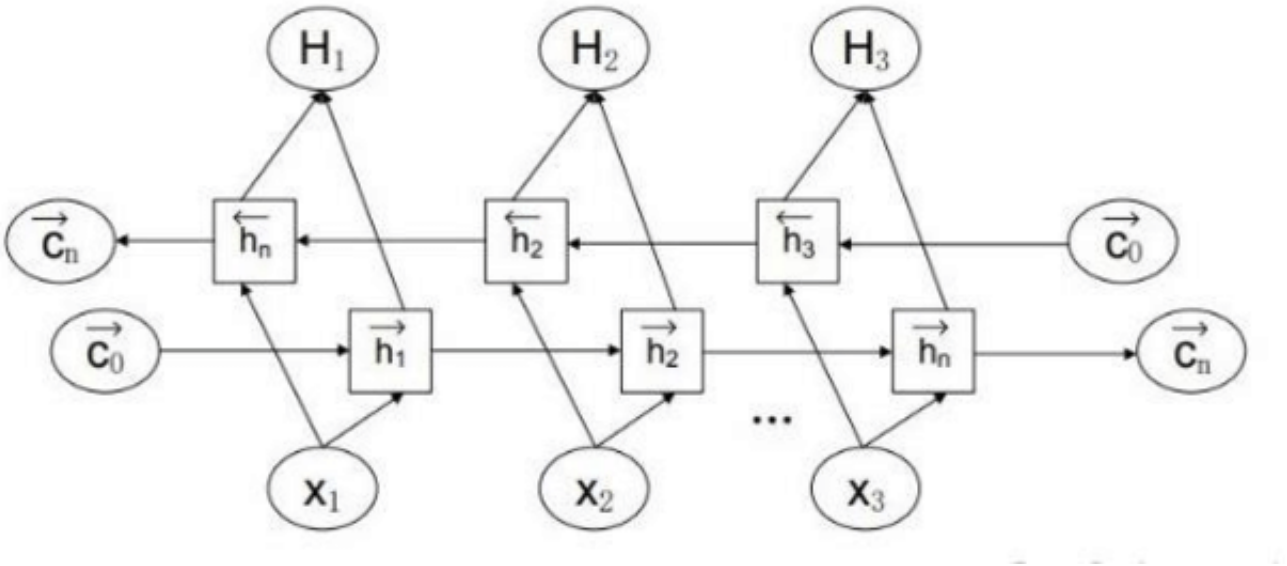


FIGURE 4.4 – La structures de BiLSTM et BiGRU [56]

4.4.1 Processus de génération du modèle de classification

L'élaboration de notre modèle respecte les phases principales suivantes : le prétraitement, l'apprentissage et la phase de test. Comme l'indique le diagramme ci-dessous Figure 4.5.

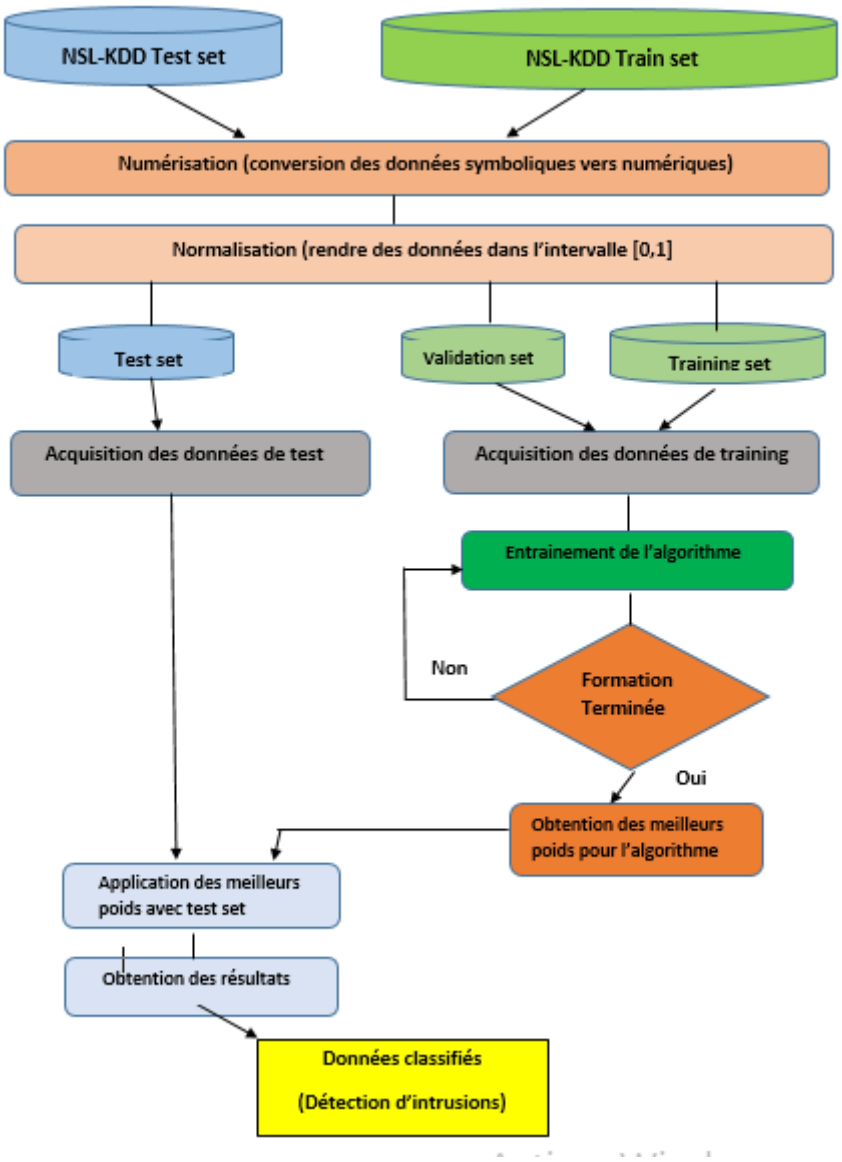


FIGURE 4.5 – Organigramme de fonctionnement de modèle de détection d'intrusion

4.4.2 Les mesures d'évaluation des modèles

On a :

- Vrai positif (TP) : une attaque correctement détectée par le test.

- Faux positif (FP) : une activité normale détectée comme attaque par le test.
- Vrai négatif (TN) : une activité anormale non détectée.
- Faux négatif (FN) : une attaque détectée comme activité normale par le test.

Les mesures :

- Précision (Pr) : le pourcentage des attaques identifiées comme des attaques TP parmi tous les exemples prédit comme attaque, cette métrique, également relative à chaque catégorie,

renseigne sur la probabilité qu'une prédiction d'une catégorie donnée soit correcte. Il est donné par :

Pr = TP/(TP+FP)

- Accuracy (le taux de réussite) : Indique la façon dont la technique de détection est correcte. C'est une métrique qui traduit également le rapport entre les détections correctes et les détections totales obtenues.

  Accuracy = (TP+TN)/(TP+TN+FP+FN)

- Recall (Rc) Le taux de détection (Rappel) ou bien TPR (True Positive Rate) : le pourcentage des attaques identifiées comme des attaques TP parmi tous les attaques dans l'ensemble de données, C'est le rapport entre le nombre d'intrusions correctement détectées et le nombre total d'intrusions. Et décrit par la formule :

  TPR ou bien Rc = TP/(TP+FN)

- False Positive Rate (FPR) : est calculé comme le rapport entre les nombres de trafic normal qui sont incorrectement classés comme intrusions et le nombre total de trafic normal, il est donné par :

  FPR = FP/(FP+TN)

- F1-score (F1) : La moyenne harmonique F combine le rappel et la précision en un nombre compris entre 0 et 1, il est donné par :

  F1 = 2*(Pr*Rc)/(Pr+Rc)

- Matrice de confusion : Est une disposition de tableau permettant de visualiser les performances d'un algorithme ML pour un problème de classification, elle est connue sous le nom de matrice d'erreur.

4.5 Implémentation et Résultat

4.5.1 Architecture des Modèles Proposés

4.5.1.1 Propriétés et Paramètres Communs

Huit modèles sont proposés dans ce projet basés sur : DNN, CNN, deux variantes de RNN et quatre modèles hybrides combinant CNN et RNN, afin d'obtenir les meilleurs résultats, et dans le but de comparer la performance des modèles, plusieurs essais ont été faits. Nous commençons par

définir des modèles basiques et simples pour chaque méthode de DL comme des expérimentations primaires, ensuite nous altérons nos modèles afin d'obtenir un modèle efficace pour la détection d'intrusion. Plusieurs modèles ont été testés et plusieurs combinaisons de propriétés et paramètres ont été ajouté ou éliminé. Néanmoins les modèles partagent certains paramètres et propriétés :

- (None,122,1) est la dimension partagée entre les couches d'entrées qui est le nombre de caractéristique (122 Features) dans le vecteur d'entrée pour chaque modèle, sauf DNN (None,122).
- (None,5) est la dimension que partage les couches de sortie de chaque modèles qui représente le nombre de classe (5 Classes).
- ReLU est la fonction d'activation utilisée pour tous les modèles, vu qu'elle donne de meilleurs résultats par rapport à d'autres fonctions qu'on a essayé tel que tanh.
- Softmax, la fonction d'activation que nous avons choisi pour la multi-classification, Elle donne une probabilité (dont la somme vaut 1) en sortie de chaque neurone, le neurone de sortie avec la probabilité la plus grande permettant alors de décider que sa classe associée est la classe prédite.
- Le dropout a été utilisé aussi, afin d'éviter le problème de sur-apprentissage (Overfitting). Cette technique s'agit de considérer aléatoirement qu'un pourcentage de neurones d'une couche dans le but d'obtenir un modèle généralisable
- categorical-cross-entropy est la fonction de perte (Loss Function) que nous avons utilisés pour tous les modèles vu qu'elle est la plus adaptée à la multi-classification.
- Adam : l'optimiseur qu'on a choisi qui est la version améliorée de SGD (Stochastic Gradient Descent), Adam performe bien avec nos modèles.

La fonction de perte va mesurer l'écart entre les prédictions de modèle et les résultats attendus. Ensuite, l'algorithme d'optimisation "Adam" va dicter comment mettre à jour les poids d'un réseau de neurones pour diminuer la perte.

4.5.1.2 Modèle DNN

Le modèle basé sur DNN qu'on a proposé, comporte 5 couches cachées avec un dropout de 0.01 pour éviter le sur-apprentissage et un nombre d'unité variés d'une couche a une autre, une couche d'entrée et une couche de sortie comme le montre FIGURE 4.6. DNN peut effectuer une extraction automatique des caractéristiques complexes correspondant à partir de données brutes.

Grâce à ces couches cachées qui contiennent un nombre important de paramètres constituant le modèle afin de déterminer les propriétés statistiques sous-jacentes des données intrues ou normales. Avec un modèles encore plus complexe avec des couches supplémentaires, peut mener a un résultat meilleur, mais également on risque d'entrainer un sur-apprentissage. Ce modèle DNN se converge plus rapidement que les autres modèles d'autres techniques de DL.
Nous avons entrainer le modèle sur 50 époques afin d'avoir une meilleure validation aux résultats de KDDTest .

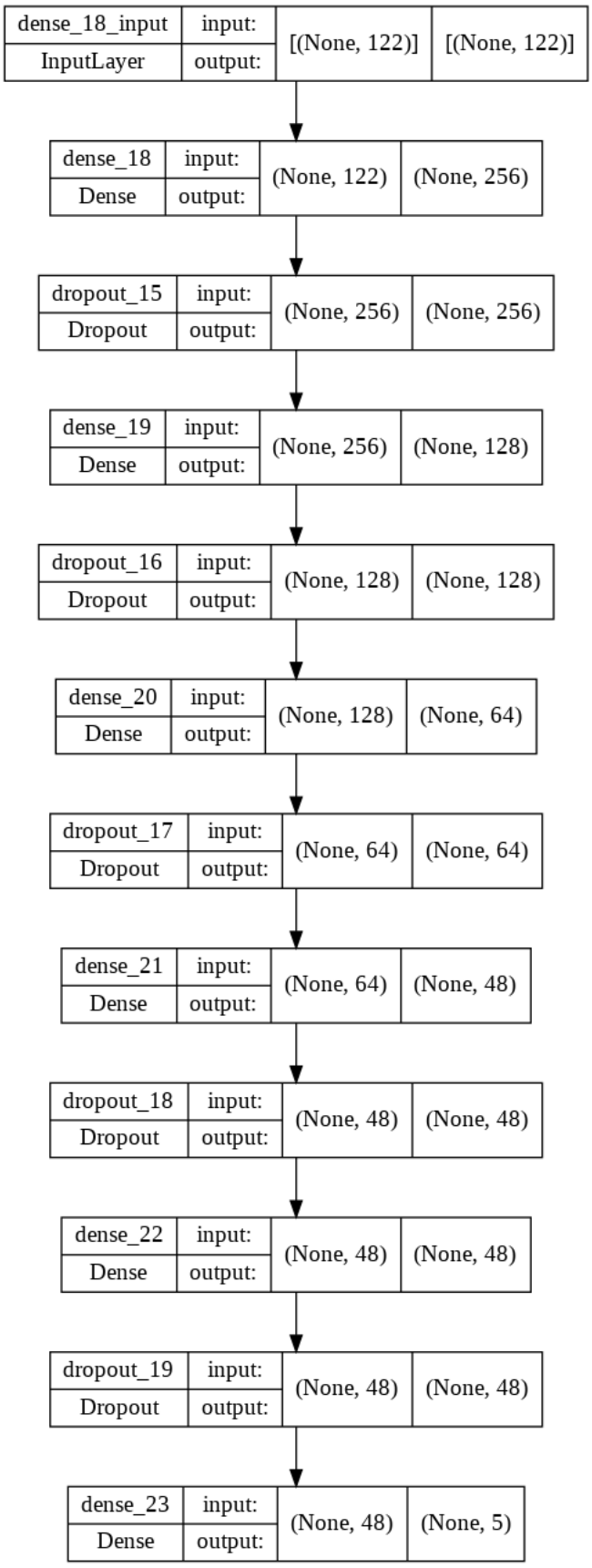


FIGURE 4.6 – Architecture du modèle basé sur DNN

4.5.1.3 Modèle CNN

Avec le modèle CNN, nous avons essayé d'extraire des caractéristiques discriminatoires spatiales en appliquant le CNN 1d car les événements du trafic réseau sont représentés comme des données d'une série chronologique sous forme 1D. Le nombre des couches de convolution nécessaires dépend généralement de la complexité des données. Plus nous avons utilisé des couches convolutifs, plus nous avons obtenu une meilleure précision.
Dans cette étude du CNN, l'architecture est illustrée dans la FIGURE 4.7. Elle est composée de 4 couches de convolution 1D, 2 couches Max-pooling 1D avec une taille de 2, une couche Flatten qui permet d'établir une connexion entre une couche de convolution et les couches de base de DL en réduisant sa dimension, 4 couches cachées, une couche convolutif d'entré et une couche cachée de sortie. Le CNN commence avec une convolution 1D, le modèle nécessite une entrée tridimensionnelle. (batch_shape, features ,steps), l'input-shape dans notre modèle c'est (None, 122, 1), ce modèle va accepter n'importe quel batch_shape avec des séquences de longueur 3 et une vecteur d'entrée de 122 caractéristiques.
Toutes couches de convolution ont des nombres de filtres variés de différentes tailles, ils ont le Padding same pour conserver la même taille des cartes de caractéristiques d'entrées. Étant donné un vecteur d'entrée 1D d'événements chronologiques du trafic réseau, l'opération de convolution 1D utilisant plusieurs filtres nous donne une carte de caractéristiques appelées (Features map), ensuite, la fonction d'activation ReLU sera appliquée à chaque valeur du features maps. Ces caractéristiques seront encore passées par 3 autres couches convolutifs où des nouvelles features maps seront construites. Après, nous appliquons l'opération Max-pooling sur chaque carte de caractéristiques afin d'obtenir des nouvelles caractéristiques plus significatives dans lesquelles la nouvelle caractéristique c'est la valeur maximale des autres valeurs de la carte. Ces nouvelles caractéristiques seront finalement transmises à 4 couches entièrement connectées, 4 couches de Dropout de 0.1 sont utilisée après chaque couche cachée, afin d'éviter le sur-apprentissage. La dernière couche contient la fonction Soft-max qui donne la distribution de probabilité sur chaque classe. Et pour une meilleure validation, nous avons formé notre modèle sur époques.

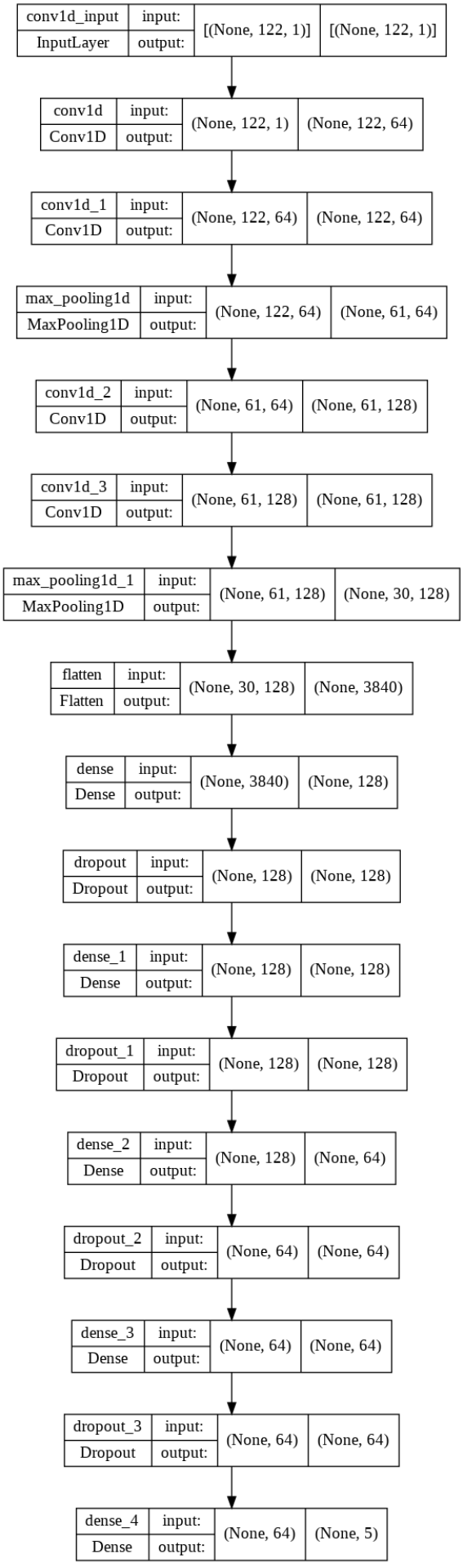


FIGURE 4.7 – Architecture du modèle basé sur CNN

4.5.1.4 Modèle RNN (LSTM et GRU)

Dans le but d'extraire des caractéristiques temporelles discriminatoires à partir des enregistrements déjà passés par une connexion trafic, Afin de les utilisés sur d'enregistrements actuels de la même connexion, Nous avons implémenté des modèle basés sur RNN, Nous avons au début implémenter un modèle basé sur le SimpleRNN qui nous à donner des résultats lors des premiers époques de formation. Mais au plus tard nous rencontrerons des Problèmes de Vanishing Gradient. Le problème qui ne se produira pas avec les deux variantes LSTM et GRU, les quelles nous avons implémenté deux modèles a chacune avec une même architecture.
Les deux modèles partagent 4 couches cachées LSTM/GRU, 2 couches cachées avec une fonction d'activation ReLU, chacune suivie par une couche dropout 10% pour éviter le sur-apprentissage, une couche de sortie pour une multi-classification opérant avec Softmax, et une couche d'entrée. Puisque l'intention est de prédire une type d'attaque pour chaque pas de temps, C'est-à-dire pour chaque type d'attaque de la séquence, l'attaque suivante doit être prédite. Ainsi l'argument return_sequences=True est appliqué sur les 3 premières couches LSTM/GRU, pour obtenir une forme de sortie de (None,122,64) (batch_size,sequence_length, steps), et return_sequences=False sur la dernière couche LSTM/GRU, pour que seule la dernière sortie serait renvoyée, donc pour un lot de None, la sortie serait (None,64), C'est-à-dire pour chaque séquence de 122 types, seul le dernier type prédit serait renvoyé.
La différence entre LSTM et GRU est que, LSTM est une cellule composée de trois « portes » : ce sont des zones de calculs qui régulent le flot d'informations (en réalisant des actions spécifiques) sont : Forget gate (porte d'oubli), Input gate (porte d'entrée) et Output gate (porte de sortie). On a également deux types de sorties (nommées états) sont : Hidden state (état caché) et Cell state (état de la cellule). Tandis que GRU pour sa part dispose de deux portes : Reset gate (porte de reset) et Update gate (porte de mise à jour), Avec un état en sortie : Cell state (état de la cellule). Ce qui fait que les calculs opérés par le GRU sont plus rapides et plus simples. Notons cependant que les capacités/l'efficacité de ce dernier ne sont plus à prouver, et il est autant utilisé que le LSTM.
Nous avons examiné les deux modèles sur 20 sur NSL-KDD dataset, avec 20 époques d'apprentissage et de validation.

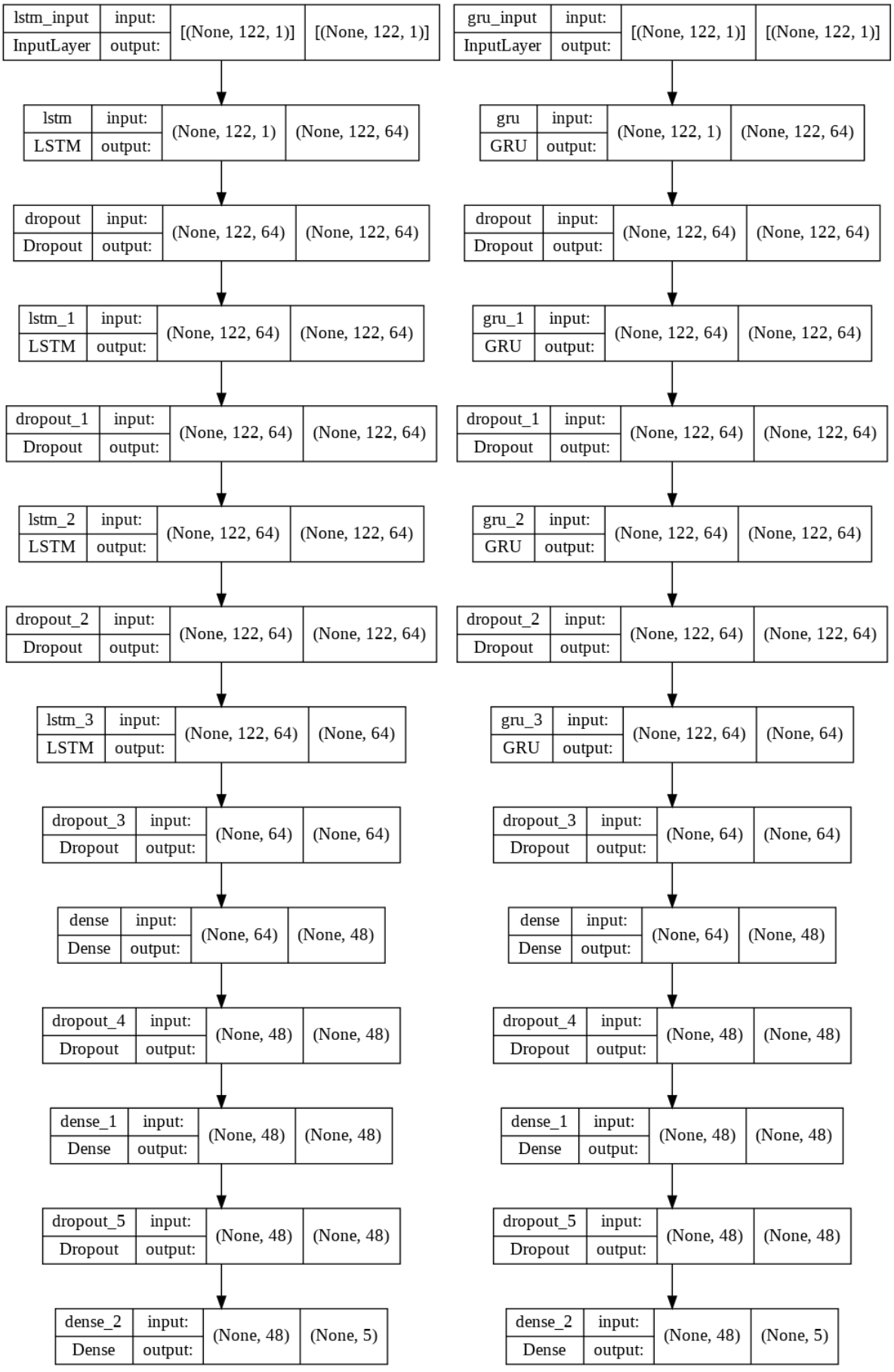


FIGURE 4.8 – Architectures des modèles basés sur RNN-LSTM et RNN-GRU

4.5.1.5 Modèles hybrides CNN-RNN

Beaucoup d'études dans le domaine de DL, utilisent des méthodes hybrides, en combinant plusieurs techniques dans un seul algorithme, Afin de bénéficier des avantages de chaque technique. Nous et après beaucoup d'expérimentations avec les techniques précédentes, Nous avons entamé la création de différents modèles à partir des méthodes hybrides, entre le CNN et les différentes variantes de RNN, du coup nous avons réussi d'avoir 4 modèles efficaces pour la détection d'intrusions. Dans cette partie de mémoire nous présentons les architectures avec leurs paramètres et propriétés.

Modèles CNN-LSTM et CNN-GRU

Vu la similarité de fonctionnement et les résultats proches entre les deux variantes de RNN, LSTM et GRU, Nous avons procéder la réalisation de deux modèles qui seront similaires en nombre de couches et propriétés, Afin de pouvoir comparer et distinguer lequel des hybrides est plus efficace avec notre jeux de données. Et dans l'espérance d'améliorer les précisions de détection précédentes, Nous avons ajouté des couches CNN's, on a sortis avec les modèles CNN-LSTM et CNN-GRU.
Comme le montre la Figure 4.9, les deux modèles partagent 4 couches de convolution 1D avec un padding='same', 2 couches Max-pooling 1D avec une taille de 2 qui se charge de redimensionner les carte caractéristiques (Feature map) avec la valeur maximale obtenue, Ils partagent également 2 couches LSTM/GRU pour CNN-LSTM/CNN-GRU respectivement, la première avec un return_sequences=True qui garde la forme de sortie, la dernière avec =False qui retourne la dernière séquence. Deux couches cachées sont également ajoutées pour une meilleure classification avec 4 couche de dropout de 0.1 qui élimine le problème de disparaissions de gradient, La couche convolutif d'entrée et la couche de sortie ont les même dimensions d'entrés et paramètres que le modèles CNN. Les deux modèles ont été entrainés et testés avec le même nombre d'époques (20 époques) et même avec le même optimiseur 'Adam' et même fonction de perte 'Categorical_crossentropy'.

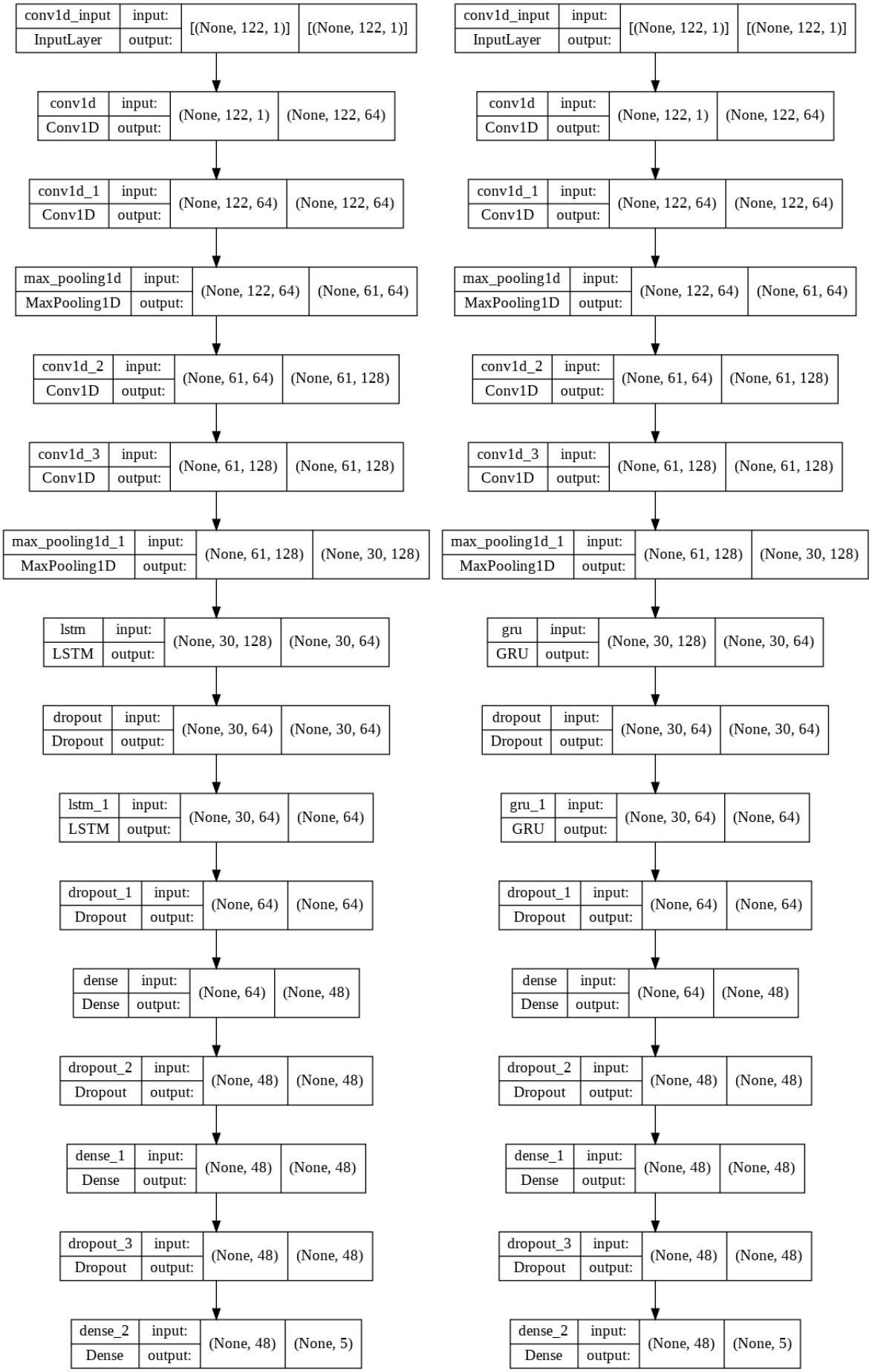


FIGURE 4.9 – Architecture des modèles CNN-LSTM et CNN-GRU

Modèles CNN-BiLSTM et CNN-BiGRU

Le BiLSTM et BiGRU sont deux variantes de RNN, malheureusement et après beaucoup d'essaies, Nous n'avons pas arrivé à y'avoir un modèle basé sur RNN seulement, qui à notre satisfaction au terme de performance, stabilité des résultats et efficacité. Au Revenge en les combinant avec le CNN, les résultats étaient à ce que nous attendions.

Ces deux méthodes interéssantes fonctionnent a base des couche LSTM et GRU cités précédemment, dans la particularité est d'avoir une deuxième couche cachée LSTM/GRU pour chacune des couche LSTM/GRU, la première analyse et traite la séquence tel qu'elle est la deuxième dans le sens inverse. BiGRU se converge plus rapide vu au nombre de porte que possède GRU par rapport LSTM. Mais les résultats dans les approches proposées sont proches. Pour cela nous avons donné deux modèles similaires et efficaces pour chacune des variantes, Afin de pouvoir comparer leurs résultats.

La Figure4.10 montre l'architecture des modèles basés les hybrides CNN-BiLSTM et CNN-BiGRU respectivement. Les deux architectures se différencies uniquement au type de couche RNN, dans le but d'une comparaison équitable comme nous avons mentionné auparavant. Les deux modèles partagent : la couche d'entrée convolutif avec une entrée traditionnelle de même dimensions règles précédentes, une couche de sortie qui filtre avec la fonction d'activation Softmax, une couche de convolutions 1D, Deux couches MaxPooling1D avec une taille de 5, Deux couches de BatchNormalization ou bien normalisation par lot qui est une méthode utilisée pour rendre l'entrainement des réseaux de neurones artificiels plus rapide et plus stable, grâce la normalisation des entrées des couches par le recentrage et remise à l'échelle, deux couches Bidirectionnelles LSTM/GRU avec un return_sequences=False dont la première est suivie d'une couche Reshape redimensionnant la sortie, ainsi qu'une couche entièrement connectée et un dropout de 50% avant la couche finale de multi-classification.

CNN-BiLSTM et CNN-BiGRU ont été entrainé et testé pour une validation de KDDTest sur 20 époques, avec le même optimiseur et fonction de perte que les méthodes précédentes.

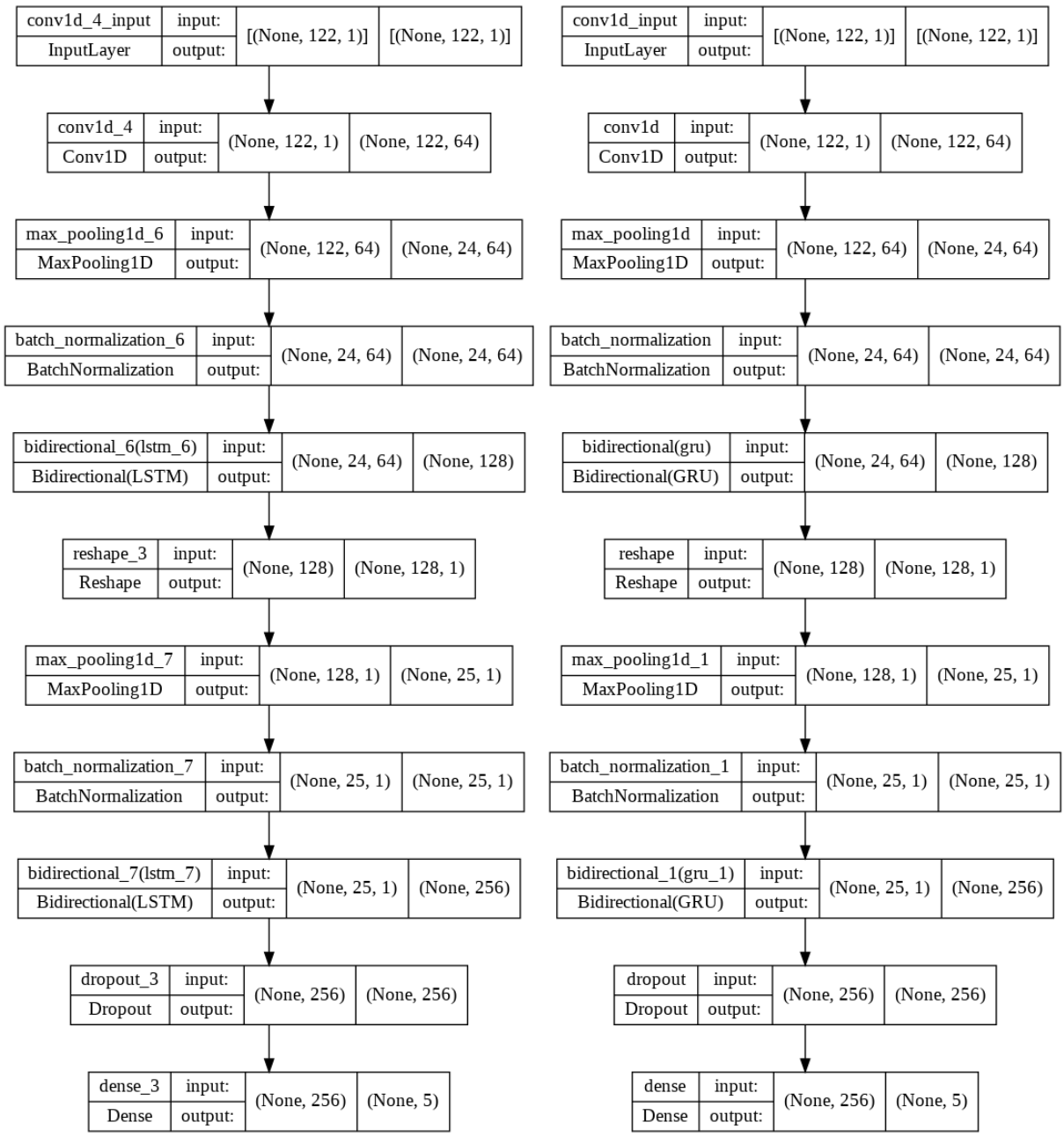


FIGURE 4.10 – Architectures des Modèles Basés sur CNN-BiLSTM et CNN-BiGRU

4.5.2 Discussion et Comparaison des Résultats

4.5.2.1 Discussion des Performances

Nous avons implémenté 8 méthodes de détection d'intrusions en se basant sur 8 méthodes de DL, 4 avec : DNN, CNN, RNN (LSTM et GRU), et 4 méthodes hybrides sont :CNN-LSTM, CNN-GRU, CNN-BiLSTM et CNN-BiGRU. L'entrainement des modèles est fait sur KDDTrain+ de 125 milles échantillons déséquilibrés, le test de validation de nos modèles a été réalisé sous un type de jeu de données NSL-KDD qui est KDDTest20+, connu par la variété de ses types d'attaques, ce qui fait que c'est difficile d'atteindre une exactitude élevée, comme le montre la Figure 4.11 . Mais d'un autre côté, c'est un moyen crédible de tester d'efficacité d'un système de détection d'intrusions. De plus aucune technique d'extraction des caractéristiques n'été faite sur le dataset dans le but de

simuler un trafic de paquets le plus réel possible, tous types de paquet.
Des dizaines de tests ont été faits pour chaque méthode. Vu aux spécifications de données que nous avons utilisées, les temps de l'exécution étaient considérablement longs, surtout pour les modèles les plus complexes. La Figure 4.12 montre les résultats d'exactitude et fonction de perte pour nos modèles proposés.

Méthode	DNN	CNN	RNN-LSTM	RNN-GRU	CNN-LSTM	CNN-GRU	CNN-BiLSTM	CNN-BiGRU
Accuracy	80.72	74.21	73.78	74.77	75.56	76.10	76.22	76.76

TABLEAU 4.4 – Comparaison des taux de réussite des 8 méthodes.

Le tableau 4.4 montre d'exactitude de nos différents modèles. D'où on voit que le DNN marque un taux de réussite plus élevé de 80.72% malgré sa simplicité de fonctionnement par rapport aux autres méthodes, ce qui est un très bon score par rapport de jeux de données utilisé. Les deux variantes de RNN marquent deux scores proches avec un petit avantage pour GRU, remarquant aussi que le temps d'entrainement et validation est également pour le GRU contre LSTM, vu aux portes supplémentaires que LSTM possède dans son architecture.
Si le RNN est exigeant au terme du cout du temps et consommation des du ressources systèmes, Le CNN de son côté est plus optimale pour notre genre d'opérations. Il fournit notamment des taux de réussite encore plus importante, 74.21% est un bon score pour KDDTest20+, Mais en combinant CNN et RNN, on voit que le pourcentage d'exactitude a considérablement monté. La combinaison CNN-GRU surpasse CNN-LSTM, ce qui confirme le résultat de RNN, GRU est plus efficace que LSTM dans notre cas. Les hybrides bidirectionnels CNN-BiGRU et CNN-BiLSTM donnent des résultats plus avancés que leurs unidirectionnels correspondants CNN-GRU et CNN-LSTM respectivement. Donc encore une autre fois une preuve que GRU est plus adapté à la détection d'intrusions avec NSL-KDD.

Afin d'avoir un aperçu sur l'efficacité de nos modèles sur une classification supervisé, Nous avons testé nos NIDS avec un dataset plus léger, qui a des paquets similaires et de même types que de KDDTrain+, et nous avons comparé les résultats avec notre dataset KDDTest20+ qui possède de nombreux nouveaux types d'attaques.

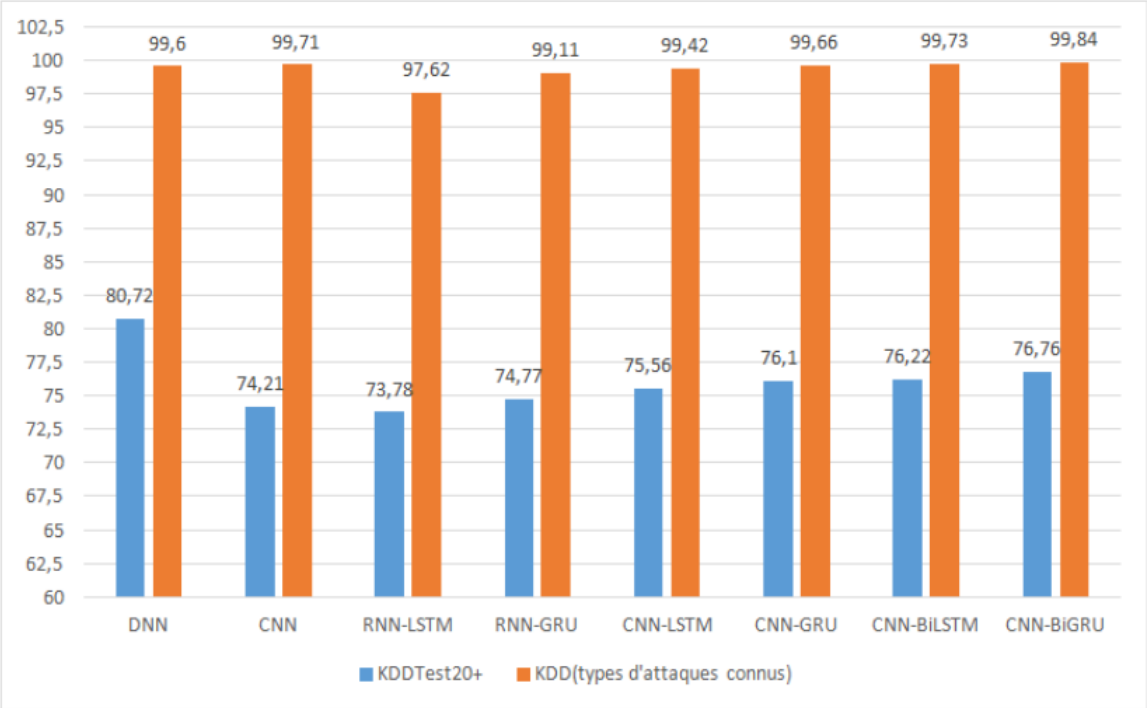
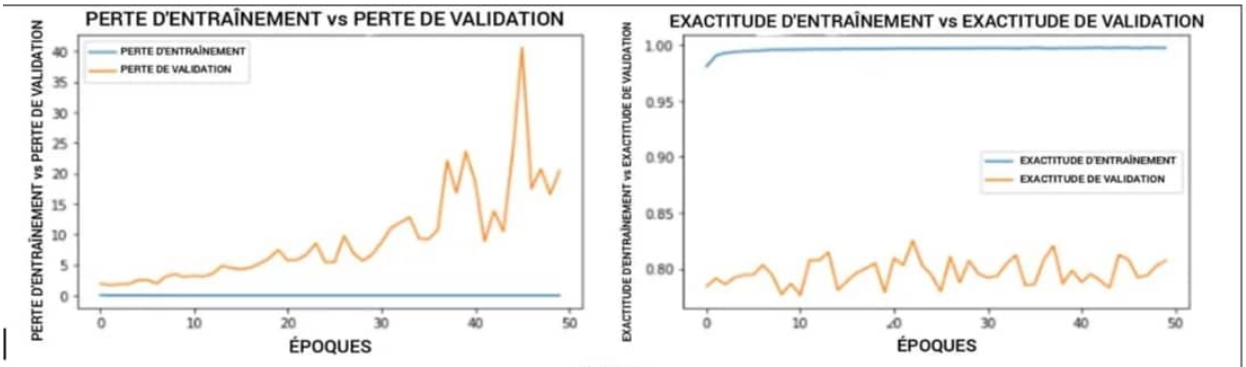
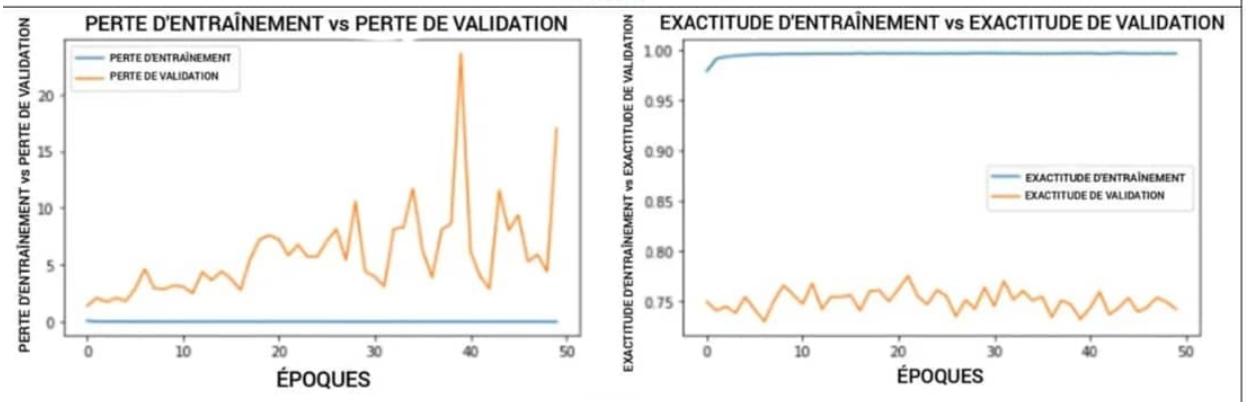


FIGURE 4.11 – L'exactitude de validation des modèles avec KDDTest20+ et KDD Legé avec types d'attaques connues

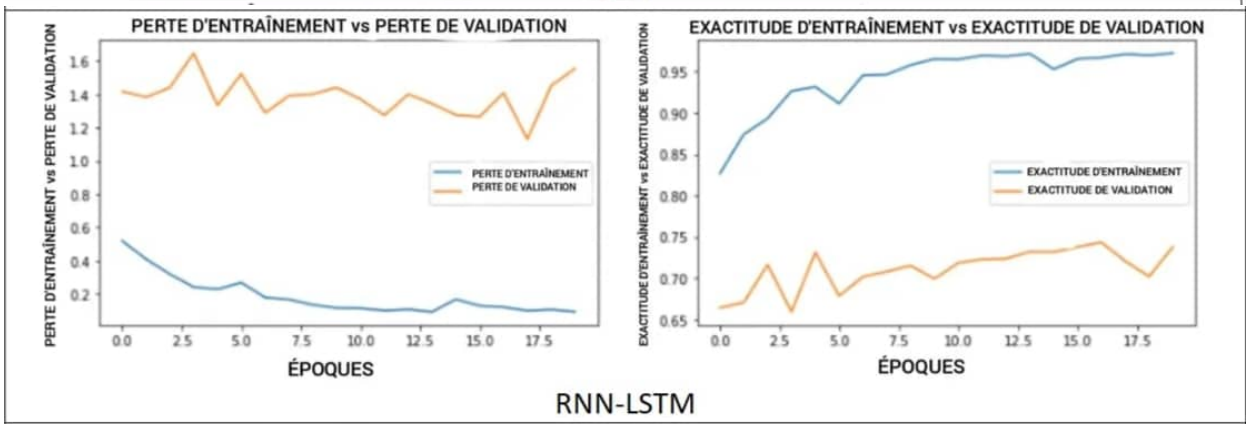
Nos modèles sont très efficaces avec des dataset d'une difficulté moyennes, et contre des attaques connues ou très similaires. Cependant l'exactude diffère d'une méthode a une autres avec un dataset qui possède une variété de paquet malveillants avec des beaucoup de types inconnus aux IDS.



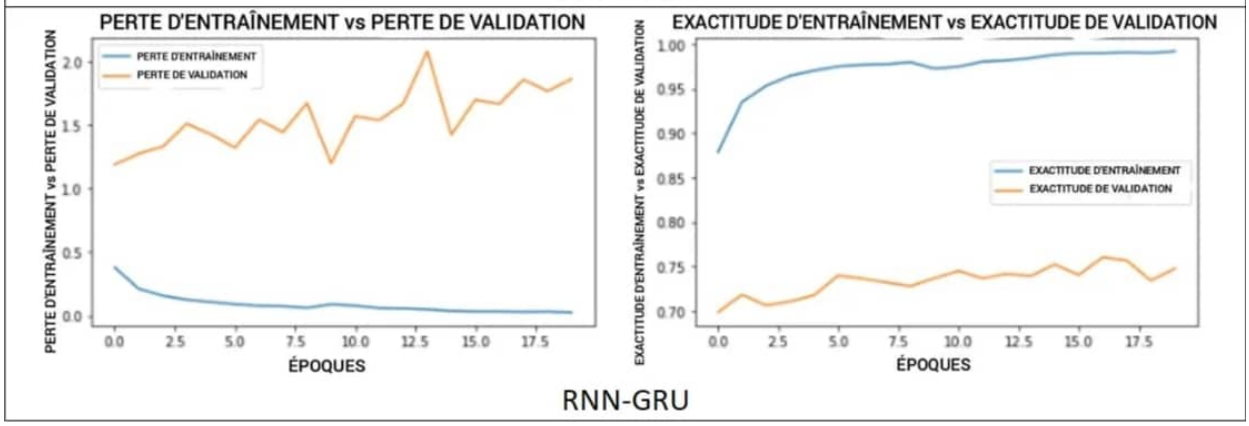
DNN



CNN



RNN-LSTM



RNN-GRU

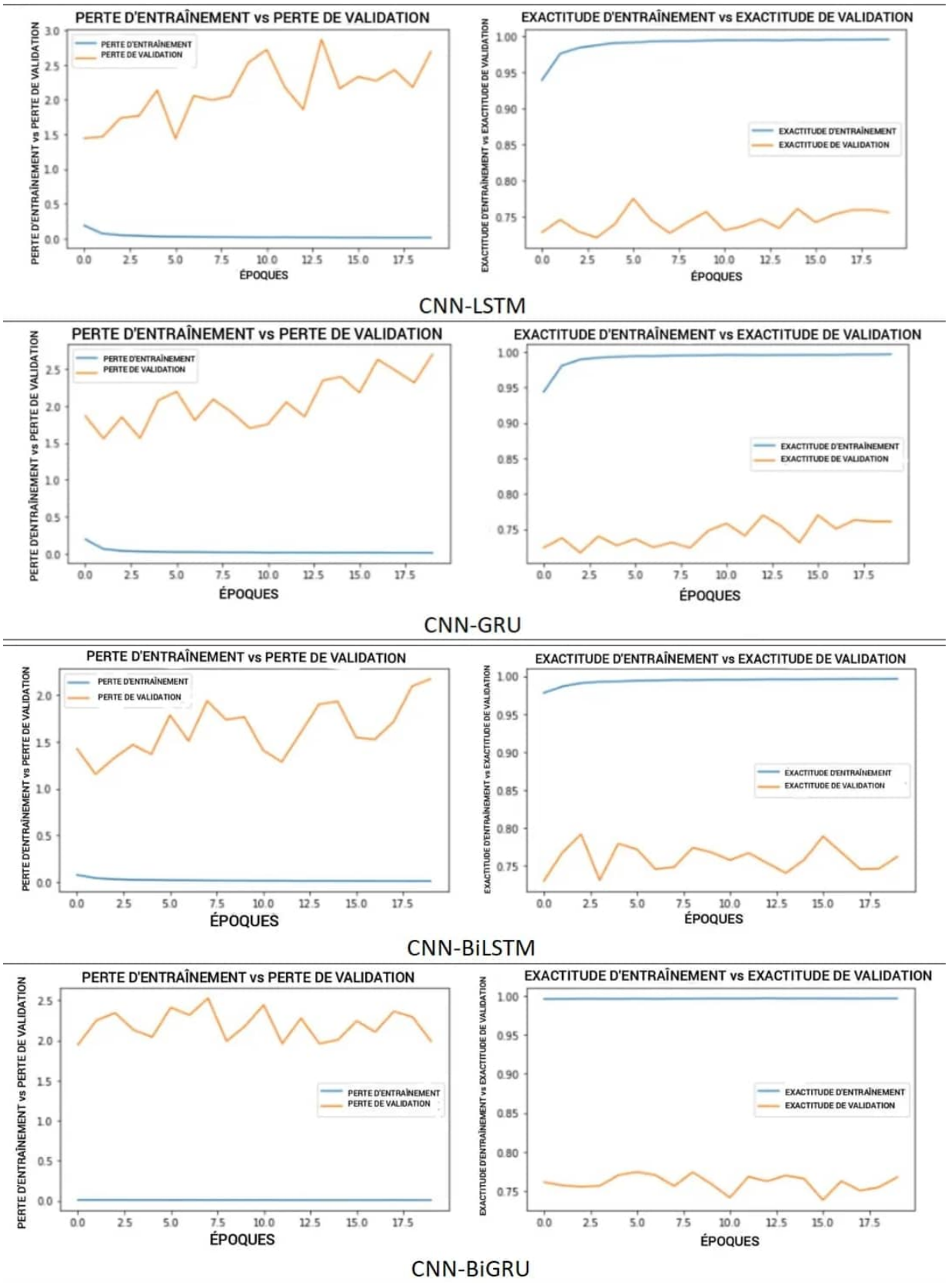


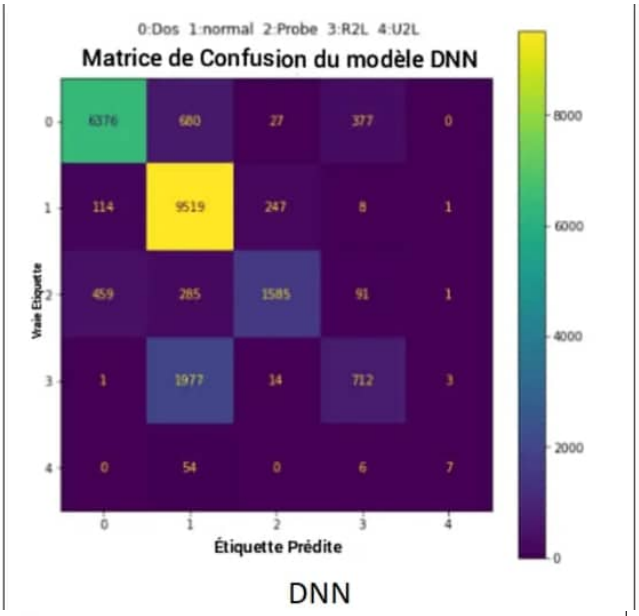
FIGURE 4.12 – Exactitude et Fonction de Perte de de KDDTrain+ et KDDTest20+

Beaucoup de tests ont été faits afin d'obtenir les bonnes Hyper-paramètres pour chaque modèle. Ces paramètres ne peuvent pas ajuster durant la phase de l'apprentissage, pourtant qu'ils ont un grand impact sur les performances des modèles durant l'apprentissage. Ils comprennent les

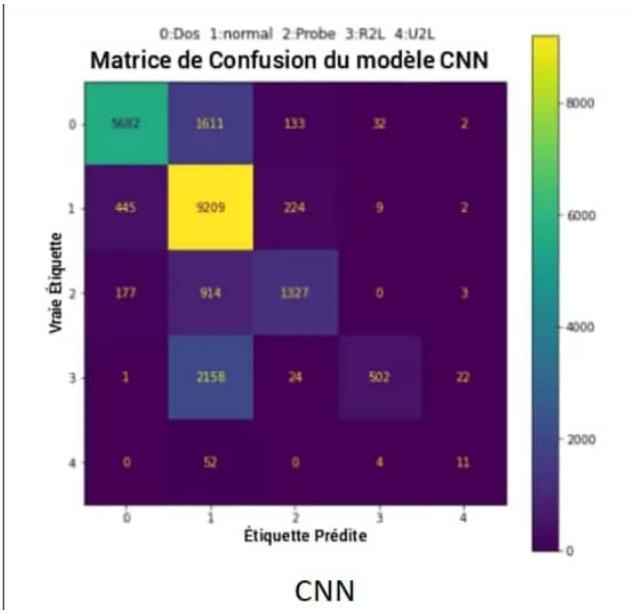
variables qui déterminent la structure du réseau (Nombre de neurones, Nombre de couches, fonction d'activation, ...), le lot d'échantillons (Batch Size) et le nombre d'itérations ...etc.
Toutes les expérimentations ont été faites sur l'ensemble de données du NSL-KDD indiquées au tableau 4.1. Les données d'entraînement ont été divisées sur 2 : 80% pour l'apprentissage et 20% pour l'évaluation. Les modèles ont été formés de 50 époques pour le DNN et CNN et 20 époques pour le reste des modèles. Lors de l'entrainement, ils ont obtenu une exactitude très bonne qui tende vers 1 pour toutes les méthodes. Nous notons ici que les 8 modèles convergent vers une valeur de perte minimale, ou ils ont presque la même valeur de perte d'apprentissage. Ce qui indique que ces modèles seront généralisés bien au-delà de l'ensemble d'apprentissage.
Ensuite, nous avons testé ces modèles sur l'ensemble de tests KDDTest20+. Comme montré dans la figure 4.12, Les modèles donnent des bons résultats de 73.78% à 80.72% par rapport la particularité et le niveau de difficulté qu'impose le dataset. Par contre à la fonction de perte qui marque de valeurs élevés et loin de des valeurs marquées en entrainements, ce qui signifier, que les modèles ont besoins encore d'améliorations pour réduire la valeur de perte au minimum.

4.5.2.2 Rapports de classifications

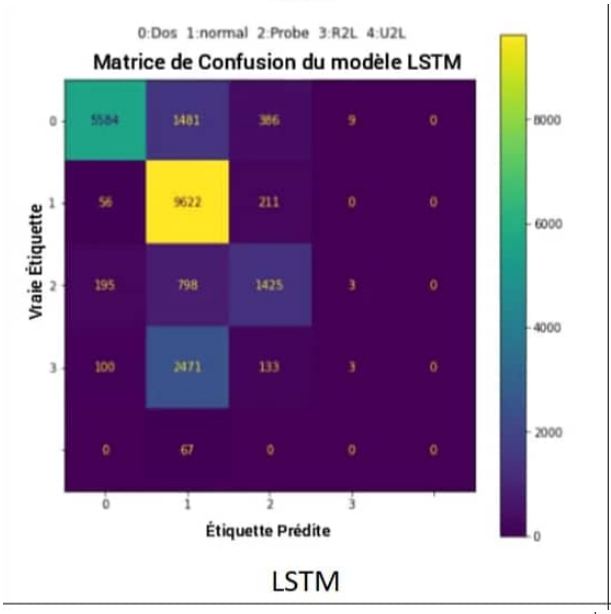
Le Tabteau 4.5 illustre les rapports de classification de nos algorithmes de classification DL. On a un aperçu sur le nombre, la précession, le rappel (recall) et le f1-score de chacune des classes.
A l'aide des matrices de confusions normalisées ci-dessous, nous avons pu analyser les résultats détaillés de nos IDS. Comme les performances des systèmes IDS reposent sur sa capacité de différencier les comportements normaux des comportements malicieux. On constate que la classe Normal qui représente le trafic normal a bien été identifiée par tous les algorithmes avec une précision de 97% pour LSTM, 96% pour DNN, 9% pour CNN-BiLSTM et 93% pour le reste des méthodes. Cette précision signifie que le nombre des fausses alarmes est minimal. Au revanche le DNN prend un avantage avec un rappel de 0.76 contre 6% à 70% pour le reste des méthodes, ce que signifié que le modèle de DNN est plus efficace que les autre modèles dans l'identification des attaques réseaux avec un taux de fausse négative (FN) réduits. Notons également pour cette classe que les f1-score (les moyennes harmoniques) de reste des modèles sont tres proches avec un petit avantage de CNN-GRU sur CNN-LSTM, CNN-BiGRU sur CNN-BiLSTM et RNN sur CNN, notons aussi que la combinaison des CNN et RNN a aider faire augmenter le taux de détection pour cette classe.



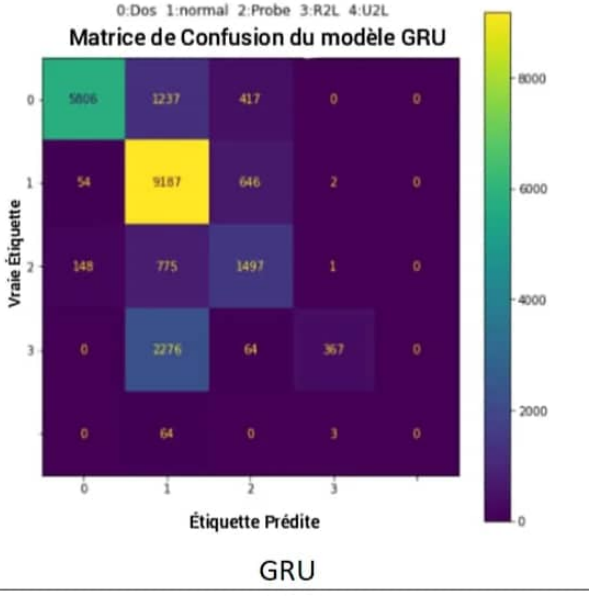
DNN



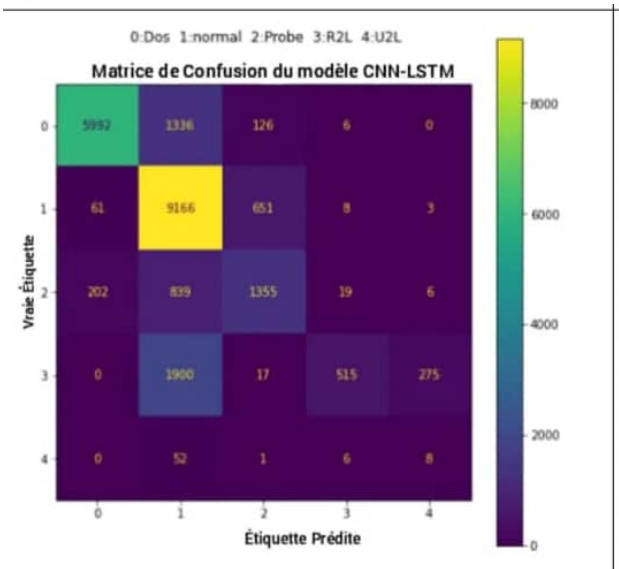
CNN



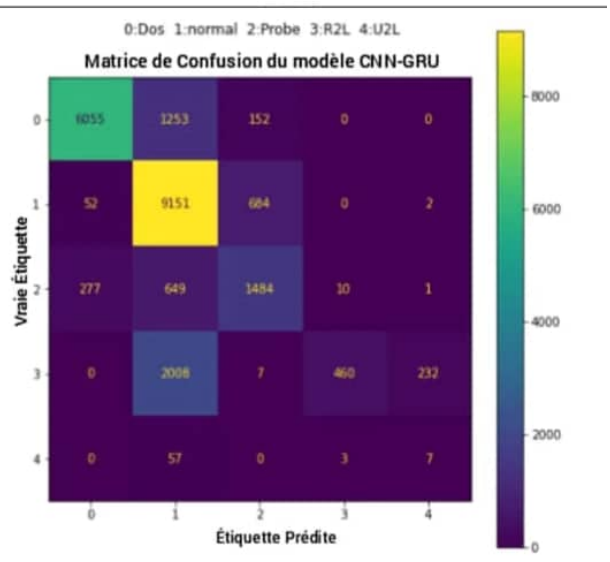
LSTM



GRU



CNN-LSTM



CNN-GRU

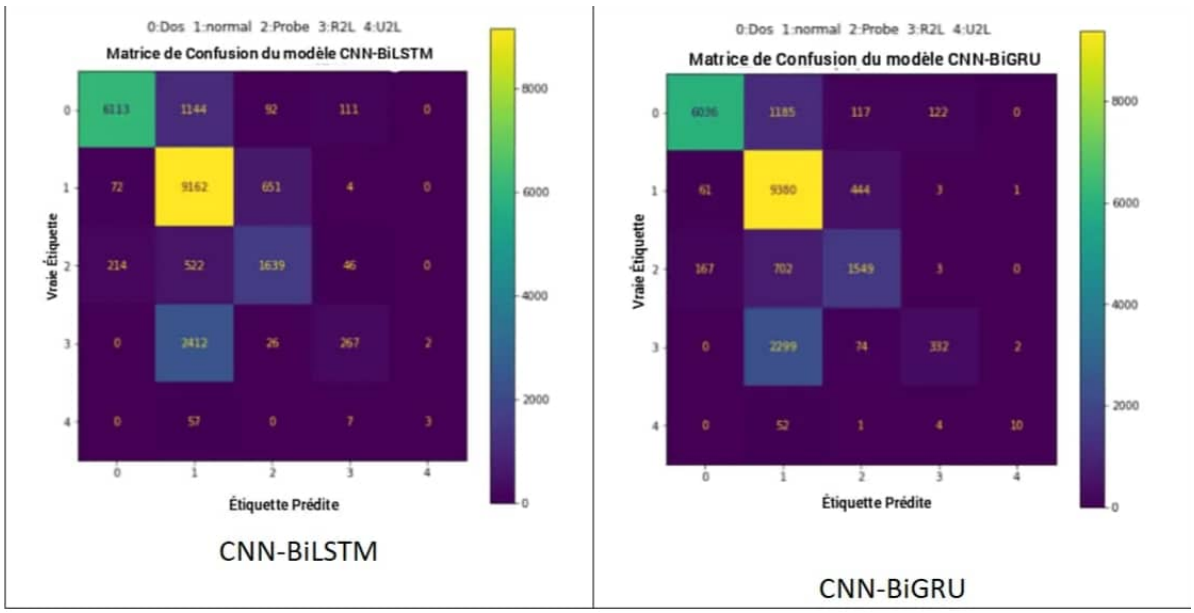


FIGURE 4.13 – Les Matrices de Confusion des 8 méthodes (NB : U2R au lieu U2L)

La classe DoS a été détectée par le DNN avec une précision de 85% et un rappel de 95%, surpassant les autres modèles proposés. On remarque une égalité entre CNN-BiGRU, CNN-BiLSTM, CNN-GRU et CNN-LSTM avec précisions des environs de 81% et rappel de 96%, Par contre le GRU surpasse son conquérant LSTM, ce dernier avec le CNN qui ont marqués les pires résultats pour cette classe d'attaques DoS avec une harmonie de 0.83.

Probe c'est la seule classe que le CNN-BiLSTM a pu surpassé son conquérant CNN-BiGRU en précision avec 68% contre 64% pour ce dernier ce que signifie que le nombre des attaques classées par CNN-BiLSTM comme des Probe est plus supérieur et donc le nombre de fausses alarmes est moins que CNN-BiGRU, Mais en terme d'efficacité l'hybride CNN-BiGRU marque un rappel plus élevé ce que fait que l'harmonie est presque similaire entre les deux. Ce qui est le cas contraire entre les deux variantes de RNN, ou on voit que GRU et plus précis et LSTM et plus efficace. Le DNN donne les meilleurs résultats en harmonie pour cette classe 0.74. Le CNN a donné le moins en précision, et la combinaison CNN-GRU surpasse CNN-LSTM en precision et rappel, ce que confirme pour ce genre de paquets le GRU est plus adapté.

Concernant les paquets d'attaque R2L et U2R qui est une classe minoritaire, nous avons marqué une détection faible et varié d'une méthode a une autres. Cela est a cause de déséquilibrement des données de NSL-KDD utilisé entre le KDDTrain+ et KDDTest20+. Ce dernier qui se compose d'une variété de niveaux types d'attaques. Cependant pour R2L, on remarque que le LSTM n'a pas reconnue du tout ce type d'intrusion et sa variante sœur 14% en précision et 98% en rappel. Notons que le DNN, CNN et CNN-GRU offrent une meilleure détection pour cette classe avec un f1-score supérieur à 0.30. Et pour la dernière classe minoritaire U2R, le RNN a échoué de la détecter avec ces deux variantes. La précision de CNN et CNN-BiGRU est le meilleur pour cette classe, avec un avantage pour ce dernier de 77% en rappel.

Classe		DNN	CNN	LSTM	GRU	CNN-LSTM	CNN-GRU	CNN-BiLSTM	CNN-BiGRU
Normal	Precision	0.96	0.93	0.97	0.93	0.93	0.93	0.93	0.95
	Recall	0.76	0.66	0.67	0.68	0.69	0.70	0.69	0.69
	f1-score	0.85	0.77	0.79	0.78	0.79	0.80	0.79	0.80
	Support	12 515	13 944	14 439	13 539	13 293	13 118	13 297	13618
DoS	Precision	0.85	0.76	0.75	0.78	0.80	0.81	0.82	0.81
	Recall	0.92	0.90	0.94	0.97	0.96	0.95	0.96	0.96
	f1-score	0.88	0.83	0.83	0.86	0.87	0.87	0.88	0.88
	Support	6 950	6 305	5 935	6 008	6 255	6 384	6 399	6 264
Probe	Precision	0.65	0.55	0.59	0.62	0.56	0.61	0.68	0.64
	Recall	0.85	0.78	0.66	0.57	0.63	0.64	0.68	0.71
	f1-score	0.74	0.64	0.62	0.59	0.59	0.63	0.68	0.67
	Support	1 873	1 708	2 155	2 624	2 150	2 327	2 408	2 185
R2L	Precision	0.26	0.19	0.00	0.14	0.19	0.17	0.10	0.12
	Recall	0.60	0.92	0.20	0.98	0.93	0.97	0.61	0.72
	f1-score	0.37	0.31	0.00	0.24	0.32	0.29	0.17	0.21
	Support	1 194	547	15	373	554	473	435	464
U2R	Precision	0.10	0.16	0.00	0.00	0.12	0.10	0.04	0.15
	Recall	0.58	0.28	0.00	0.00	0.03	0.03	0.60	0.77
	f1-score	0.18	0.21	0.00	0.00	0.04	0.05	0.08	0.25
	Support	12	40	00	00	292	242	5	13

TABLEAU 4.5 – Comparaison des Rapports de classifications des 5 classes pour les 8 méthodes implémentées.

En comparant dans le tableau 4.6 la classification des 22 544 échantillons de KDDTest20+ par modèles de multi-classification, On constate la puissance des modèles DNN à la bonne détection des paquets malveillants avec un TPR de 85%. En contrepartie sa faiblisse réside la détection des paquets non-malveillants comme étant des paquets suspects, dont le FPR et de 3.80%. Les modèles hybrides de leurs côtés montrent des très bons résultats à la vraie détection des attaques avec TPR entre 80.32 et 81.94%, d'où ils surpassent le simple CNN qui a 76.16% et les méthodes RNN qui ont 74.85% pour LSTM et 77.82% pour GRU. Doc la combinaison des deux algorithmes a importé ses fruits, surtout que le FPR de CNN est le plus élevé à 4.13% et celui de GRU et le moins avec 1.33%.

	TP	FP	TN	FN	TPR%	FPR%
DNN	6 376	574	14 510	1 084	85.46	3.80
CNN	5 682	623	14 461	1 776	76.16	4.13
LSTM	5 584	351	14 733	1 876	74.85	2.32
GRU	5 806	202	14 882	1 654	77.82	1.33
CNN-LSTM	5 992	263	14 821	1 468	80.32	1.74
CNN-GRU	6 055	329	14 755	1 405	81.16	2.18
CNN-BiLSTM	6113	286	14 796	1 347	81.94	1.89
CNN-BiGRU	6036	228	14 856	1 424	80.91	1.51

TABLEAU 4.6 – Comparaison des Rapports d'évaluation de classification.

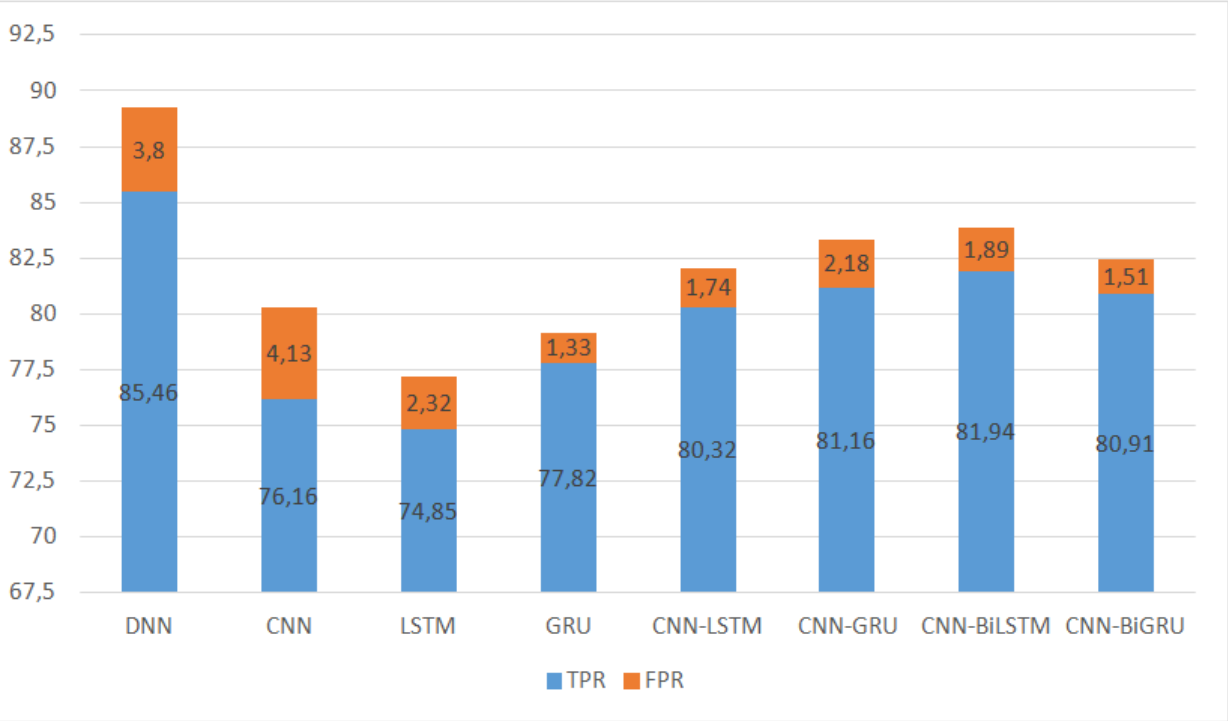


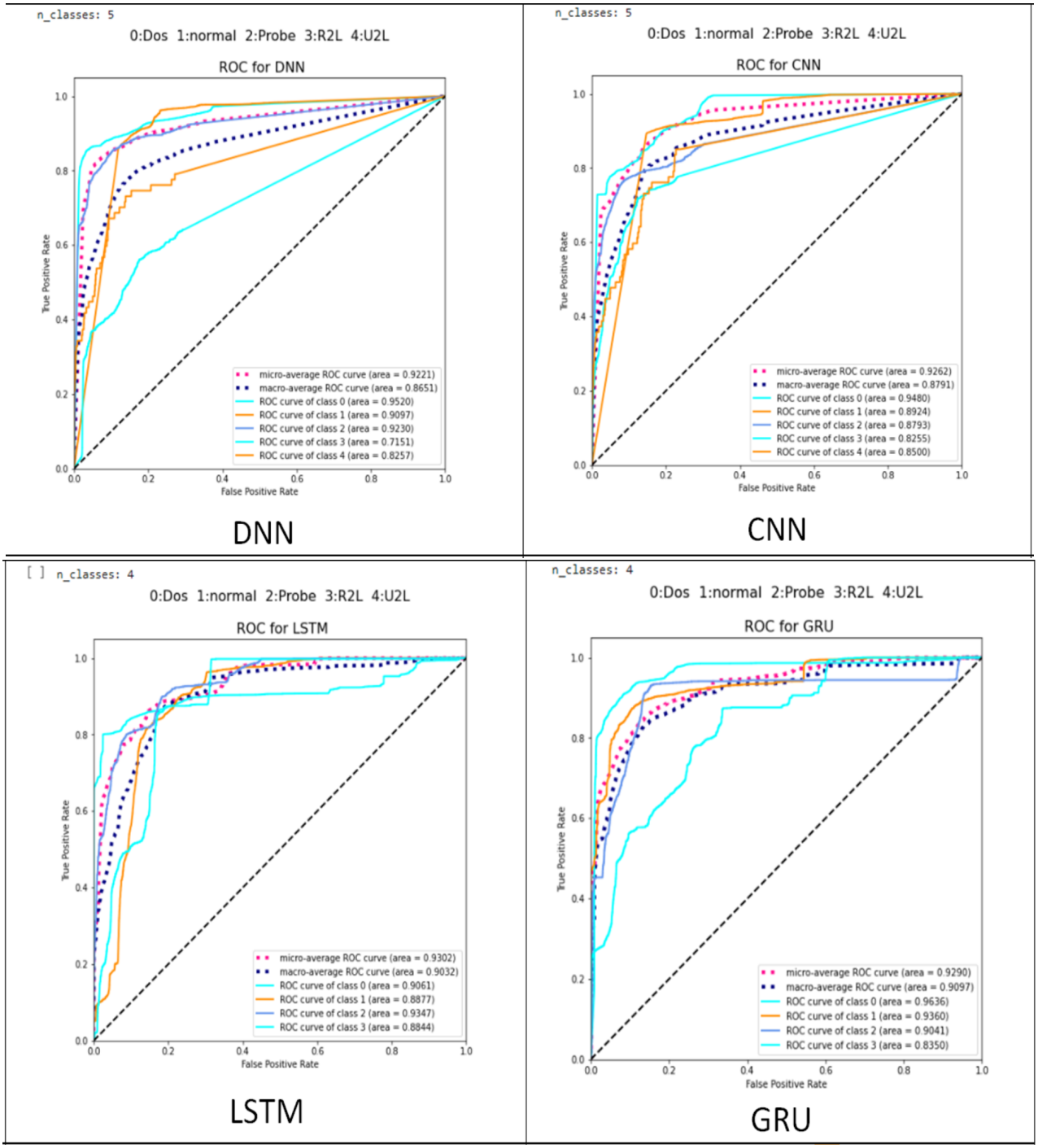
FIGURE 4.14 – Comparaison de TPR et FPR des modèles proposés.

ROC curves

La courbe Receiver Operating Characteristic (ROC) est une mesure de performance souvent utilisé pour les classifieurs binaires et surtout dans le cas où les classes sont déséquilibrées. Elle trace le taux de vrai positif en fonction du taux de faux positif à différents seuils de classification, afin de montrer le compromis entre la sensibilité (TPR) et la spécificité (FPR). Mathématiquement, il est calculé par la surface sous la courbe appelée Area Under Curve (AUC), Dans le cas idéal, nous aimerions avoir un Auc = 1, de sorte que la courbe ROC passe par le coin supérieur gauche du carré au point (FPR = 0, TPR = 1).

Les courbes ROCs illustrées dans la figure 4.15, nous permettent de bien évaluer la précision de nos classifieurs avec des classes déséquilibrées, et nous permet de visualiser les résultats dont nous avons

parlé précédemment. Nous pouvons constater que nos modèles sont performants à la détection des deux classes Normal et DoS, et d'une façon moins avec la classe Probe. Tandis que les deux classes R2L et U2R sont mal-détectés par nos modèles, plus précisément les modèles basés sur RNN qui marquent des taux de détection très faibles.



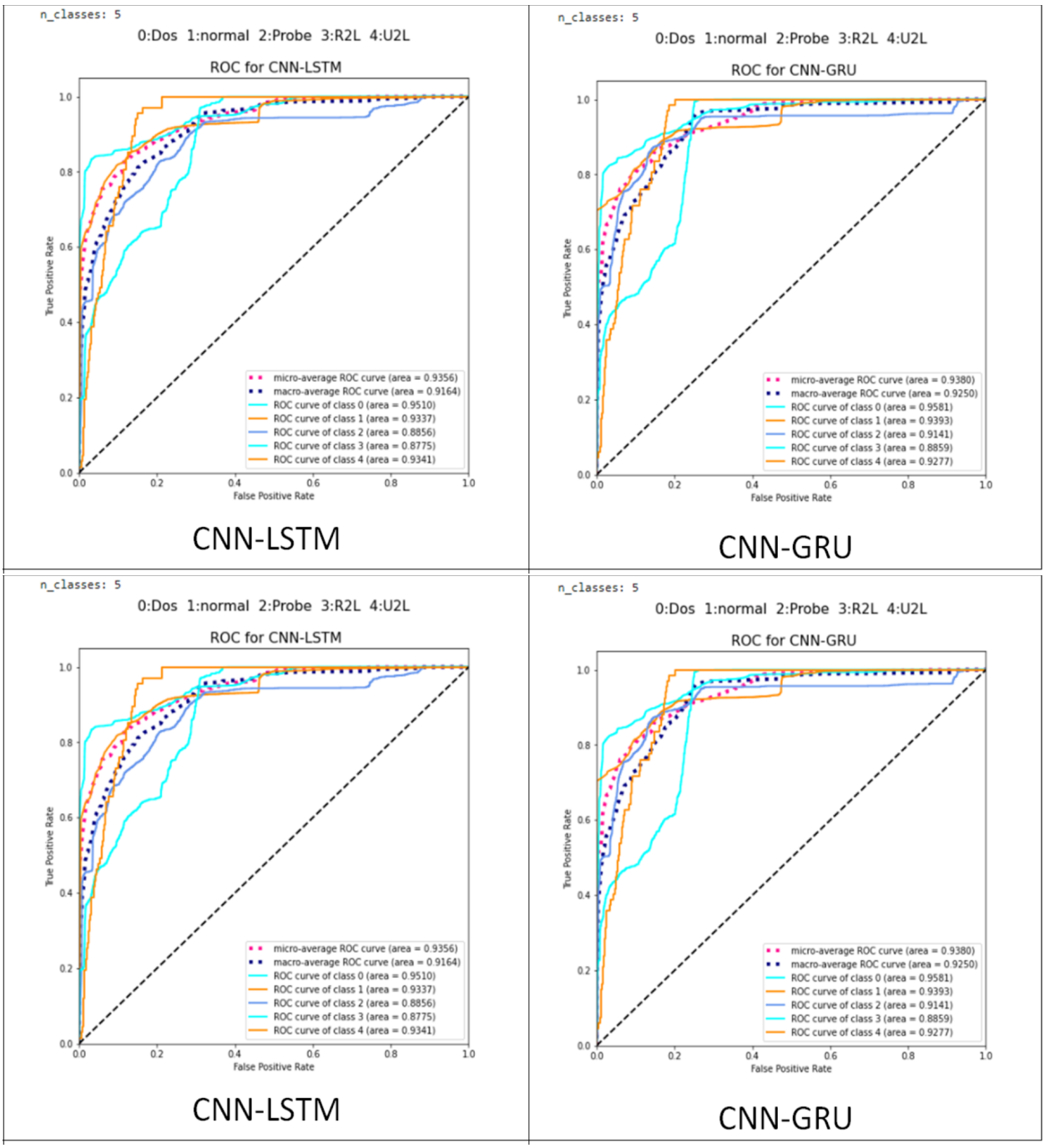


FIGURE 4.15 – Les Courbes du ROC des Modèles Implémentés.

4.6 Conclusion

Dans ce travail, nous avons implémenté 8 modèles de multi-classification, basés sur 8 méthodes de Deep Learning, Ces modèles ont été entrainés et testés sur les dataset NSL-KDD avec une classification de 5 classes (Normal, Dos, Probe, R2L et U2R).

	Accuracy %	Precision %	F1-Score	TPR (Rc) %	FPR %
DNN	80.72	91.74	0.88	85.46	3.80
CNN	74.21	90.11	0.82	76.16	4.13
LSTM	73.78	94.08	0.83	74.85	2.32
GRU	74.77	96.63	0.86	77.82	1.33
CNN-LSTM	75.56	95.79	0.87	80.32	1.74
CNN-GRU	76.10	94.84	0.87	81.16	2.18
CNN-BiLSTM	76.22	95.53	0.88	81.94	1.89
CNN-BiGRU	76.76	96.36	088	80.91	1.51

TABLEAU 4.7 – Résultats de nos modèles proposés.

Plus précisément, nous avons utilisé pour l'entrainement un sous-ensemble de dataset KDDTrain+, qui contient des données déséquilibrés un avec un petit nombre d'échantillons. Pour le test, KDDTest20+, qui contient une variété de types et nombre d'attaques que KDDTrain+ ne possède pas. Plusieurs expérimentations ont été faites pour chacune des méthodes, et plusieurs propriétés et paramètres ont été testés, afin de trouvé les hyper-paramètres pour chaque modèle d'une des méthodes de DL. Nous avons rencontré beaucoup de problèmes lors de programmation et l'implémentation des modèles. Beaucoup du temps a été investi dans ce travail, l'exécution et l'entrainement des modèles ont pris du temps considérable, vu qu'on a voulu testé l'efficacité de nos classifieurs sur des dataset tel qu'ils sont, sans une extractions des caractéristiques ou rééquilibrage des échantillons , dans le but de simuler le plus possible un flux de données réseaux réel avec tout paquet important ou de bruit.

Les résultats étaient très bons par rapport à la spécificité des données, et ils ont montré des taux de précision élevés supérieurs à 90% pour tous les modèles. Et des taux de réussite de 73.78% pour LSTM, jusqu'à 80.72% pour le DNN, avec des fausses détections réduite pour la majorité des modèles comme montré dans tableau 4.7. La complexité de dataset de test nous nous a poussé a testé nos modèles entrainé avec un KDDTest20+ réduit au niveau de difficulté en contenant des attaques similaires des mêmes types que celui d'entrainement. La tâche a été très facile nos IDS's et les résultats d'exactitude de tous les modèles convergent vers 1 avec sauf pour LSTM. Ce qui prouve l'efficacité de nos modèles sur les différents niveaux de difficulté.

Conclusion générale

Les systèmes de détection d'intrusions font partie des pratiques qui consistent à sécuriser les éléments vulnérables grâce aux technologies de l'information et de communication (TIC). Les IDS's couvrent l'insuffisance des logiciels d'anti-virus et firewall face à l'évolution des nouvelles menaces plus sophistiquées. Cette étude a été menée afin de démontrer l'efficacité du Deep Learning pour le domaine détection d'intrusions réseaux en utilisant des IDS réseaux NIDS (Network Intrusion Detection System). Notre objectif est d'implémenter des méthodes de détection d'intrusions basées sur l'apprentissage profond supervisé et évaluer ses performances.

Nous avons commencé par le choix de dataset, ou nous avons choisi NSL-KDD, plus précisément un sous-ensemble d'entrainement KDDTrain+, et de test KDDTest20+, les deux qui marquent les moins taux d'exactitudes dans les recherches menées auparavant. Nous voulions tester le flux de données tel qu'il est, afin de garantir une crédibilité sur un trafic réel.

Ensuite nous avons procédé a implémenter 8 modèles qui se basent sur 8 méthodes de multi-classification DL pour un apprentissage supervisé. Au début 4 modèles : le DNN, CNN, et deux variantes de RNN ont été implémentés LSTM et GRU individuellement, Ensuite nous avons combiné le CNN et RNN, et nous avons sorti avec 4 autres modèles, en se basant sur des hybride-méthodes : CNN-LSTM, CNN-GRU, CNN-BiLSTM et CNN-BiGRU. Le choix de ces méthodes a été fait lorsqu'elles conviennent avec un ensemble de données similaires. Ainsi, lorsqu'elles sont fait de bon résultat dans des travaux antérieurs connexes. . Afin d'obtenir les bons hyper-paramètres, plus expérimentations et tests d'entrainement et évaluations ont été faites.

Les résultats obtenus sont très satisfaisants pour la totalité des méthodes, dont CNN-BiGRU et CNN-BiLSTM montrent une amélioration par rapport aux travaux antérieurs, où seules les caractéristiques du trafic réel d'un réseau public sont prises en compte sans aucune information relative aux terminaux connectés, ce qui nous donne à penser que le taux de détection serait encore plus élevé si nous appliquons ces méthodes sur un réseau spécifique. En mesure d'harmonie F1-Score DNN, CNN-BiLSTM et CNN-BiGRU ont marqués les meilleurs résultats avec 0.88, dont l'avantage de DNN sur le reste des modèles réside dans sa capacité de détecter les vrais positifs TPR=85.46Nos NIDS seront intégrés dans un réseau, après avoir sauvegarder les poids du modèle et mettre en place un capteur de réseau et un analyseur où les flux peuvent être lus en temps réel et envoyés dans le modèle pour la prédiction. Le temps de réponses pour une seule prédiction dépend de la complexité de modèles (nombre de paramètres du modèle) qui doit suffisamment minimal pour l'utiliser comme un système de détection d'alarme en temps réel.

Perceptive

Nous voulions dans nos prochains travaux, travailler sur l'amélioration de nos modèles pour le NSL-KDD, et l'utilisation d'autres types de jeux de données, tel que CIC-IDS avec des algorithmes d'apprentissage profond non-supervisés tels que autodidacte, l'encodage automatiques, et d'autres algorithmes intéressants.

Bibliographie

[1] Applying convolutional neural network for network intrusion detection. In *Conférence internationale 2017 sur les progrès de l'informatique, des communications et de l'informatique (ICACCI)*, pages 1222–1228.

[2] Détection d'intrusion et big data hétérogène : une enquête. volume 2, pages 1–41.

[3] Un tutoriel sur les architectures, les algorithmes et les applications pour le deep learning. *Transactions APSIPA sur le traitement du signal et de l'information*, 3.

[4] Une enquête sur les attaques réseau et les systèmes de détection d'intrusion. In *2017 4e Conférence internationale sur les systèmes avancés de calcul et de communication (ICACCS)*, pages 1–7.

[5] `https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be`, Consulté le 10 aout 2022.

[6] `http://www.ordinateur.cc/réseaux/sécurité-des-réseaux/76484.html`, Consulté le 11 mars 2022.

[7] `https://www.prevention-cybercrime.ca/systeme-detection-intrusion`, Consulté le 11 mars 2022.

[8] `httphttps://datatracker.ietf.org/doc/rfc4766/`, Consulté le 11 mars 2022.

[9] `https://ichi.pro/fr/alors-comment-l-ia-est-elle-utilisee-pour-detecter-le-covid-19-229660245198879`, Consulté le 11 mars 2022.

[10] `https://ichi.pro/fr/alors-comment-l-ia-est-elle-utilisee-pour-detecter-le-covid-19-229660245198879`, Consulté le 11 mars 2022.

[11] `https://web.maths.unsw.edu.au/~lafaye/CCM/attaques/buffer-overflow.htm`, Consulté le 12 mars 2022.

[12] `https://www.universalis.fr/encyclopedie/virus-informatique/1-virus-cheval-de-troie-et-ver/`, Consulté le 12 mars 2022.

[13] https://www.lemagit.fr/definition/Botnet#:~:text=Un%20botnet%20(ou%20réseau%20de,autres%20ordinateurs%20reliés%20à%20Internet, Consulté le 13 mars 2022.

[14] https://www.mdpi.com/2076-3417/10/8/2870/htm, Consulté le 14 mars 2022.

[15] https://www.researchgate.net/figure/Presentation-dun-auto-encodeur-La-partie-en-bleu-correspond-a-lencodeur-avec-en_fig20_342876843, Consulté le 14 mars 2022.

[16] https://mobiskill.fr/blog/conseils-emploi-tech/quels-sont-les-algorithmes-de-deep-learning/, Consulté le 14 mars 2022.

[17] https://penseeartificielle.fr/focus-reseau-neurones-artificiels-perceptron-multicouche/, Consulté le 15 mars 2022.

[18] https://fr.blog.businessdecision.com/tutoriel-deep-learning-le-reseau-neuronal-convolutif-cnn/, Consulté le 15 mars 2022.

[19] https://datavalue-consulting.com/deep-learning-reseaux-neurones-recurrents-rnn/, Consulté le 15 mars 2022.

[20] https://dataanalyticspost.com/Lexique/auto-encodeur/, Consulté le 15 mars 2022.

[21] https://www.lemagit.fr/definition/Reseau-antagoniste-generatif-GAN, Consulté le 15 mars 2022.

[22] https://penseeartificielle.fr/focus-reseau-neurones-artificiels-perceptron-multicouche/#2_Presentation_generale, Consulté le 16 mars 2022.

[23] https://mobiskill.fr/blog/conseils-emploi-tech/quels-sont-les-algorithmes-de-deep-learning/, Consulté le 17 mars 2022.

[24] https://fr.wikipedia.org/wiki/R%C3%A9seau_neuronal_convolutif, Consulté le 18 mars 2022.

[25] https://fr.m.wikipedia.org/wiki/Fichier:Recurrent_neural_network_unfold.svg, Consulté le 18 mars 2022.

[26] https://kongakura.fr/article/Reseau-antagoniste-generatif-Generative-adversarial-Network, Consulté le 18 mars 2022.

[27] https://moov.ai/fr/blog/deep-learning-avec-google-colab/, Consulté le 29 août2022.

[28] https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1445304-python-definition-et-utilisation-de-ce-langage-informatique/, Consulté le 29 août2022.

[29] https://datascientest.com/pandas-python-data-science, Consulté le 29 août2022.

[30] https://fr.wikipedia.org/wiki/NumPy, Consulté le 29 août2022.

[31] https://www.lebigdata.fr/tensorflow-definition-tout-savoir, Consulté le 29 août2022.

[32] https://datascientest.com/keras, Consulté le 29 août2022.

[33] https://learn.saylor.org/mod/book/view.php?id=29755&chapterid=5443, Consulté le 29 août2022.

[34] BENSIAH Oussama Akram. *proposition d'une nouvelle approche basé Deep Learning pour la prédiction du cancer du sain*. mémoire de fin d'études, département mathématique et informatique, université L'arbi Ben M'hidi Oum El Bouaghi, Algérie, octobre 2019-2020.

[35] Md Zahangir Alom, VenkataRamesh Bontupalli, and Tarek M Taha. Intrusion detection using deep belief networks. In *2015 National Aerospace and Electronics Conference (NAECON)*, pages 339–344. IEEE, 2015.

[36] J. P. ANDERSON. Computer security threat monitoring and surveillance. *Technical Report, James P. Anderson Company*, 1980.

[37] Asmaa Shaker Ashoor. Importance of intrusion detection system (ids). *Scientific Engineering Research*, 2, 2011.

[38] LAIB Hamz BABAALI Baligh. *Approche basé sur l'apprentissage profond pour la détection d'intrusion réseau*. Projet de fin d'études, département mathématiques et informatique, Université yahia fares, Algerie, 2018-2019.

[39] Ritu Bala and Ritu Nagpal. A review on kdd cup99 and nsl nsl-kdd dataset. *International Journal of Advanced Research in Computer Science*, 10(2), 2019.

[40] Andrew Hay ; Daniel Cid ; Rory Bray. *OSSEC host-based intrusion detection guide*. Burlington, Mass, livre numérique : document : anglais edition, 2008.

[41] Jason Brownlee. A gentle introduction to long short-term memory networks by the experts. *Machine Learning Mastery*, 19, 2017.

[42] Okan Can and Ozgur Koray Sahingoz. A survey of intrusion detection systems in wireless sensor networks. In *2015 6th international conference on modeling, simulation, and applied optimization (ICMSAO)*, pages 1–6. IEEE, 2015.

[43] Bo Cao, Chenghai Li, Yafei Song, and Xiaoshi Fan. Network intrusion detection technology based on convolutional neural network and bigru. *Computational Intelligence and Neuroscience*, 2022, 2022.

[44] Marc et Wespi Andreas Debar, Hervéé et Dacier. Une taxonomie révisée pour les systèmes de détection d'intrusion. In *Annales des télécommunications*, volume 55, pages 361–378.

[45] Dhanabal and Shantharajah. A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. *International journal of advanced research in computer and communication engineering*, 4(6) :446–452, 2015.

[46] Hamouda Djallel. *Un système de détection d'intrusion pour la cyber sécurité*. mémoire de fin d'études master, département informatique, université de 8 Mai 1945, Guelma, Algérie, octobre 2020.

[47] Vokorokos et Baláž. Système de détection d'intrusion basé sur l'hôte. In *2010 IEEE 14th International Conference on Intelligent Engineering Systems*, pages 43–47.

[48] Leandros et Moschoyiannis Sotiris et Janicke Helge Ferrag, Mohamed Amine et Maglaras. Apprentissage en profondeur pour la détection d'intrusion dans la cybersécurité : Approches, ensembles de données et étude comparative. *Journal de la sécurité de l'information et des applications*, 50 :102419.

[49] Ni Gao, Ling Gao, Quanli Gao, and Hai Wang. An intrusion detection model based on deep belief networks. In *2014 Second international conference on advanced cloud and big data*, pages 247–252. IEEE, 2014.

[50] Tal Garfinkel, Mendel Rosenblum, et al. A virtual machine introspection based architecture for intrusion detection. In *Ndss*, volume 3, pages 191–206. Citeseer, 2003.

[51] László Göcs and Zsolt Csaba Johanyák. Survey on intrusion detection systems. In *7th International Scientific and Expert Conference TEAM 2015 Technique, Education, Agriculture & Management*, 2015.

[52] Sandeep Gurung, Mirnal Kanti Ghose, and Aroj Subedi. Deep learning approach on network intrusion detection system using nsl-kdd dataset. *International Journal of Computer Network and Information Security*, 11(3) :8–14, 2019.

[53] Sandeep Gurung, Mirnal Kanti Ghose, and Aroj Subedi. Deep learning approach on network intrusion detection system using nsl-kdd dataset. *International Journal of Computer Network and Information Security*, 11(3) :8–14, 2019.

[54] Simon Hansman and Ray Hunt. A taxonomy of network and computer attacks. *Computers & Security*, 24(1) :31–43, 2005.

[55] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02) :107–116, 1998.

[56] Islam. A novel bigrubilstm model for multilevel sentiment analysis using deep neural network with bigru- bilstm. *Lecture Notes in Electrical Engineering*, 730 :403–414, 07 2021.

[57] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. A deep learning approach for network intrusion detection system. *Eai Endorsed Transactions on Security and Safety*, 3(9) :e2, 2016.

[58] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. A deep learning approach for network intrusion detection system. *Eai Endorsed Transactions on Security and Safety*, 3(9) :21–26, 2016.

[59] Sindhu Kakuru. Behavior based network traffic analysis tool. In *2011 IEEE 3rd International Conference on Communication Software and Networks*, pages 649–652. IEEE, 2011.

[60] Min-Joo Kang and Je-Won Kang. Intrusion detection system using deep neural network for in-vehicle network security. *PLOS ONE*, 11 :1–17, 06 2016.

[61] Min-Joo Kang and Je-Won Kang. Intrusion detection system using deep neural network for in-vehicle network security. *PloS one*, 11(6) :e0155781, 2016.

[62] Jihyun Kim and Howon Kim. Applying recurrent neural network to intrusion detection with hessian free optimization. In *International workshop on information security applications*, pages 357–369. Springer, 2015.

[63] Harley Kozushko. Intrusion detection : Host-based and network-based intrusion detection systems. 2003.

[64] Deepak Kshirsagar, Suraj Sawant, Ravi Wadje, and Pravin Gayal. Distributed intrusion detection system for tcp flood attack. In *Proceeding of International Conference on Intelligent Communication, Control and Devices*, pages 951–958. Springer, 2017.

[65] Collins Achepsah Leke and Tshilidzi Marwala. *Deep learning and missing data in engineering systems*. 2019.

[66] Chao Li, Wei Jiang, and Xin Zou. Botnet : Survey and case study. In *2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC)*, pages 1184–1187, 2009.

[67] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system : A comprehensive review. *Journal of Network and Computer Applications*, 36(1) :16–24, 2013.

[68] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system : A comprehensive review. *Journal of Network and Computer Applications*, 36(1) :16–24, 2013.

[69] Anyi Liu, Yi Yuan, Duminda Wijesekera, and Angelos Stavrou. Sqlprob : a proxy-based architecture towards preventing sql injection attacks. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 2054–2061, 2009.

[70] Paola Stone Martinez. Virtual machines and security. *2013*, pages 1–7, 2013.

[71] Johan Mazel, Pedro Casas, Yann Labit, and Philippe Owezarski. Détection non supervisée d'anomalies du trafic. In *CFIP 2011-Colloque Francophone sur l Ingénierie des Protocoles*, 2011.

[72] Azizjon Meliboev, Jumabek Alikhanov, and Wooseong Kim. Performance evaluation of deep learning based network intrusion detection system across multiple balanced and imbalanced datasets. *Electronics*, 11(4) :515, 2022.

[73] Cédric Michel. *Langage de description d'attaques pour la détection d'intrusions par corrélation d'événements ou d'alertes en environnement réseau hétérogène*. thèse en informatique, Universite Rennes 1, Décembre 2003.

[74] Preeti Mishra, Emmanuel S Pilli, Vijay Varadharajan, and Udaya Tupakula. Intrusion detection techniques in cloud environment : A survey. *Journal of Network and Computer Applications*, 77 :18–47, 2017.

[75] Chirag Modi, Dhiren Patel, Bhavesh Borisaniya, Hiren Patel, Avi Patel, and Muttukrishnan Rajarajan. A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications*, 36(1) :42–57, 2013.

[76] Dietmar PF Möller. Machine learning and deep learning. In *Cybersecurity in Digital Transformation*, pages 77–87. Springer, 2020.

[77] Nour Moustafa and Jill Slay. The evaluation of network anomaly detection systems : Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set. *Information Security Journal : A Global Perspective*, 25(1-3) :18–31, 2016.

[78] Tiwari Nitin, Solanki Rajdeep Singh, Pandya Gajaraj Singh, et al. Intrusion detection and prevention system (idps) technology-network behavior analysis system (nbas). *ISCA J. Engineering Sci*, 1(1) :51–56, 2012.

[79] Suad Mohammed Othman, Nabeel Alsohybe, Fadl Mutaher Ba-Alwi, and Ammar Thabit Zahary. Survey on intrusion detection system types. *International Journal of Cyber-Security and Digital Forensics*, 7(4) :444–463, 2018.

[80] Suad Mohammed Othman, Nabeel T Alsohybe, Fadl Mutaher Ba-Alwi, and Ammar Thabit Zahary. Survey on intrusion detection system types. *International Journal of Cyber-Security and Digital Forensics*, 7(4) :444–463, 2018.

[81] Ahmed Patel, Mona Taghavi, Kaveh Bakhtiyari, and Joaquim Celestino Júnior. An intrusion detection and prevention system in cloud computing : A systematic review. *Journal of network and computer applications*, 36(1) :25–41, 2013.

[82] Muhammad Shakil Pervez and Dewan Farid. Feature selection and intrusion classification in nsl-kdd cup 99 dataset employing svms. In *The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014)*, pages 1–6. IEEE, 2014.

[83] Shahadate Rezvy, Miltos Petridis, Aboubaker Lasebae, and Tahmina Zebin. Intrusion detection and classification with autoencoded deep neural network. In *International Conference on Security for Information Technology and Communications*, pages 142–156. Springer, 2018.

[84] Mostafa A Salama, Heba F Eid, Rabie A Ramadan, Ashraf Darwish, and Aboul Ella Hassanien. Hybrid intelligent intrusion detection scheme. In *Soft computing in industrial applications*, pages 293–303. Springer, 2011.

[85] Mostafa A Salama, Heba F Eid, Rabie A Ramadan, Ashraf Darwish, and Aboul Ella Hassanien. Hybrid intelligent intrusion detection scheme. In *Soft computing in industrial applications*, pages 293–303. Springer, 2011.

[86] Karen Scarfone, Peter Mell, et al. Guide to intrusion detection and prevention systems (idps). *NIST special publication*, 800(2007) :94, 2007.

[87] Seethalakshmi and Nasira. Detecting and preventing intrusion in multi-tier web applications using double guard. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 3124–3127. IEEE, 2016.

[88] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. The performance of lstm and bilstm in forecasting time series. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3285–3292. IEEE, 2019.

[89] Aurobindo Sundaram. An introduction to intrusion detection. *XRDS*, 2 :3–7, 1996.

[90] Mayur Suramwar and Bansode. A survey on different types of intrusion detection systems. *International Journal of Computer Applications*, 122(16), 2015.

[91] Yacine Tamoudi, Djibrilla Amadou Kountché, Alain Ribault, and Sylvain Gombault. Reconstruction de spécifications pour la détection d'intrusion web.

[92] Tuan A Tang, Lotfi Mhamdi, Des McLernon, Syed Ali Raza Zaidi, and Mounir Ghogho. Deep learning approach for network intrusion detection in software defined networking. In *2016 international conference on wireless networks and mobile communications (WINCOM)*, pages 258–263. IEEE, 2016.

[93] Jitendra Tembhurne and Tausif Diwan. Sentiment analysis in textual, visual and multimodal inputs using recurrent neural networks. *Multimedia Tools and Applications*, 80 :1–40, 02 2021.

[94] Ankit Thakkar and Ritika Lohiya. A review of the advancement in intrusion detection datasets. *Procedia Computer Science*, 167 :636–645, 2020.

[95] Pablo Torres, Carlos Catania, Sebastian Garcia, and Carlos Garcia Garino. An analysis of recurrent neural networks for botnet detection behavior. In *2016 IEEE biennial congress of Argentina (ARGENCON)*, pages 1–6. IEEE, 2016.

[96] Rahul Vigneswaran, Vinayakumar, Soman, and Prabaharan Poornachandran. Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security. In *2018 9th International conference on computing, communication and networking technologies (ICCCNT)*, pages 1–6. IEEE, 2018.

[97] Vinayakumar, Soman, and Prabaharan Poornachandran. Applying convolutional neural network for network intrusion detection. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1222–1228. IEEE, 2017.

[98] Kehe Wu, Zuge Chen, and Wei Li. A novel intrusion detection model for a massive network using convolutional neural networks. *Ieee Access*, 6 :50850–50859, 2018.

[99] Richard Zuech, Taghi Khoshgoftaar, and Randall Wald. Intrusion detection and big heterogeneous data : a survey. *Journal of Big Data*, 2(1) :1–41, 2015.